Data organization & tidy data

Eleutherodactylus Research Studio · Fall 2026

Tom Devitt

2026-08-31

This semester runs on tables

Every product this class ships is built on a **measurement table**:

- Team 1's calling-phenology analysis: a table of detections + weather
- Teams 2–4's call descriptions: tables of call parameters, one per species
- Your AmphibiaWeb account: a table of literature records behind every claim

The published paper shows the *summary* (mean ± SD, range, n). The science lives in the table underneath it — and whether that table is usable depends on decisions you make **before you type the first number**.

Today: how to make those decisions well.

The first question: one row per *what*?

Sound data are nested: a **recording** contains **calls**, made of **notes**, made of pulses.

Each level is a different *unit of observation*, and each answers different questions:

- One row per **recording** — when and where is *E. marnockii* calling? (Team 1)
- One row per **call** — how long are calls? how often are they repeated?
- One row per **note** — note duration, dominant frequency (Köhler et al. 2017)

Your team will need tables at more than one of these levels. Deciding “one row = one X” — and writing it down — is the single most important data decision you’ll make.

Tidy data: three rules

1. Every **column** is a variable
2. Every **row** is an observation
3. Every **cell** holds exactly one value

That's it. (Wickham 2014 — the paper that named the idea.)

Sounds obvious. Almost every spreadsheet ever made by a biologist violates at least one of them — including, historically, mine.

A field-notebook special

Real spreadsheets tend to look like this:

Species	**June 22**	**June 23**	**June 24**
E. marnockii (BFL)	12 calls (24 °C)	8 calls	—
E. pipilans	3 calls (26.5)		n/a
E. nitidus	see notes	15	15

Take 60 seconds with your neighbor: **how many distinct problems can you find?**

The sins, named

Problem	Where
Data stored in column *headers*	the dates
Two variables in one cell	“12 calls (24 °C)”
Two variables in one *column*	species + locality: “*E. marnockii* (BFL)”
Inconsistent units/format	“24 °C” vs “26.5”
Three different codes for missing	“—”, blank, “n/a”
A value that isn’t a value	“see notes”

None of these confuse a human reader. **All of them stop a computer cold** — and the computer is who reads this table 500 times this semester.

The same data, tidy

species	locality	date	temp_c	n_calls
E. marnockii	BFL	2026-06-22	24.0	12
E. marnockii	BFL	2026-06-23	NA	8
E. pipilans	Guerrero	2026-06-22	26.5	3
E. nitidus	Oaxaca	2026-06-23	NA	15

- One row = one **species × site × night** (and we wrote that down)
- Every variable has its own column; units live in the column *name* (`temp_c`)
- Missing is always `NA` — visibly missing, not silently blank
- Ugly to a human? Slightly. **Instantly usable by R? Completely.**

Wide vs. long

You'll often *measure* in wide format — it's natural at the keyboard:

```
calls_wide <- tribble(
  ~voucher,  ~species,      ~call_1_s, ~call_2_s, ~call_3_s,
  "TJD 839", "E. pipilans",      0.21,      0.19,      0.22,
  "TJD 848", "E. nitidus",       0.48,      0.51,      0.47
)
calls_wide
```

```
# A tibble: 2 × 5
  voucher species     call_1_s call_2_s call_3_s
  <chr>   <chr>          <dbl>    <dbl>    <dbl>
1 TJD 839 E. pipilans     0.21     0.19     0.22
2 TJD 848 E. nitidus      0.48     0.51     0.47
```

But R's plotting and stats tools want **one measurement per row** (long format).

`pivot_longer()`: ten seconds of code

```
calls_long <- calls_wide |>
  pivot_longer(starts_with("call_"),
               names_to = "call", values_to = "duration_s")
calls_long
```

```
# A tibble: 6 × 4
  voucher species     call     duration_s
  <chr>   <chr>       <chr>         <dbl>
1 TJD 839 E. pipilans call_1_s       0.21
2 TJD 839 E. pipilans call_2_s       0.19
3 TJD 839 E. pipilans call_3_s       0.22
4 TJD 848 E. nitidus  call_1_s       0.48
5 TJD 848 E. nitidus  call_2_s       0.51
6 TJD 848 E. nitidus  call_3_s       0.47
```

Enter data in whichever shape is least error-prone — and let **code**, never copy-paste, do the reshaping.

Rules for the spreadsheet itself

From Broman & Woo (2018), *Data Organization in Spreadsheets* — assigned skim, link on Resources:

- **Be consistent.** Same species spelling, same category names, same units, every row
- **One header row**, then data. No titles, no merged cells, no summary rows mixed in
- **No color as data.** A yellow cell means nothing after export — add a `notes` or `flag` column instead
- **Explicit `NA`** for missing, never blank, never 0, never "-"
- **No calculations in the data file.** Means and figures happen in R, from the raw table
- **Make it a rectangle.** Every row has the same columns

Dates will betray you

- The only safe format: `YYYY-MM-DD` (2026-08-31). It sorts correctly as text and every program parses it the same way
- "6/8/26" is June 8 or August 6 depending on who's reading
- Excel is so aggressive about converting things to dates that **geneticists renamed human genes** to escape it (*SEPT2* kept becoming September 2nd — true story, Ziemann et al. 2016)

Closer to home: our field catalog once recorded recording numbers as ranges like "1026-7." One value? Two? A date? A human knows. Software guessed — and we paid for it in cleanup.

Names: columns and files

Columns: lowercase, no spaces, units in the name — `snake_case`

```
dominant_freq_khz   note_duration_s   temp_c   n_notes
```

not `Dominant Frequency (kHz)` — spaces and symbols force quoting gymnastics in R forever after.

Files: no spaces, ISO dates, and names that still mean something in December:

```
2026-09-14_marnockii_call-measurements.csv     ✓
FINAL_data_v2_REALLY_final(1).xlsx             ✗
```

Raw data is sacred

- The file that came off the recorder / out of first data entry is **never edited**. Fixes, filtering, and reshaping happen in an R script that reads raw and writes derived
- Why: an edit to a raw file is invisible and unrepeatable. A script is both documentation *and* the undo button
- Every data file gets a **data dictionary** — a plain-text README naming each column, its units, its allowed values, and what one row means
- Your team's **data steward** owns this. But everyone follows it

Why CSV

- **Plain text**: readable by R, Python, Excel, Raven, a web browser, and you — in 20 years
- Plays perfectly with **GitHub**: changes show up line by line, like edits to code
- `.xlsx` is fine as a *data-entry* surface. It is not an archive format: export to CSV, and the CSV is the version of record

```
calls <- read_csv("~/data/marnockii_calls.csv")
glimpse(calls)
```

`glimpse()` is your first move on any new dataset: every column, its type, the first values. A number column arriving as `chr` means someone upstream broke a rule from today.

This isn’t hypothetical

The **recording inventory** — the table your team assignments are being built from *this week* — is exactly this kind of object: one row per recording, with voucher number, species, locality, date, format, and duration, assembled by a script.

And a cautionary tale from my own catalog: frog calls are **temperature-dependent**, so a call measurement without temperature is half a measurement. Of ~67 call-vouchered specimens, the catalog has ambient temperature for about **4** — the rest must be recovered from field notebooks, by hand, years later.

That’s what an “I’ll remember it” decision costs. If it isn’t in the table, it doesn’t exist.

This week

- **Team preferences due today** — if you haven't submitted, do it before midnight
- **AmphibiaWeb species claiming closes Wednesday** (link on Canvas). Claimed = yours
- **Wednesday:** ggplot2 — your first figures, in class, from a tidy table (bring your laptop)
- Team assignments announced **end of this week**

Readings (short, on the Resources page):

- Broman & Woo 2018, *Data Organization in Spreadsheets*
- Wickham 2014, *Tidy Data* (skim §1–3)