Data management in R

23rd May

Charles Darwin House

What is ‘data management’?

Data management is the process of controlling the information generated during a research project. Any research will require some level of **data management**, and funding agencies are increasingly requiring scholars to plan and execute good **data management** practices.

Key principles

- Full documentation of all data, metadata and protocols
- Data processing is fully replicable, version controlled
- Unique and persistent identifiers
- Data are stored in a permanent repository
- Data can be found, interpreted and used by someone else

Why is R useful?

- Scripting increases reproducibility
- Easy to allow version control e.g. by using Git to store code
- Easy to manipulate data and create new datasets

Why you might need more than R

- Limited support for relational databases
- Limited ability to store metadata
- Users need to put effort in to ensure unique IDs etc (this is controlled in some other programs)

This course – how to get the most out of R for data management

1. Utilising data from multiple sources
2. Manipulating data within R
3. Documentation

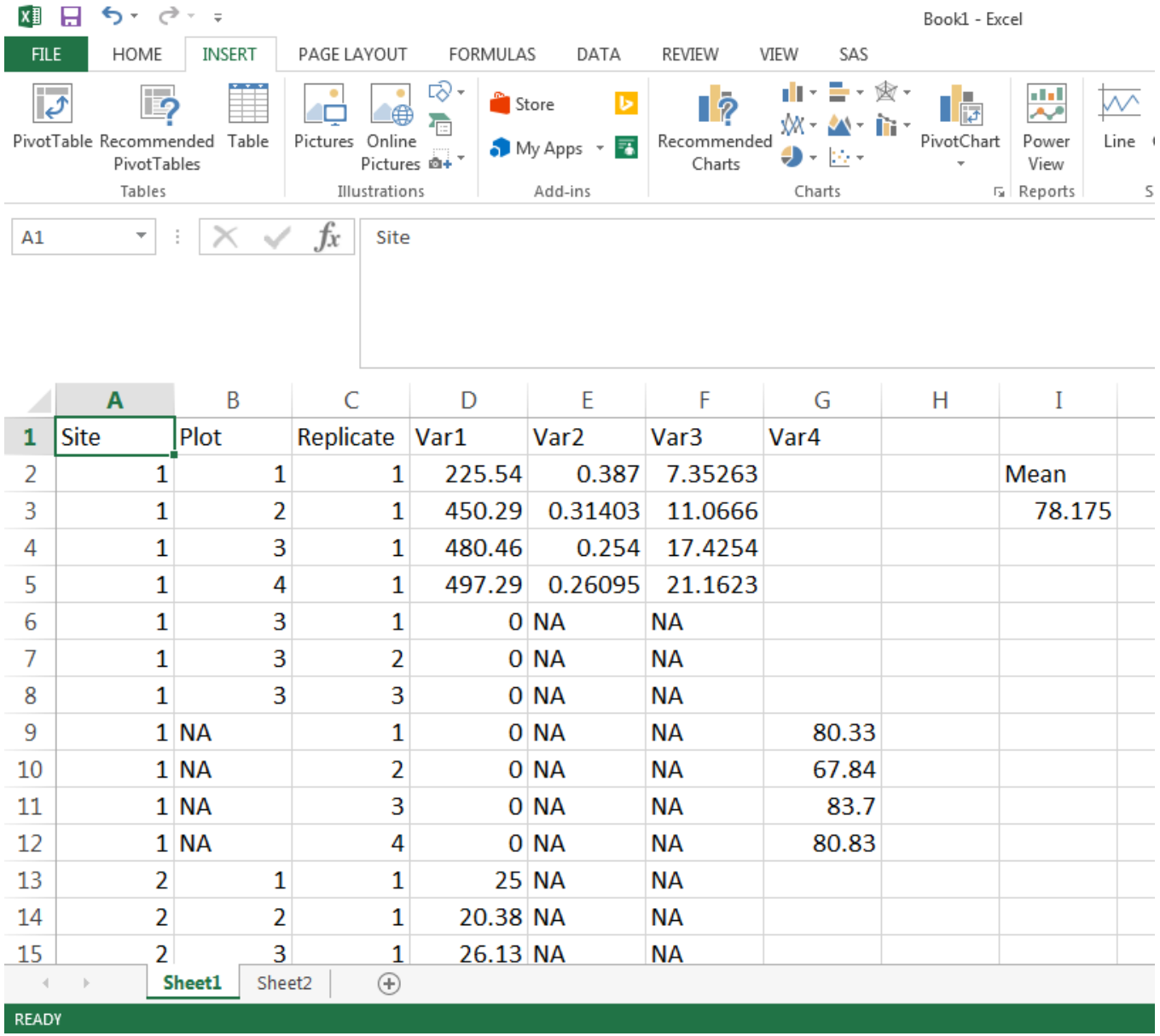
1. Utilising data from multiple sources

- Import data into R without changing the original data source
- Never manipulate the original data
- If new datasets are created e.g. errors are found, new data is generated create a new dataset with a new version number e.g. v1.1
- Keep a note (either in the file or in a README file in the directory) of any changes to the dataset

1. Utilising data from multiple sources

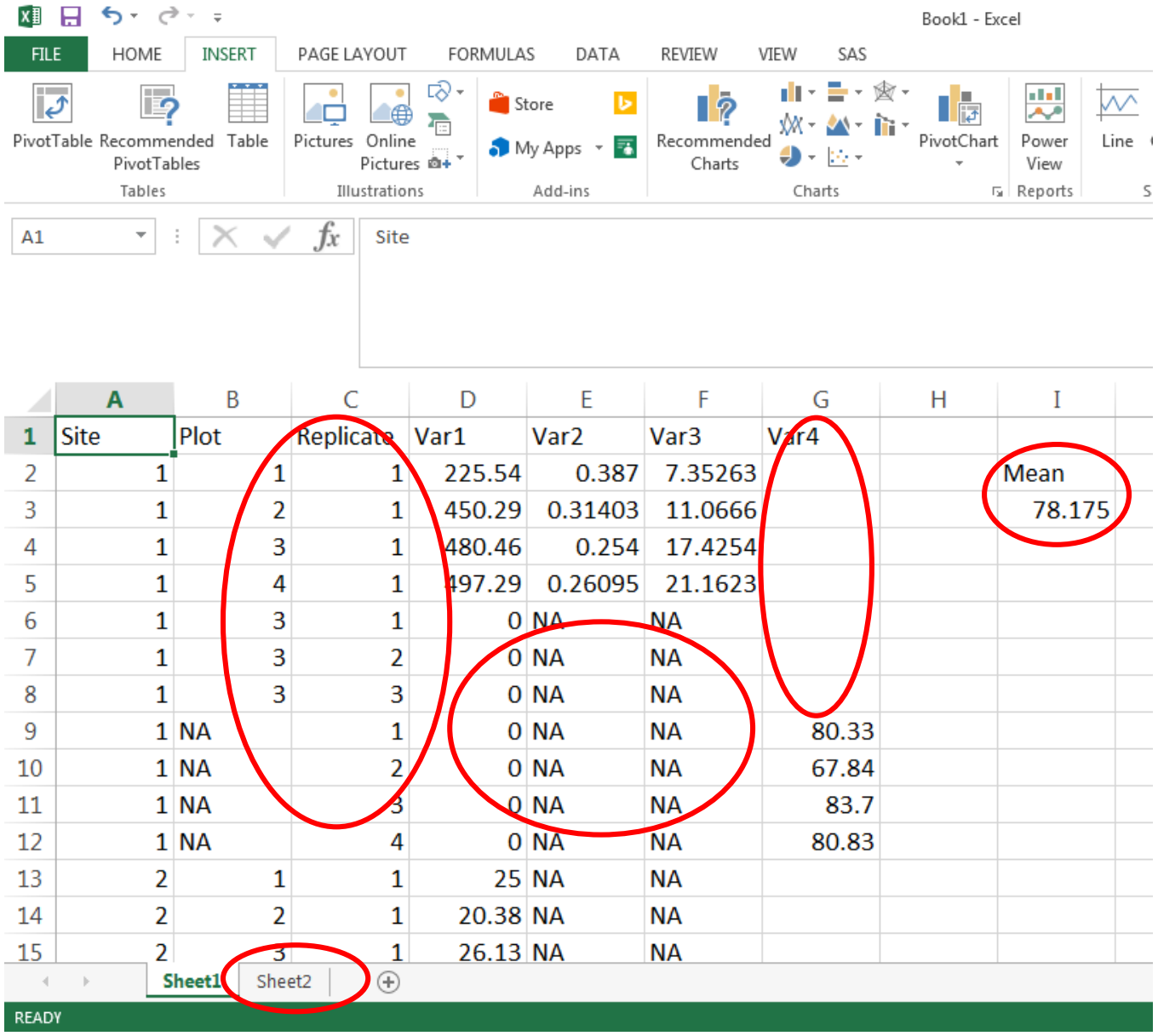
- Where might data be stored:
  - Excel
  - Access
  - SAS
  - Oracle
  - NetCDF
  - Spatial data formats – shapefiles/rasters

Excel



	A	B	C	D	E	F	G	H	I
1	Site	Plot	Replicate	Var1	Var2	Var3	Var4		
2	1	1	1	225.54	0.387	7.35263			Mean
3	1	2	1	450.29	0.31403	11.0666			78.175
4	1	3	1	480.46	0.254	17.4254			
5	1	4	1	497.29	0.26095	21.1623			
6	1	3	1	0	NA	NA			
7	1	3	2	0	NA	NA			
8	1	3	3	0	NA	NA			
9	1	NA	1	0	NA	NA	80.33		
10	1	NA	2	0	NA	NA	67.84		
11	1	NA	3	0	NA	NA	83.7		
12	1	NA	4	0	NA	NA	80.83		
13	2	1	1	25	NA	NA			
14	2	2	1	20.38	NA	NA			
15	2	3	1	26.13	NA	NA			

Excel

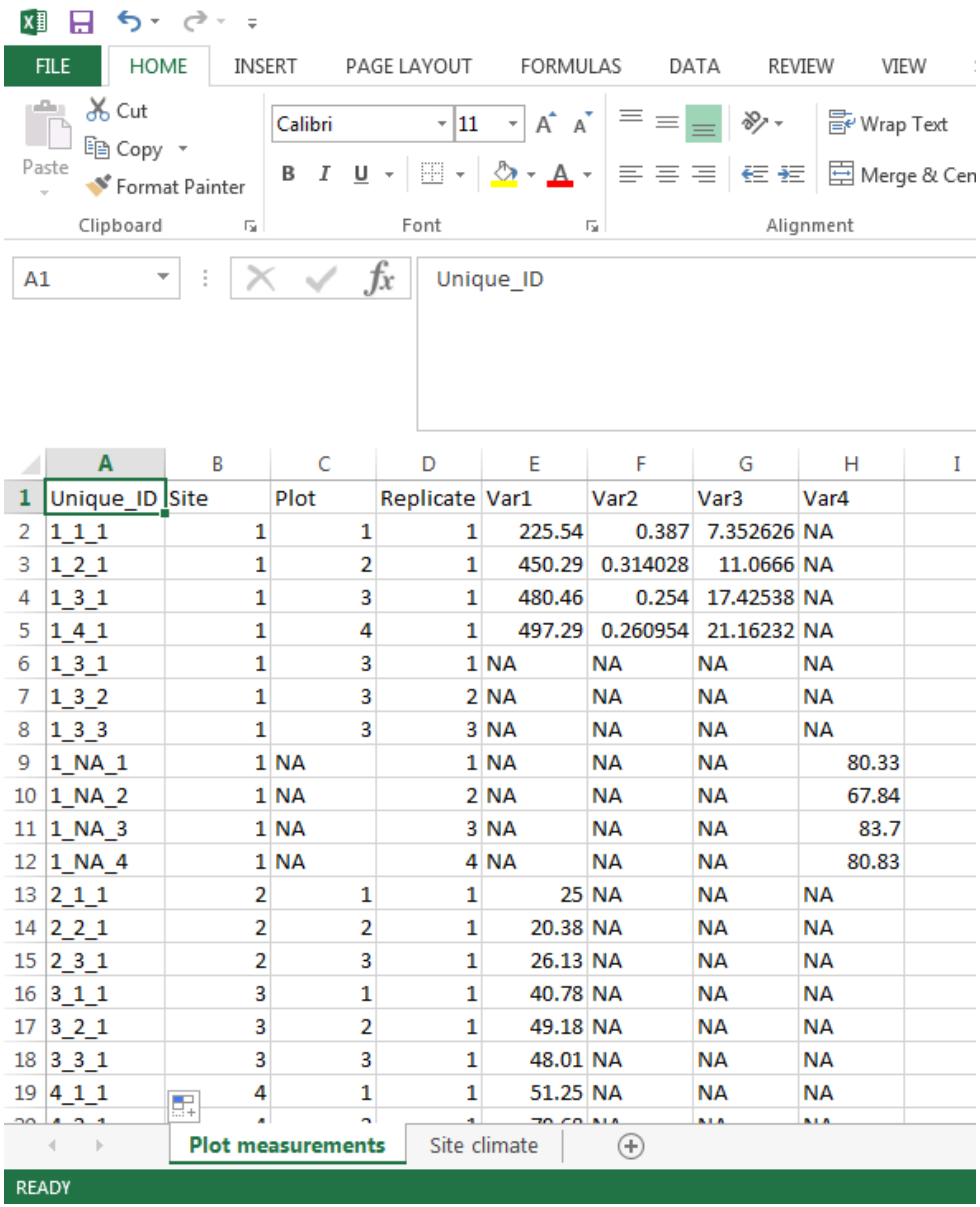


	A	B	C	D	E	F	G	H	I
1	Site	Plot	Replicate	Var1	Var2	Var3	Var4		
2	1	1	1	225.54	0.387	7.35263			
3	1	2	1	450.29	0.31403	11.0666			
4	1	3	1	480.46	0.254	17.4254			
5	1	4	1	497.29	0.26095	21.1623			
6	1	3	1	0	NA	NA			
7	1	3	2	0	NA	NA			
8	1	3	3	0	NA	NA			
9	1	NA	1	0	NA	NA	80.33		
10	1	NA	2	0	NA	NA	67.84		
11	1	NA	3	0	NA	NA	83.7		
12	1	NA	4	0	NA	NA	80.83		
13	2	1	1	25	NA	NA			
14	2	2	1	20.38	NA	NA			
15	2	3	1	26.13	NA	NA			

Before inputting from Excel:

- Standardise ‘no data’ as NA
- Remove in-sheet calculations/plots
- Ensure one column has a unique identifier for each row of data
- If data are in multiple sheets, give sheets descriptive names

Excel



	A	B	C	D	E	F	G	H	I
1	Unique_ID	Site	Plot	Replicate	Var1	Var2	Var3	Var4	
2	1_1_1	1	1	1	225.54	0.387	7.352626	NA	
3	1_2_1	1	2	1	450.29	0.314028	11.0666	NA	
4	1_3_1	1	3	1	480.46	0.254	17.42538	NA	
5	1_4_1	1	4	1	497.29	0.260954	21.16232	NA	
6	1_3_1	1	3	1	NA	NA	NA	NA	
7	1_3_2	1	3	2	NA	NA	NA	NA	
8	1_3_3	1	3	3	NA	NA	NA	NA	
9	1_NA_1	1	NA	1	NA	NA	NA	80.33	
10	1_NA_2	1	NA	2	NA	NA	NA	67.84	
11	1_NA_3	1	NA	3	NA	NA	NA	83.7	
12	1_NA_4	1	NA	4	NA	NA	NA	80.83	
13	2_1_1	2	1	1	25	NA	NA	NA	
14	2_2_1	2	2	1	20.38	NA	NA	NA	
15	2_3_1	2	3	1	26.13	NA	NA	NA	
16	3_1_1	3	1	1	40.78	NA	NA	NA	
17	3_2_1	3	2	1	49.18	NA	NA	NA	
18	3_3_1	3	3	1	48.01	NA	NA	NA	
19	4_1_1	4	1	1	51.25	NA	NA	NA	

Inputting data from Excel

- Save each sheet as .csv and use read.csv
- Import directly from Excel with xlsx package

Access and Oracle

- Databases have several advantages over Excel:
  - Not ‘flat format’ i.e. can have different tables with different information and specify how the tables relate to each other – prevents empty fields
  - Require you to specify unique row identifiers
  - Can store very large amounts of data

Access and Oracle

- Existing databases (and some other formats) can be accessed directly using your computer's ODBC driver and the RODBC package
- There is no need to export any data from the database and the original data is not modified
- Important to ensure 32-bit/64-bit continuity between driver, R version and system architecture

Access and Oracle

- Step 1 – work out which version of R you are using

  sessionInfo()

- Step 2 – change to 32-bit R if necessary

  R studio – Global Options/R version

- Step 3 – find your 32-bit ODBC connector

  Type ‘ODBC’ in Start menu

  Check whether you can add an Access connection

  If not use version here: C:\Windows\SysWoW64\odbcad32.exe

Access and Oracle

- Step 4 – set up a new ODBC connection
  - Use the Access driver allowing .accdb formats
  - Choose the database downloaded from GitHub
  - Call the new connection “Example”
- Step 5 – use the RODBC package to connect to the database

SAS

- SAS is preferred by some as it has good data visualisation, scripting and database capabilities
- R packages ‘haven’ and ‘sas7bdat’ can be used to read directly from SAS data files
- SAS data can be exported as csv – this is my preferred method

NetCDF

- Very useful data format for 3-d data e.g. daily outputs of a spatial model
- Self-describing format
- Packages ‘RNetCDF’ and ‘ncdf4’ can read NetCDF files

Spatial data

- Most commonly analysed in GIS software such as ArcGIS
- Why might you want to use R instead?
  - Integrate spatial analyses into R-based workflow
  - Lots of packages available - flexibility
  - Reproducibility
- However:
  - Python is often faster than R and is also flexible and reproducible
  - R most useful when small GIS tasks are needed as part of a larger analysis

Spatial data packages

- Huge number available for different geoprocessing tasks
- Lots of help available online…

Summary – part 1

- R can read in data from a huge number of data formats
- R can do most of the jobs that Excel, Access, SAS, GIS can do
- R scripts can be reproducible and auditable so the entire workflow can be repeated
- My opinion – use Excel for data entry, SAS for manipulating large databases, ArcGIS/arcpy for large geoprocessing tasks – everything else in R!

2. Manipulating data in R

- Once data is in R, what can we do with it?
  - Single datasets
  - Multiple datasets

- Keeping track of data in R – don’t overwrite dataset names!
  - data <- read.csv(“data_v2.1.csv”)
  - data <- data[data$Value < 10,]

Single dataset operations

You might want to:

- Change format – lists, tables, matrices
- Sort data
- Remove duplicates
- Remove missing data
- Reshape datasets (reshape)
- Summarise (aggregate, apply)
- Apply functions to data

Multiple dataset operations

You might want to:

- Append datasets
- Match data from different datasets

3. Documentation

Now that we've input and processed all our data efficiently in R we need to make sure we document everything we've done:

- So someone else can use our code
- So that we can remember what we did (and why)
- To conform with quality assurance standards

Things we need to document

- Lines of code in R – these should be commented ‘#’ with text that allows someone else to understand your code
- Your R scripts – if you have multiple scripts you should know what each one does and the order in which they are run. Should also be version controlled
- Your datasets – you may wish to circulate the results of your data processing. These should be linked to a script, with a DOI and metadata

Documenting code within scripts

- Can be as simple as #commenting your code
- Integrating code and notes e.g. using RMarkdown is easy and intuitive – in Rstudio
- Markdown can be used to document both code and/or outputs depending on the arguments used – very flexible and reproducible

Documenting scripts

Two key elements:

1. Workflow
2. Version control

R workflow documentation

- No “proper” way to construct workflows in R
- Single script – workflow is embedded
- Multiple scripts:
  - Number them
  - Use source() in scripts to call previous script
  - Split into different components:
    - Load data
    - Clean data
    - Define custom functions
    - Run analysis
- Document workflow in a readme file

R workflow documentation

- The best (but most time consuming) option – writing your own R package!
- Inbuilt documentation
- Not as hard as you might think…but not necessarily useful for day-to-day work
- Or you could try http://projecttemplate.net/index.html for a template system for project organisation

Workflows with multiple programs

- Where possible, limit the number of programs involved
- Unless completely unavoidable, make sure other programs can be run from a saved script (e.g. Python for ArcGIS) or workflow (Process Flows in SAS)
- If you need to point-and-click anything write down the exact sequence of actions in a text file

Workflows with multiple programs

- In some cases it is possible to call other languages from R or vice versa
- R and Python are a good example of this – you can call a script written in one language with the other language
- If you have programs written in e.g. Fortran they can be packaged as DLLs and run from R that way (in theory)

Workflows with multiple programs AND multiple people

- The worst case scenario…
- Agree on a workflow BEFORE you start analysis and make sure it is documented in a shared directory
- Make everyone documents everything!
- Independent tests are key – if you are clever these can be automated

Version control

- Generally refers to using a system like Git
- Advance on creating new versions of code every time you make a notable change
  - Rscriptv1.1.R
  - Rscriptv1.2.R
  - Etc…
- VERY important to allow rollbacks to previous versions e.g “hmm that R^2 is now smaller, I wonder what the difference is?”

Version control with Git/SVN etc

There are lots of advantages to using a system like Git:

- Tracks the exact changes you make to code
- Roll back to previous versions
- Can fork and merge changes
- Easy to collaborate with others
- Only store the changes, avoids large numbers of file copies

Git and Github

- Git is one of the most popular version control systems
- Github is an online repository where you can store your code and contribute to collaborators projects
- Most R packages are stored on Github
- You can use Github without using Git but Git is needed for the best functionality

Github

- https://github.com/

1. Set up an account
2. Have a look at other people's accounts!
3. You can create a repository here but we'll do this later

Git

1. Install Git:
   - Windows: http://git-scm.com/download/win.
   - OS X: http://git-scm.com/download/mac.
   - Debian/Ubuntu: sudo apt-get install git-core
   - Other linux: http://git-scm.com/download/linux

2. Show RStudio where Git is installed:
   - Tools > Global Options > Git/SVN
   - Under Git executable navigate to where you downloaded Git.exe (should be C:/Program Files/Git/bin/git.exe unless you changed this)
   - Ok

Setting up Git

3. Git uses the shell…
    - Rstudio Tools > Shell
    - No Rstudio? https://www.rstudio.com/
    - Note shell with open up in working directory

```
git config --global user.name "YOUR FULL NAME“
git config --global user.email "YOUR EMAIL ADDRESS“
```

Check setup:

```
git config –-global --list
```

Git setup

4. Set up an SSH RSA key
   - Go to Rstudio > Tools > Global Options > Git/SVN
   - Under SSH RSA key click ‘Create RSA key…’
   - Follow instructions then once complete click ‘View public key’
   - Copy the public key onto your clipboard
   - Go to Github.com
   - Under your account go to Settings > SSH and GPG keys
   - Click ‘New SSH key’
   - Name your key and paste the public key from your clipboard

Git setup

- You will now need to restart RStudio and should see a new tab in the Environment pane called Git
- Or View > Show Git if you can't see it
- Now create a new R project
- File > New Project > New directory > Empty project
- Call your project a sensible name and make sure 'Create a git repository' is selected
- Note where you created the repository on your computer…

New project

- By setting up in RStudio you have created a new folder in the directory on your computer
- You can simply copy and paste any scripts/data you want into the new repository
- Now we need to commit our changes
- In the Git window, click on ‘Commit’
- In the top left you will see the new files you added
- Click ‘Stage’ to find new files to add
- Add a message in the window in top right
- Click ‘Commit’

Pushing your project to Github

- Create a new repository on Github with the same name as your project
- Other options can be left as defaults
- Open a shell from RStudio
- Copy the two lines under '…push an existing repository from the command line' into the shell
- You might need to enter your Github credentials
- Ta dah! Should be all uploaded!

Making edits

- When you edit a file an ‘M’ symbol will appear in the Git window
- Simply repeat the steps of staging and commiting
- Use the ‘Push’ arrow to push changes to Github
- View changes in the History window
- Can view and save previous file versions
- Don’t push to Github if you might want to revert to a previous version – makes it more complicated!

Git commit issues

- I find the RStudio git window very slow and use the shell instead with the basic commands:
  - git status
  - git add -u
  - git commit –m “commit message”
  - git push

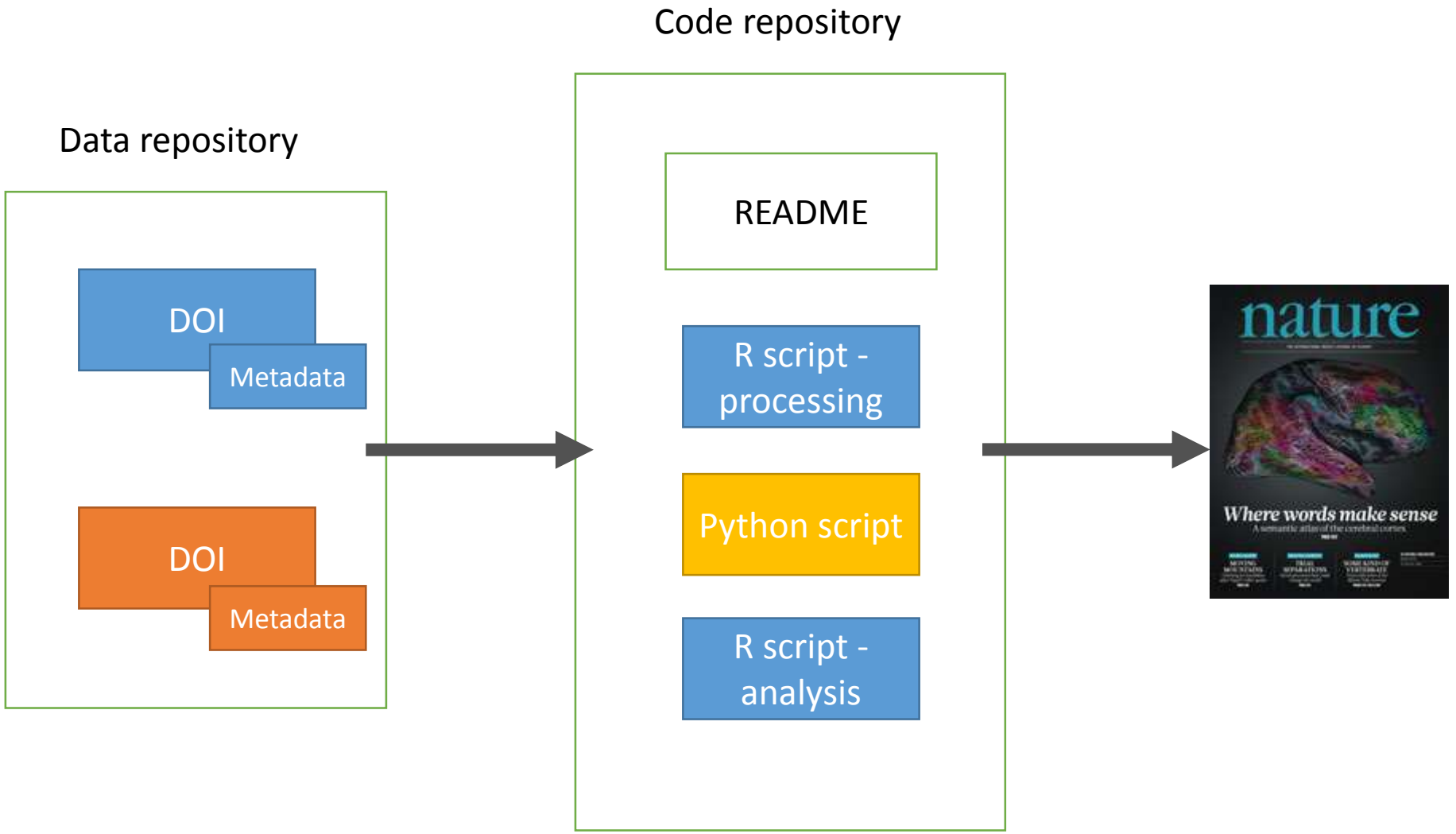
Documenting your data

- Also need to document your datasets
- Many journals now expect you to deposit data and obtain a DOI
  - Long term storage
  - Available to others
  - Citable
- Examples – Environmental Information Data Centre, Dryad

Documenting your data

- Remember – your raw data should be unchanged through the entire process
- BUT your scripts may well have created an edited dataset as part of the processing – the processes are already documented in the script
- Generally want to store raw data but could store processed copy if e.g. errors in raw data are corrected during processing

All together now…



Dates for your diary

- 10th October, Cambridge – Data Analysis of Citizen Science Data
- 11th-14th December, Liverpool – BES Annual Meeting 2016
  - Quantitative Ecology SIG social
  - Quantitative Ecology SIG wikithon
  - Methods in Ecology and Evolution workshop “Best practise for code archiving”

Keep in touch

 @BES_QE_SIG

quantitative@britishecologicalsociety.org

https://besquantitativeecology.wordpress.com/

FEEDBACK: http://goo.gl/forms/h9DjBui1z17ICFKC3