Week-2-Tidying-Data-Peer-Graded-Assignment

Vickie Gray

2023-01-02

Tidy Data

Q1. The built in billboard dataset is not tidy. Describe why it is not tidy and then tidy the dataset.

While the exercise asked for a different output, to me, the most important data is the ranking of tracks THIS WEEK. Rank per week should be a single column, with date descending as a second column, and perhaps a number of weeks on the chart variable calculated from the rest of the data. This way, as tracks drop off the chart, they do not take up space with NAs.

What would the first few entries look like if it were tidy?

A:

Table 1: Billboard

Artist	Track	Week	Rank	Weeks_Charted
Vickie Gray	Data Darling	2022-12-28	1	43
Jane Wall	Pivot Party	2022-12-28	2	19
Arthur Dent	Don't Panic	2022-12-28	3	34
…				
Jane Wall	Pivot Party	2022-12-21	1	18
Arthur Dent	Don't Panic	2022-12-21	2	33
…				

Billboard Tidied:

```r
billboard # The original dataset
```

```
## # A tibble: 317 x 79
##    artist track date.ent~1   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8   wk9
##    <chr>  <chr> <date>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 2 Pac  Baby~ 2000-02-26    87    82    72    77    87    94    99    NA    NA
##  2 2Ge+h~ The ~ 2000-09-02    91    87    92    NA    NA    NA    NA    NA    NA
##  3 3 Doo~ Kryp~ 2000-04-08    81    70    68    67    66    57    54    53    51
##  4 3 Doo~ Loser 2000-10-21    76    76    72    69    67    65    55    59    62
##  5 504 B~ Wobb~ 2000-04-15    57    34    25    17    17    31    36    49    53
##  6 98^0   Give~ 2000-08-19    51    39    34    26    26    19     2     2     3
##  7 A*Tee~ Danc~ 2000-07-08    97    97    96    95   100    NA    NA    NA    NA
##  8 Aaliy~ I Do~ 2000-01-29    84    62    51    41    38    35    35    38    38
##  9 Aaliy~ Try ~ 2000-03-18    59    53    38    28    21    18    16    14    12
## 10 Adams~ Open~ 2000-08-26    76    76    74    69    68    67    61    58    57
## # ... with 307 more rows, 67 more variables: wk10 <dbl>, wk11 <dbl>,
## #   wk12 <dbl>, wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>,
## #   wk18 <dbl>, wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>,
## #   wk24 <dbl>, wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>,
## #   wk30 <dbl>, wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>,
## #   wk36 <dbl>, wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>,
## #   wk42 <dbl>, wk43 <dbl>, wk44 <dbl>, wk45 <dbl>, wk46 <dbl>, wk47 <dbl>, ...
```

```r
tidy_billboard <- billboard %>%
  pivot_longer(
    cols = starts_with("wk"), # gather the data currently in multiple wk* columns
    names_to = "week", # into one column called week
    names_prefix = "wk", # remove the prefix from the names to leave the week number
    values_to = "rank", # create a rank column and put the values previously in wk* columns there.
    values_drop_na = TRUE # drop rows with NA to tidy the dataset.
    # This is safe, as a rank of NA indicates the song was not on the chart at all.
  )

tidy_billboard #The tidied dataset.
```

```
## # A tibble: 5,307 x 5
##    artist  track                   date.entered week   rank
##    <chr>   <chr>                   <date>       <chr> <dbl>
##  1 2 Pac   Baby Don't Cry (Keep... 2000-02-26   1        87
##  2 2 Pac   Baby Don't Cry (Keep... 2000-02-26   2        82
##  3 2 Pac   Baby Don't Cry (Keep... 2000-02-26   3        72
##  4 2 Pac   Baby Don't Cry (Keep... 2000-02-26   4        77
##  5 2 Pac   Baby Don't Cry (Keep... 2000-02-26   5        87
##  6 2 Pac   Baby Don't Cry (Keep... 2000-02-26   6        94
##  7 2 Pac   Baby Don't Cry (Keep... 2000-02-26   7        99
##  8 2Ge+her The Hardest Part Of ... 2000-09-02   1        91
##  9 2Ge+her The Hardest Part Of ... 2000-09-02   2        87
## 10 2Ge+her The Hardest Part Of ... 2000-09-02   3        92
## # ... with 5,297 more rows
```

```r
# Now we can add the calculated week column:

tidy_billboard2 <- tidy_billboard %>%
  mutate(
    date = (date.entered) + 7 * (as.numeric(week) - 1)
  ) # Mutate the table to add the calculated week
tidy_billboard2
```

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

Q2. Tidy the "fish_encounters" dataset of fish spotting by monitoring stations. Make the NA into 0 using the option "values_fill = list(seen = 0)"

```
# The fish encounters dataset has multiple observations per variable (station)
# We can tidy the data by pivoting wider, showing each station as a variable
tidy_fish <- fish_encounters %>%
  pivot_wider(
    names_from = "station",  # the new columns will be stations recording the fish encounter
    values_from = "seen", # each station will have an entry for the "seen" status of each fish
    values_fill = list(seen = 0) # missing data in this case means the fish was not seen so we can fill
  )

tidy_fish
```

```
## # 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
```

Q3. Import the flowers1 dataset. Tidy and pivot the data. Hint: use "read_csv2()" to read in the dataset

I can't find the data dictionary for this dataset, but with a little hunting, found that research into apple and other fruit trees is concerned with the effect of the intensity of light on flowering time and plant quality.

I will assume, then, that the variables in this dataset - Time, Replication, Variable, and Value - are headings for studies into the flowering time (1, 2) of flowers (variable), and the associated light intensity (variable), in 12 replications of each observation.

I have therefore turned each observation into a record of the flowers and intensity within each replication of the observation for each flowering time.

I have read in the data using read_csv2, as this dataset uses semicolons as separators.

```
flowers1 <- read_csv2("https://raw.githubusercontent.com/JaneWall/data_STAT412612/master/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
```

```
## # A tibble: 48 x 4
##     Time replication Variable Value
##    <dbl>       <dbl> <chr>    <dbl>
##  1     1           1 Flowers   62.3
##  2     1           2 Flowers   77.4
##  3     1           3 Flowers   55.3
##  4     1           4 Flowers   54.2
##  5     1           5 Flowers   49.6
##  6     1           6 Flowers   61.9
##  7     1           7 Flowers   39.4
##  8     1           8 Flowers   45.7
##  9     1           9 Flowers   31.3
## 10     1          10 Flowers   44.9
## # ... with 38 more rows
```

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

tidy_flowers
```

```
## # 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
## # ... with 14 more rows
```

Q4. Import the flowers2 dataset. Tidy the dataset by turning the one column into 3 separate columns.

I have read in the data using read_csv2, as this dataset uses semicolons as separators.

```
flowers2 <- read_csv2("https://raw.githubusercontent.com/JaneWall/data_STAT412612/master/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 # The original dataset
```

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

```
tidy_flowers_2 <- flowers2 %>% #a new object gets the resulting dataset
  separate(
    "Flowers/Intensity", # separate the original variable name
    into = c("flowers", "intensity"), # into two new variables
    sep = "/", # at the separator character
    convert = TRUE) # and convert the resulting data types

tidy_flowers_2 # The separated dataset
```

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

Q5. In the following dataset, turn the implicit missing values to explicit

```r
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
```

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

```r
tidy_output <- output %>%
  group_by(treatment) %>% #Within each treatment, show all possible combinations
  complete(
    gender,
    explicit = TRUE)

tidy_output
```

```
## # A tibble: 9 x 3
## # Groups:   treatment [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
```

Q6.

a. Import the weather dataset as weather.

b. Use “pivot_longer()” to put the days all in one column, then
c. use “pivot_wider” to separate tmax and tmin into separate columns.
d. Print the summary of the final resulting dataset

```
weather <- read_csv("https://raw.githubusercontent.com/JaneWall/data_STAT412612/master/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
```

```
## # A tibble: 22 x 35
##    id        year month element    d1    d2    d3    d4    d5    d6    d7    d8
##    <chr>    <dbl> <dbl> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 MX17004  2010     1 tmax       NA  NA    NA       NA  NA       NA    NA    NA
##  2 MX17004  2010     1 tmin       NA  NA    NA       NA  NA       NA    NA    NA
##  3 MX17004  2010     2 tmax       NA  27.3  24.1     NA  NA       NA    NA    NA
##  4 MX17004  2010     2 tmin       NA  14.4  14.4     NA  NA       NA    NA    NA
##  5 MX17004  2010     3 tmax       NA  NA    NA       NA  32.1     NA    NA    NA
##  6 MX17004  2010     3 tmin       NA  NA    NA       NA  14.2     NA    NA    NA
##  7 MX17004  2010     4 tmax       NA  NA    NA       NA  NA       NA    NA    NA
##  8 MX17004  2010     4 tmin       NA  NA    NA       NA  NA       NA    NA    NA
##  9 MX17004  2010     5 tmax       NA  NA    NA       NA  NA       NA    NA    NA
## 10 MX17004  2010     5 tmin       NA  NA    NA       NA  NA       NA    NA    NA
## # ... with 12 more rows, and 23 more variables: d9 <lgl>, d10 <dbl>, d11 <dbl>,
## #   d12 <lgl>, d13 <dbl>, d14 <dbl>, d15 <dbl>, d16 <dbl>, d17 <dbl>,
## #   d18 <lgl>, d19 <lgl>, d20 <lgl>, d21 <lgl>, d22 <lgl>, d23 <dbl>,
## #   d24 <lgl>, d25 <dbl>, d26 <dbl>, d27 <dbl>, d28 <dbl>, d29 <dbl>,
## #   d30 <dbl>, d31 <dbl>
```

```
tidy_weather <- weather %>%
  pivot_longer( # b.
    cols = d1:d31,
    names_to = c("day"),
```

```r
    names_pattern = "d(.*)",
    values_to = "temp",
    values_drop_na = TRUE # there are a lot of cells with no data, so this made the dataset much smalle
    )

tidy_weather
```

```
## # A tibble: 66 x 6
##    id       year month element day    temp
##    <chr>   <dbl> <dbl> <chr>   <chr> <dbl>
##  1 MX17004  2010     1 tmax    30     27.8
##  2 MX17004  2010     1 tmin    30     14.5
##  3 MX17004  2010     2 tmax    2      27.3
##  4 MX17004  2010     2 tmax    3      24.1
##  5 MX17004  2010     2 tmax    11     29.7
##  6 MX17004  2010     2 tmax    23     29.9
##  7 MX17004  2010     2 tmin    2      14.4
##  8 MX17004  2010     2 tmin    3      14.4
##  9 MX17004  2010     2 tmin    11     13.4
## 10 MX17004  2010     2 tmin    23     10.7
## # ... with 56 more rows
```

```r
tidy_weather2 <- tidy_weather %>%
  pivot_wider( # c.
    names_from = "element", # We're splitting the max and min temperatures in element into two columns
    values_from = "temp"
  )

tidy_weather2
```

```
## # A tibble: 33 x 6
##    id       year month day    tmax  tmin
##    <chr>   <dbl> <dbl> <chr> <dbl> <dbl>
##  1 MX17004  2010     1 30     27.8  14.5
##  2 MX17004  2010     2 2      27.3  14.4
##  3 MX17004  2010     2 3      24.1  14.4
##  4 MX17004  2010     2 11     29.7  13.4
##  5 MX17004  2010     2 23     29.9  10.7
##  6 MX17004  2010     3 5      32.1  14.2
##  7 MX17004  2010     3 10     34.5  16.8
##  8 MX17004  2010     3 16     31.1  17.6
##  9 MX17004  2010     4 27     36.3  16.7
## 10 MX17004  2010     5 27     33.2  18.2
## # ... with 23 more rows
```

```r
# A summary of temperature data would make sense if showed average maximum and minimum temperatures by

tidy_weather2_sum <- tidy_weather2 %>%
  group_by(month) %>%
  summarize(m_max = mean(tmax), m_min = mean(tmin))

tidy_weather2_sum
```

```
## # A tibble: 11 x 3
##    month m_max m_min
##    <dbl> <dbl> <dbl>
##  1     1  27.8  14.5
##  2     2  27.8  13.2
##  3     3  32.6  16.2
##  4     4  36.3  16.7
##  5     5  33.2  18.2
##  6     6  29.0  17.8
##  7     7  29.2  17
##  8     8  28.3  15.8
##  9    10  28.9  13.1
## 10    11  28.1  12.5
## 11    12  28.8  12.2
```

Q7. 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

```
as_tibble(anscombe) # The original data set
```

```
## # A tibble: 11 x 8
##       x1    x2    x3    x4    y1    y2    y3    y4
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1    10    10    10     8  8.04  9.14  7.46  6.58
##  2     8     8     8     8  6.95  8.14  6.77  5.76
##  3    13    13    13     8  7.58  8.74 12.7   7.71
##  4     9     9     9     8  8.81  8.77  7.11  8.84
##  5    11    11    11     8  8.33  9.26  7.81  8.47
##  6    14    14    14     8  9.96  8.1   8.84  7.04
##  7     6     6     6     8  7.24  6.13  6.08  5.25
##  8     4     4     4    19  4.26  3.1   5.39 12.5
##  9    12    12    12     8 10.8   9.13  8.15  5.56
## 10     7     7     7     8  4.82  7.26  6.42  7.91
## 11     5     5     5     8  5.68  4.74  5.73  6.89
```

```
tidy_anscombe <- anscombe %>%
  pivot_longer(
    everything(), #All columns
    names_to = c(".value", "set"), # `".value"` indicates that the corresponding component of the colum
# See https://rdrr.io/cran/tidyr/src/R/pivot-long.R for more explanation
    names_pattern = "(.)(.)" # divide the column names into two parts with no divider
  ) %>%
  arrange(set) # Arrange the result by set

tidy_anscombe
```

```
## # A tibble: 44 x 3
##    set       x     y
##    <chr> <dbl> <dbl>
##  1 1        10  8.04
##  2 1         8  6.95
##  3 1        13  7.58
##  4 1         9  8.81
##  5 1        11  8.33
##  6 1        14  9.96
##  7 1         6  7.24
##  8 1         4  4.26
##  9 1        12 10.8
## 10 1         7  4.82
## # ... with 34 more rows
```

“‘