Week 5: Joins & Data Cleaning

September 23, 2026

WHY CLEANING IS MOST OF THE WORK

Real data rarely arrives ready to plot. Columns come in as the wrong type, names have spaces and symbols, categories are spelled five ways, dates are text, and the information you need is split across two files. This week is the checklist for turning raw data into something you can trust: **merge it, type it, name it, recode it, date it** — then check the result.

JOINS: MERGING TWO DATA FRAMES

Never glue columns side by side with `bind_cols()` and hope the rows line up — that's how a map ends up with Texas's data drawn on Tennessee. Match rows on a shared **key** column with a join.

Join	Keeps
`inner_join(x, y)`	Only rows with a match in **both**
`left_join(x, y)`	Every row of `x`; `NA` where `y` has no match
`right_join(x, y)`	Every row of `y`; `NA` where `x` has no match
`full_join(x, y)`	Every row of **both**, `NA`-filled

Always name the key with `by`. If the key column is named differently in each table, map one name to the other (or `rename()` first):

```
band_members %>%
  left_join(band_instruments, by = 'name')

band_members %>%
  left_join(band_instruments2,
            by = c("name" = "artist"))
```

Check a join: compare row counts before and after. A `left_join()` that *adds* rows means the key repeats in `y` — use `distinct()` on the lookup table first. New `NA`s mean keys that didn't match (e.g. `"N/A"`, typos, case).

RIGHT TYPE?

Always `glimpse()` after reading data. Numbers read as `<chr>` won't sort, sum, or plot on a continuous axis.

Function	`"$4.50"` becomes	Use when
`as.numeric()`	`NA` (with a warning)	Text is already a clean number
`parse_number()`	`4.5`	Number is wrapped in `$`, `%`, commas, `*`

`parse_number()` grabs the *first* number it finds — `"1-800-123-4567"` becomes `1`, so check what it dropped. In `read_excel()`, `col_types` sets types on read.

RIGHT NAME?

`janitor::clean_names()` turns `Total Investment ($ Millions)` into `total_investment_millions`: lowercase, underscores, no symbols. Change the style with `case = 'lower_camel'` or `'screaming_snake'`.

`select()` helper	Picks
`name:order`	A range of columns
`-(name:order)`	Everything *except* that range
`starts_with("sleep")`	Names with that prefix
`ends_with("wt")` / `contains("eep")`	Suffix / substring
`select_if(is.numeric)`	Columns of one type
`select(a, b, everything())`	Moves `a`, `b` to the front
`select(new = old)`	Renames **and drops** the rest
`rename(new = old)`	Renames, keeps everything

RECODING VARIABLES

`case_when()` checks conditions top to bottom; the first match wins. End with `TRUE ~` as a catch-all, or unmatched rows become `NA`. Prefer it over nested `ifelse()`, which gets unreadable past two levels.

```
wildlife_impacts %>%
  mutate(season = case_when(
    between(incident_month, 3, 5)  ~ 'spring',
    between(incident_month, 6, 8)  ~ 'summer',
    incident_month %in% c(9, 10, 11) ~ 'fall',
    TRUE ~ 'winter'
  ))
```

Collapse spelling variants before recoding: `str_to_lower()` first, so `"Climb"` and `"climb"` land in the same group.

Split and combine columns

```
tb_rates %>%
  separate(rate, into = c("cases", "population"),
           sep = "/", convert = TRUE)
```

Argument / function	Does
`sep = "/"`	Split at that character
`sep = 2`	Split after position 2
`convert = TRUE`	Re-guess the new columns' types
`unite(new, a, b)`	Paste columns back together

DATES WITH LUBRIDATE

To parse a string, pick the function whose letters match the **order** of the parts. The format is irrelevant — only the order matters.

Function	Parses
`ymd()`	`'2026-09-09'`, `'2026 Sep. 9'`
`mdy()`	`'September 9, 2026'`, `'Sep 9 2026'`
`dmy()`	`'9 September 2026'`

Extract	Returns
`year(date)`, `day(date)`	Numbers
`month(date, label = TRUE, abbr = FALSE)`	`September`
`wday(date, label = TRUE)`	`Wed`

Assigning works too: `year(date) <- 2016`. Watch impossible dates — setting day 30 on Feb. 28 rolls over into March. `today()` gives the current date.

MESSY EXCEL FILES

Repeated column groups

Men's and women's hot-dog winners sit side by side (`mens`, `dogs_eaten_3`, `country_4`, `womens`, …). Two routes to one tidy table:

1. **Divide & conquer**: `select()` each group into the same column names, tag it with `mutate(competition = 'Mens')`, then stack with `bind_rows()`.
2. **Long, separate, wide**: rename to `dogs_eaten.mens`, etc., `pivot_longer()` everything, `separate()` on `'\\.'`, then `pivot_wider()`.

Divide & conquer, end to end:

```
hot_dogs_m <- hot_dogs %>%
  select(year, competitor = mens,
         dogs_eaten = dogs_eaten_3,
         country = country_4) %>%
  mutate(competition = 'Mens')
# ...same for womens -> hot_dogs_w
bind_rows(hot_dogs_m, hot_dogs_w) %>%
  mutate(
    new_record = str_detect(dogs_eaten, "\\*"),
    dogs_eaten = parse_number(dogs_eaten)
  )
```

The asterisk marking a new record is information, not noise — save it in its own column *before* `parse_number()` strips it.

Long, separate, wide — after renaming to `competitor.mens`, `dogs_eaten.mens`, …, `country.womens`:

```
hot_dogs %>%
  pivot_longer(competitor.mens:country.womens,
               names_to = 'variable',
               values_to = 'value') %>%
  separate(variable, sep = '\\.',
           into = c('variable', 'competition')) %>%
  pivot_wider(names_from = variable,
              values_from = value)
```

Sub-headers and notes

The OICA car-sales sheet has title rows, blank columns, and region totals mixed in with countries:

```
read_excel(file.path('data', 'pc_sales_2018.xlsx'),
           sheet = 'pc_sales', skip = 5) %>%
  clean_names() %>%
  select(-c(x2:x4)) %>%       # blank columns
  filter(!country %in% drop,  # region totals
         !is.na(country)) %>%
  pivot_longer(cols = x2005:x2018,
               names_to = 'year',
               values_to = 'num_cars') %>%
  separate(year, into = c('drop', 'year'),
           sep = 'x', convert = TRUE)
```

Need the regions back? Don't parse them out of the rows — `left_join()` a small country-to-region lookup table. Build short tables like that by hand, or copy them from the web and paste as code with `datapasta::tribble_paste()`.

Cleaning isn't done when the code runs — it's done when you've checked the result. After every join, recode, or parse: count the rows, look for new `NA`s, and `distinct()` the recoded column. That check is your job, even when an agent wrote the code.

VERIFY BEFORE YOU TRUST IT

Cleaning code that runs without errors can still be wrong. Run these checks after each step — and ask an agent to show you them, not just tell you.

Check	How
Rows gained or lost?	`nrow()` before and after
New missing values?	`summary()` or `sum(is.na(x))`
Categories what you expect?	`count(df, phase_of_flt)`
Types right?	`glimpse()`
Totals still add up?	Compare a `sum()` to the source
Spot-check	Pick three raw rows; find them in the output

Directing an agent to clean

You decide the strategy; the agent types the code. Be specific about the judgment calls, and ask for evidence:

- *"Drop the region-total rows (EUROPE, NAFTA, …) — keep only countries. List every row you dropped."*
- *"Convert `dogs_eaten` to numeric, but first save which values had an asterisk in a `new_record` column."*
- *"Join on state abbreviation. Report row counts before and after, and any keys that didn't match."*

ERRORS YOU MAY HIT THIS WEEK

Message / symptom	Usual cause
"NAs introduced by coercion"	`as.numeric()` hit a $, comma, or note — try `parse_number()`
`Joining with by = join_by(...)`	Not an error; join guessed the key. Set `by` yourself
Row count jumps after a join	Duplicate keys in the lookup table — `distinct()` it
New category of `NA` after `case_when()`	No `TRUE ~` catch-all
"Expected 2 pieces" from `separate()`	Some values have more or fewer separators

BEFORE NEXT CLASS

- **HW4: Exploring Data**, due **Sept 28**.
- **MP1: Clean & Verify**, due **Oct 4** (9% of grade).
- **Quiz 2** at the start of next class (Sept 30), covering Weeks 3–5.

You should be able to:

- Pick the right join, set `by`, and check the row count afterward.
- Spot numbers stored as text and fix them with `parse_number()`.
- Clean column names with `clean_names()` and reorganize them with `select()`.
- Recode categories with `case_when()` and split columns with `separate()`.
- Parse dates in any order and pull out year, month, or weekday.
- Turn a messy Excel sheet into one tidy table.