Data management for computational research

Project organization, file formats, tidy data

Instructor: Tobin Magle, PhD

Date: July 6, 2023

Materials: https://github.com/nuitrcs/rdm-for-coding

Northwestern
INFORMATION TECHNOLOGY

Data* Management

How you store, organize, and document your work so it's understandable and reusable **.

* can extend to documents, code, notes, etc

** by you and others

Goal:

"I can walk away from the project and come back to it a year later and resume work fairly quickly"

Real life R example

the 2 other people (the post-doc whose project it is + the bioinformatician for that lab) were able to figure out what I did and decide which files they needed to look at, etc.

GOOD ENOUGH!

- Jenny Bryan, RStudio Developer

http://www2.stat.duke.edu/~rcs46/lectures_2015/01-markdown-git/slides/organization-slides/organization-slides.pdf

Real life Python example

When a potential user or contributor lands on your repository's page, they see a few things:

- Project Name
- Project Description
- Bunch O' Files

Only when they scroll below the fold will the user see your project's README.

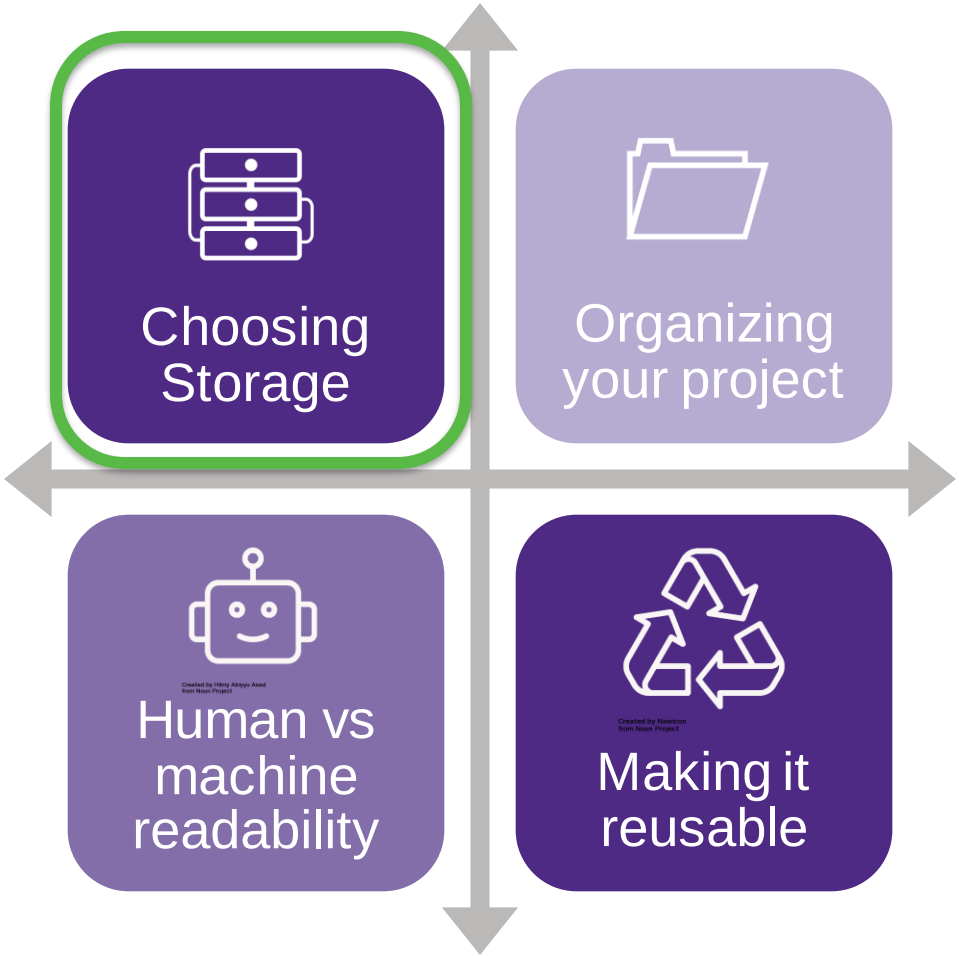
If your repo is a massive dump of files or a nested mess of directories, they might look elsewhere before even reading your beautiful documentation.

> Dress for the job you want, not the job you have.

- Hitchhikers Guide to Python

https://docs.python-guide.org/writing/structure/

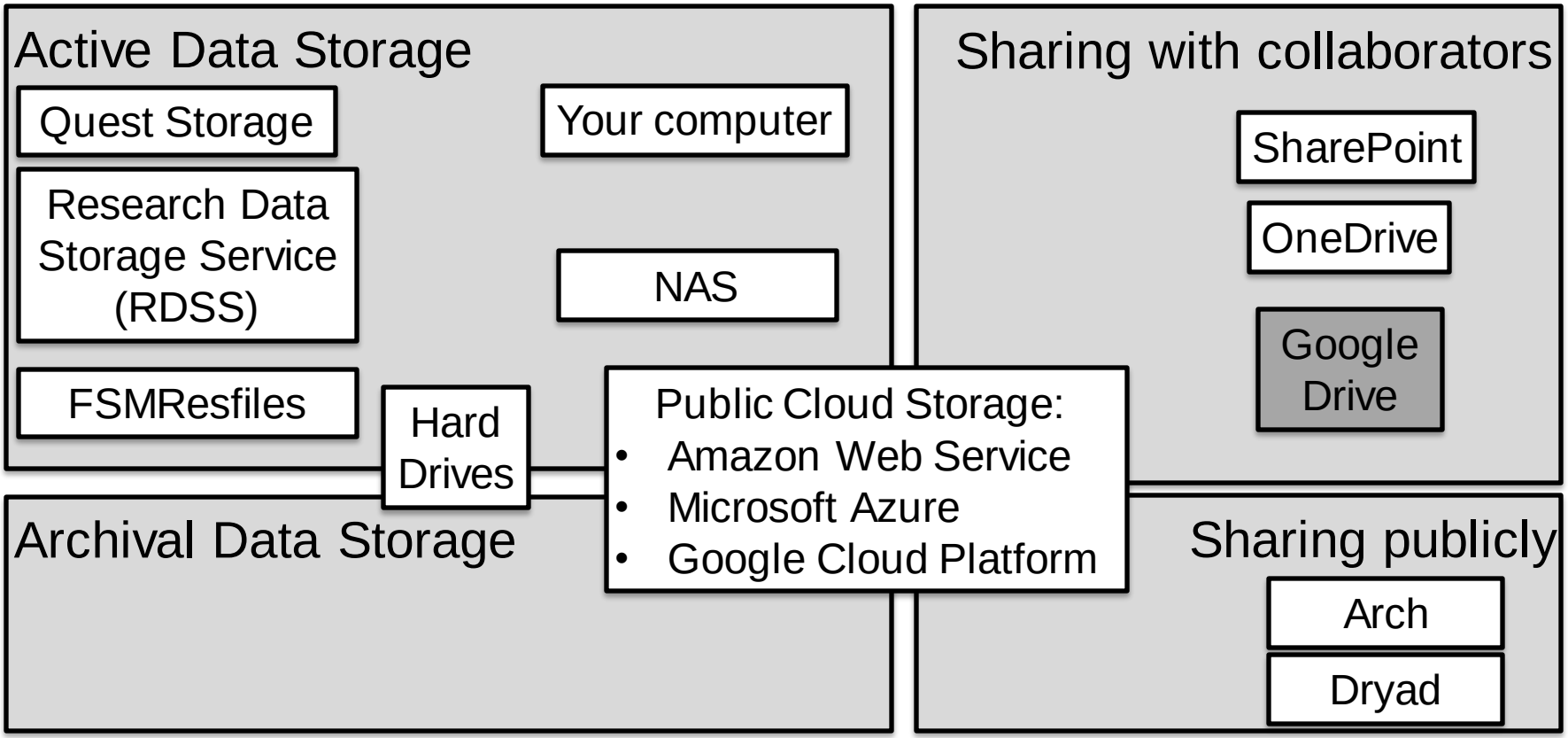
Topics


Choosing Storage
Organizing your project
Human vs machine readability
Making it reusable

Data storage

where to store it depends on what you want to do with it

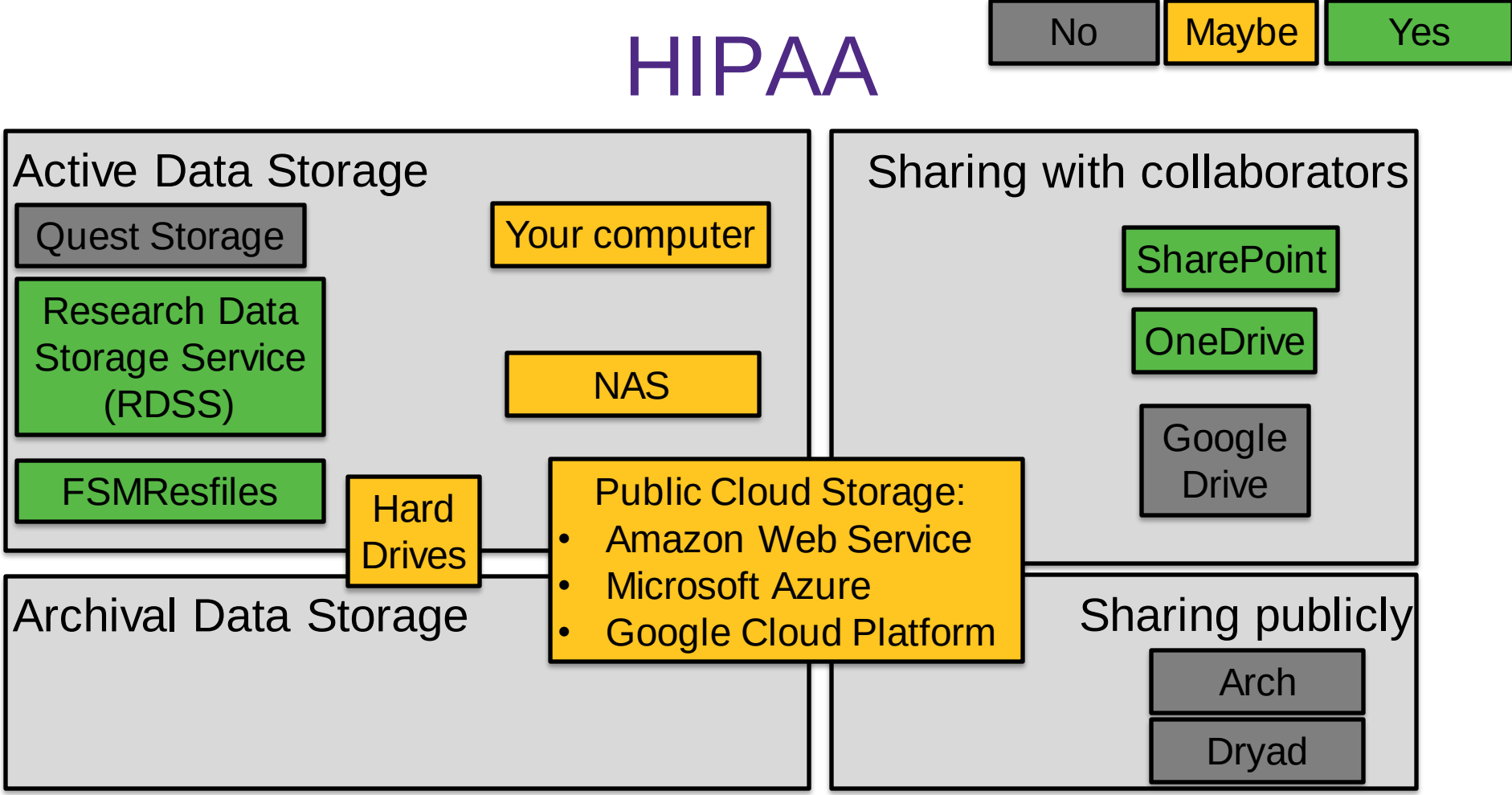
Places to store your data



Choosing data storage

- **Security**: Is your data "legally or contractually restricted"? What requirements must be satisfied?

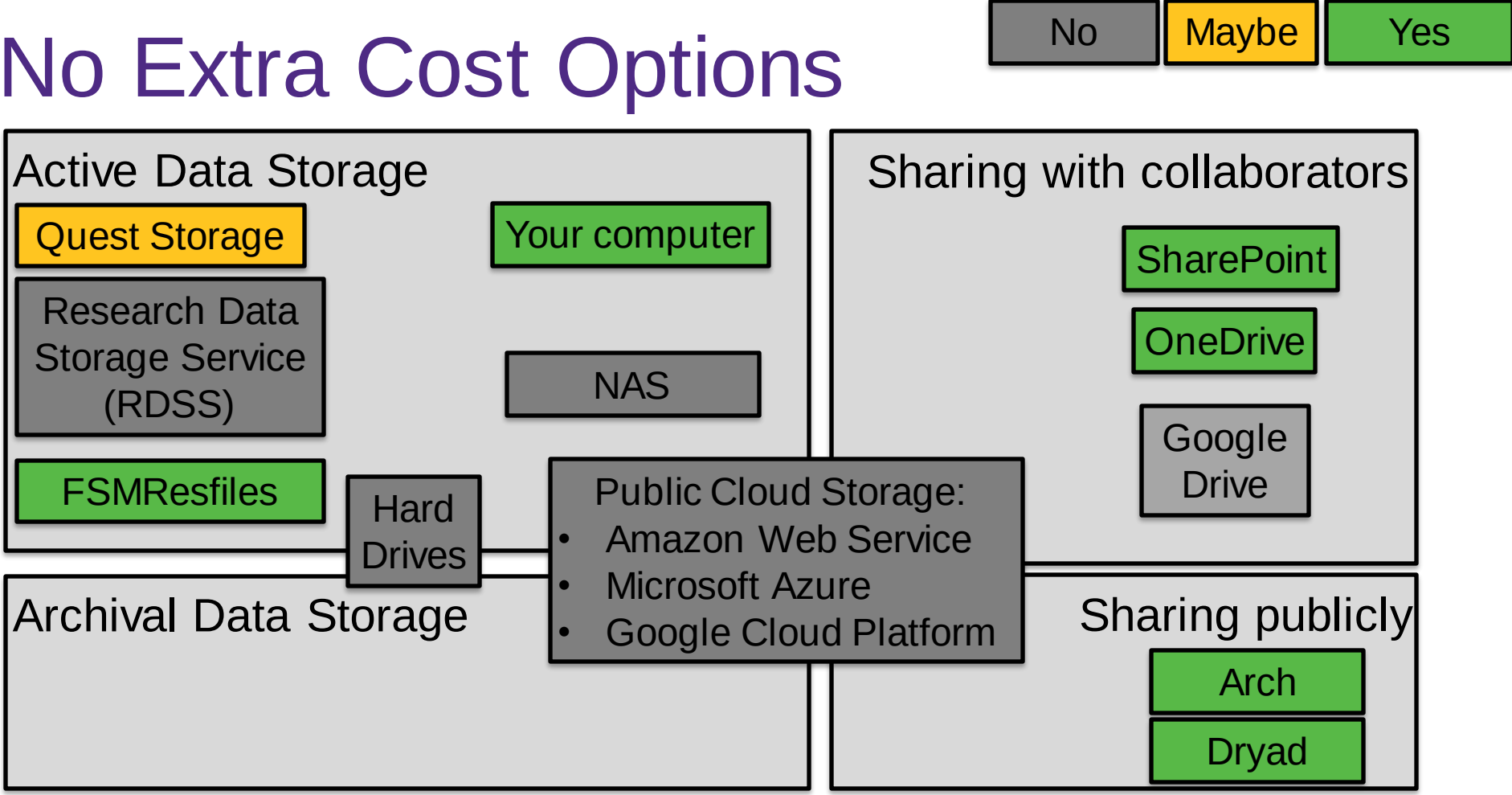
HIPAA


No
Maybe
Yes
Active Data Storage
Quest Storage
Research Data Storage Service (RDSS)
FSMResfiles
Your computer
NAS
Hard Drives
Public Cloud Storage:
• Amazon Web Service
• Microsoft Azure
• Google Cloud Platform
Sharing with collaborators
SharePoint
OneDrive
Google Drive
Archival Data Storage
Sharing publicly
Arch
Dryad

Choosing data storage

- **Security**: Is your data "legally or contractually restricted"? What requirements must be satisfied?
- **Volume:** How much data do you have? Where will it fit? How much will it cost?

No Extra Cost Options



Capacity limits

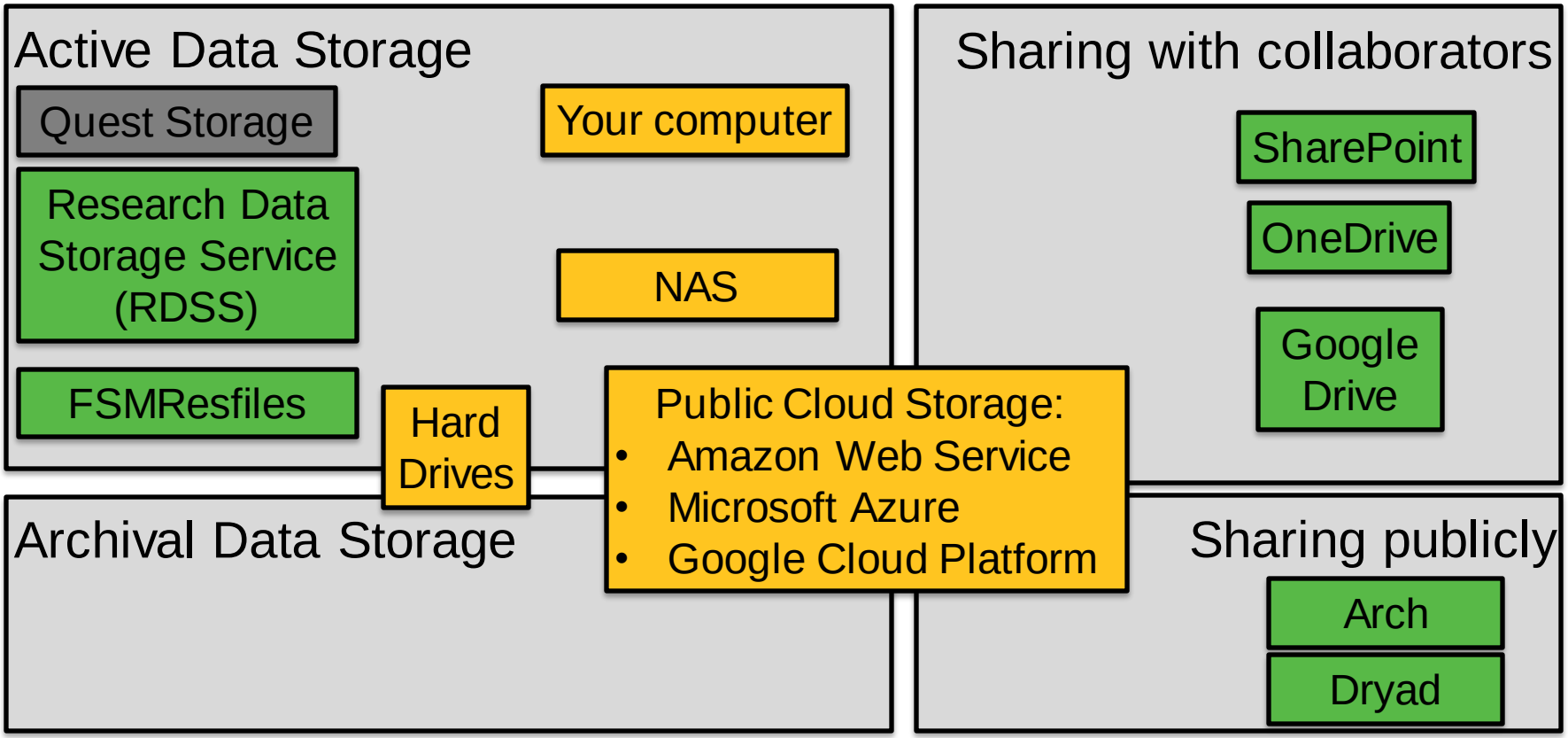
Platform	OneDrive	SharePoint	GoogleDrive	Dryad
Total Capacity	5 TB/user	25 TB/site	Unlimited*	300 GB/ submission
Individual file size	250 GB	250 GB	5 TB	10 GB
Number of files	30,000,000/ user	30,000,000/site	500,000/folder	none
Movement	250 GB or 10,000 files download		750 GB/ day upload	NA

*Not for long

Choosing data storage

- **Security**: Is your data "legally or contractually restricted"? What requirements must be satisfied?
- **Volume:** How much data do you have? Where will it fit? How much will it cost?
- **Durability**: Does your data storage back it up for you?

Storage that has backups



Choosing data storage

- **Security**: Is your data "legally or contractually restricted"? What requirements must be satisfied?
- **Volume:** How much data do you have? Where will it fit? How much will it cost?
- **Durability**: Does your data storage back it up for you?
- **Access**: Who needs access to your data? What level?

Access

Platform	Quest	RDSS	SharePoint	GoogleDrive	Dryad
Who can get access?	Anyone with a Quest account	Anyone with a NetID	Anyone with Microsoft account	Anyone with Google account	Everyone (public)
Types of access	Whole project, permissions can be edited by file/folder	Whole Share	Whole library, By file or folder	By file/folder	public
Permissions	read/write/ execute	read only or read/write	View, edit or review	View, edit, comment	Read only

Choosing data storage

- **Security**: Is your data "legally or contractually restricted"? What requirements must be satisfied?
- **Volume:** How much data do you have? Where will it fit? How much will it cost?
- **Durability**: Does your data storage back it up for you?
- **Access**: Who needs access to your data? What level?
- **Workflow**: Where is the data produced? Where do you analyze it?

Data workflows

Make sure your data is **accessable to your compute**

or

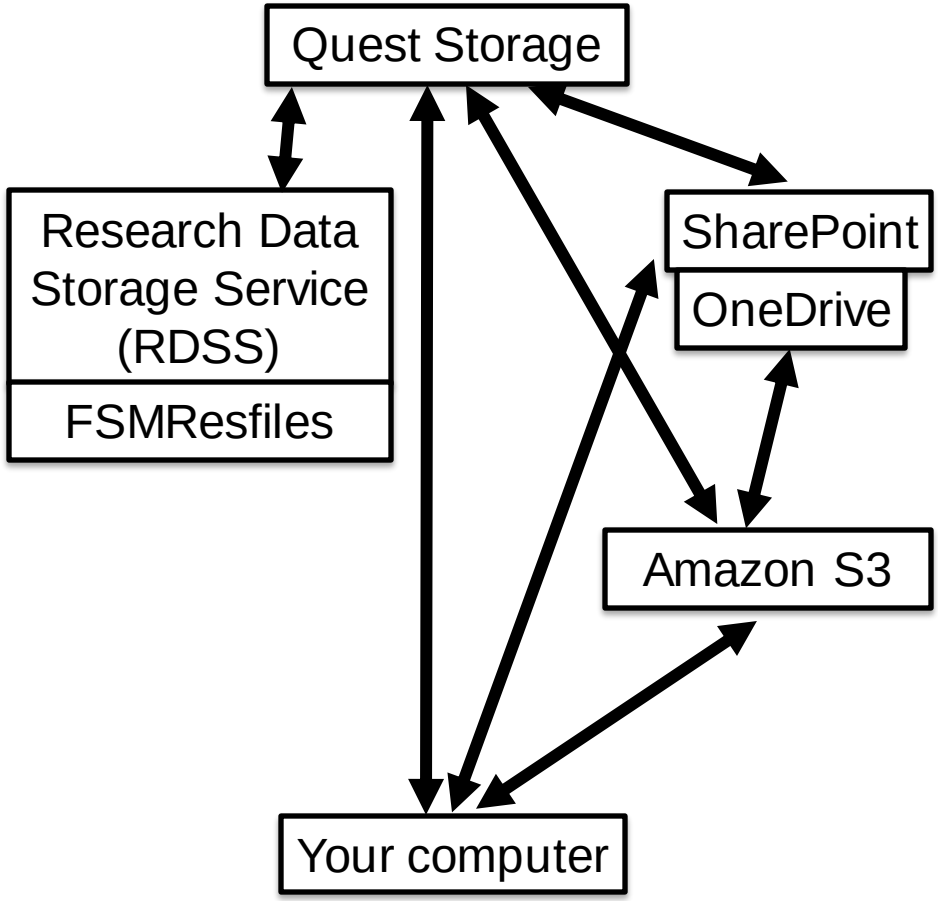
Have a plan to **move it**

Compute Sources

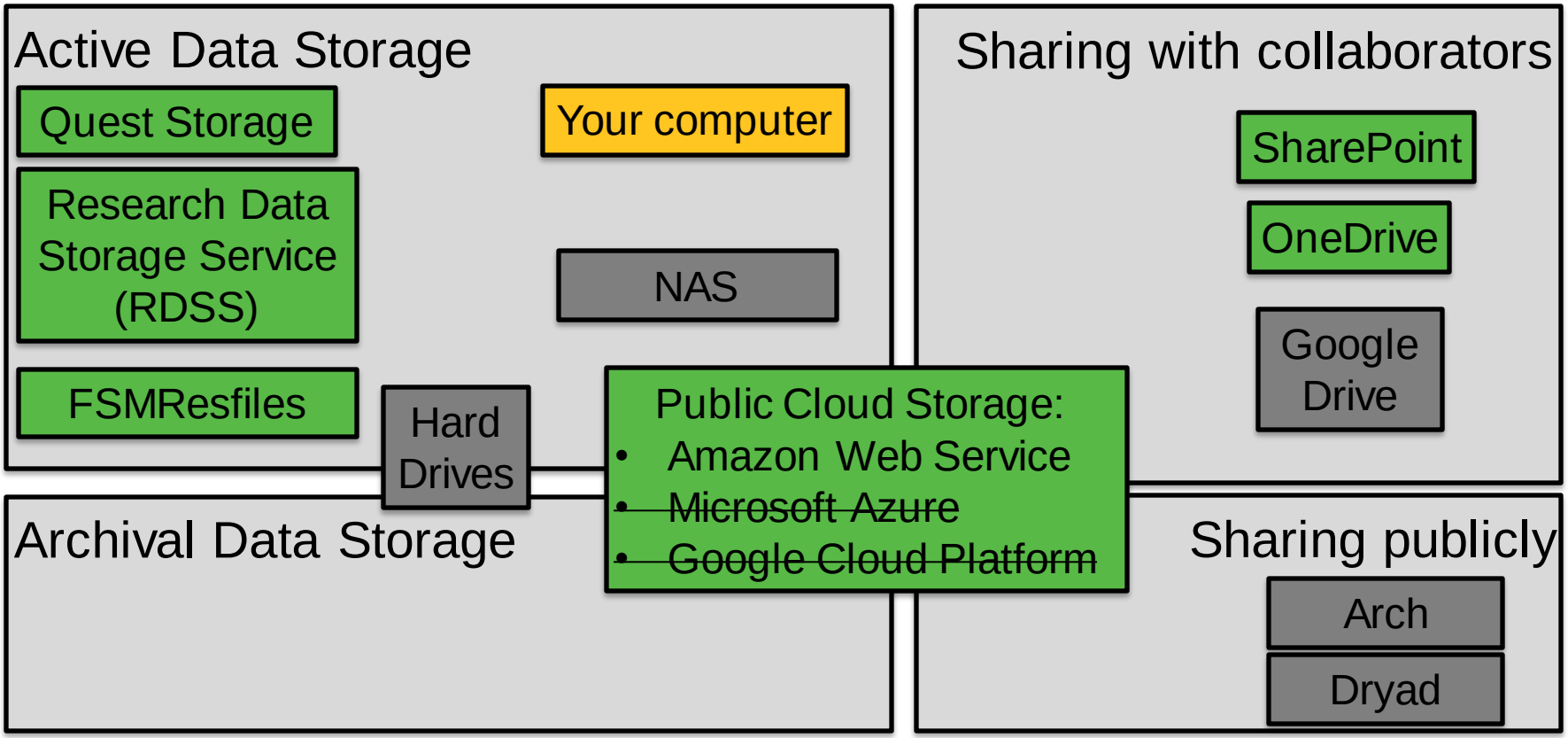
Compute Source	Where to store data	Comments
Your computer	Your hard drive	Might not be big enough Needs a backup
	External drive	Might not be fast enough
	Network Drive (RDSS)	Might not be fast enough
Quest	Quest storage	Only option Needs a backup location
"The Cloud"	In the cloud	Pay per use Moving data out costs $$

Tool: Globus

- Designed for large file transfer
- Transfer and sync data between "collections"
  - Many NU storage systems
  - Your computer
  - More to come this fall
- Error handling



Globus connected



Example Workflow

Instrument

You're using data produced at a core facility.

?

Quest Storage

SharePoint

You need to analyze it on Quest, then transfer the results to SharePoint to collaborate on a paper

Direct to Quest

Instrument

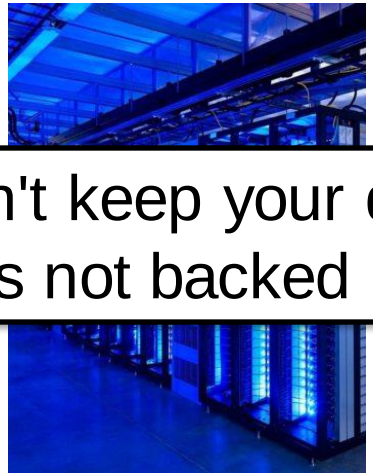
Quest Storage

SharePoint

Direct to Quest

Instrument

Quest Storage



SharePoint

- Core facility won't keep your data
- Quest Storage is not backed up

Via your computer

Instrument

Your computer

Quest Storage

SharePoint

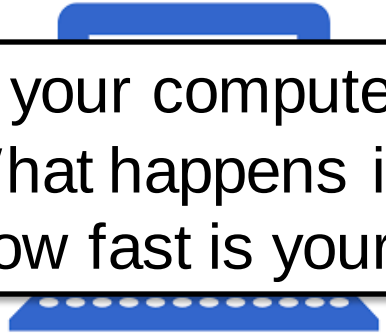
Via your computer

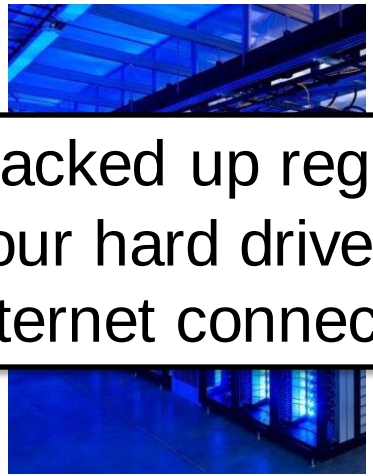
Instrument

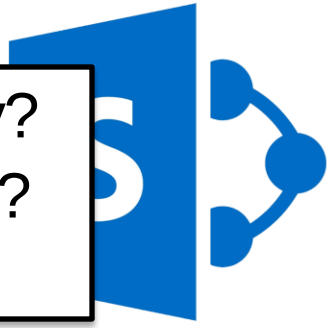
Your computer

Quest Storage

SharePoint





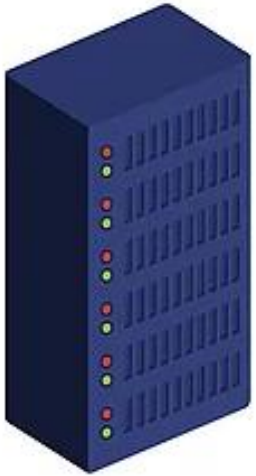


- Is your computer backed up regularly?
- What happens if your hard drive dies?
- How fast is your internet connection?

Via RDSS/FSMResfiles

Instrument

RDSS

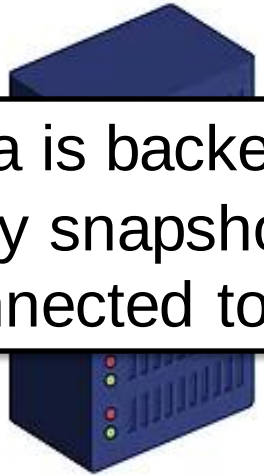


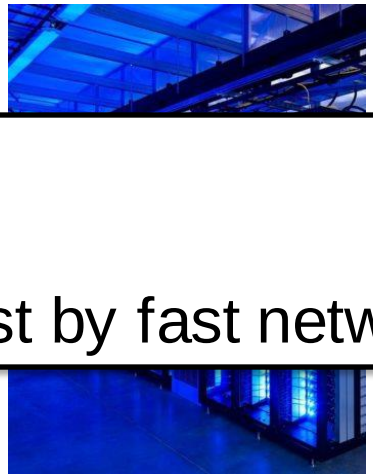
Quest Storage

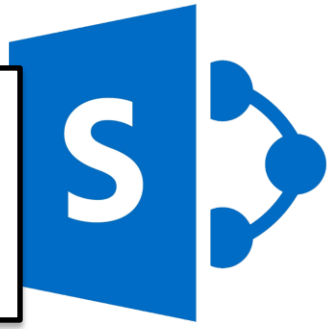
SharePoint

Via RDSS/FSMResfiles

Instrument | RDSS | Quest Storage | SharePoint

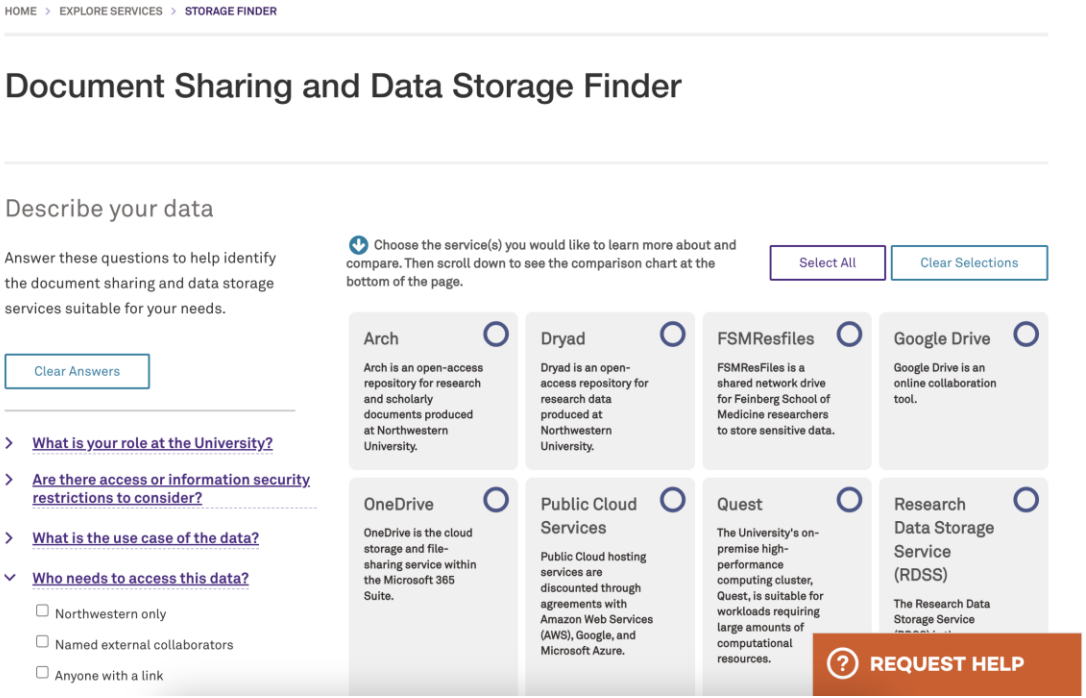






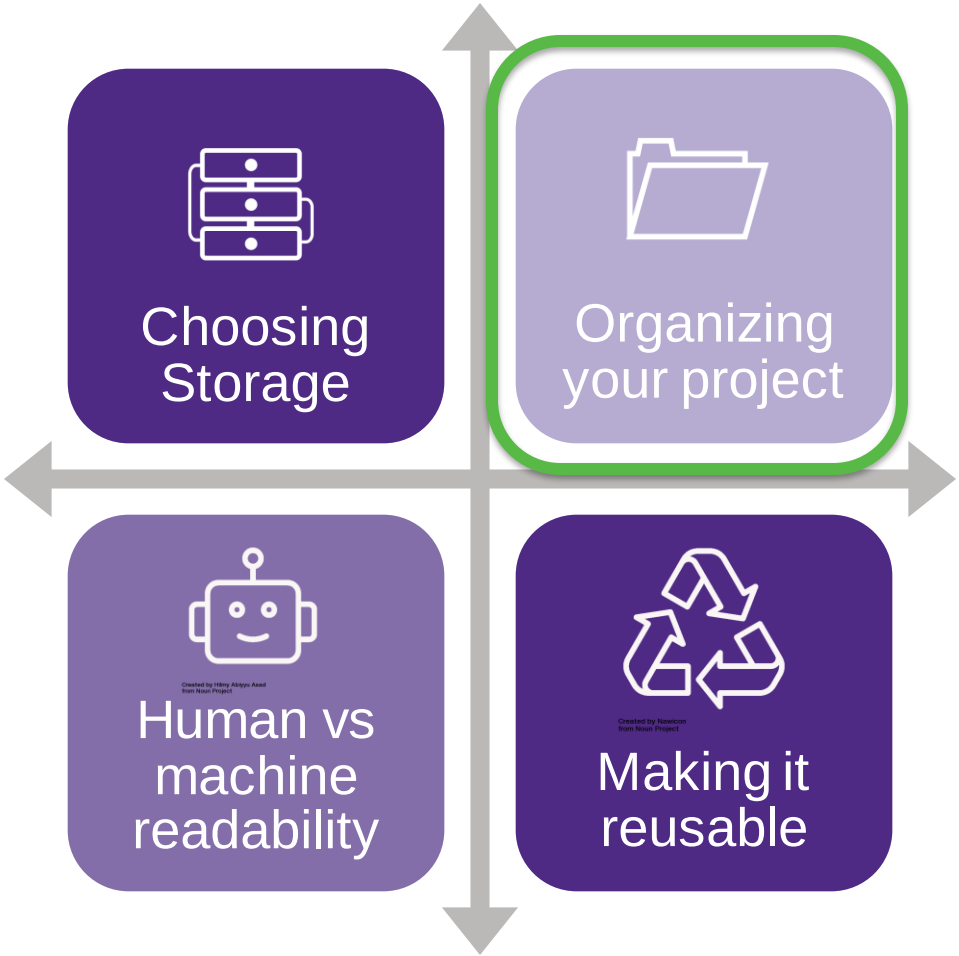
- Data is backed up
- Daily snapshots
- Connected to Quest by fast network

Finding data storage at Northwestern



https://www.it.northwestern.edu/services/storage-finder/

Topics


Choosing Storage
Organizing your project
Human vs machine readability
Making it reusable

Organization

Takes time up front
Saves time and frustration later

Based on material from https://kbroman.org/

Good Organization Practices

- Put everything for a project in one folder
- Create a subfolder for each file type
- Use descriptive names (more on this later)
- Make a README (more later)

Goal: Avoid Chaos

```
AimeeNullSims/     Deuterium/
AimeeResults/      ExtractData4Gary/
AnnotationFiles/   FromAimee/
Brian/             GoldStandard/
Chr6_extrageno/    HumanGWAS/
Chr6_segdis/       Insulin/
ChrisPlaisier/     Int2_for_Mark/
Code4Aimee/        Islet_2011-05/
CompAnnot/         MappingProbes/
CondScans/         MultiProbes/
D20_2012-02-14/    NewMap/
D20_cellcycle/     Notes/
D20corr/           NullSims/
Data4Aimee/        NullSims_2009-09-10/
Data4Tram/         PepIns_2012-02-09/
```

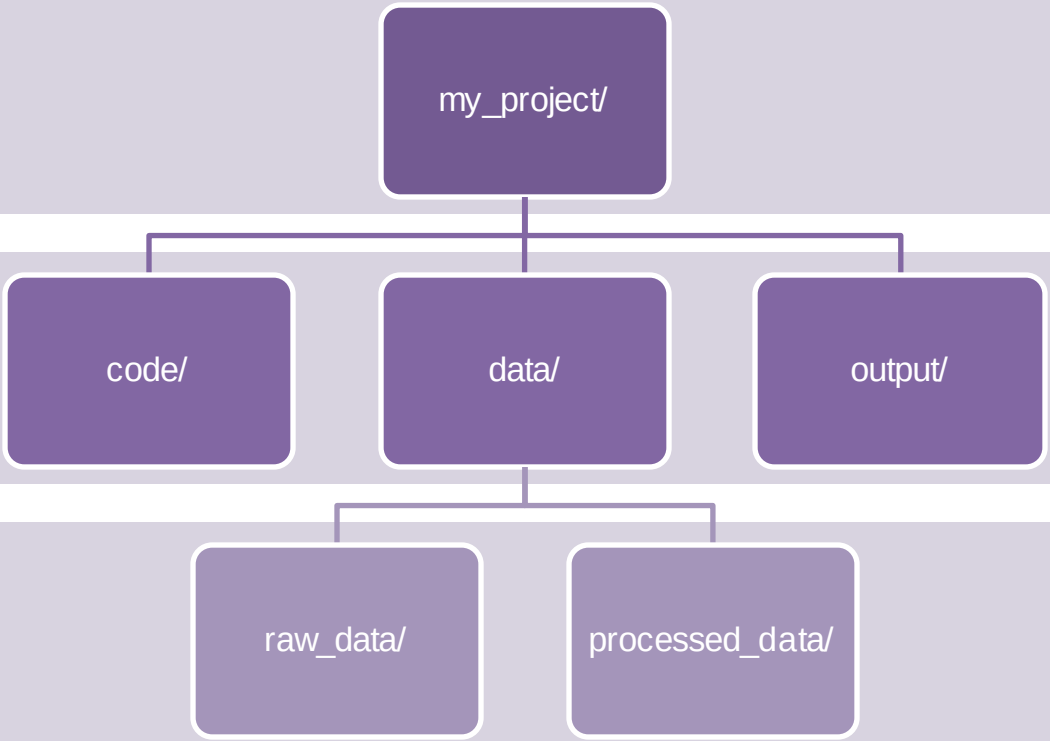
https://kbroman.org/Tools4RR/assets/lectures/06_org_eda_withnotes.pdf

Example Basic Project

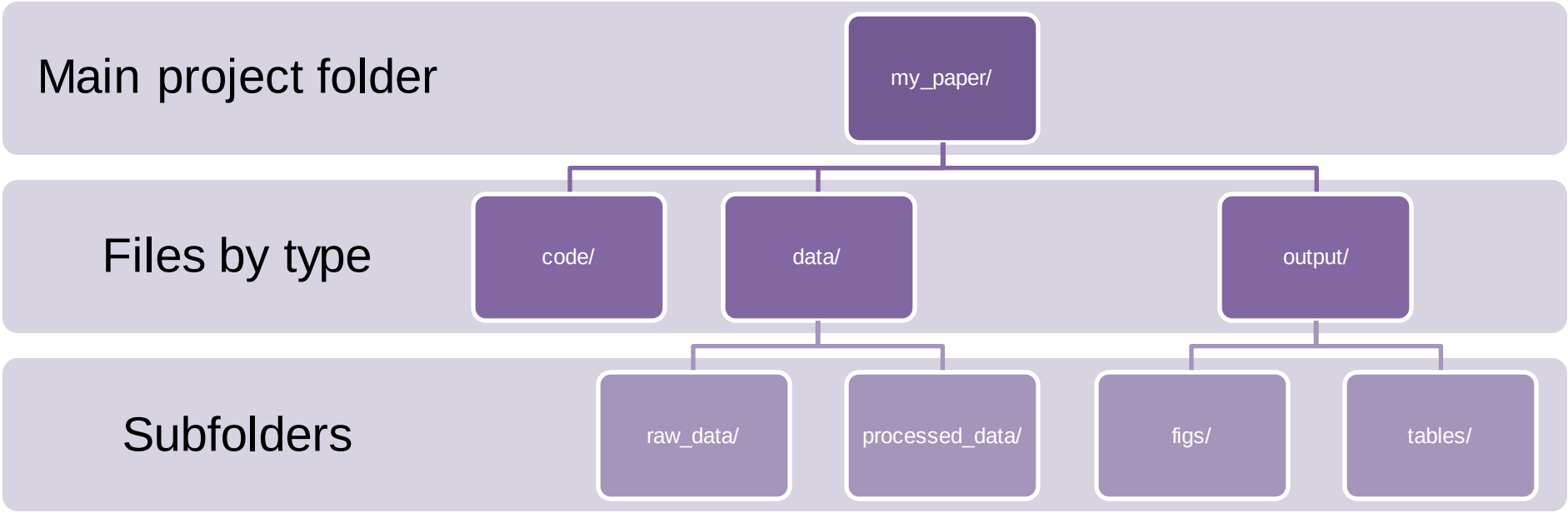
Main project folder

Files by type

Subfolders



Example Paper Project



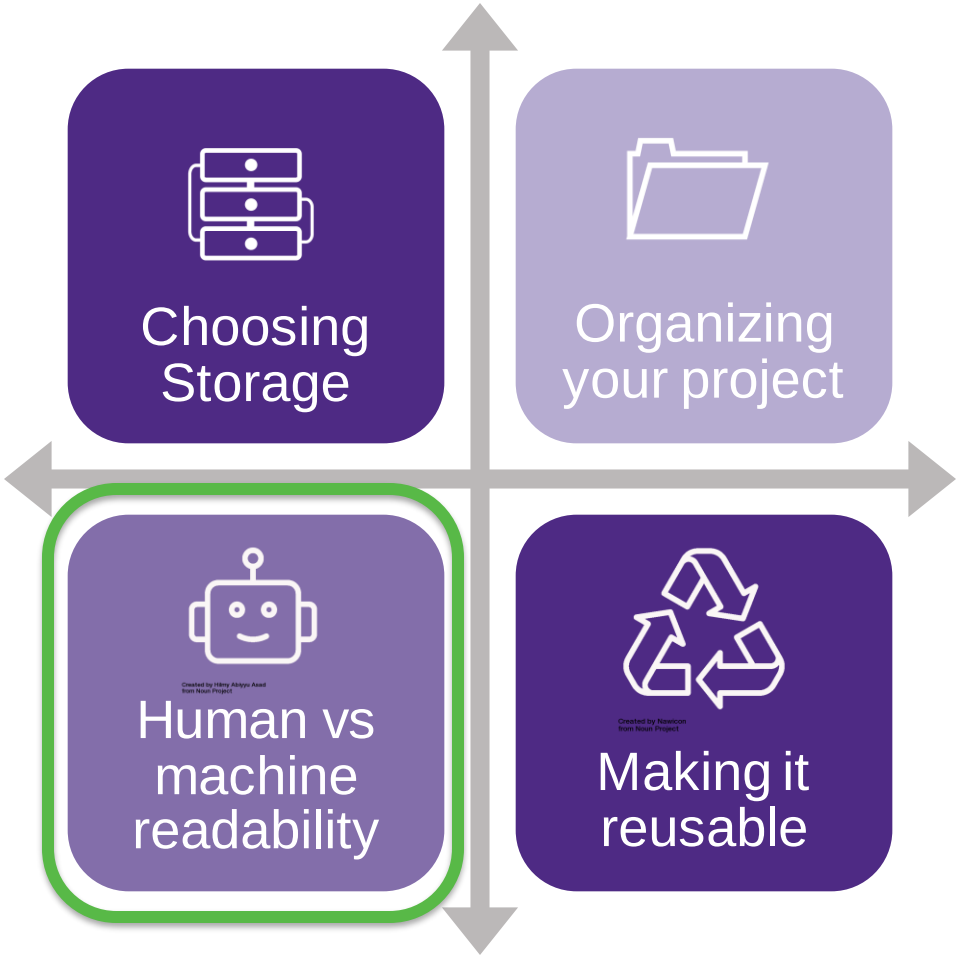
Project Organization Advice

There's no one right answer

- Make a system that works for you
- Be consistent!
- Stick to it

Examples in R, python and bash
https://github.com/moldach/project-directory

Topics

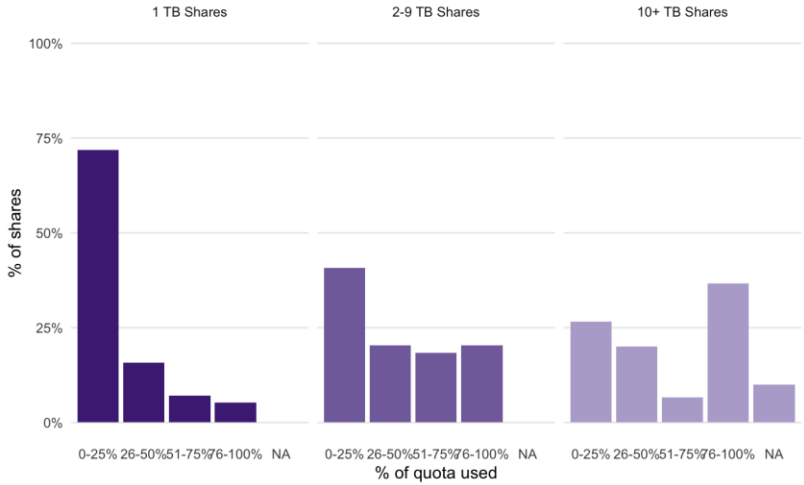

Choosing Storage
Organizing your project
Human vs machine readability
Making it reusable

Human vs. Machine readability

Machine readable

```
rdss_shares<-read_csv(combined_shares)%>%
  #filter(!is.na(quota_isilon))%>%
  mutate(quota = quota_isilon,
         used = used*byte_conversion,
         percent_used = round(100*used/quota,10))%>%
  select("share", quota, "used", "percent_used", "zone", "unit", "department",
         "chart_string")%>% #,
         #"rw_users", "ro_users")
  distinct()%>%
  mutate(size = fct(case_when(quota == 1 ~ "1 TB",
                              quota <10 ~ "2-9 TB",
                              TRUE~"10+ TB"), levels = c("1 TB", "2-9 TB", "10+ TB")),
         sharebin = case_when(quota <= 1 ~ "1",
                              quota < 10 ~ "2-9",
                              quota < 20 ~ "10-19",
                              quota < 50 ~ "20-49",
                              quota < 100 ~ "50-99",
                              TRUE ~ "100+"),
         # have to factor to get them in the right order in the plot
         sharebin = factor(sharebin, levels=c("1", "2-9", "10-19", "20-49", "50-99", "100+")
                           ),
```

Human Readable

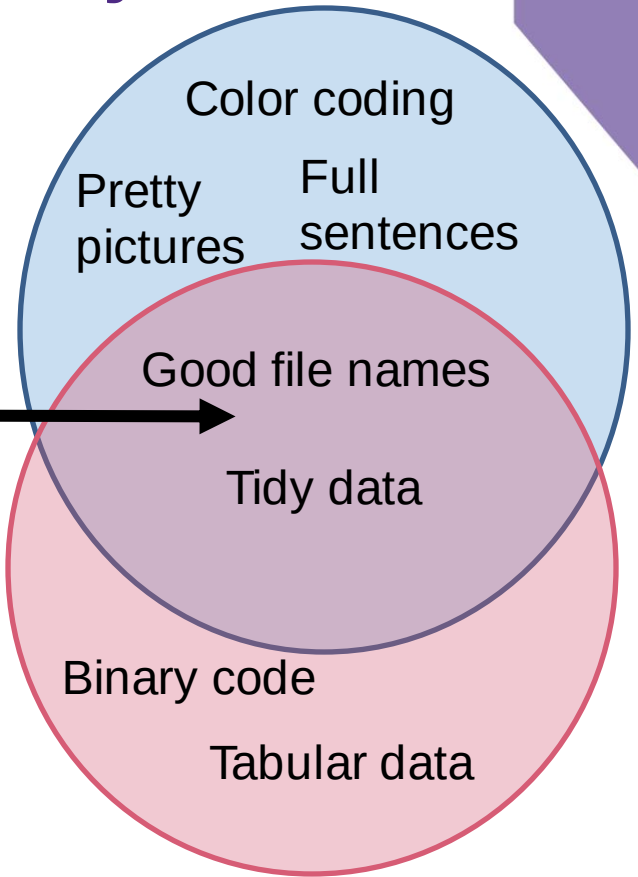


Goal: Make your project...

human readable

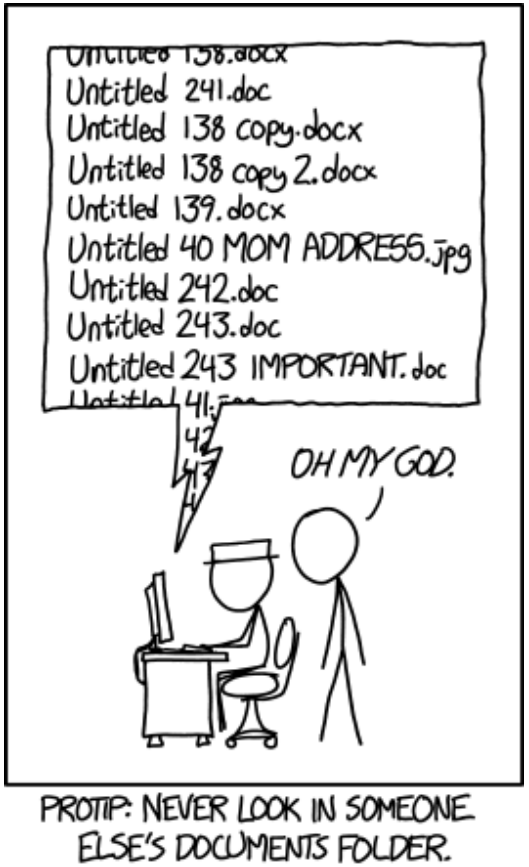
AND

machine readable



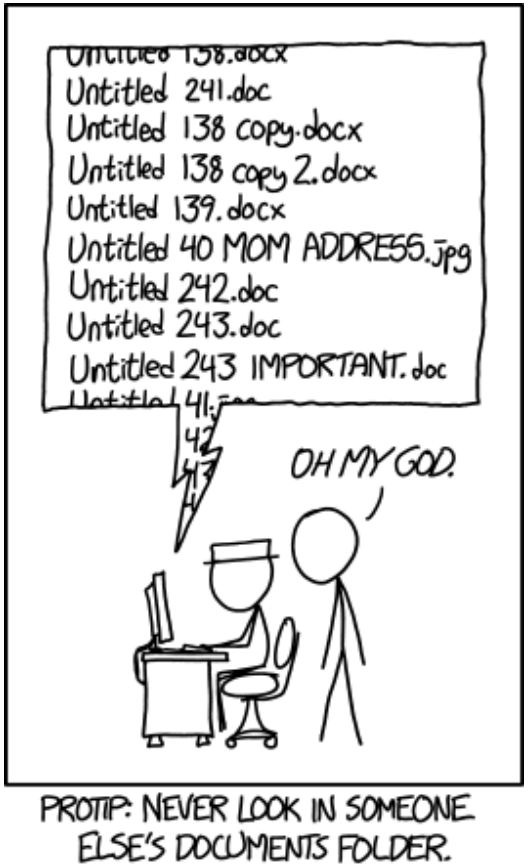
File naming

- Programs do not produce good default names
- Bad file names can waste time
- Create a file naming convention that works for you and stick to it



File naming

- Programs do not produce good default names
- Bad file names can waste time
- Create a file naming convention that works for you and stick to it



Good file naming practices

- Use **descriptive names**
- **No spaces** in file names
- **Limit special characters** in file names
- Be **consistent**
- **Use existing standards**
- Use **sort** to your advantage

Use descriptive names

- Names should tell you what's in the file
- Non-specific names don't help you or the computer
- Use appropriate file extensions
- Make sure to include information that differentiates files

Not specific

script.R
document 1.txt
script.txt
notes.md

Specific

clean-the-data.R
references.txt
outlier-analysis.py
README.md

Spaces in file names

- Human readable, but..
- Computers use spaces as a delimiter
- Replace with _ or -

```
$ ls
file 1.txt file2.txt

$ ls file 1.txt
ls: 1.txt: No such file or directory
ls: file: No such file or directory

$ ls file2.txt
file2.txt
$
```

Limit special characters

- Characters have special meanings to operating systems use
- Don't confuse your OS
- Ex: bash
- Underscore (_) and dash (-) are ok

Character	Meaning
#	Comment
$	Variable expression
&	Background job
*	Wildcard
( )	Groups commands
;	Separates commands
' or "	Quotes phrases
/	Pathname directory separator
?	Single-character wildcard
!	Pipeline logical NOT

https://www.oreilly.com/library/view/learning-the-bash/

Be consistent

- Make it easy to find the information you need
- Things to check for consistency: dates, capital letters, order, abbreviations
- Use existing standards

Inconsistent

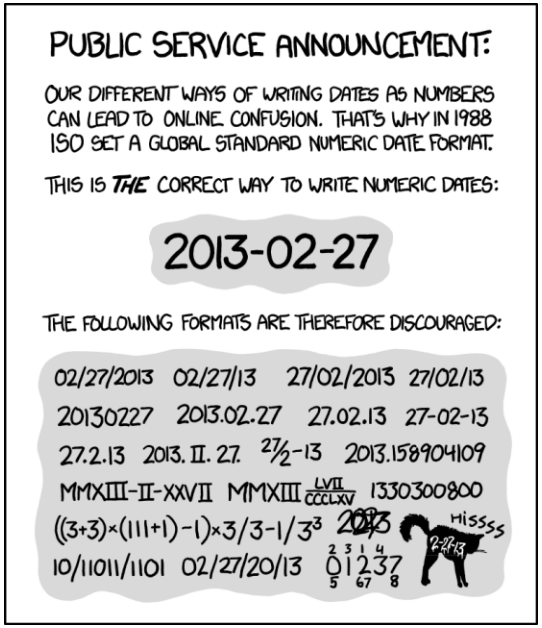
2023-07-05-geo-Chicago.csv
Chicago-June2023-Geospatial.csv
July22-23-Austin-geo.csv
Austin-July-2023-income.csv

Consistent

07-23-Chicago-geospatial.csv
06-23-Chicago-geospatial.csv
07-23-Austin-geospatial.csv
07-23-Austin-income.csv

Use existing standards

- Dates!
- What do other people in your research group do?
- What do people in your field do?



https://xkcd.com/1179/

Use sort to your advantage

- Your computer is good at sorting
- Name your files so it sorts them in a useful way
- General to specific

Not useful

06-23-Chicago-geospatial.csv
07-23-Austin-Geospatial.csv
07-23-Austin-income.csv
07-23-Chicago-geospatial.csv

Useful

2023-06-geospatial-Chicago.csv
2023-07-geospatial-Chicago.csv
2023-07-geospatial-Austin.csv
2023-07-income-Austin.csv

File naming advice

- Name your files so you know what it is before you open it
- Don't use spaces and special characters
- Pick a convention and stick to it

Tidy Data

Tidy datasets are easy to manipulate, model and visualize

- Many reasons
  - Different collection methods
  - Inconsistent units
  - Human error
- Cleaning takes a lot of time

Tidy datasets are all alike but every messy dataset is messy in its own way.

-Hadley Wickham

Tidy data structure

- Each **variable** forms a column.
- Each **observation** forms a row.
- Each type of **observational unit** forms a table.

https://vita.had.co.nz/papers/tidy-data.pdf

Common problems

- **Multiple variables** in one column
- Variables spread **across rows**
- Variables spread **across columns**
- **Mixing observational units** in a table
- **Spreading an observational unit** across tables.

https://vita.had.co.nz/papers/tidy-data.pdf

Multiple Variables in a column

Monitoring birds in local parks

- Observation: one bird
- Variables: park, date, bird
  - Bird = species + sex
- Can't analyze species and sex separately

park	date	bird
Penny	2023-07-04	Cardinal-M
Penny	2023-07-04	Cardinal-F
James	2023-07-05	Cardinal-M
James	2023-07-05	Sparrow-F
James	2023-07-05	Sparrow-M
James	2023-07-05	RWBB-F

Multiple Variables in a column

Solution

- Make a column for each variable

park	date	species	sex
Penny	2023-07-04	Cardinal	M
Penny	2023-07-04	Cardinal	F
James	2023-07-05	Cardinal	M
James	2023-07-05	Sparrow	F
James	2023-07-05	Sparrow	M
James	2023-07-05	RWBB	F

Variables spread across rows

Measuring pet weight and height at a vet clinic

- Observation: 1 pet
- 2 rows per pet
- How do we fix it?

Name	species	variable	value
Tybalt	cat	weight	5
Madden	dog	weight	4
Tybalt	cat	height	3
Madden	dog	height	5

https://vita.had.co.nz/papers/tidy-data.pdf

Variables spread across rows

Solution

- Move unique values in 'variable' to column headers
- Move values to corresponding cells

Name	species	height	weight
Tybalt	cat	3	5
Madden	dog	5	4

https://vita.had.co.nz/papers/tidy-data.pdf

Variables spread across columns

Experiment with 3 treatments and 3 replicates

- Good for compactness
- Human readable
- Can't analyze the replicates together

Treatment	1	2	3
control	0	1	0.5
A	5	6	4
B	4	3	1

https://vita.had.co.nz/papers/tidy-data.pdf

Variables spread across columns

Solution

- Add a column for "replicate" (variable)
- Pivot the values into one column

treatment	replicate	value
control	1	0
A	1	5
B	1	4
control	2	1
A	2	6
B	2	3
control	3	0.5
A	3	4
B	3	1

Mixing observational units

Measuring pet weight **over time**

- Duplicates information (owner, species)
- Not a problem until you get into really big data

day	owner	name	species	weight
1	Toby	Tybalt	cat	8.0
1	Arden	Madden	dog	30.3
7	Toby	Tybalt	cat	8.1
7	Arden	Madden	dog	28.5
14	Toby	Tybalt	cat	8.2
14	Arden	Madden	dog	29.8

Mixing observational units

Solution

- Make 2 tables:
  - Pet table
  - Weight table
- Foundation of creating relational databases

owner	name	species
Toby	Tybalt	cat
Arden	Madden	dog

day	name	weight
1	Tybalt	8.0
1	Madden	30.3
7	Tybalt	8.1
7	Madden	28.5
14	Tybalt	8.2
14	Madden	29.8

Splitting observational units

Measuring pet weight **over time**

- Splitting up data by variable categories or by time
- Intuitive, but inefficient
- Harder to catch inconsistencies

Day 1	
name	weight
Tybolt	8.0
Madden	30.3

D7	
name	weight
Tybalt	8.1
Maddan	28.5

Day 14	
name	wt
Tybalt	8.2
Madden	29.8

Splitting observational units

Solution

- Make sure format is consistent across tables
- Combine into 1 dataset

day	name	weight
1	Tybalt	8.0
1	Madden	30.3
7	Tybalt	8.1
7	Madden	28.5
14	Tybalt	8.2
14	Madden	29.8

Column Names

- Column headers become variable names
- **Human readable**: Aim for descriptive name
  - Avoid abbreviations
- **Machine readable**: Avoid spaces and most special characters
- Be consistent!

Naming convention	Example
Camel case	speciesName
Snake case	species_name
Kabob case	species-name
Dot case	species.name*

* Can cause issues in Python with pandas

How can we improve this table?

Measuring animal weight by sex and species

- Is each column a separate variable?
- Is each value in the column the same format
- Can a computer read the column headers?

Date collected	plot	Species-sex	Weight
Jan 9, 1978	1	DM-M	40
1/9/1978	1	DM-F	36 g
1/9/78	1	DS-F	135
1/20/78	2	DM-M	38g
1/20/78	2	DS-f	.144 kg
03/13/1978	2	DM-F	44
3/13/78	2	DS-F	146

Structural Changes

- Use consistent header format
- Make a column for each variable

date_collected	plot	species	sex	weight
Jan 9, 1978	1	DM	M	40
1/9/1978	1	DM	F	36 g
1/9/78	1	DS	F	135
1/20/78	2	DM	M	38g
1/20/78	2	DS	f	.144 kg
03/13/1978	2	DM	F	44
3/13/78	2	DS	F	146

Content Changes

- Make sure codes for male/female are consistent
- Use a consistent format within columns
- Make sure weights are numbers

date_collected	plot	species	sex	weight_g
1/9/1978	1	DM	M	40
1/9/1978	1	DM	F	36
1/9/1978	1	DS	F	135
1/20/1978	2	DM	M	38
1/20/1978	2	DS	F	144
3/13/1978	2	DM	F	44
3/13/1978	2	DS	F	146

Topics


Choosing Storage
Organizing your project
Human vs machine readability
Making it reusable

Reusability

You mostly collaborate with yourself, and me-from-two-months-ago never responds to email.
- @mtholder

The results in Table 1 don't seem to correspond to those in Figure 2.

How did I make that figure?

"Your script is now giving an error."

```
Karl -- this is very interesting,
however you used an old version of
the data (n=143 rather than n=226).

I'm really sorry you did all that
work on the incomplete dataset.

Bruce
```

In what order do I run these scripts?

Where did we get this data file?

"The attached is similar to the code we used."

Which image goes with which experiment?

Why did I omit those samples?

Adapted from https://www.biostat.wisc.edu/~kbroman/presentations/steps2rr.pdf

Basic: Make it reproducible

- Automation
  - Keep the raw data raw
- Documentation
  - README / code books
- Portability
  - File paths
- Version control

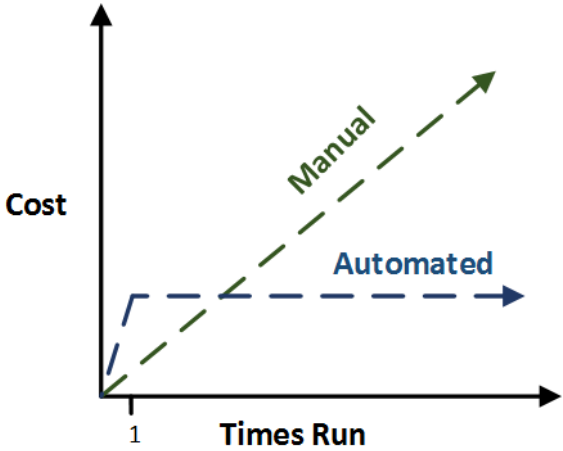
organize the data and code in a way
that you can hand them to someone else
and they can re-run the code
and get the same results

- @kbroman

Automate everything

(even the things that take longer to code than if you did it by hand)

- It's really hard to document things you do by hand
- Don't touch the raw data – mark as read only
- Write code that cleans the data and makes any output you need
- Then you have a pipeline for when you get another similar file



What goes in a README file?

Tell you-from-6-months-ago how to use the project

- **Context**: How the data was produced
- **Data inventory**: list of files and folders and what they contain
- **Codebook**: Describe the contents of data files
- **Instructions**: how to use the code and data together

What goes in a Codebook?

Describe the contents of data files

- What variables measure
- What type is it?
- Units and any other format (dates, geographic coordinates)
- Define valid data range and missing values

Portability

Will your code work for someone else?

- Do you use file paths only you have?
- Are all the components you need to run the code in one folder?

If the first line of your R script is

```
setwd("C:\Users\jenny\path\that\only\I\have")
```

I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.

https://www.tidyverse.org/blog/2017/12/workflow-vs-script/

Version control

Show how you got from A to B

By hand: Save As

- file naming
  - V1,V2 etc
  - Dates
  - Never "Final"
- Takes precision
- Creates so many files

Formal: Version control system (Git)

- Unlimited undo
- Creates an ID for each "commit"
- Good for collaboration (who did what when)
- Steep learning curve
- Versions hidden behind latest

Reusability Advice

A little bit reusable is better than nothing

- Keep the raw data raw – read only
- Document what is there
- Use self-contained projects
- Use version control

Think about your workflow + back it up.

Choosing Storage

Organizing your project

Keep it all in one place + be consistent.

Names + data structure are important.

Human vs machine readability

Making it reusable

Automate + document everything.

Questions?

References

- Organization by Jenny Bryan for the Reproducible Science Workshop at Duke University
- Structuring your project by Kenneth Reitz and Tanya Schlusser in the Hitchhiker's Guide to Python
- Document Sharing and Data Storage Finder from Northwestern IT
- Tools for reproducible research by Karl Broman from https://kbroman.org/
- Best Practices for Data Science Project Workflows and File Organization by Matthew Oldach
- Naming things by Jenny Bryan for the Reproducible Science Workshop
- Tidy data by Hadley Wickham in the Journal of Statistical Software (2014) *59*(10), 1–23
- Steps Toward Reproducible Research by Karl Broman from https://kbroman.org/
- Project-oriented workflow by Jenny Bryan from tidyverse.org blog

Thank You