Tidy data

Denis Schluppeck

Introduction

- a lot of data we work with is *tabular*
- can be represented in a table with *rows* and *columns*
- maybe particular important for reporting data from repeated trials, experiments, conditions (neuroscience)
- links to *statistical reports* and *visualisations* we often want/need

Examples:

You probably have your own, but eg:

- rating in a questionnaire [per item, participant]
- reaction times [per trial, subject, condition]
- % fMRI signal change [per brain region across, subject, conditions]
- spike rate [per neuron, animal, task]

Just put them in a table, right!?

Anna Karenina principle

> "Happy families are all alike; every unhappy family is unhappy in its own way." — Leo Tolstoy

Figure 1: The ~~files~~ are in the ~~computer~~?

> "Tidy datasets are all alike, but every messy dataset is messy in its own way." — Hadley Wickham

Example table A

number of TB cases in country, population

```
table1 %>% gt()
```

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Example table B

```
table2 %>% gt()
```

2

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737

Example table C

```
table3 %>% gt()
```

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

"Tidy" means

Wickham and Grolemund (2023)

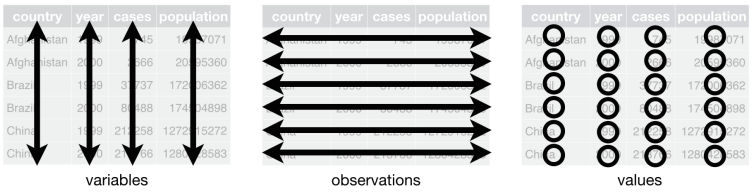


Figure 2: Tidy data illustration from R4DS

- each column represents a *variable*
- each row an *observation*
- each cell entry a *value* (number, text, …)

Benefits

- this layout leads to a series of elegant ways to manipulate table
- it's a standard (so tool builders can make code to work with it)

Wickham, H, and G Grolemund. 2023. "R for Data Science (2e)." 2023. https://r4ds.hadley.nz/.

- it plays nicely with storage (files) and visualisation (*grammar of graphics* ideas)

`Manipulating tables: concepts`

Some ideas that crop up in

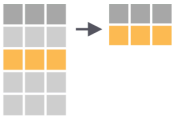
- `sql`
- `dplyr` **(a popular library in `r`),**
- `pandas` (in `python`)
- `QueryVerse.jl` (in `julia`)
- `tables` in `matlab`

`Main ideas`

A really good summary on this cheatsheet – using `r` syntax, but good for ideas!

- subsetting (rows, columns)
- mutating (calculating new values)
- aggregating (grouping, summarising)
- combining (including *relational* data, `join()`)

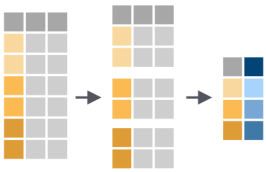
`taking rows, filter()`



filter(.data, …, .preserve = FALSE**)** Extract rows that meet logical criteria.
filter(mtcars, mpg > 20)

select(.data, …**)** Extract columns as a table.
select(mtcars, mpg, wt)

Use **group_by(**.data, …, .add = FALSE, .drop = TRUE**)** to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.

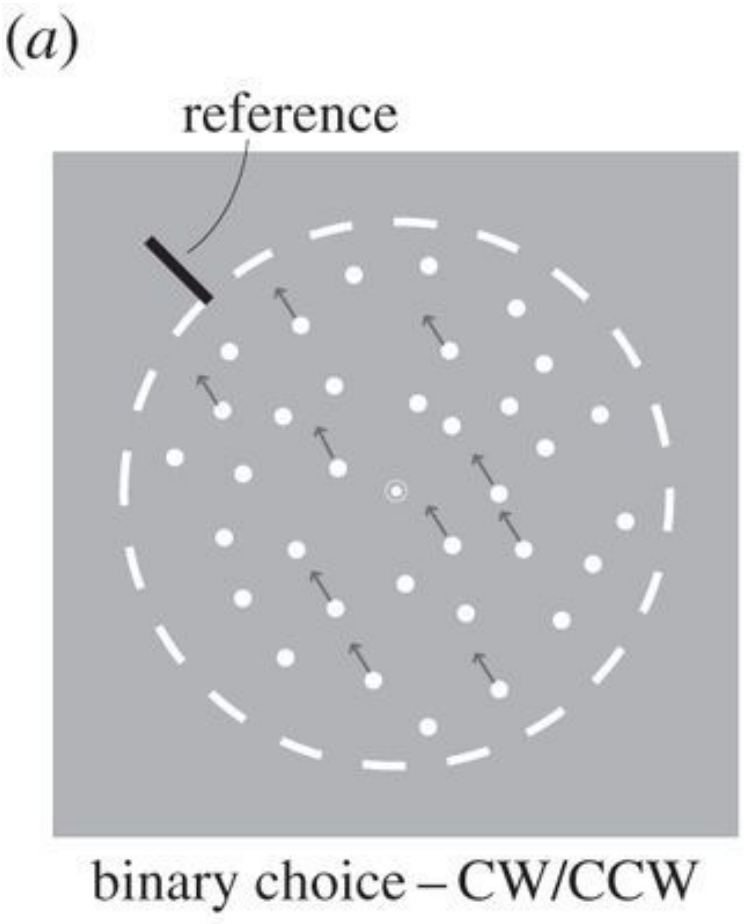


```
mtcars  %>%
  group_by(cyl) %>%
  summarise(avg = mean(mpg))
```

taking columns, `select()`

aggregate, `groupby()`, `summarize()`

By example (Psychophysics data)



As a table

```
d |> gt()
```

direction	p_cw	se	coherence	subject
-19.5	0.114	0.023	0.04	A
-15.5	0.173	0.030	0.04	A
-11.5	0.236	0.032	0.04	A
-7.5	0.276	0.033	0.04	A
-3.5	0.390	0.036	0.04	A
0.5	0.430	0.037	0.04	A
4.5	0.516	0.037	0.04	A
8.5	0.599	0.035	0.04	A
12.5	0.719	0.033	0.04	A
16.5	0.748	0.031	0.04	A
20.5	0.780	0.031	0.04	A
-19.5	0.048	0.016	0.07	A
-15.5	0.089	0.021	0.07	A
-11.5	0.106	0.023	0.07	A
-7.5	0.152	0.026	0.07	A
-3.5	0.304	0.034	0.07	A
0.5	0.397	0.036	0.07	A
4.5	0.592	0.034	0.07	A
8.5	0.695	0.033	0.07	A
12.5	0.823	0.029	0.07	A
16.5	0.831	0.029	0.07	A
20.5	0.923	0.021	0.07	A
-19.5	0.010	0.007	0.13	A
-15.5	0.049	0.015	0.13	A
-11.5	0.098	0.022	0.13	A
-7.5	0.121	0.024	0.13	A
-3.5	0.218	0.030	0.13	A
0.5	0.424	0.038	0.13	A
4.5	0.611	0.038	0.13	A
8.5	0.715	0.035	0.13	A
12.5	0.820	0.028	0.13	A
16.5	0.924	0.020	0.13	A
20.5	0.950	0.015	0.13	A
-19.5	0.005	0.005	0.25	A
-15.5	0.022	0.010	0.25	A
-11.5	0.047	0.015	0.25	A
-7.5	0.073	0.020	0.25	A
-3.5	0.140	0.026	0.25	A

0.5	0.375	0.034	0.25	A
4.5	0.593	0.037	0.25	A
8.5	0.825	0.029	0.25	A
12.5	0.904	0.021	0.25	A
16.5	0.945	0.017	0.25	A
20.5	0.972	0.012	0.25	A
-19.5	0.290	0.036	0.04	C
-15.5	0.345	0.037	0.04	C
-11.5	0.371	0.039	0.04	C
-7.5	0.393	0.040	0.04	C
-3.5	0.400	0.039	0.04	C
0.5	0.523	0.040	0.04	C
4.5	0.594	0.039	0.04	C
8.5	0.633	0.041	0.04	C
12.5	0.675	0.040	0.04	C
16.5	0.683	0.039	0.04	C
20.5	0.744	0.038	0.04	C
-19.5	0.172	0.032	0.07	C
-15.5	0.203	0.031	0.07	C
-11.5	0.236	0.035	0.07	C
-7.5	0.373	0.040	0.07	C
-3.5	0.417	0.041	0.07	C
0.5	0.493	0.041	0.07	C
4.5	0.595	0.042	0.07	C
8.5	0.725	0.036	0.07	C
12.5	0.740	0.035	0.07	C
16.5	0.800	0.035	0.07	C
20.5	0.804	0.032	0.07	C
-19.5	0.092	0.025	0.13	C
-15.5	0.131	0.030	0.13	C
-11.5	0.234	0.035	0.13	C
-7.5	0.333	0.040	0.13	C
-3.5	0.385	0.043	0.13	C
0.5	0.531	0.042	0.13	C
4.5	0.672	0.039	0.13	C
8.5	0.745	0.036	0.13	C
12.5	0.796	0.034	0.13	C
16.5	0.777	0.032	0.13	C
20.5	0.908	0.023	0.13	C
-19.5	0.051	0.018	0.25	C
-15.5	0.082	0.024	0.25	C

-11.5	0.150	0.030	0.25	C
-7.5	0.261	0.035	0.25	C
-3.5	0.364	0.039	0.25	C
0.5	0.383	0.041	0.25	C
4.5	0.623	0.040	0.25	C
8.5	0.739	0.035	0.25	C
12.5	0.762	0.035	0.25	C
16.5	0.800	0.033	0.25	C
20.5	0.924	0.021	0.25	C
-19.5	0.174	0.035	0.04	D
-15.5	0.231	0.038	0.04	D
-11.5	0.222	0.036	0.04	D
-7.5	0.284	0.040	0.04	D
-3.5	0.375	0.043	0.04	D
0.5	0.485	0.044	0.04	D
4.5	0.605	0.042	0.04	D
8.5	0.762	0.040	0.04	D
12.5	0.858	0.031	0.04	D
16.5	0.879	0.029	0.04	D
20.5	0.897	0.028	0.04	D
-19.5	0.064	0.022	0.07	D
-15.5	0.070	0.023	0.07	D
-11.5	0.138	0.030	0.07	D
-7.5	0.278	0.040	0.07	D
-3.5	0.360	0.044	0.07	D
0.5	0.504	0.045	0.07	D
4.5	0.639	0.043	0.07	D
8.5	0.776	0.036	0.07	D
12.5	0.832	0.033	0.07	D
16.5	0.944	0.021	0.07	D
20.5	0.959	0.018	0.07	D
-19.5	0.017	0.011	0.13	D
-15.5	0.065	0.022	0.13	D
-11.5	0.108	0.029	0.13	D
-7.5	0.252	0.039	0.13	D
-3.5	0.327	0.045	0.13	D
0.5	0.450	0.044	0.13	D
4.5	0.696	0.044	0.13	D
8.5	0.855	0.031	0.13	D
12.5	0.933	0.021	0.13	D
16.5	0.969	0.016	0.13	D

20.5	0.992	0.008	0.13	D
-19.5	0.015	0.010	0.25	D
-15.5	0.030	0.016	0.25	D
-11.5	0.067	0.023	0.25	D
-7.5	0.105	0.028	0.25	D
-3.5	0.271	0.038	0.25	D
0.5	0.440	0.046	0.25	D
4.5	0.818	0.034	0.25	D
8.5	0.868	0.031	0.25	D
12.5	0.940	0.023	0.25	D
16.5	1.000	0.000	0.25	D
20.5	1.000	0.000	0.25	D
-19.5	0.169	0.030	0.04	E
-15.5	0.136	0.027	0.04	E
-11.5	0.214	0.033	0.04	E
-7.5	0.290	0.039	0.04	E
-3.5	0.413	0.044	0.04	E
0.5	0.474	0.043	0.04	E
4.5	0.586	0.044	0.04	E
8.5	0.681	0.039	0.04	E
12.5	0.682	0.037	0.04	E
16.5	0.791	0.036	0.04	E
20.5	0.831	0.033	0.04	E
-19.5	0.101	0.026	0.07	E
-15.5	0.103	0.026	0.07	E
-11.5	0.129	0.030	0.07	E
-7.5	0.222	0.035	0.07	E
-3.5	0.307	0.040	0.07	E
0.5	0.469	0.042	0.07	E
4.5	0.634	0.038	0.07	E
8.5	0.755	0.034	0.07	E
12.5	0.748	0.038	0.07	E
16.5	0.865	0.028	0.07	E
20.5	0.910	0.023	0.07	E
-19.5	0.039	0.016	0.13	E
-15.5	0.036	0.016	0.13	E
-11.5	0.066	0.021	0.13	E
-7.5	0.131	0.029	0.13	E
-3.5	0.287	0.037	0.13	E
0.5	0.477	0.040	0.13	E
4.5	0.642	0.042	0.13	E

8.5	0.843	0.032	0.13	E
12.5	0.936	0.021	0.13	E
16.5	0.953	0.018	0.13	E
20.5	0.978	0.012	0.13	E
-19.5	0.000	0.000	0.25	E
-15.5	0.020	0.011	0.25	E
-11.5	0.035	0.015	0.25	E
-7.5	0.066	0.020	0.25	E
-3.5	0.217	0.033	0.25	E
0.5	0.407	0.042	0.25	E
4.5	0.752	0.035	0.25	E
8.5	0.888	0.027	0.25	E
12.5	0.946	0.019	0.25	E
16.5	0.974	0.013	0.25	E
20.5	1.000	0.000	0.25	E
-19.5	0.446	0.043	0.04	F
-15.5	0.486	0.044	0.04	F
-11.5	0.577	0.042	0.04	F
-7.5	0.532	0.047	0.04	F
-3.5	0.559	0.045	0.04	F
0.5	0.593	0.045	0.04	F
4.5	0.595	0.041	0.04	F
8.5	0.565	0.045	0.04	F
12.5	0.612	0.041	0.04	F
16.5	0.615	0.042	0.04	F
20.5	0.684	0.038	0.04	F
-19.5	0.378	0.041	0.07	F
-15.5	0.518	0.042	0.07	F
-11.5	0.397	0.040	0.07	F
-7.5	0.470	0.044	0.07	F
-3.5	0.500	0.043	0.07	F
0.5	0.528	0.045	0.07	F
4.5	0.597	0.044	0.07	F
8.5	0.664	0.039	0.07	F
12.5	0.707	0.038	0.07	F
16.5	0.621	0.041	0.07	F
20.5	0.678	0.043	0.07	F
-19.5	0.272	0.039	0.13	F
-15.5	0.276	0.038	0.13	F
-11.5	0.375	0.042	0.13	F
-7.5	0.489	0.041	0.13	F

-3.5	0.446	0.043	0.13	F
0.5	0.577	0.042	0.13	F
4.5	0.602	0.043	0.13	F
8.5	0.611	0.042	0.13	F
12.5	0.727	0.040	0.13	F
16.5	0.746	0.039	0.13	F
20.5	0.805	0.034	0.13	F
-19.5	0.178	0.033	0.25	F
-15.5	0.203	0.036	0.25	F
-11.5	0.276	0.041	0.25	F
-7.5	0.281	0.037	0.25	F
-3.5	0.457	0.041	0.25	F
0.5	0.543	0.039	0.25	F
4.5	0.626	0.041	0.25	F
8.5	0.738	0.039	0.25	F
12.5	0.779	0.038	0.25	F
16.5	0.857	0.030	0.25	F
20.5	0.852	0.031	0.25	F

`group_by` and `summarize`

- pick one coherence level
- group by direction
- summarise across all observers

````
```{r}
d |> filter(coherence == 0.25) |>
     group_by(direction) |>
     summarise(mean_p_cw = mean(p_cw)) |>
     gt()
```
````

result

```
d |> filter(coherence == 0.25) |>
     group_by(direction) |>
     summarise(mean_p_cw = mean(p_cw)) |>
     gt()
```

direction	mean_p_cw
-19.5	0.0498
-15.5	0.0714
-11.5	0.1150
-7.5	0.1572
-3.5	0.2898
0.5	0.4296
4.5	0.6824
8.5	0.8116
12.5	0.8662
16.5	0.9152
20.5	0.9496

plotting can follow same ideas

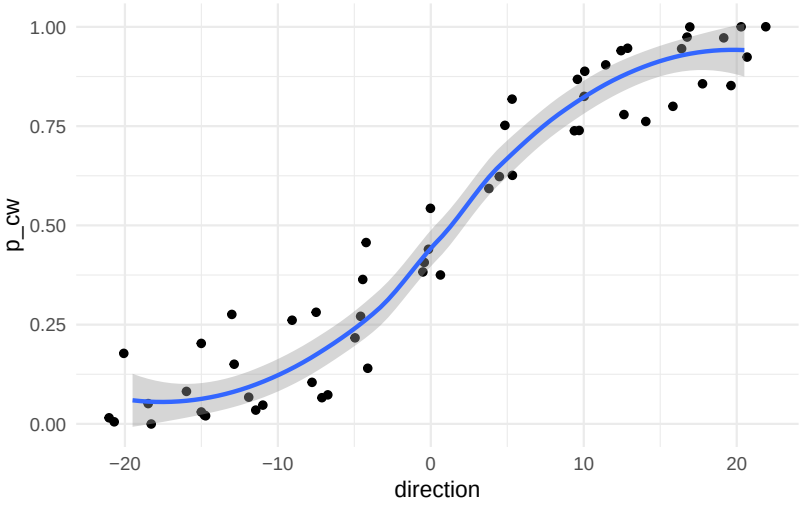
- *declarative* style (`ggplot`) versus
- *imperative* style (`matlab`, `matplotlib`, ...)[1]

[1] what I used to use before I hit on / read the `tidyverse` stuff.

plot example

5,6 lines of code to get this

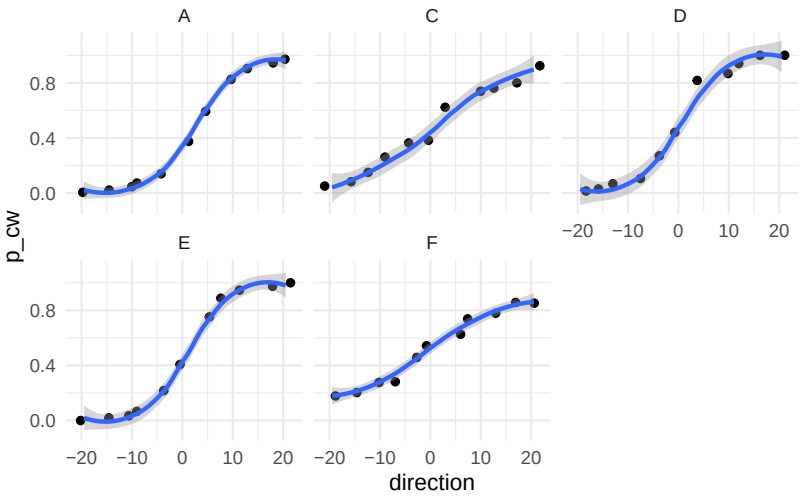
```
d |> filter(coherence == 0.25) |>
     ggplot(aes(x = direction, y = p_cw)) +
      geom_jitter() +
      geom_smooth() +
      theme_minimal()
```



one additional line…

````
```{r}
      ...
      facet_wrap(~subject) +
      ...
```
````

```
d |> filter(coherence == 0.25) |>
     ggplot(aes(x = direction, y = p_cw)) +
      geom_jitter() +
      geom_smooth() +
      facet_wrap(~subject) +
      theme_minimal()
```



Discussion

- data files (`csv`, `parquet`, `feather` ??)
- what do people do (hand-wrap their own? other libraries)
- how uses an actual **database**?
- should we teach this at UG/PG level more??

References