Tidying Data in the Tidyverse

Megan Lewis

2025-05-29

Introduction to session

- What is “untidy” data?
- What is “tidy” data?
- An intro to the `tidyverse`
- A couple `tidyverse` packages

The problem with data

- Real world data often messy
- Often not collected in tidy formats
- Can slow down analysis and increase frustration

Messy, untidy data

> “Happy families are all alike; every unhappy family is unhappy in its own way.” - Leo Tolstoy

> “Tidy datasets are all alike, but every messy dataset is messy in its own way.” - Hadley Wickham

According to an article in Forbes:

> “Data preparation accounts for about 80% of the work of data scientists”

Common problems with messy data sets

- Column headers are values but should be variable names
- Single column with multiple variables
- Variables entered in both rows and columns
- Multiple types of observational units stored in the same table
- A single observational unit is stored in multiple tables

Examples from: Hadley Wickham (2014)

Column headers are values but should be variables

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

Single column with multiple variables

country	year	column	cases
AD	2000	m014	0
AD	2000	m1524	0
AD	2000	m2534	1
AD	2000	m3544	0
AD	2000	m4554	0
AD	2000	m5564	0
AD	2000	m65	0
AE	2000	m014	2
AE	2000	m1524	4
AE	2000	m2534	4
AE	2000	m3544	6
AE	2000	m4554	5
AE	2000	m5564	12
AE	2000	m65	10
AE	2000	f014	3

Variables entered in both rows and columns

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	—	—	—	—	—	—	—	—

Multiple types of observational units stored in the same table

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

One type in mutiple tables

- Values about a single type of observational unit
- In several tables, split by another variable (e.g., year, person, location etc.)

Other issues - Excel formatting…

- Merged cells
- Explanatory text (should be elsewhere)
- Unnecessary table *decoration*
- Summary stats/info

What can untidy data look like?

AutoSave Off | 1.4-untidy.xlsx | Search

File Home Insert Draw Page Layout Formulas Data Review View Help Acrobat

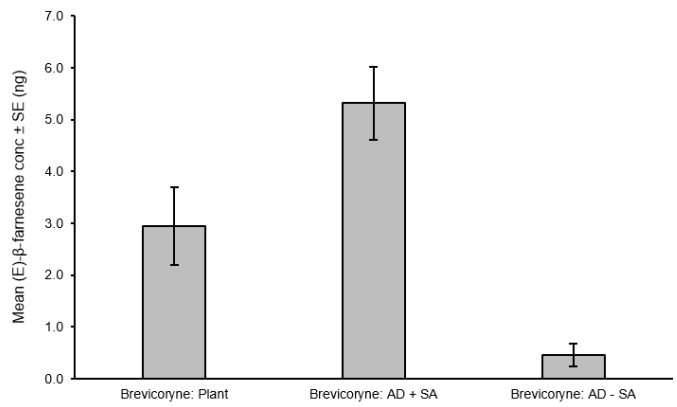
Cut Copy Format Painter Paste | Calibri 11 | Wrap Text Merge & Center | General | Conditional Formatting Format as Table | Normal Bad Good Neutral | Insert Delete Format | AutoSum Fill Clear

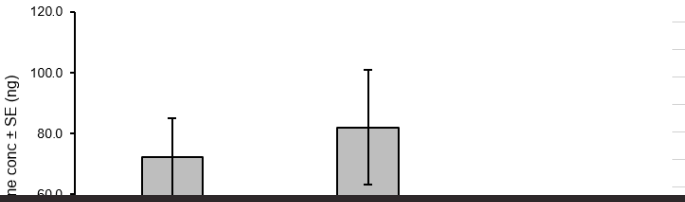
Clipboard Font Alignment Number Styles Cells

11 | Brevicoryne: AD + SA

Sample ID	Sample	RT	Peak Area	Total Conc (ng)	Individual Conc (ng)	Peak Area (1 ng)
CG19-05	2 ul EBF Std @ 1000 ng/ul	16.25	32876919	2000	-	16438
CG19-14	Brevicoryne: Plant	16.25	154123	9.4	3.1	-
CG19-23	Brevicoryne: Plant	16.24	77839	4.7	1.6	-
CG19-24	Brevicoryne: Plant	16.24	204238	12.4	4.1	-
CG19-09	Myzus: Plant	16.24	4783237	291.0	97.0	-
CG19-13	Myzus: Plant	16.25	2653330	161.4	53.8	-
CG19-26	Myzus: Plant	16.24	3224152	196.1	65.4	-
CG19-28	Brevicoryne: AD + SA	16.24	295414	18.0	6.0	-
CG19-31	Brevicoryne: AD + SA	16.24	298222	18.1	6.0	-
CG19-34	Brevicoryne: AD + SA	16.24	192785	11.7	3.9	-
CG19-30	Brevicoryne: AD - SA	N/A	0	0.0	0.0	-
CG19-32	Brevicoryne: AD - SA	16.24	33358	2.0	0.7	-
CG19-33	Brevicoryne: AD - SA	16.24	34018	2.1	0.7	-
CG19-15	Myzus: AD + SA	16.24	4034920	245.5	81.8	-
CG19-18	Myzus: AD + SA	16.24	2442438	148.6	49.5	-
CG19-22	Myzus: AD + SA	16.24	5650064	343.7	114.6	-
CG19-17	Myzus: AD - SA	16.24	51450	3.1	1.0	-
CG19-20	Myzus: AD - SA	16.24	846354	51.5	17.2	-
CG19-21	Myzus: AD - SA	16.24	110176	6.7	2.2	-

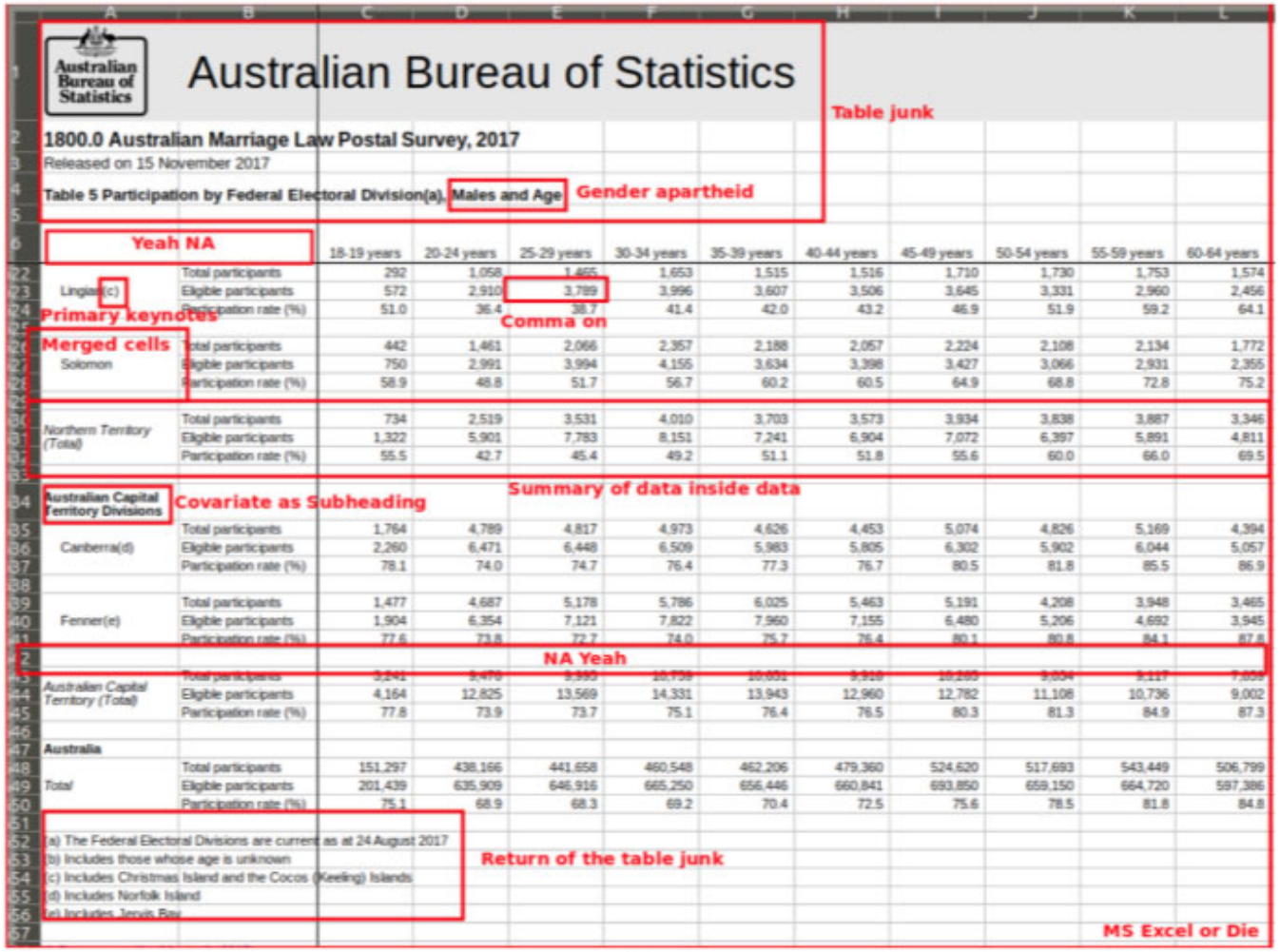
Sample	Mean Conc	SE Conc
Brevicoryne: Plant	2.9	0.7
Myzus: Plant	72.1	12.9
Brevicoryne: AD + SA	5.3	0.7
Brevicoryne: AD - SA	0.5	0.2
Myzus: AD + SA	82.0	18.8
Myzus: AD - SA	6.8	5.2





Harper Adams R Users Group

What can untidy data look like?



Australian Bureau of Statistics

1800.0 Australian Marriage Law Postal Survey, 2017

Released on 15 November 2017

Table 5 Participation by Federal Electoral Division(a), Males and Age

Table junk

Gender apartheid

Yeah NA

Primary keynotes

Merged cells

Comma on

Summary of data inside data

Covariate as Subheading

NA Yeah

Return of the table junk

MS Excel or Die

	18-19 years	20-24 years	25-29 years	30-34 years	35-39 years	40-44 years	45-49 years	50-54 years	55-59 years	60-64 years
Lingiari(c) – Total participants	292	1,058	1,465	1,653	1,515	1,516	1,710	1,730	1,753	1,574
Lingiari(c) – Eligible participants	572	2,910	3,789	3,996	3,607	3,506	3,645	3,331	2,960	2,456
Lingiari(c) – Participation rate (%)	51.0	36.4	38.7	41.4	42.0	43.2	46.9	51.9	59.2	64.1
Solomon – Total participants	442	1,461	2,066	2,357	2,188	2,057	2,224	2,108	2,134	1,772
Solomon – Eligible participants	750	2,991	3,994	4,155	3,634	3,398	3,427	3,066	2,931	2,355
Solomon – Participation rate (%)	58.9	48.8	51.7	56.7	60.2	60.5	64.9	68.8	72.8	75.2
Northern Territory (Total) – Total participants	734	2,519	3,531	4,010	3,703	3,573	3,934	3,838	3,887	3,346
Northern Territory (Total) – Eligible participants	1,322	5,901	7,783	8,151	7,241	6,904	7,072	6,397	5,891	4,811
Northern Territory (Total) – Participation rate (%)	55.5	42.7	45.4	49.2	51.1	51.8	55.6	60.0	66.0	69.5
Australian Capital Territory Divisions										
Canberra(d) – Total participants	1,764	4,789	4,817	4,973	4,626	4,453	5,074	4,826	5,169	4,394
Canberra(d) – Eligible participants	2,260	6,471	6,448	6,509	5,983	5,805	6,302	5,902	6,044	5,057
Canberra(d) – Participation rate (%)	78.1	74.0	74.7	76.4	77.3	76.7	80.5	81.8	85.5	86.9
Fenner(e) – Total participants	1,477	4,687	5,178	5,786	6,025	5,463	5,191	4,208	3,948	3,465
Fenner(e) – Eligible participants	1,904	6,354	7,121	7,822	7,960	7,155	6,480	5,206	4,692	3,945
Fenner(e) – Participation rate (%)	77.6	73.8	72.7	74.0	75.7	76.4	80.1	80.8	84.1	87.8
Australian Capital Territory (Total) – Total participants	3,241	9,476	9,995	10,736	10,601	9,918	10,265	9,034	9,117	7,859
Australian Capital Territory (Total) – Eligible participants	4,164	12,825	13,569	14,331	13,943	12,960	12,782	11,108	10,736	9,002
Australian Capital Territory (Total) – Participation rate (%)	77.8	73.9	73.7	75.1	76.4	76.5	80.3	81.3	84.9	87.3
Australia										
Total – Total participants	151,297	438,166	441,658	460,548	462,206	479,360	524,620	517,693	543,449	506,799
Total – Eligible participants	201,439	635,909	646,916	665,250	656,446	660,841	693,850	659,150	664,720	597,386
Total – Participation rate (%)	75.1	68.9	68.3	69.2	70.4	72.5	75.6	78.5	81.8	84.8

(a) The Federal Electoral Divisions are current as at 24 August 2017
(b) Includes those whose age is unknown
(c) Includes Christmas Island and the Cocos (Keeling) Islands
(d) Includes Norfolk Island
(e) Includes Jervis Bay

https://medium.com/@miles.mcbain/tidying-the-australian-same-sex-marriage-postal-survey-data-with-r-5d35cea07962

Why we don’t want messy data

- Difficult to analyse
- Easy to make mistakes
- Not reproducible

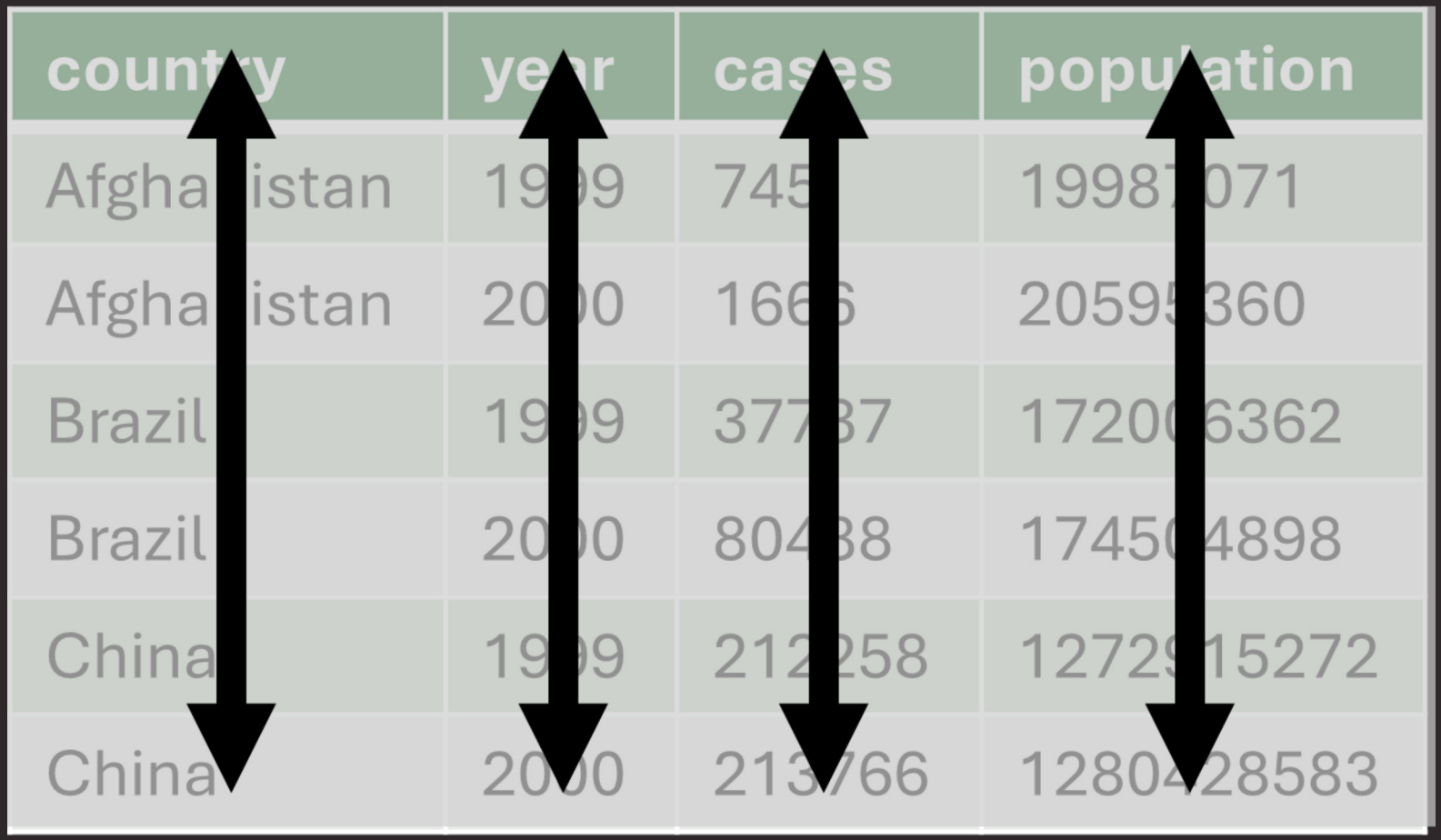
Messy data might not be perfect, but we can work with it…

But what does “tidy” data look like?

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	1666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Tidy data

One *column*: One *variable*



country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	1666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Variables

One *row*: One *observation*

One *cell*: One *value*

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	1666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

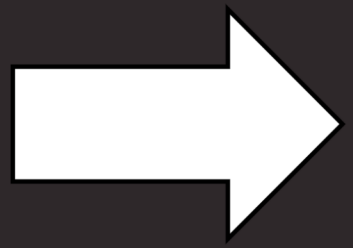
Values

Tidy data visual example

ID	BP1	BP2
A	100	120
B	140	115
C	120	125

Tidy data visual example

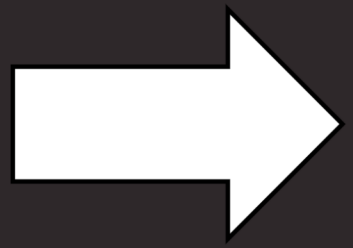
ID	BP1	BP2
A	100	120
B	140	115
C	120	125



ID	measurement	value
A	BP1	100
A	BP2	120
B	BP1	140
B	BP2	115
C	BP1	120
C	BP2	125

Tidy data visual example

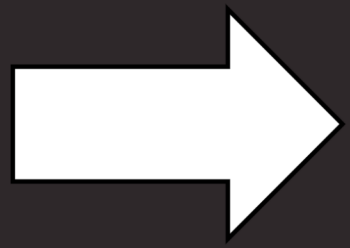
ID	BP1	BP2
A	100	120
B	140	115
C	120	125



ID	measurement	value
A	BP1	100
A	BP2	120
B	BP1	140
B	BP2	115
C	BP1	120
C	BP2	125

Tidy data visual example

ID	BP1	BP2
A	100	120
B	140	115
C	120	125



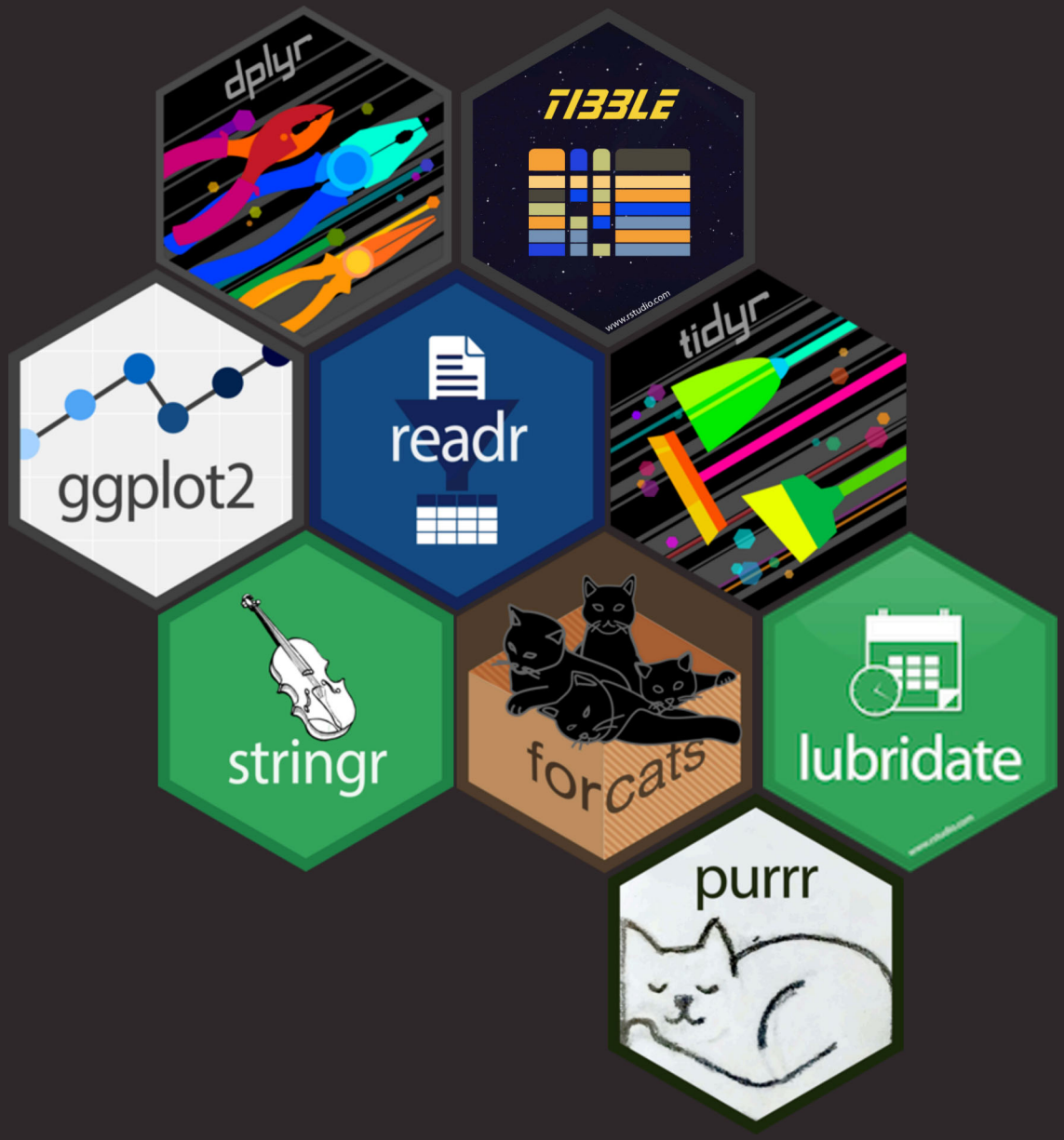
ID	measurement	value
A	BP1	100
A	BP2	120
B	BP1	140
B	BP2	115
C	BP1	120
C	BP2	125

Why we like ‘tidy’ data

- Easier to wrangle, plot and model
- Gives us a standard structure that works beautifully with the `tidyverse`

Dramatic entrance from the Tidyverse…

- Collection of R packages for data science
- Built around tidy data principles
- Consistent syntax & philosophy



What's in the Tidyverse?

Visualize your data

Manipulate your data

Tidy your data

What's in the Tidyverse?

Read rectangular data

Work with functions and vectors

Re-imagining the data frame

What's in the Tidyverse?

Work with strings

Work with factors

Work with dates and times

Today's focus

Tidying

Wrangling

A couple quirks…

The tidyverse can have a couple weird but helpful features…

- Tibbles
- Pipes

Tibbles

- Special type of data frame used in the tidyverse
- Designed for large datasets
  - Only show the first few rows and columns that fit on one screen
  - can use `View()` to see data in interactive, scrollable and filterable view
  - or `glimpse()` to see all variables in console

Pipes

- Two forms:
  - Native pipe `|>`
  - `magrittr` package pipe `%>%`
    - (*package loaded as part of tidyverse*)
- Keyboard short cut
  - `ctrl` + `shift` + `m`
  - `cmd` + `shift` + `m`
  - Need to change R settings to use native pipe with shortcut

Why pipes?

- Takes whats on the left and passes it along to the function on it's right
- Reduces need for nesting or use of intermediate objects
- Native `|>` vs magrittr `%>%`
  - generally behave identically
  - `%>%` was introduced in 2014 in R
  - `|>` included in R in 2021 but now part of base R

Quick syntax with and without pipes

No pipe

- `function(data, arguments)`
- Data is first argument
- Function **wraps around** data

With Pipe

- `data |> verb(arguments)`
- Data **flows into** function
- Easier to read and chain multiple steps

The `tidyr` package

- Help tidy messy data
- Focus on reshaping and reorganizing
- Built around tidy data principles

`tidyr` functions

- `pivot_longer()`: Reshapes wide data into long format
- `pivot_wider()`: Turns long data into wide format
- `separate()`: Splits one column into multiple based on a delimiter
- `unite()`: Combines multiple columns into one
- `drop_na()` / `replace_na()`: Remove or fill in missing values in a tidy-friendly way

pivot_longer()

tidyr

```r
# Pivot week columns into long format
billboard_long <- billboard_sm |>
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank"
  )
```

Base R

```r
# Reshape wide week columns to long
billboard_long_base <- reshape(
  billboard_sm,
  varying = grep("^wk", names(billboard_sm), value = TRUE),
  v.names = "rank",
  timevar = "week",
  times = grep("^wk", names(billboard_sm), value = TRUE),
  direction = "long"
)
```

separate()

tidyr

```r
# Split date.entered into components
billboard_sep_date <- billboard_long |>
  separate(date.entered, into = c("year", "month", "day"), sep = "-")
```

Base R

```r
# Use strsplit to separate date into components
date_parts <- do.call(rbind, strsplit(as.character(billboard_long_base$date
billboard_long_base$year <- date_parts[, 1]
billboard_long_base$month <- date_parts[, 2]
billboard_long_base$day <- date_parts[, 3]
```

Demo ft. glittery dung data

Does the addition of glitter to dung affect brood ball production in dung beetles?

Ecological Entomology

Royal Entomological Society

METHODS | Open Access

A simple technique to assess resource use in dung beetle breeding studies

Megan J. Lewis, Jacob D. Berson, Raphael K. Didham, Theodore A. Evans

First published: 11 September 2023 | https://doi.org/10.1111/een.13277

Demo ft. glittery dung data

- Independent Variables
  - Dung beetle species (*Onthophagus taurus* & *Euoniticellus fulvus*)
  - Four glitter colours + no glitter control
- Response variable
  - Number of brood balls
- Individual dung beetles in individual tubes

Demo!

Introducing `dplyr`

- Data manipulation in the tidyverse
- Consistent, readable grammar
  - First argument is always a data frame
  - Other arguments describe which column to operate on
  - Output always a new data frame
- Works with tibbles and pipes

Core `dplyr` verbs

- Each verb does one thing
- Solving complex problems require combining multiple verbs with pipes
- Organised into four groups based on what they operate on:
  - rows
  - columns
  - groups
  - tables

Row verbs

These verbs work across or modify **rows** of your dataset:

- `filter()` – keep only rows that match a condition
- `arrange()` – reorder rows
- `distinct()` – keep only unique rows

Filtering

dplyr

```r
# Filter to include only humans
starwars |>
  filter(species == "Human")
```

Base R

```r
# Filter using indexing
starwars[starwars$species == "Human", ]
```

Arranging

dplyr

```r
# Arrange by height in ascending order
starwars |>
  arrange(height)
```

Base R

```r
# Use order() to sort by height
starwars[order(starwars$height), ]
```

Distinct

dplyr

```r
# Get unique species values in a data frame
starwars |>
  distinct(species)
```

Base R

```r
# To get a data frame equivalent to dplyr
data.frame(species = unique(starwars$species))
```

Column verbs

These verbs help you work with **columns** of data:

- `mutate()` – create or transform columns
- `select()` – keep/drop columns
- `rename()` – rename columns
- `relocate()` – reorder columns

Mutate

dplyr

```
# Create a new column with height in meters
starwars |>
  mutate(height_m = height / 100)
```

Base R ::: {.fragment}

```
# Same result using base R column creation
starwars$height_m <- starwars$height / 100
```

:::

Select

dplyr

```r
# Select specific columns
starwars |>
  select(name, species, height)
```

Base R

```r
# Use column indexing
starwars[c("name", "species", "height")]
```

Rename

dplyr

```r
# Rename the 'name' column
starwars |>
  rename(character_name = name)
```

Base R

```r
# Rename column manually
names(starwars)[names(starwars) == "name"] <- "character_name"
```

Relocate

dplyr

```
# Move 'species' before 'name'
starwars |>
  relocate(species, .before = name)
```

Base R

```
# Reorder columns manually
cols <- names(starwars)
new_order <- c("species", setdiff(cols, "species"))
starwars[new_order]
```

Group verbs

These verbs are for **grouped operations**, often used in summarising:

- `group_by()` – group dataset by one or more variables
- `summarise()` – calculate summary statistics per group
- `slice_*()` – select specific rows within groups (e.g., `slice_max()`, `slice_head()`)

Group by & Summarise

dplyr

```r
# Group by species and calculate average height
starwars |>
  group_by(species) |>
  summarise(mean_height = mean(height, na.rm = TRUE))
```

Base R

```r
# Use aggregate to get same result
aggregate(height ~ species, data = starwars, FUN = mean, na.rm = TRUE)
```

Slice

dplyr

```
1 # Take first 3 rows
2 starwars |>
3   slice(1:3)
```

Base R

```
1 # Base R equivalent
2 starwars[1:3, ]
```

Piping it all together…

dplyr

```r
starwars |>
  filter(!is.na(mass)) |>          # Remove rows where mass is NA
  group_by(species) |>             # Group data by species
  summarise(avg_mass = mean(mass)) |>  # Mean mass for each species
  arrange(desc(avg_mass))          # Sort results by mean mass, descending
```

Base R

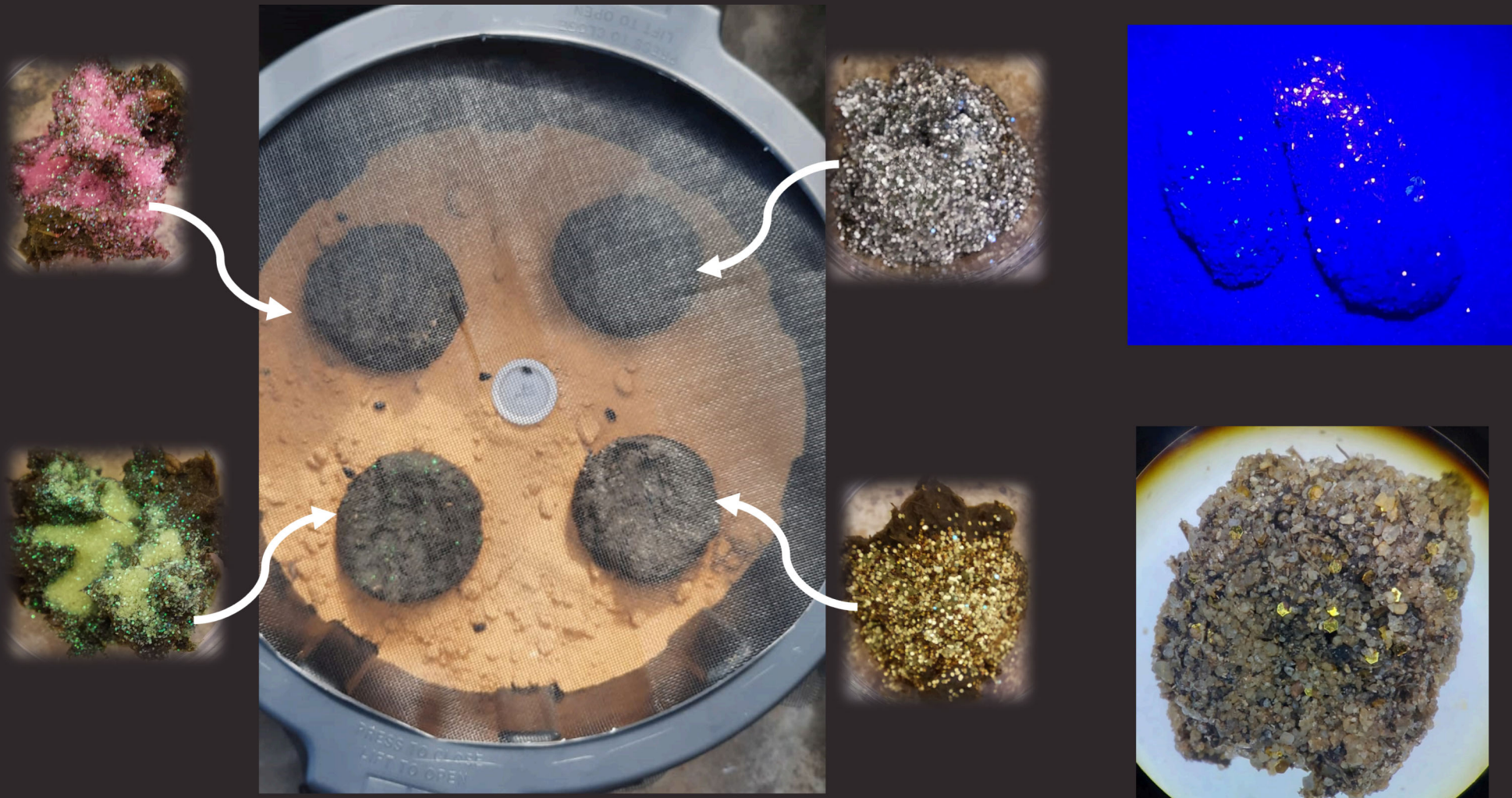
```r
# Filter out NAs
filtered <- starwars[!is.na(starwars$mass), ]

# Mmean mass by species
avg_mass <- aggregate(mass ~ species, data = filtered, FUN = mean)

# Sort descending
avg_mass <- avg_mass[order(-avg_mass$mass), ]
```

Live demo - ft. more glittery dung

Will dung beetles show a dung preference based on glitter colour when offered multiple options?



Harper Adams R Users Group

Live demo - ft. more glittery dung

- Subset of data from larger experiment (5 out of 100 bins)
- Four dung pats per bin
  - Each dung pat with different colour glitter
- Six females, six males added per bin
- Seven days to produce brood balls

Demo!

Wrap up

- Messy data is normal – but it can be tamed!
- Tidy data: Smoother analysis, faster, and more reproducible
- Tidyverse: a consistent, powerful toolkit data prep
- Pipes: build readable, step-by-step workflows
- Small, simple verbs — like filter(), mutate(), group_by() — combine to solve complex problems

> You can’t polish a turd… but you can roll it in glitter and pipe it through dplyr()

Questions

Resources

- Hadley Wickham (2014)
- R for Data Science (2e)
- Tidyverse documentation
- R Studio cheat sheets
- Tidy Tuesday Project