Tidying Data Lab

Anthony Tetreault

2025-03-20

1. The built in billboard dataset is not tidy. Describe why it is not tidy and then tidy the dataset.

```
# This dataset is not tidy because the columns are values of a variable, weeks, not each variable has i
billboard %>%
    pivot_longer(
        cols = starts_with("wk"),
        names_to = "week",
        names_prefix = "wk",
        names_transform = list(week = as.integer),
        values_drop_na = TRUE
    ) %>%
        mutate(date = date.entered + weeks(week)) %>%
            arrange(artist, track, week)
```

```
## # A tibble: 5,307 x 6
##    artist  track                   date.entered  week value date
##    <chr>   <chr>                   <date>       <int> <dbl> <date>
##  1 2 Pac   Baby Don't Cry (Keep... 2000-02-26       1    87 2000-03-04
##  2 2 Pac   Baby Don't Cry (Keep... 2000-02-26       2    82 2000-03-11
##  3 2 Pac   Baby Don't Cry (Keep... 2000-02-26       3    72 2000-03-18
##  4 2 Pac   Baby Don't Cry (Keep... 2000-02-26       4    77 2000-03-25
##  5 2 Pac   Baby Don't Cry (Keep... 2000-02-26       5    87 2000-04-01
##  6 2 Pac   Baby Don't Cry (Keep... 2000-02-26       6    94 2000-04-08
##  7 2 Pac   Baby Don't Cry (Keep... 2000-02-26       7    99 2000-04-15
##  8 2Ge+her The Hardest Part Of ... 2000-09-02       1    91 2000-09-09
##  9 2Ge+her The Hardest Part Of ... 2000-09-02       2    87 2000-09-16
## 10 2Ge+her The Hardest Part Of ... 2000-09-02       3    92 2000-09-23
## # i 5,297 more rows
```

2. Tidy the "fish_encounters" dataset of fish spotting by monitoring stations. Make the NA into 0 using the option "values_fill = list(seen = 0)"

```
fish_encounters %>%
    pivot_wider(
        names_from = "station",
        values_from = "seen",
        values_fill = list(seen = 0)
    )
```

```
## # A tibble: 19 x 12
##    fish  Release I80_1 Lisbon  Rstr Base_TD   BCE   BCW  BCE2  BCW2   MAE   MAW
```

```
##    <fct>   <int> <int>  <int> <int>   <int> <int> <int> <int> <int> <int> <int>
##  1 4842        1     1      1     1       1     1     1     1     1     1     1
##  2 4843        1     1      1     1       1     1     1     1     1     1     1
##  3 4844        1     1      1     1       1     1     1     1     1     1     1
##  4 4845        1     1      1     1       1     0     0     0     0     0     0
##  5 4847        1     1      1     0       0     0     0     0     0     0     0
##  6 4848        1     1      1     1       0     0     0     0     0     0     0
##  7 4849        1     1      0     0       0     0     0     0     0     0     0
##  8 4850        1     1      0     1       1     1     1     0     0     0     0
##  9 4851        1     1      0     0       0     0     0     0     0     0     0
## 10 4854        1     1      0     0       0     0     0     0     0     0     0
## 11 4855        1     1      1     1       1     0     0     0     0     0     0
## 12 4857        1     1      1     1       1     1     1     1     1     0     0
## 13 4858        1     1      1     1       1     1     1     1     1     1     1
## 14 4859        1     1      1     1       1     0     0     0     0     0     0
## 15 4861        1     1      1     1       1     1     1     1     1     1     1
## 16 4862        1     1      1     1       1     1     1     1     1     0     0
## 17 4863        1     1      0     0       0     0     0     0     0     0     0
## 18 4864        1     1      0     0       0     0     0     0     0     0     0
## 19 4865        1     1      1     0       0     0     0     0     0     0     0
```

3. Import the flowers1 dataset. Tidy and pivot the data. Hint: use “read_csv2()” to read in the dataset

```
flowers1 <- read_csv2("./flowers1.csv")
```

```
## i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
```

```
## Rows: 48 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (1): Variable
## dbl (3): Time, replication, Value
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
flowers1 %>%
    pivot_wider(
        names_from = "Variable",
        values_from = "Value"
    )
```

```
## # A tibble: 24 x 4
##     Time replication Flowers Intensity
##    <dbl>       <dbl>   <dbl>     <dbl>
##  1     1           1    62.3       150
##  2     1           2    77.4       150
##  3     1           3    55.3       300
##  4     1           4    54.2       300
##  5     1           5    49.6       450
##  6     1           6    61.9       450
##  7     1           7    39.4       600
```

```
##  8     1           8    45.7       600
##  9     1           9    31.3       750
## 10     1          10    44.9       750
## # i 14 more rows
```

4. Import the flowers2 dataset. Tidy the dataset by turning the one column into 3 separate columns

```
flowers2 <- read_csv2("./flowers2.csv")
```

```
## i Using "','" as decimal and "'.'" as grouping mark. Use ‘read_delim()‘ for more control.

## Rows: 24 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (1): Flowers/Intensity
## dbl (1): Time
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
flowers2 %>%
    separate_wider_delim(
        cols = "Flowers/Intensity",
        delim = "/",
        names = c("Flowers", "Intensity")
    ) %>% select(Time, Flowers, Intensity)
```

```
## # A tibble: 24 x 3
##     Time Flowers Intensity
##    <dbl> <chr>   <chr>
##  1     1 62.3    150
##  2     1 77.4    150
##  3     1 55.3    300
##  4     1 54.2    300
##  5     1 49.6    450
##  6     1 61.9    450
##  7     1 39.4    600
##  8     1 45.7    600
##  9     1 31.3    750
## 10     1 44.9    750
## # i 14 more rows
```

5. In the following dataset, turn the implicit missing values to explicit

```
output <- tibble(
      treatment   = c("a", "b", "a", "c", "b"),
      gender   = factor(c("M", "F", "F", "M", "M"), levels = c("M", "F", "O")),
      return = c(1.5, 0.75,  0.5,  1.8,  NA)
    )
output %>%
    complete(treatment, gender)
```

```
## # A tibble: 9 x 3
##   treatment gender return
##   <chr>     <fct>   <dbl>
## 1 a         M        1.5
## 2 a         F        0.5
## 3 a         O       NA
## 4 b         M       NA
## 5 b         F        0.75
## 6 b         O       NA
## 7 c         M        1.8
## 8 c         F       NA
## 9 c         O       NA
```

6. Import the weather dataset as weather. Use "pivot_longer()" to put the days all in one column, then use "pivot_wider" to separate tmax and tmin into separate columns. Print the summary of the final resulting dataset

```
weather <- read_csv("./weather.csv")
```

```
## Rows: 22 Columns: 35
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): id, element
## dbl (25): year, month, d1, d2, d3, d4, d5, d6, d7, d8, d10, d11, d13, d14, d...
## lgl  (8): d9, d12, d18, d19, d20, d21, d22, d24
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
weather %>%
    pivot_longer(
        cols = starts_with("d"),
        names_to = "day",
        names_prefix = "d",
        values_drop_na = TRUE
    ) %>%
        pivot_wider(
            names_from = "element",
            values_from = "value"
        ) %>%
            summary()
```

```
##       id                 year          month            day
##  Length:33          Min.   :2010   Min.   : 1.000   Length:33
##  Class :character   1st Qu.:2010   1st Qu.: 4.000   Class :character
##  Mode  :character   Median :2010   Median : 8.000   Mode  :character
##                     Mean   :2010   Mean   : 7.212
##                     3rd Qu.:2010   3rd Qu.:10.000
##                     Max.   :2010   Max.   :12.000
##       tmax            tmin
##  Min.   :24.10   Min.   : 7.90
##  1st Qu.:27.80   1st Qu.:13.40
```

```
##  Median :29.00   Median :15.00
##  Mean   :29.19   Mean   :14.65
##  3rd Qu.:29.90   3rd Qu.:16.50
##  Max.   :36.30   Max.   :18.20
```

7. Load the built in "anscombe" data frame and use "pivot_longer()" to separate all the x and y columns and categorize them into 4 sets.

```
anscombe %>%
    pivot_longer(
        cols = starts_with(c("x","y")),
        names_to = c(".value", "set"),
        names_pattern = "(.)(.)"
    )
```

```
## # A tibble: 44 x 3
##    set       x     y
##    <chr> <dbl> <dbl>
##  1 1        10  8.04
##  2 2        10  9.14
##  3 3        10  7.46
##  4 4         8  6.58
##  5 1         8  6.95
##  6 2         8  8.14
##  7 3         8  6.77
##  8 4         8  5.76
##  9 1        13  7.58
## 10 2        13  8.74
## # i 34 more rows
```