Data collection, curation and validation on the traits wheel™

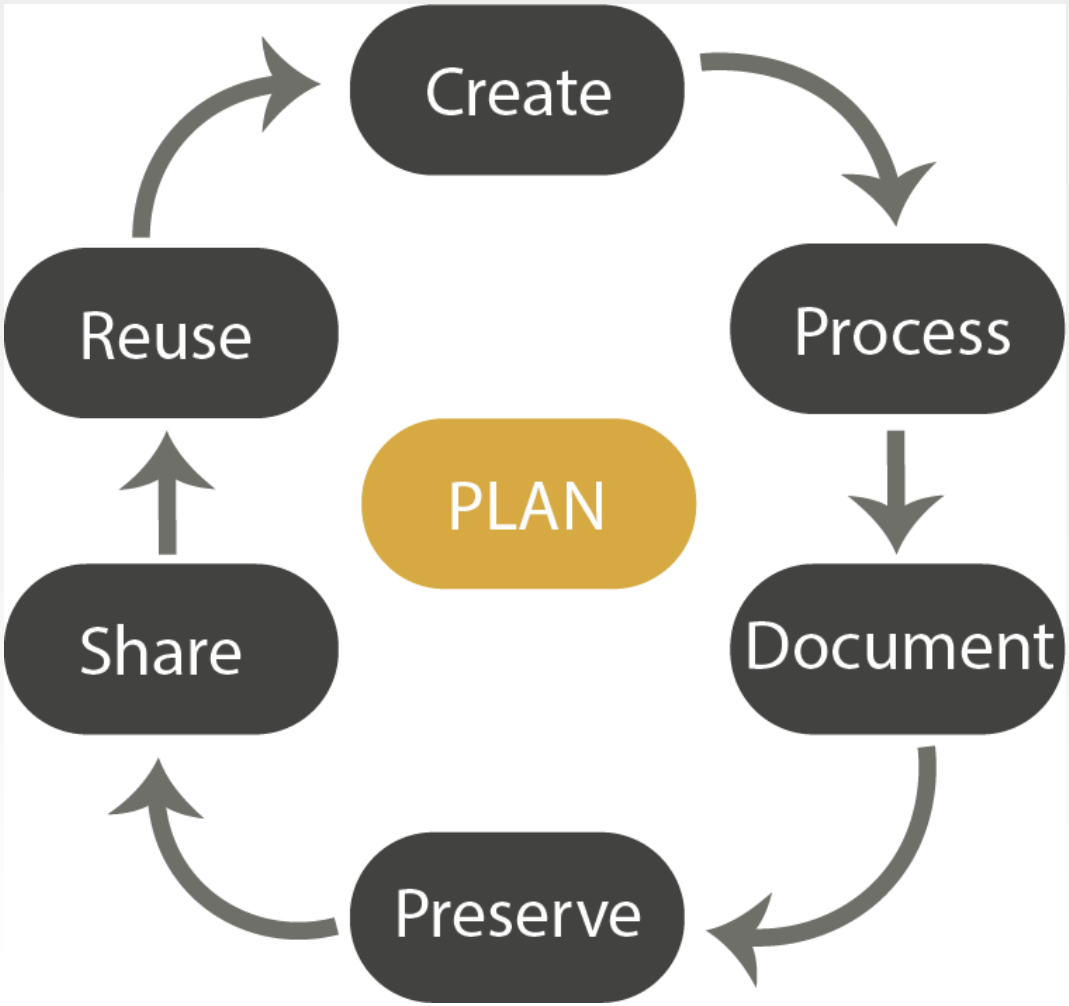
PFTC5 – Peru

Aud Halbritter

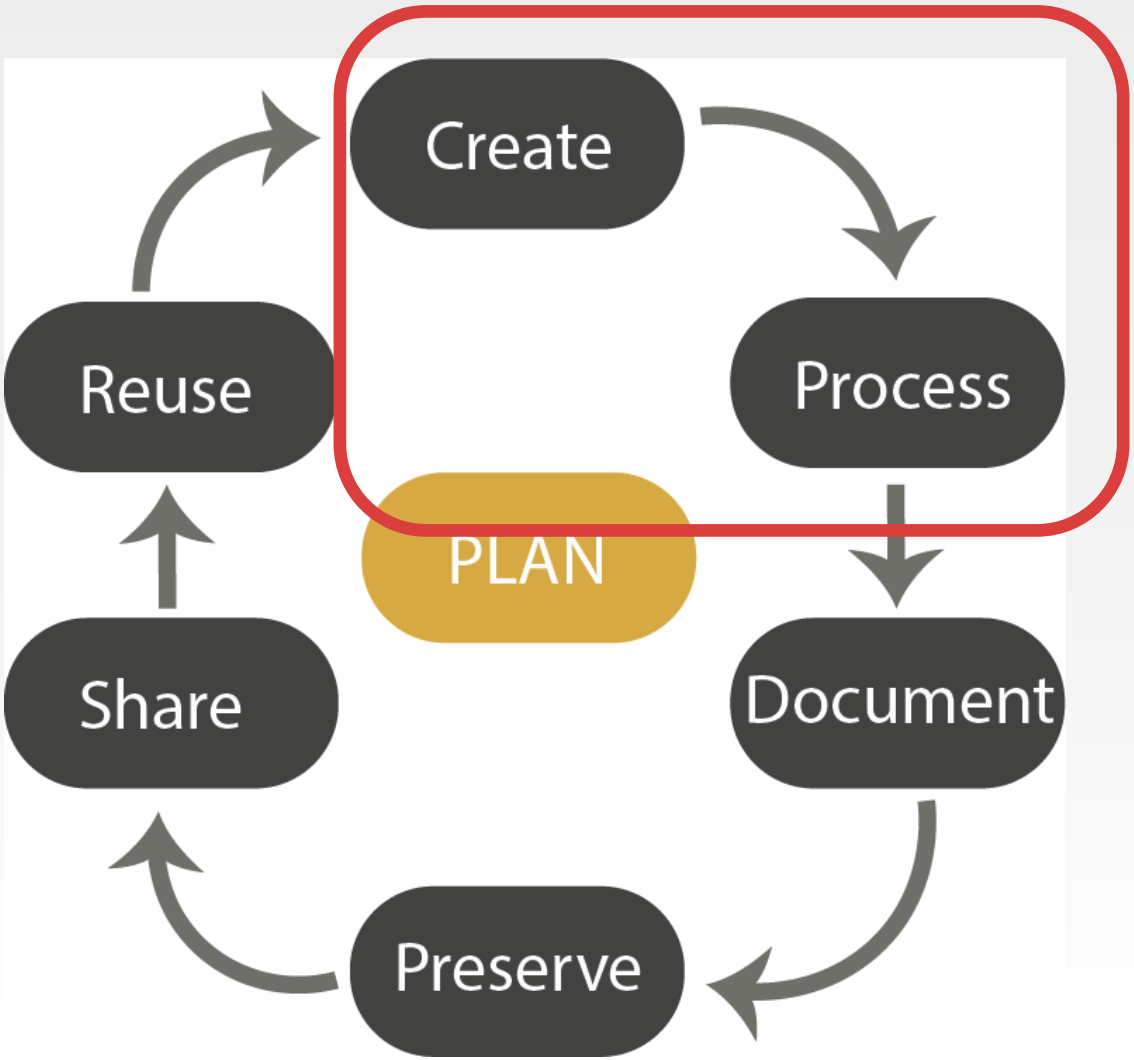
aud.halbritter@uib.no
@audhalbritter

UNIVERSITY OF BERGEN

Science is based on data



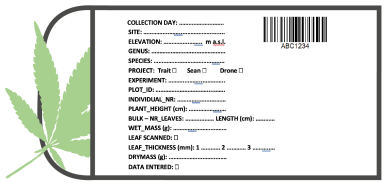
Science is based on data



THE TRAITS WHEEL

Collect

Prepare leaves

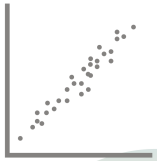


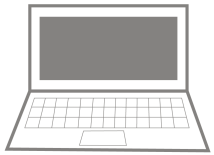
Weight (wet)

Scan (area)

Thickness

Data checking



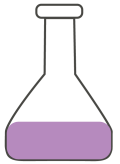


Data entry

Data doc

Weight (dry)

Chemical traits

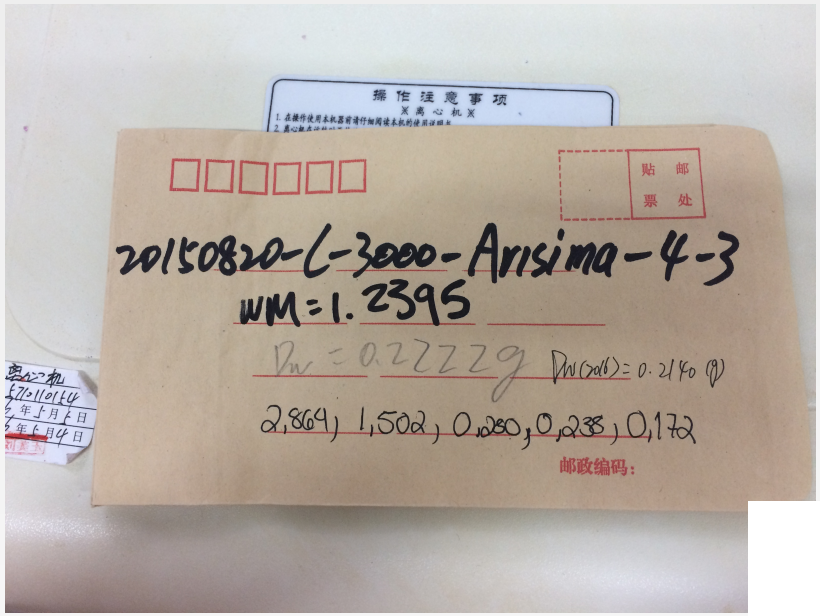


Create – collect leaves

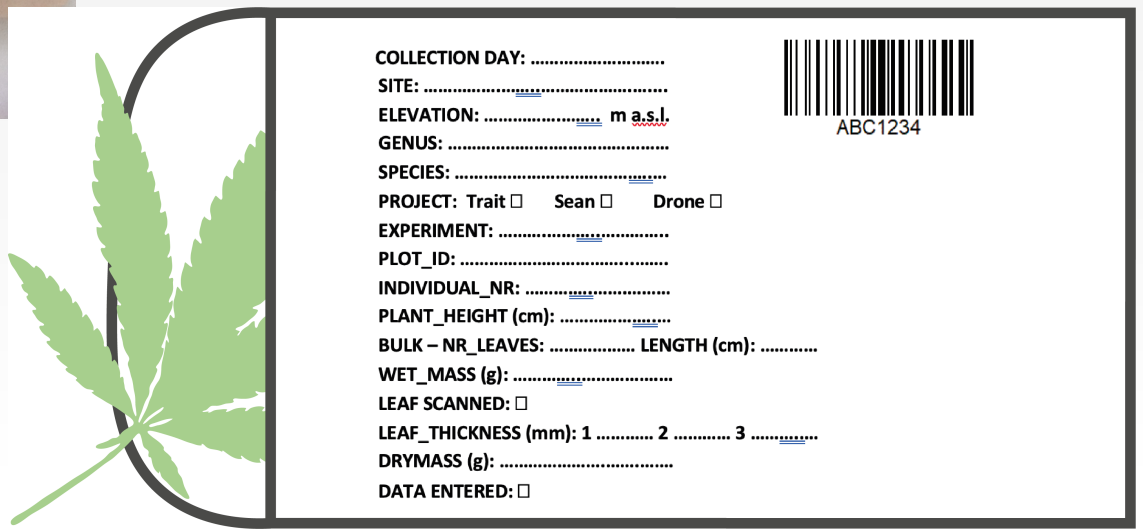
Collect „good“ plant material

Create – Prepare the leaves

Label your samples
Take notes!

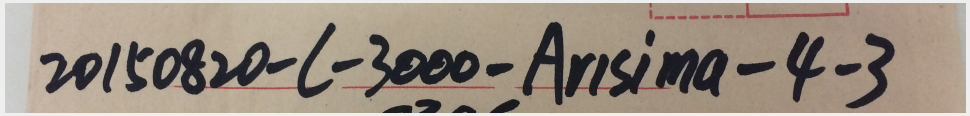


COLLECTION DAY: .....
SITE: .....
ELEVATION: ..... m a.s.l.
GENUS: .....
SPECIES: .....
PROJECT: Trait Sean Drone
EXPERIMENT: .....
PLOT_ID: .....
INDIVIDUAL_NR: .....
PLANT_HEIGHT (cm): .....
BULK – NR_LEAVES: ..... LENGTH (cm): .....
WET_MASS (g): .....
LEAF SCANNED:
LEAF_THICKNESS (mm): 1 ..... 2 ..... 3 .....
DRYMASS (g): .....
DATA ENTERED:



Create – Prepare

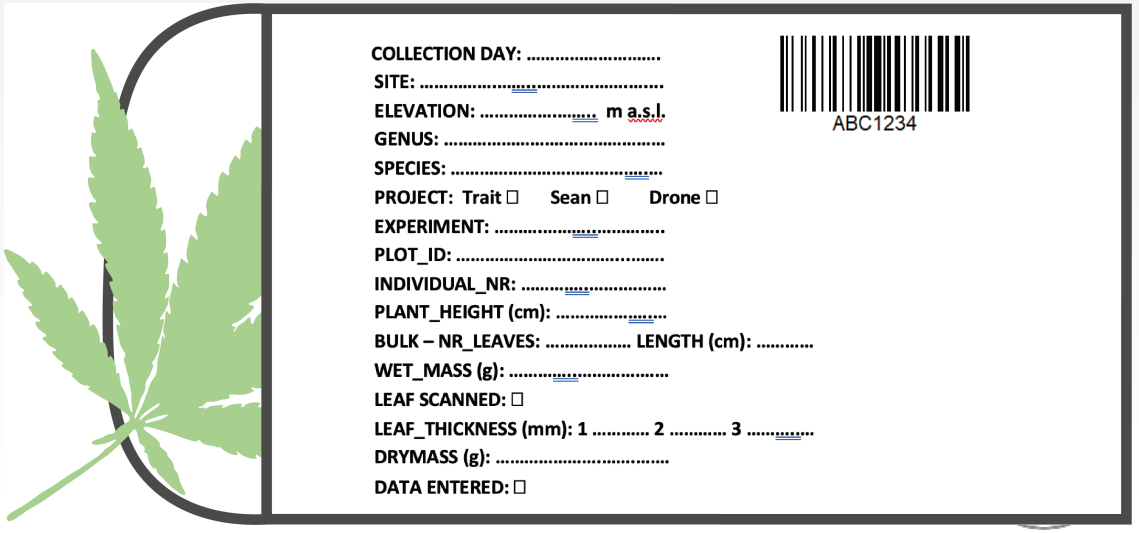
- Unique sample ID



- AAA1234
- Barcode

https://github.com/EnquistLab/PFTC3_Peru/blob/master/traits/Rdatagathering/CreateBarecodes.R

R package: baRcodeR

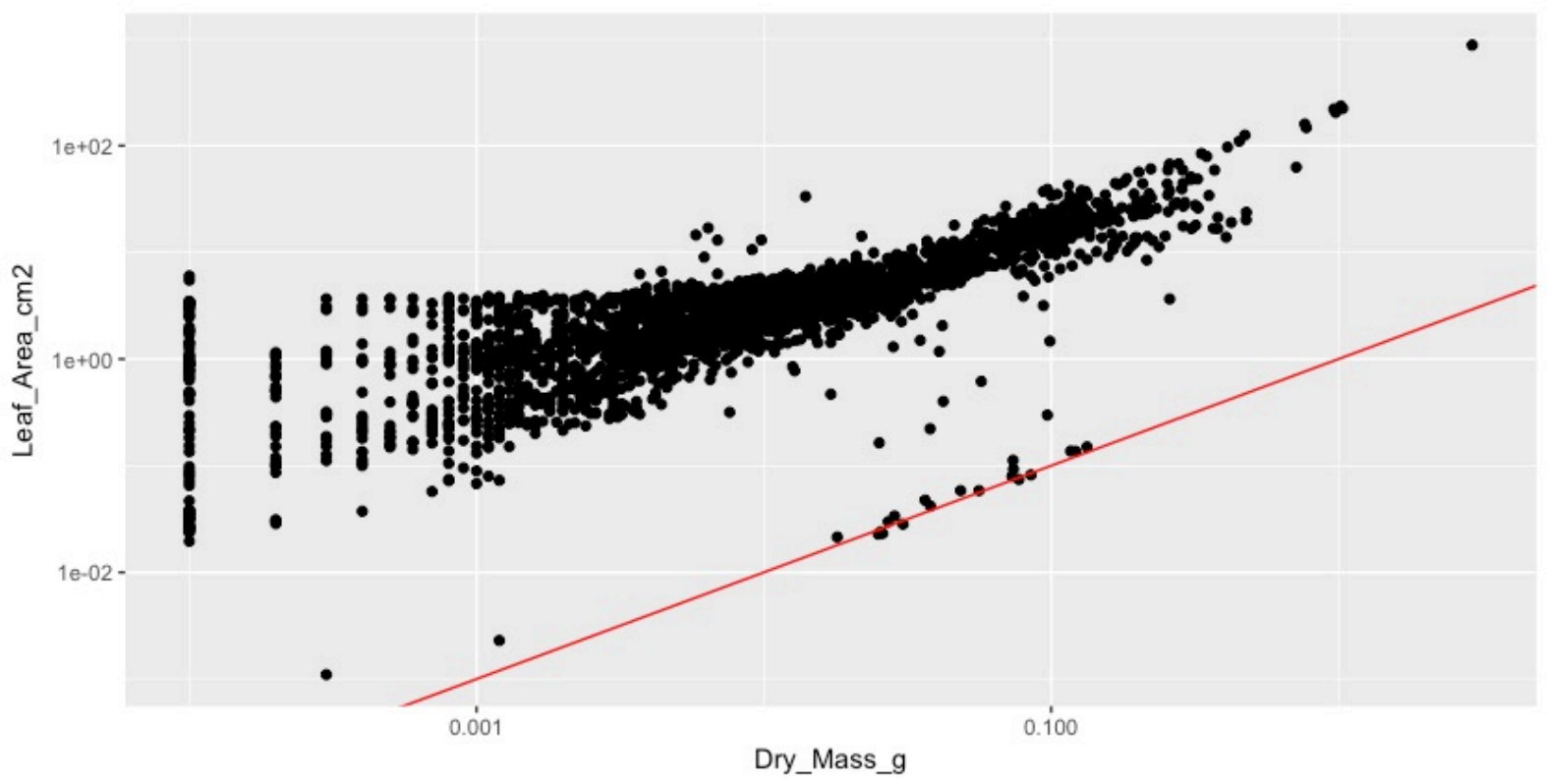


Create - Weight

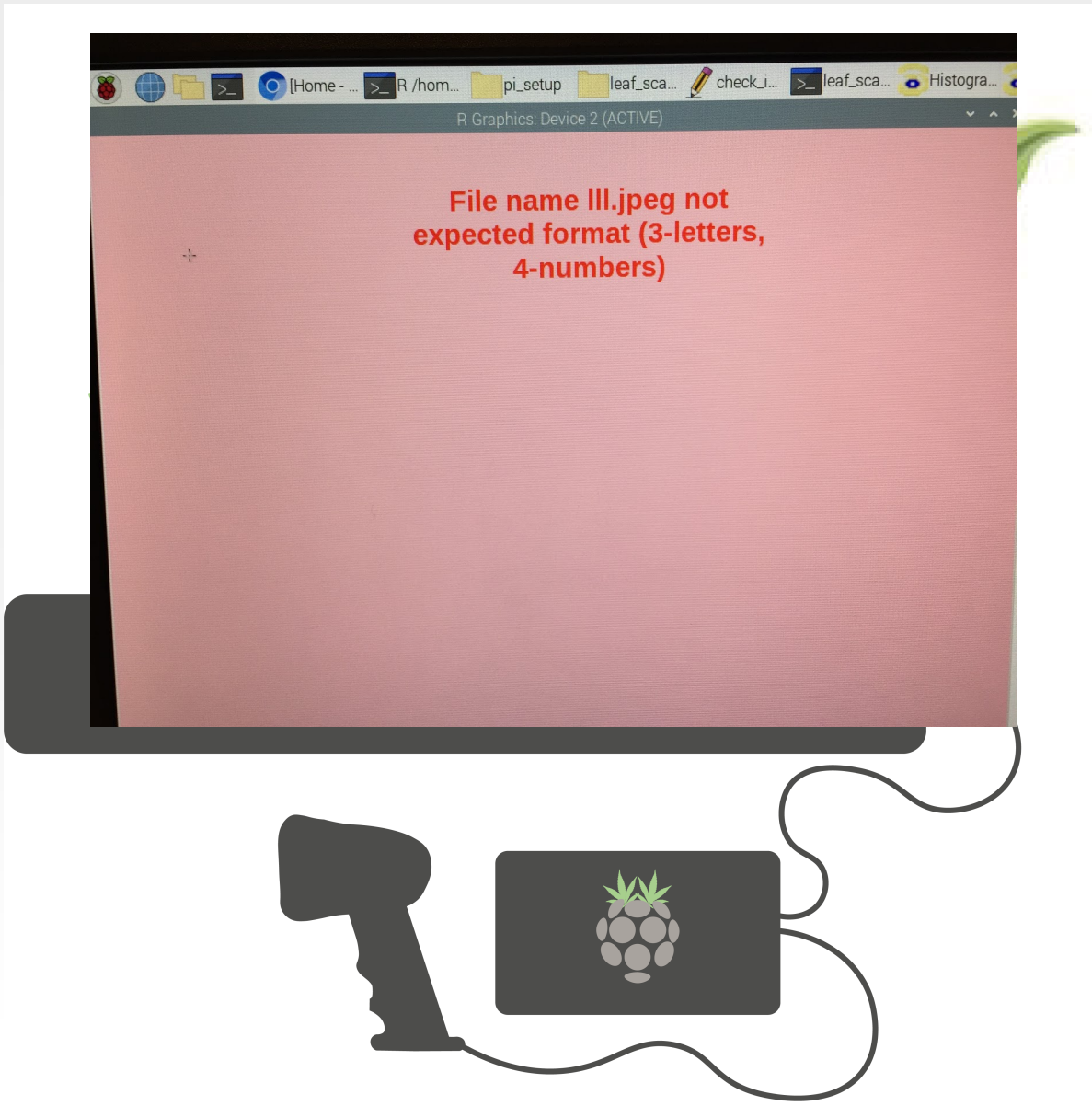
Write down the weight with all digits.

Create - Scanning

Process – Data validation

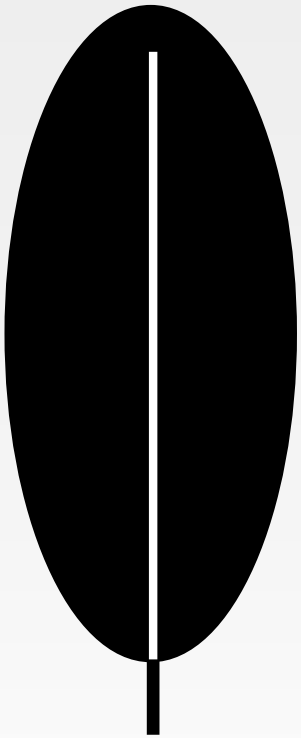


Create - Scanning



Create - Scanning

Create - Scanning



Create - Scanning

Create – Calculate Leaf Area

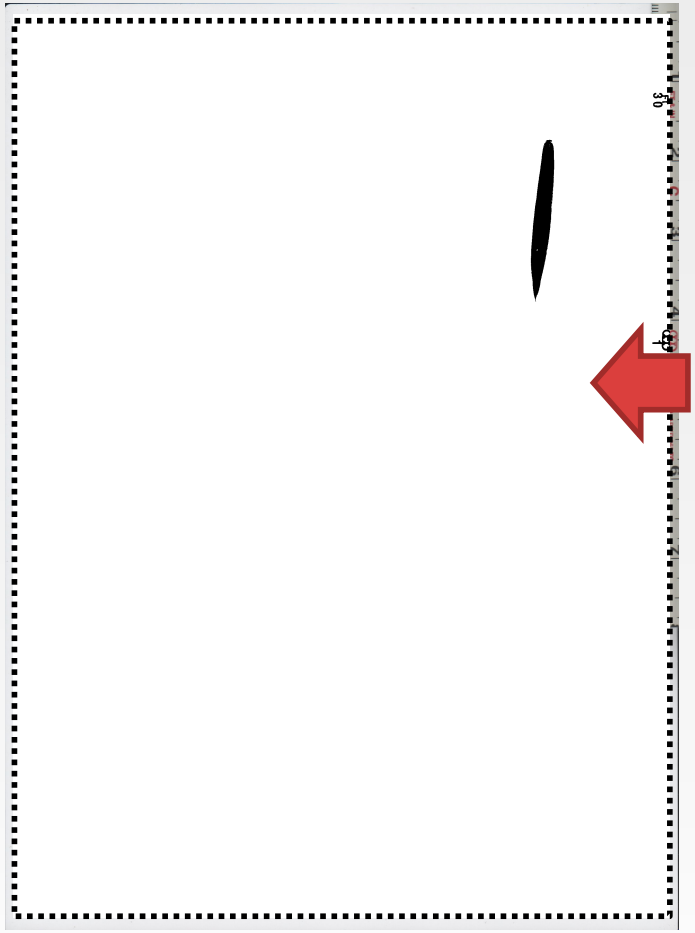
Code:
https://github.com/EnquistLab/PFTC3_Peru/blob/master/traits/Rdatagathering/CalculateLeafArea.R

ImageJ
R package: LeafArea

Custom function:

```
devtools::install_github("richardjtelford/LeafArea")
library(LeafArea)
```

Function to process multiple scans (loop.files)



Create - Calculate Leaf Area

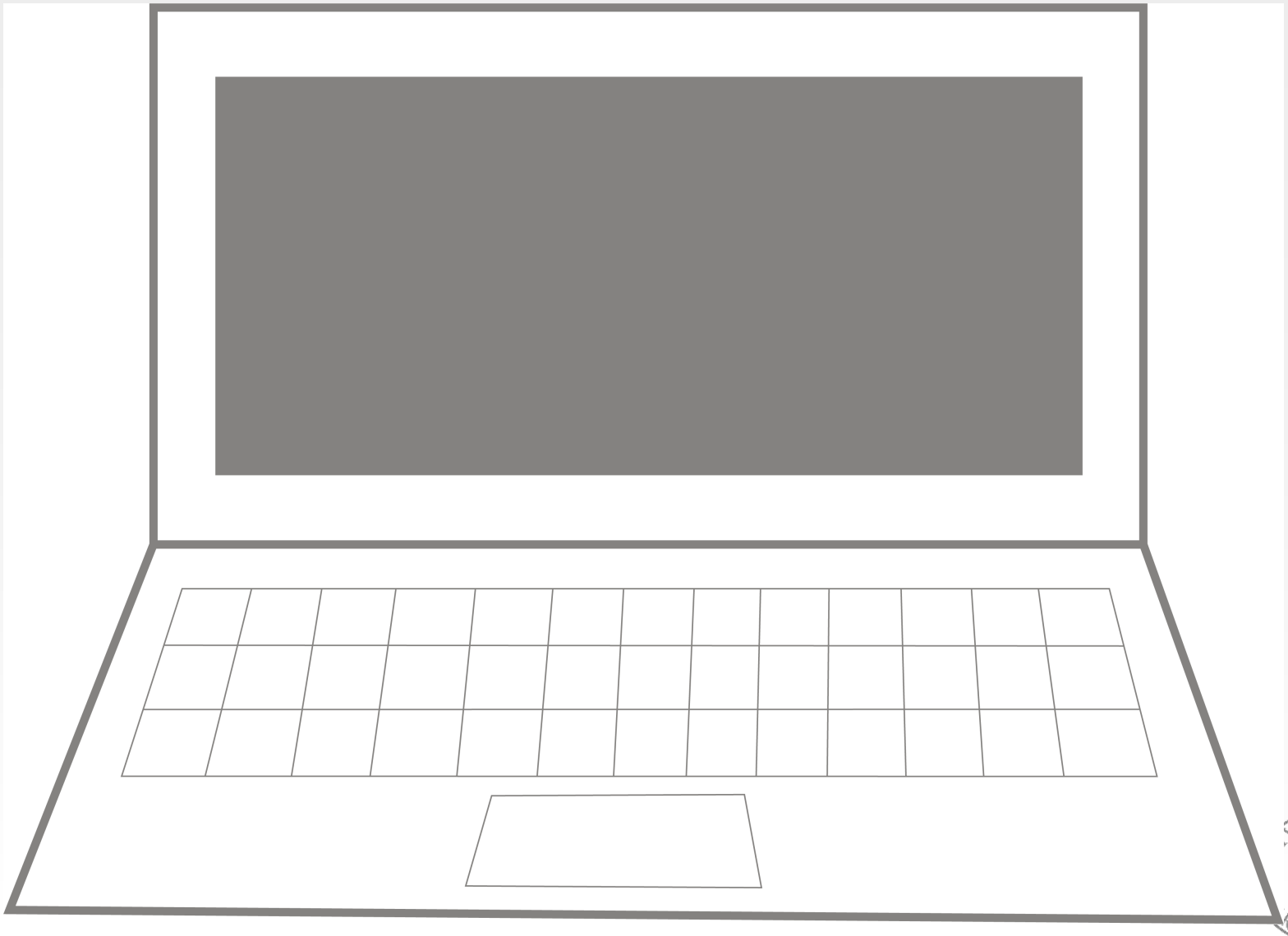
```
#### Function to calculate leaf area
 loop.files <- function(files){
  file.copy(files, new.folder)
  print(files)
  area <- try(run.ij(set.directory = new.folder, distance.pixel =
237, known.distance = 2, log = TRUE, low.size = 0.005, trim.pixel
= 60, trim.pixel2 = 150, save.image = TRUE))
  if(inherits(area, "try-error")){
    return(data.frame(LeafArea = NA))
  }
  file.copy(dir(new.folder, full.names = TRUE, pattern =
"\\.tif"), output.folder)
  Sys.sleep(0.1)
  if(any(!file.remove(dir(new.folder, full.names = TRUE) )))
stop()
  res <- data.frame(ID = names(unlist(area[[2]])), LeafArea =
(unlist(area[[2]])))
  return(res)
}
```

Create - Thickness

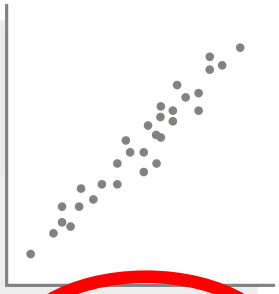
Measure leaf thickness 3 times for each sample avoiding the mid rib.

If the leaf is too small make less measurements. The measurments should not overlap.

Process – Data entry



Process – Import and checking



EnquistLab / PFTC3_Peru

Watch 18 | Star 2 | Fork 3

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Security | Insights

Join GitHub today

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

No description, website, or topics provided.

139 commits | 1 branch | 0 packages | 0 releases | 3 contributors

Branch: master | New pull request | Find file | Clone or download

audhalbritter initial commit — Latest commit a79cb6e 16 hours ago

Process – data checking

assertr
Provides functionality to assert conditions that have to be met so that errors in data used in analysis pipelines can fail quickly. Similar to 'stopifnot()' but more powerful, friendly, and easier for use in pipelines.

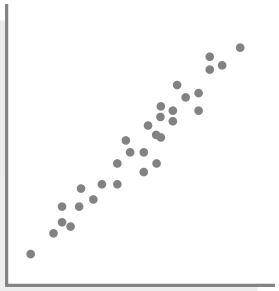
```
site.list <- c("WAY", "ACJ", "PIL", "TRE", "QUE")

… function(dat){
        …
        assert(in_set(site.list), Site, error_fun = error_report)
        …
}
```

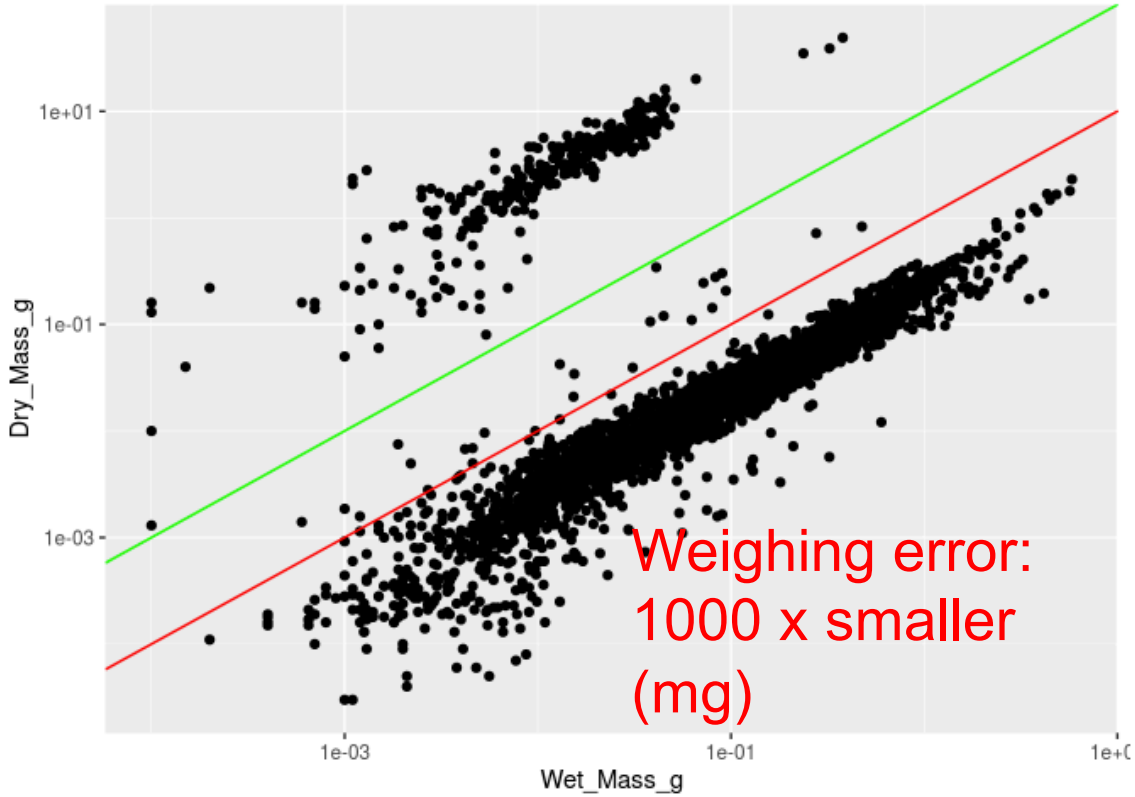
tidylog
feedback for basic dplyr operations

```
mutate: changed 6 values (<1%) of 'ID' (0 new NA)
mutate: changed 826 values (47%) of 'Project' (0 new NA)
mutate: changed 351 values (20%) of 'Experiment' (0 new NA)
mutate: changed 2 values (<1%) of 'Individual_nr' (0 new NA)
        changed 2 values (<1%) of 'Leaf_nr' (2 fewer NA)
filter: removed 52 rows (3%), 1,689 rows remaining
mutate: changed one value (<1%) of 'Site' (1 fewer NA)
mutate: changed 5 values (<1%) of 'Plot_ID' (4 fewer NA)
mutate: changed 4 values (<1%) of 'Individual_nr' (0 new NA)
mutate: changed 13 values (1%) of 'Leaf_nr' (10 fewer NA)
```

Process – Data validation



- Plot data
  - wet mass vs. dry mass
  - wet mass vs. leaf area
  - Leaf thickness 1 vs leaf thickness 2 etc.
- Log scale if necessary
- Check outliers
- Check unrealistic values
  - SLA > 500 g/cm2
  - SLA < 5 g/cm2
  - LDMC > 1

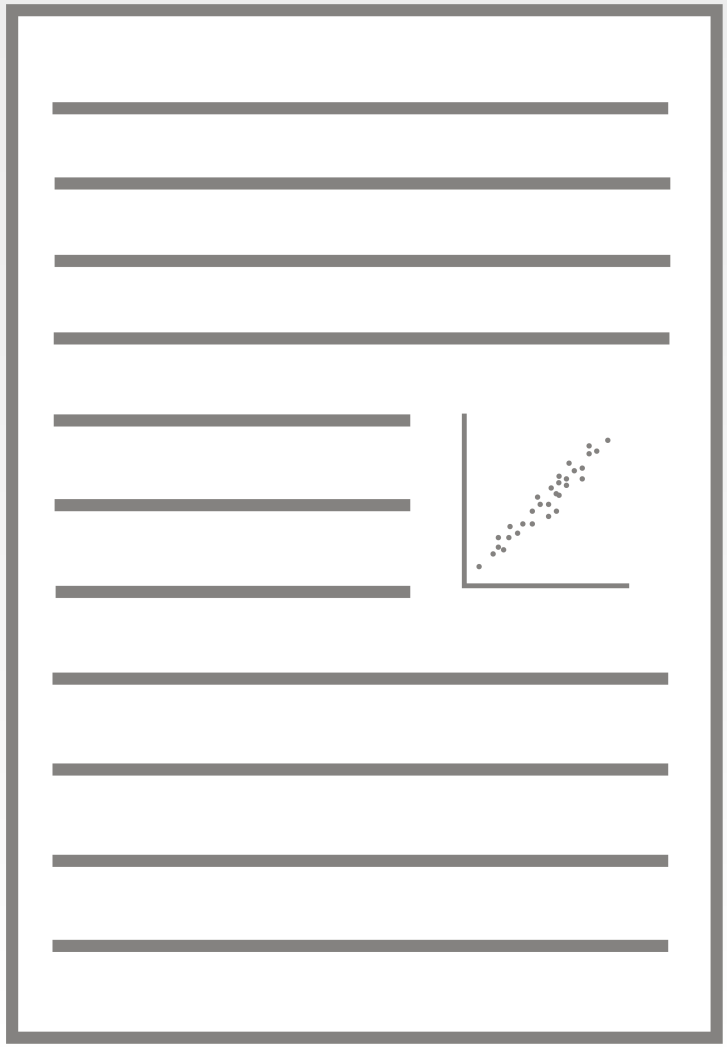


Process – some numbers

	China (2x)	Peru 1*	Svalbard	Peru 2**
# leaves	6734	3071	1469	1534
Total wet mass (g)	1141	836	132	248
Total leaf area (cm^2)	42353	23021	5666	7505
Lines of R code	522 + 1 year playing with Excel	393 ++	399	205

*not all chemical traits collected yet
** not all leaves processed yet

Document – Data documentation



Process data – Why reproducible workflow

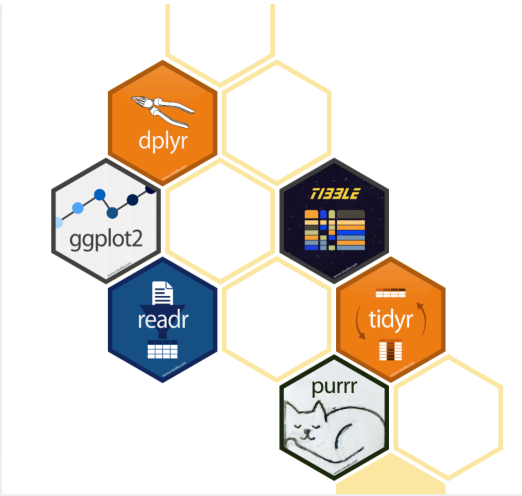
Clean, repeatable and script-based workflow

- It makes **returning to the code** much easier a few months down the line; whether revisiting an old project, or making revisions following peer review.
- The results of your analysis are **more easily scrutinised by the readers** of your paper, meaning it is easier to show their validity.
- Having clean and reproducible code available can **encourage greater uptake** of new methods that you have developed.

Repeatable and script-based workflow

- Start your analysis from your **raw data**
- Any cleaning, merging, transforming, etc. of data should be done in **scripts**, not manually
- **Split your workflow** (scripts) into logical thematic units. For example, you might separate your code into scripts that
  - (i) load, merge and clean data
  - (ii) analyse data
  - (iii) produce outputs like figures and tables
- Eliminate code duplication by packaging up useful code into custom **functions**.
- Programm defensively and **test** your code
- Make sure to **comment your functions** thoroughly, explaining their expected inputs and outputs, and what they are doing and why.
- **Document your code** and data as comments in your scripts or by producing separate documentation.
- Any intermediary outputs generated by your workflow should be kept separate from raw data.

Tidy(uni)verse



The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Package drake

It analyzes your workflow, skips steps with up-to-date results, and orchestrates the rest with optional distributed computing. At the end, drake provides evidence that your results match the underlying code and data, which increases your ability to trust your research.

Programmig style guide

- Concise, descriptive and menaingful names
- Spacing, split long code and intendations
- Use #comments
- Do not repeat code -> use functions
- Use relative not absolute path
- Defensive programming: test your code

```
### Good
pos <- function(x) {
  if (is.null(dim(x))) {
    x[x > 0]
  } else{
    x[, colSums(x) > 0, drop  = FALSE]
  }
}


### Bad
pos <- function(x){
  if(is.null(dim(x)))
  {x[x >0]}
  else{
    x[, colSums(x) > 0, drop  = FALSE]
  }}
```

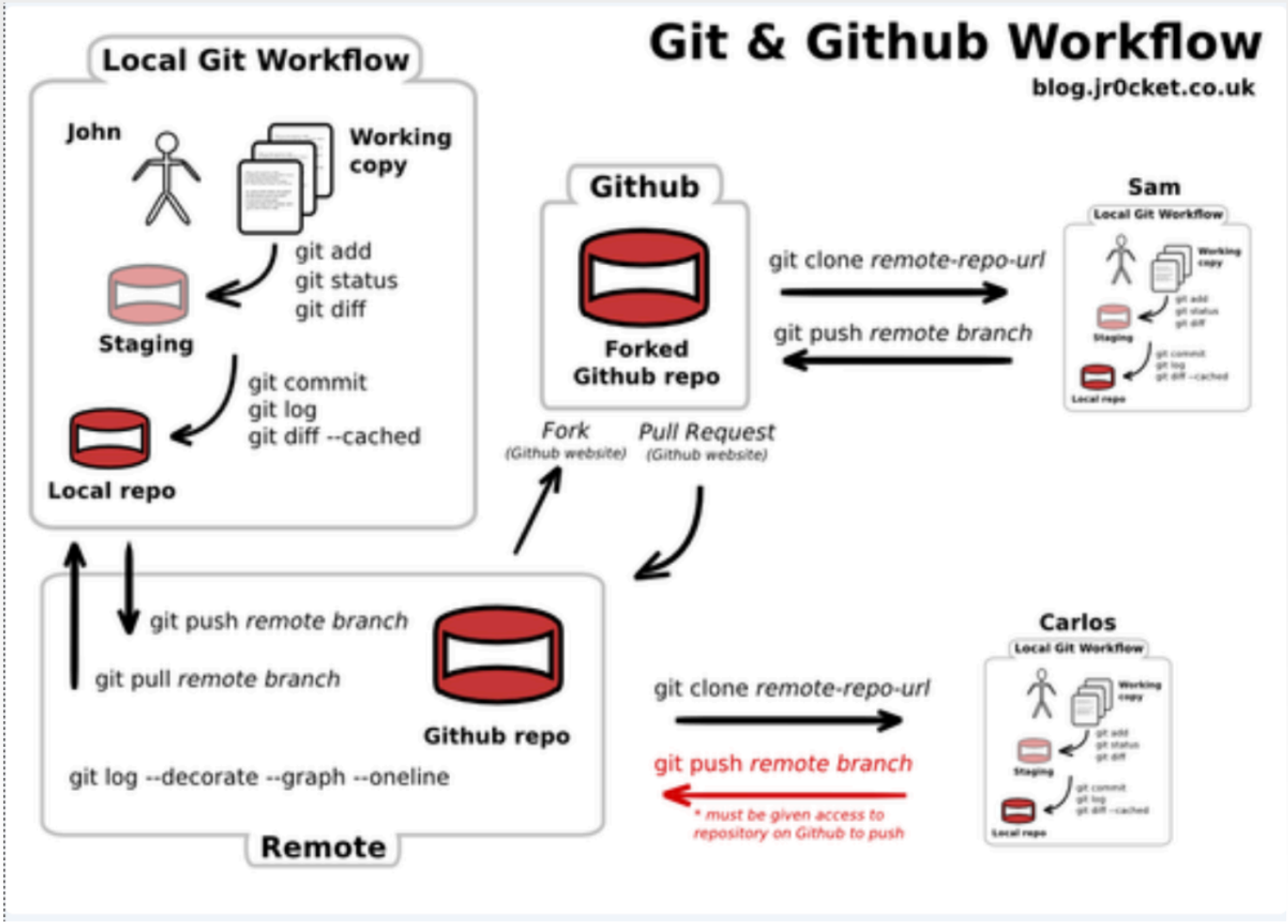
Git and GitHub

- Git is a **version control system**. Git manages the evolution of a set of files – called a **repository** – in a sane, highly structured way. If you have no idea what I'm talking about, think of it as the "Track Changes" features from Microsoft Word on steroids.
- GitHub (Bitbucket, GitLab) provides a home for your Git-based projects on the internet.

Why Git and GitHub

- Easy to share code with collaborators, students, etc.
- Collaborators can work on the same code at the same time
- Keeping track of changes in the code

Workflow


Git & Github Workflow
blog.jr0cket.co.uk
Local Git Workflow
John
Working copy
git add
git status
git diff
Staging
git commit
git log
git diff --cached
Local repo
Github
Forked Github repo
Fork
(Github website)
Pull Request
(Github website)
git clone remote-repo-url
git push remote branch
Sam
Local Git Workflow
git push remote branch
git pull remote branch
Github repo
git log --decorate --graph --oneline
Remote
git clone remote-repo-url
git push remote branch
* must be given access to repository on Github to push
Carlos
Local Git Workflow

Commit - some rules

- Commit often and provide useful messages so you can keep track of what you are doing.
- Commit code and plain text
- Don’t upload large files (e.g. data files)
- Don’t upload output files (figures)
- You can create a **gitignore** file where you can define rules, which files will be uploaded to git.

Further reading

- https://happygitwithr.com/ Jenny Bryan

Let's Git started

Github – R studio Cheat Sheet

Why use version control?

- Easy to share code
- Work on the same code at the same time
- Keeping track of changes in the code

Preparation:

1. Download Git or similar: https://git-scm.com/
   For UiB users git is on the Software Centre.
2. Get a github account: https://github.com/
3. Connect RStudio with Github

Further reading