SOLES codeRs

Reproducible research with tidy projects, data, and code

Thomas White

Hi!

Thomas (Tom) E. White

- Email: thomas.white@sydney.edu.au
- Office: A08 337
- Lab: The Sensory and Evolutionary Ecology (SEE) Lab

The Sensory and Evolutionary Ecology (SEE) Lab

The evo-ecology of information

Behaviour

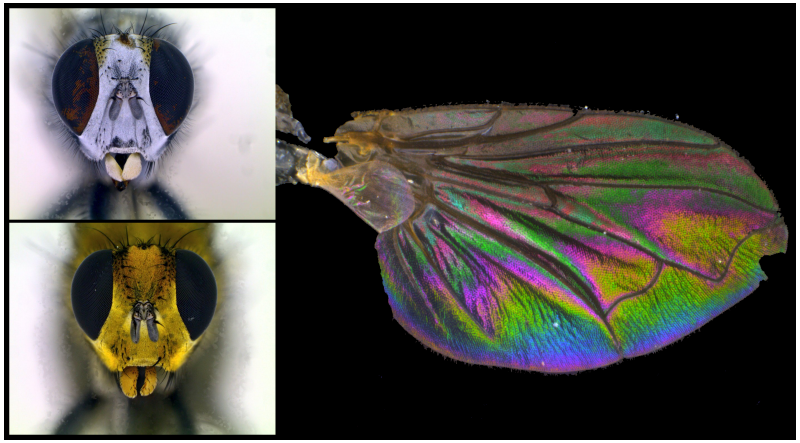
- Communication
- Perception
- Decision-making

Evo-Ecology

- Sexual selection
- Insect <-> plant
- Predator <-> prey

Meta-science

- Tools and methods
- Meta-analysis
- Evidence synthesis



codeRs?

- Nice to keep in touch, hear what's happening
- Fun to learn stuff from and with others
- Have a spot (physical & online) to ask for ideas, help, direction
- Food
- Ask each
- Total work-in-progress, will evolve continuously, so ideas welcome!
- solescodeRs.github.io
- Slack

Tidying up: Outcomes

- Understand the principles and importance of reproducibility in science
- Learn the key steps in producing reproducible research
- Create and detail the structure and value of ‘tidy’ projects, data, and code

Science is open & robust

'Open' science is the practice of making everything in the discovery process fully and openly available, creating transparency and driving further discovery by allowing others to build on existing work

Features of robust science

Reproducible:

Replicable:

Features of robust science

Reproducible: The same result can be independently reached given the same data & analysis pipeline.

Replicable: The same result can be independently reached given independent data & analysis pipeline.

Why conduct reproducible science?

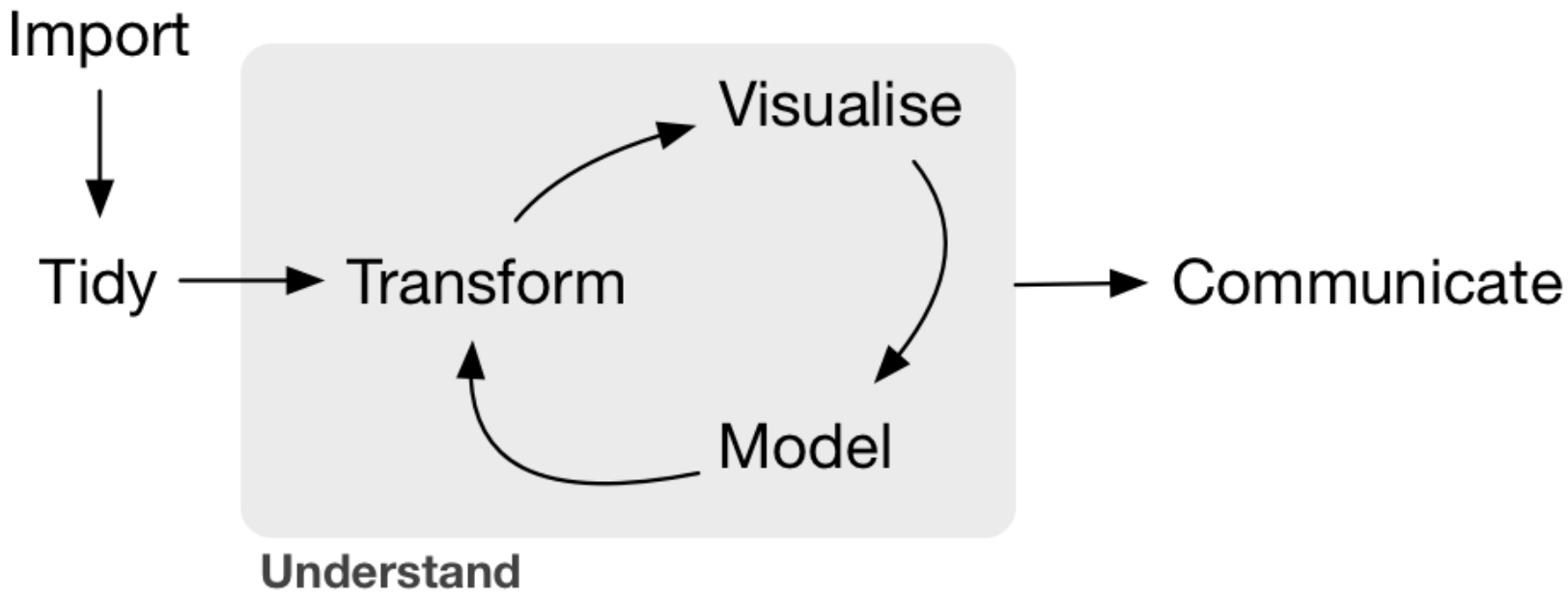
Huge practical benefits!

- Easier to share and reuse, as projects are richly documented and detailed, which is the point of science!
- Newly-collected data can be integrated easily into existing projects and analyses
- Mistakes are easier to detect and remedy
- Documents (e.g. manuscripts) are easier to revise & update as data and/or analyses change
- The exact steps you took to produce an end-result (e.g. a manuscript) will be richly documented and self-contained forever & ever
- You can more easily steal from yourself & minimise the duplication of effort into the future
- Easier to collaborate (once everyone's on-board)

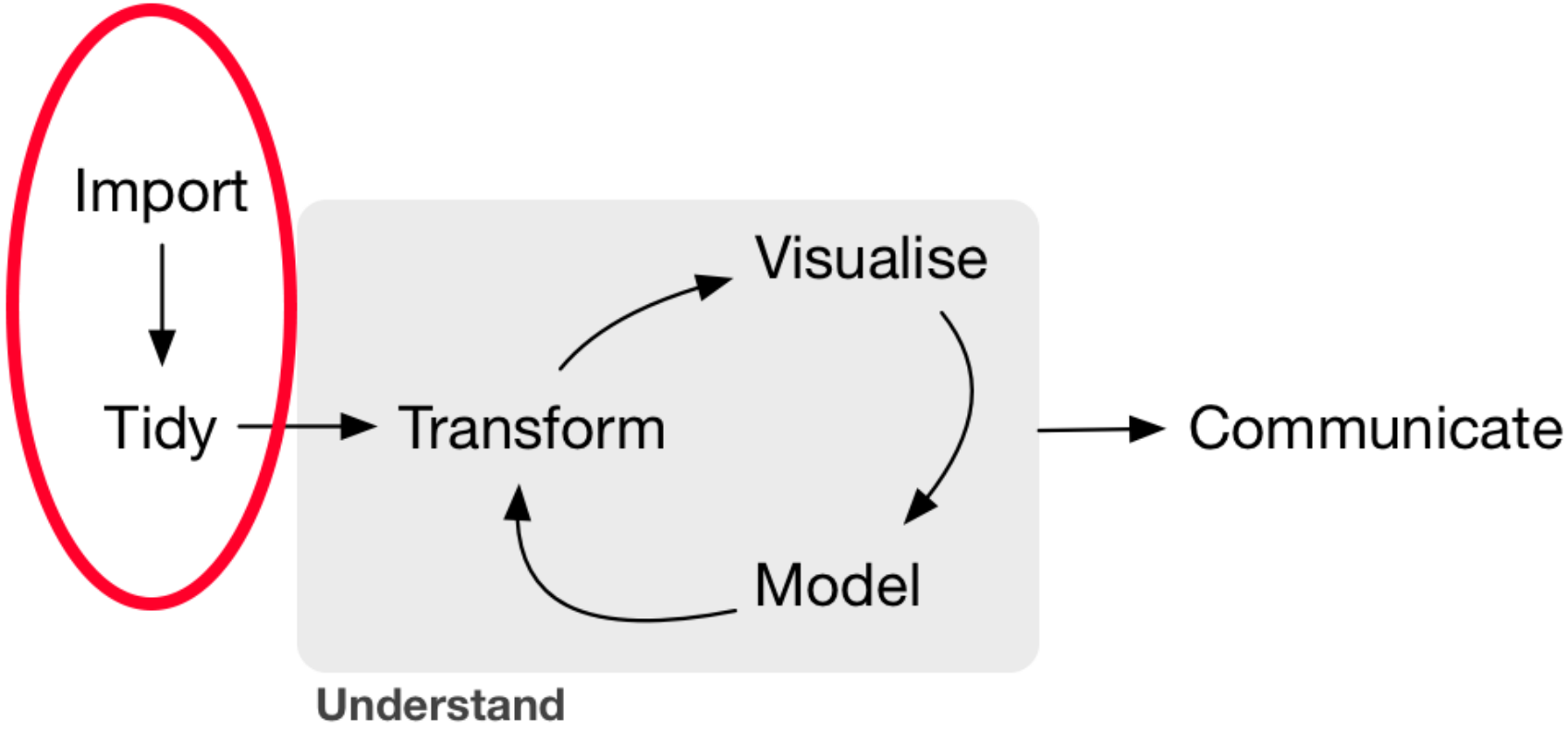
Practical principles for reproducible research

- **Principle 1:** Someone unfamiliar with your project should be able to look at your computer files and understand in detail what you did and why. (N.B 'Someone' includes future-you).
- **Principle 2:** Everything you do, you will probably have to do over again.

Practical steps to reproducible research



Practical steps to reproducible research



Practical steps to reproducible research

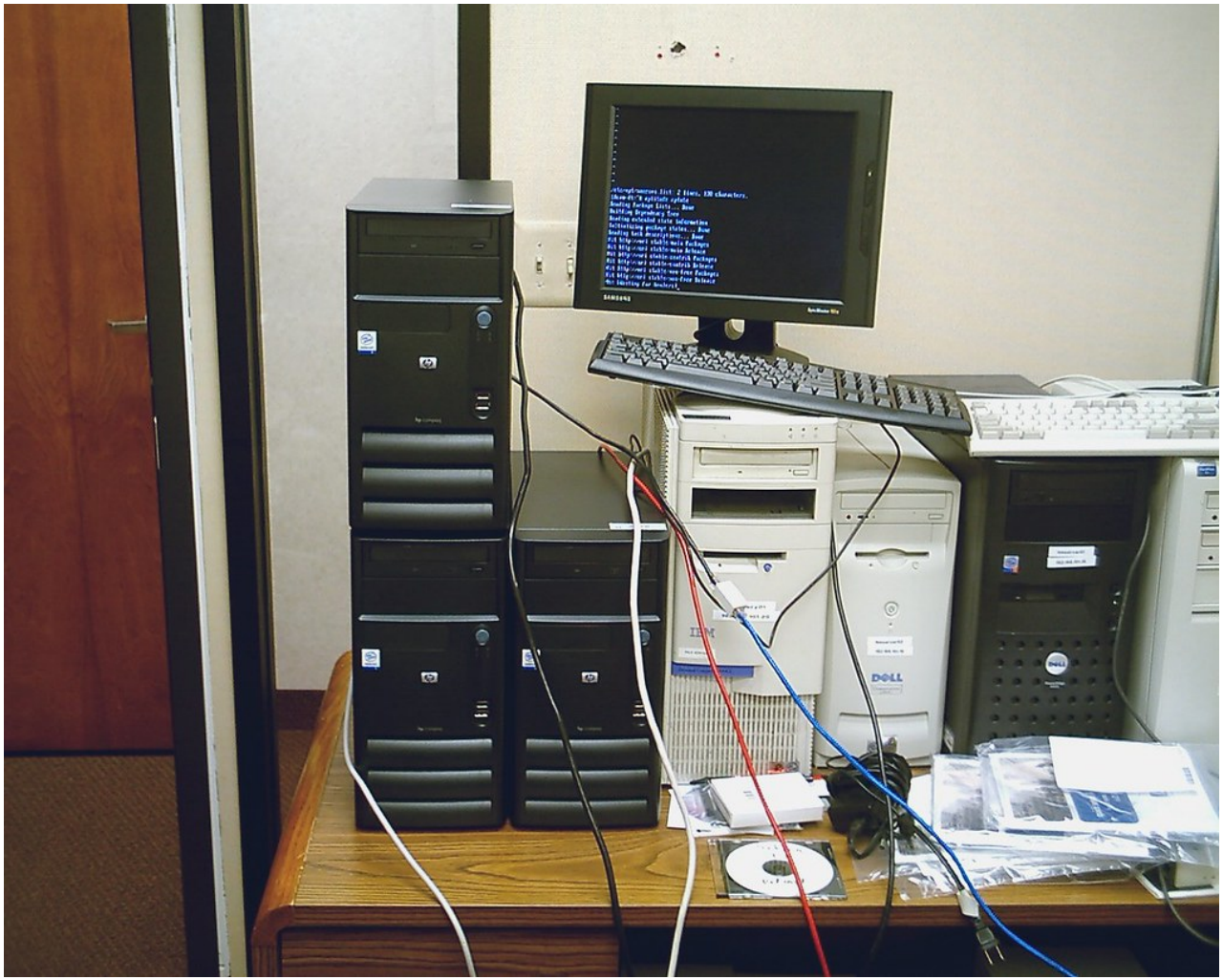
Tidy projects

Tidy data

Tidy code

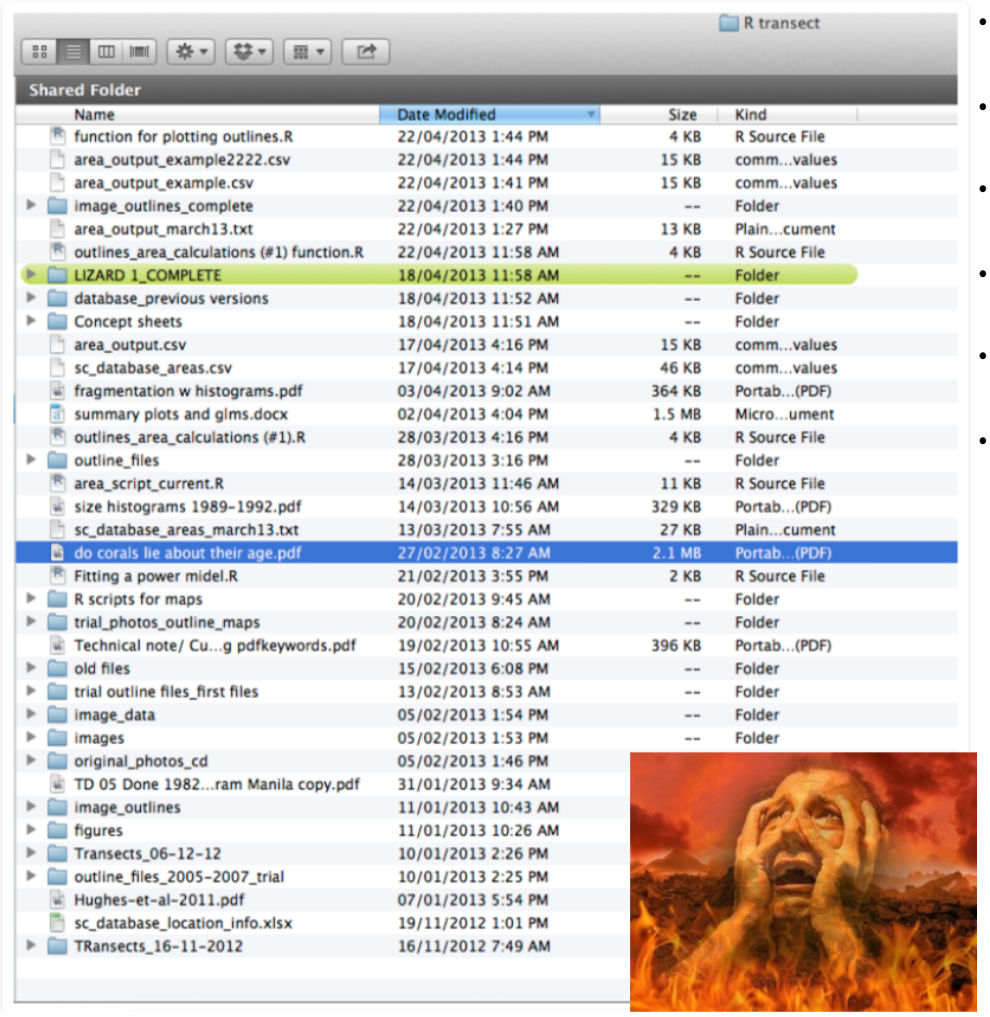
Tidy projects

What is a tidy project?



Source: phil_g

What is a tidy project?



- Hard to find anything
- What's canonical?
- What's important?
- Errors guaranteed & tough to trace
- Unshareable
- It's just awful

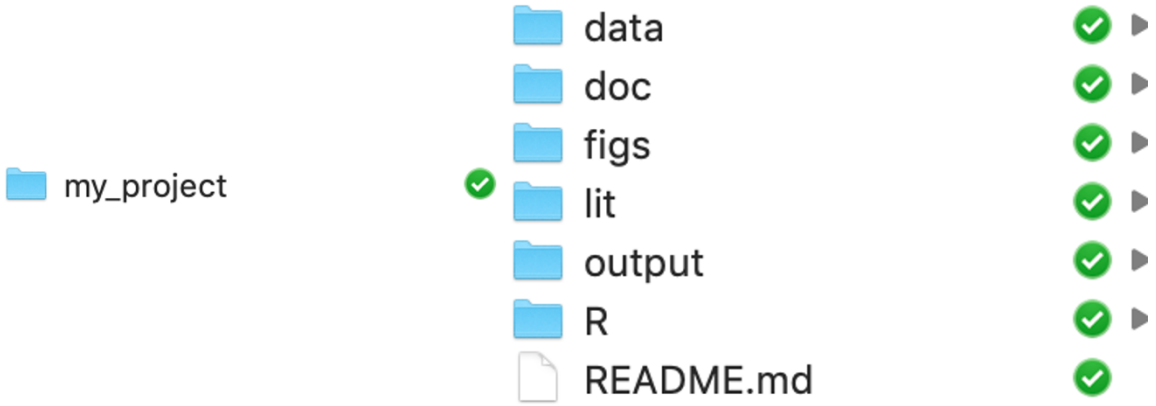
What is a tidy project?

my_project

- data
- doc
- figs
- lit
- output
- R
- README.md

Three tips for tidy projects

- Make it self-contained
- Create a consistent, sensibly-named directory structure
- Include a readme, documenting the layout & contents



For example

- **informative_project_name**
  - **README.txt** (a readme text file at the top-level of the directory which outlines the broad structure/details of the project)
  - **/data** (raw data, such as images or videos, as well as the processed products for analysis)
  - **/doc** (all notes and the draft manuscript associated with the project)
  - **/figs** (figures to be included in the manuscript, typically generated via code)
  - **/output** (programmatically-generated output from data handling and analysis such as tables of statistical results, which can be re-generated at any time)
  - **/R** (code for processing and analysing data)

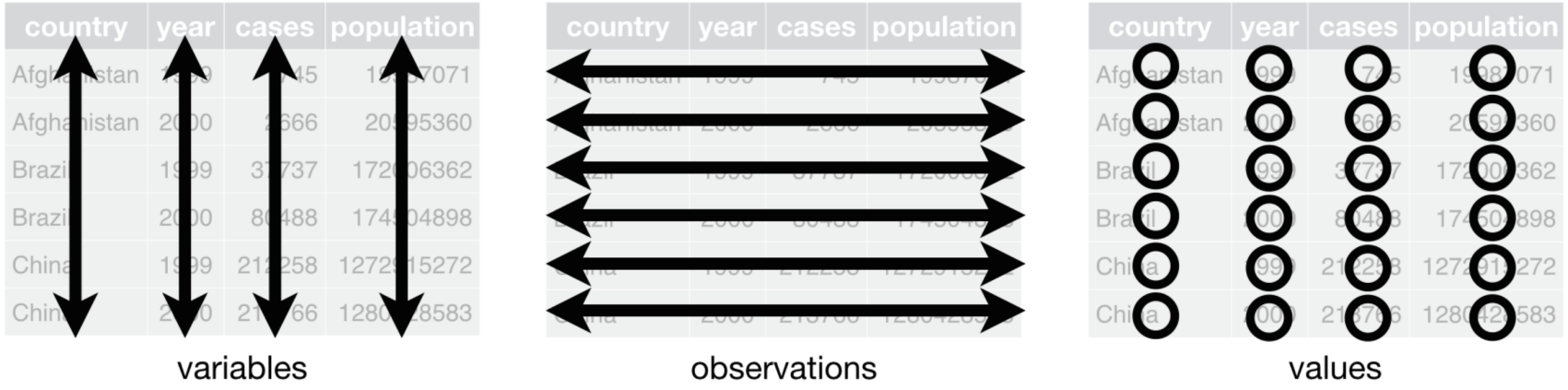
Tidy data

What is tidy *data*?

Eight golden rules for data organisation

1. Each variable forms a column, each observation a row
2. Use plain text
3. Choose good names
4. No empty cells
5. Use metadata
6. Treat raw data as read-only
7. Be consistent
8. Dates are awful

1. Each variable a column, observation a row



1. Each variable a column, observation a row

Messy

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Source: Wickham (2009) & Pew Research Center

1. Each variable a column, observation a row

Tidy

religion	income	freq
Agnostic	<$10k	27
Agnostic	$10-20k	34
Agnostic	$20-30k	60
Agnostic	$30-40k	81
Agnostic	$40-50k	76
Agnostic	$50-75k	137
Agnostic	$75-100k	122
Agnostic	$100-150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Source: Wickham (2009) & Pew Research Center

1. Each variable a column, observation a row

Messy

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

Source: Wickham (2009)

1. Each variable a column, observation a row

Tidy

id	date	tmax	tmin
MX17004	2010-01-30	27.8	14.5
MX17004	2010-02-02	27.3	14.4
MX17004	2010-02-03	24.1	14.4
MX17004	2010-02-11	29.7	13.4
MX17004	2010-02-23	29.9	10.7
MX17004	2010-03-05	32.1	14.2
MX17004	2010-03-10	34.5	16.8
MX17004	2010-03-16	31.1	17.6
MX17004	2010-04-27	36.3	16.7
MX17004	2010-05-27	33.2	18.2

Source: Wickham (2009)

2. Use plain text

Microsoft excel through the ages

- `.xls`
- `.xlt`
- `.xlm`
- `.xlam`
- `.xltm`
- `.xlsx`
- `.xltx`
- `...`

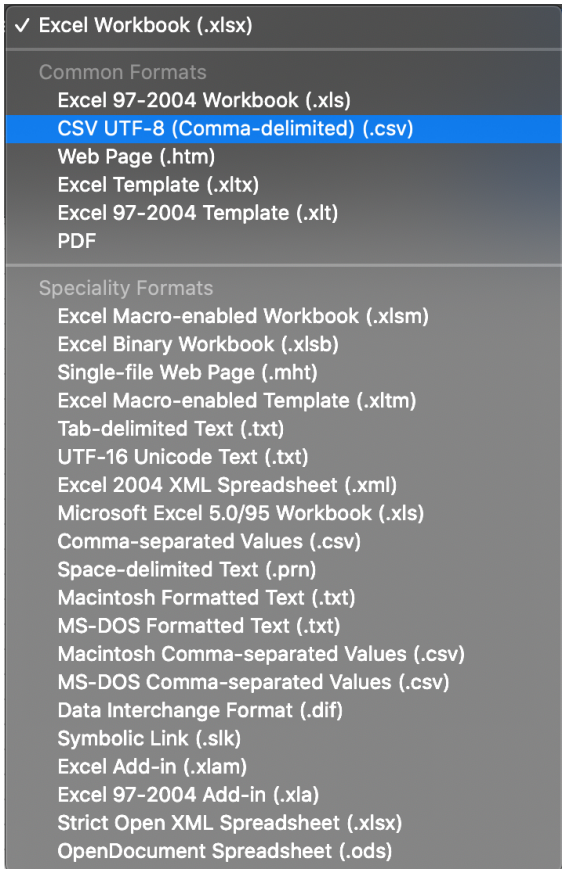
Text through the ages

- `.txt`

2. Use plain text

Types of text file

- `.csv`: comma-separated values. Great all-purpose format.
- `.txt` or `.tsv`: plain-text/tab-delimited.
- Future-proof
- Can be opened with anything/anywhere



3. Choose good names

Untidy

- myabstract.docx
- Tom's best ideas.docx
- figure 1.png
- newNEWv2_dontdelete_forREAL_dont_FINALfinal_v2.xlsx

Tidy

- 2020_abstract_for_hons_conf.docx
- toms_ideas.docx
- fig_01_scatterplot_length_width.png
- 2019-08-07_raw_data_LIFE4000.xlsx

3. Choose good names

Good names are

- Machine readable
- Human readable
- Nicely ordered

3. Choose good names

Good names are

- Machine readable
  - No special characters or formatting

Don’t use: `! @ # $ % ^ & * ( ) ~ + =`

3. Choose good names

Good names are

- Machine readable
  - No special characters or formatting

Do use: _ -

for `separating_metadata` and `splitting-up-words`

3. Choose good names

Good names are

- Machine readable
- Human readable
  - Names contain information on **content**

Nay: `data 1.csv`

3. Choose good names

Good names are

- Machine readable
- Human readable
  - Names contain information on **content**

Yay: `2020-08-09_field-data_heights-weights.csv`

3. Choose good names

Good names are

- Machine readable
- Human readable
- Nicely ordered
  - Think about sorting

Chronological

```
2020-08-09_field-data_heights-weights.csv

2020-08-12_field-data_heights-weights.csv

2020-08-18_field-data_heights-weights.csv
```

3. Choose good names

Good names are

- Machine readable
- Human readable
- Nicely ordered
    - Think about sorting

Logical

```
01_load_functions.R

02_clean_data.R

03_analysis.R
```

4. No empty cells

Or special characters

```
##    cow_ID milk_volume weight
## 1     moo          12   1100
## 2   bumbo           2   1201
## 3    spot           ?   1084
## 4 jeffrey               1044
## 5    holy          16   1244
## 6   daisy           -   1093
```

4. No empty cells

Use NA if NA, or 0 if 0

```
##    cow_ID milk_volume weight
## 1     moo          12   1100
## 2   bumbo           2   1201
## 3    spot          NA   1084
## 4 jeffrey           0   1044
## 5    holy          16   1244
## 6   daisy           0   1093
```

5. Use metadata

or a ‘data dictionary’

Data

Meta-data

variable	type	description
author	character	authors
year	numeric	publication year
obs	character	observation level ID
study	character	study level ID
group	character	group level ID

5. Use metadata

or a ‘data dictionary’

- Metadata = data *about* data
- A file describing the contents & structure of a separate file
- The richer & more detailed the better
- Essential to reproducibility (not least for yourself)

6. Treat raw data as read-only

Hands off!

n	r	r_transformed
39	0.19	0.42
36	0.22	0.11
36	0.12	
38	0.38	

6. Treat raw data as read-only

Modify by hand (only when unavoidable)

- Create a work on a copy
- Document *every* change you make in a separate file

Modify via code (whenever possible)

n	r	
39	0.19	
36	0.22	
36	0.12	
36	0.38	
37	0.03	

```
data$r_transformed <- sqrt(data$r)
```

n	r	r_transformed	
39	0.19	0.43588989	
36	0.22	0.46904158	
36	0.12	0.34641016	
36	0.38	0.6164414	

7. Be consistent

e.g. Naming conventions

- snake_case
- camelCase
- SCREAMING_SNAKE_CASE
- kebab-case
- Train-Case

PICK-one_AndUse_ItCONSISTENTLY

8. Dates are awful

- `MM/DD/YY`
- `DD/MM/YY`
- `YY/MM/DD`
- `DD-MM-YYYY`
- `MM-YY`
- Not to mention excel's handling of them

Instead, split up the variables:

day	month	year
12	11	2019
13	10	2019
15	9	2019

Or if you must, use the ISO standard: `YYYY-MM-DD`

R tools to help along the way

- **`library(janitor)`**
  - `clean_names()`: creates consistent, tidy-rule-following variable names
  - `remove_empty()`: remove rows/columns/both containing missing or empty data
  - `convert_to_date()`: take the fight to Excel's concept of a date
- **`library(tidyverse)`**
  - Set of ~25 packages for cleaning/wrangling/visualising…
  - `tidyr`: reshaping data
  - `dplyr`: manipulating data

Tidy code

Four steps to code cleanliness

1. Choose good names & be consistent
2. Write human-readable code
3. Keep it self-contained
4. Keep it well-styled (and use help)

1. Choose good names & be consistent

Good:

```
dat_heights_2020 <- read.csv('2020_field_data_heights.csv')
```

Less good (maybe)

```
dat_field <- read.csv('2020_field_data_heights.csv')
```

Bad

```
dat <- read.csv('2020_field_data_heights.csv')
```

2. Write human-readable code

Comment liberally & richly

```
## ----------------- Load data ----------------- ##

`dat_heights_2020 <- read.csv('2020_field_data_heights.csv')`  # Summer 2020
`dat_heights_2021 <- read.csv('2021_field_data_heights.csv')`  # Winter 2021

## ----------------- Summarise data ----------------- ##

# Calculate mean +- SD heights
dat_heights_summary %>%
  summarise(mean = mean(),
            sd = sd(),
            n = n())
```

2. Write human-readable code

Use space

Good

```
height <- cm * 6 + mm
mean(x, na.rm = TRUE)
```

Bad

```
height<-cm*6+mm
mean(x,na.rm=TRUE)
```

2. Write human-readable code

Use space

Good

```
do_something_very_complicated(
  something = "that",
  requires = many,
  arguments = "which may be long"
)
```

Bad

```
do_something_very_complicated("that", requires, many, arguments, "which may be long")
```

3. Keep it self-contained

Use relative file paths, never absolute.

- Forget `setwd()` exists
- Assume a script is being run from the ‘root’ of the project
- Use paths *relative* to that

3. Keep it self-contained

Use relative file paths, never absolute.

Bad

```
`data <- read.csv('C:/tomscomputer/projects/feeding_experiment/data/feeding_data.csv')`
```

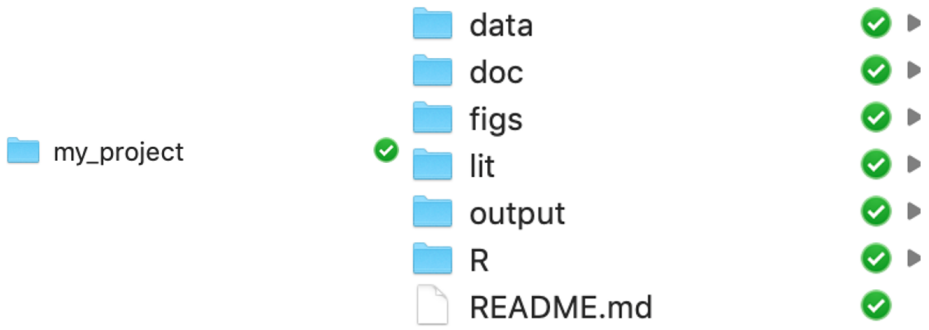
- Won't run on any other computer
- Won't run on *my* computer, if I ever move it or modify my filesystem

3. Keep it self-contained

Use relative file paths, never absolute.

Good

```
`data <- read.csv('data/feeding_data.csv')`
```



Also see `here::here()`

4. Keep it well-styled (and use help)

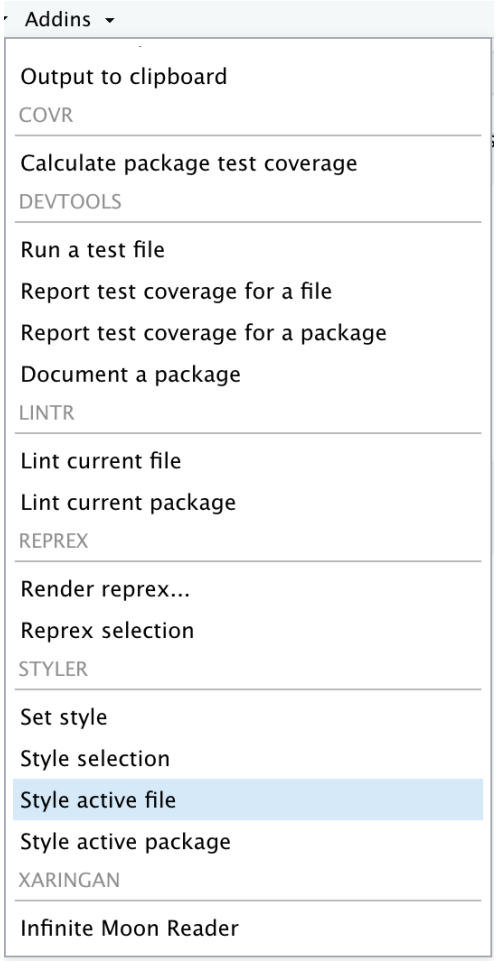
```
styler::style_file()
```

Before

```
height<-cm*6+mm+2; mean(x,na.rm=TRUE)
```

After

```
height <- cm * 6 + mm + 2
mean(x, na.rm = TRUE)
```



Outcomes

- Understand the principles and importance of reproducibility in science
- Learn the key steps in producing reproducible research
- Create and detail the structure and value of ‘tidy’ projects, data, and code

Thanks!