Zeyd Khalil HW8, October 15, 2020

```
library(tidyverse)
```

```
## -- Attaching packages --------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Exercise 1 - The following built-in datasets are not tidy. For each one, describe why it is not tidy and write out what the first five entries would look like once it is in a tidy format.

a. relig_income
b. billboard
c. us_rent_income

```
head(relig_income, n = 5)
```

```
## # A tibble: 5 x 11
##   religion `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-50k` `$50-75k` `$75-100k`
##   <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>      <dbl>
## 1 Agnostic      27        34        60        81        76       137        122
## 2 Atheist       12        27        37        52        35        70         73
## 3 Buddhist      27        21        30        34        33        58         62
## 4 Catholic     418       617       732       670       638      1116        949
## 5 Don't k~      15        14        15        11        10        35         21
## # ... with 3 more variables: `$100-150k` <dbl>, `>150k` <dbl>, `Don't
## #   know/refused` <dbl>
```

The reason why this data is not tidy is because the column headings are values and not variable names. In order to tidy this data, I would use count() in order to distribute the income intervals in each row, and count the number of cases in those intervals. The result would consist of 3 columns; religion, income interval, and count. That would significantly increase the number of rows, but decrease the number of columns.

```
head(billboard, n = 5)
```

```
## # A tibble: 5 x 79
##   artist track date.entered   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8
##   <chr>  <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2 Pac  Baby~ 2000-02-26      87    82    72    77    87    94    99    NA
## 2 2Ge+h~ The ~ 2000-09-02      91    87    92    NA    NA    NA    NA    NA
## 3 3 Doo~ Kryp~ 2000-04-08      81    70    68    67    66    57    54    53
## 4 3 Doo~ Loser 2000-10-21      76    76    72    69    67    65    55    59
## 5 504 B~ Wobb~ 2000-04-15      57    34    25    17    17    31    36    49
## # ... with 68 more variables: wk9 <dbl>, 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>, wk48 <dbl>,
## #   wk49 <dbl>, wk50 <dbl>, wk51 <dbl>, wk52 <dbl>, wk53 <dbl>, wk54 <dbl>,
## #   wk55 <dbl>, wk56 <dbl>, wk57 <dbl>, wk58 <dbl>, wk59 <dbl>, wk60 <dbl>,
## #   wk61 <dbl>, wk62 <dbl>, wk63 <dbl>, wk64 <dbl>, wk65 <dbl>, wk66 <lgl>,
## #   wk67 <lgl>, wk68 <lgl>, wk69 <lgl>, wk70 <lgl>, wk71 <lgl>, wk72 <lgl>,
## #   wk73 <lgl>, wk74 <lgl>, wk75 <lgl>, wk76 <lgl>
```

The reason why this data is not tidy is the same as in the previous example, because most of the columns in this dataset are numerical values instead of variable names. In order to tidy this data, I would use count() in order to distribute the week number in each row, and count the number of cases in those intervals. The result would consist of 4 columns; artist, track, date entered, week number, and count. That would significantly increase the number of rows, but decrease the number of columns.

```
head(us_rent_income, n = 5)
```

```
## # A tibble: 5 x 5
##   GEOID NAME    variable estimate   moe
##   <chr> <chr>   <chr>       <dbl> <dbl>
## 1 01    Alabama income      24476   136
## 2 01    Alabama rent          747     3
## 3 02    Alaska  income      32940   508
## 4 02    Alaska  rent         1200    13
## 5 04    Arizona income      27517   148
```

The reason why this data is not tidy is because each observational unit is spread across multiple rows. In order to make this data tidy, what I would do is use mutate() in order to create a new column for rent. Once that's done, the first five entries of the data will consist of 5 columns and only one row per state; the columns will be GEOID, NAME, Income, Rent, and moe.

Exercise 2 - Try on your own to do the same thing to table4b.

```
tidy4b <- table4b %>% pivot_longer(cols = c(`1999`, `2000`), names_to = "year", values_to = "cases")

tidy4b
```

```
## # A tibble: 6 x 3
##   country     year       cases
##   <chr>       <chr>      <int>
## 1 Afghanistan 1999    19987071
## 2 Afghanistan 2000    20595360
## 3 Brazil      1999   172006362
## 4 Brazil      2000   174504898
## 5 China       1999  1272915272
## 6 China       2000  1280428583
```

Exercise 3 - Tidy built-in dataset relig_income

```
tidy_relig_income <- relig_income %>% pivot_longer(cols = c(`<$10k`, `$10-20k`, `$20-30k`, `$30-40k`, `$
tidy_relig_income
```

```
## # A tibble: 180 x 3
##    religion `religion income`  count
##    <chr>    <chr>              <dbl>
##  1 Agnostic <$10k                 27
##  2 Agnostic $10-20k               34
##  3 Agnostic $20-30k               60
##  4 Agnostic $30-40k               81
##  5 Agnostic $40-50k               76
##  6 Agnostic $50-75k              137
##  7 Agnostic $75-100k             122
##  8 Agnostic $100-150k            109
##  9 Agnostic >150k                 84
## 10 Agnostic Don't know/refused    96
## # ... with 170 more rows
```

Exercise 4 -

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

```
## Parsed with column specification:
## cols(
##   Monkey = col_character(),
##   Treatment = col_character(),
##   Week2 = col_double(),
##   Week4 = col_double(),
##   Week8 = col_double(),
```

```
##   Week12 = col_double(),
##   Week16 = col_double()
## )
```

```
tidy_monkeymem <- monkeymem %>% pivot_longer(cols = c(Week2, Week4, Week8, Week12, Week16), names_to =

tidy_monkeymem
```

```
## # A tibble: 90 x 4
##    Monkey Treatment Week   percent
##    <chr>  <chr>     <chr>    <dbl>
##  1 Spank  Control   Week2       95
##  2 Spank  Control   Week4       75
##  3 Spank  Control   Week8       80
##  4 Spank  Control   Week12      65
##  5 Spank  Control   Week16      70
##  6 Chim   Control   Week2       85
##  7 Chim   Control   Week4       75
##  8 Chim   Control   Week8       55
##  9 Chim   Control   Week12      75
## 10 Chim   Control   Week16      85
## # ... with 80 more rows
```

Exercise 5 - Tidy the fish_encounters dataset of fish spotting by monitoring stations. Make the NA into 0 using the option values_fill = list(seen = 0)

```
tidy_fishencounters <- fish_encounters %>% pivot_wider(names_from = station, values_from = seen, values

tidy_fishencounters
```

```
## # 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
```

Exercise 6 - Spread the flowers1 data frame. Hint: use read_csv2() to read in the dataset.

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

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

```
## Parsed with column specification:
## cols(
##   Time = col_double(),
##   replication = col_double(),
##   Variable = col_character(),
##   Value = col_double()
## )
```

```
spread_flowers1 <- flowers1 %>% pivot_wider(names_from = Variable, values_from = Value)

spread_flowers1
```

```
## # 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
```

Exercise 7 - Tidy the dataset flowers2.csv by turning the one column into 3 separate columns.

```
flowers2 <- read_csv("https://raw.githubusercontent.com/JaneWall/data_STAT412612/master/flowers2.csv")
```

```
## Parsed with column specification:
## cols(
##   `Flowers/Intensity;Time` = col_character()
## )
```

```
sep_flowers2 <- flowers2 %>% separate(`Flowers/Intensity;Time`, into = c("Flowers", "Intensity", "Time"
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 24 rows [1, 2, 3,
## 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
sep_flowers2
```

```
## # A tibble: 24 x 3
##    Flowers Intensity Time
##    <chr>   <chr>     <chr>
##  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
```

Exercise 8 - Re-unite the data frame you separated from the flowers2 exercise. Use a comma for the separator.

```
unite_flowers2 <- sep_flowers2 %>% unite(`Flowers,Intensity`, Flowers, Intensity, sep = ",")

unite_flowers2
```

```
## # A tibble: 24 x 2
##    `Flowers,Intensity` Time
##    <chr>               <chr>
##  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
```

Exercise 9 - 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
```

```
## # 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
```

```
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
```

Exercise 10 - Use pivot_longer() to put the days all in one column. Then, rearrange the data.

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

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   id = col_character(),
##   element = col_character(),
##   d9 = col_logical(),
##   d12 = col_logical(),
##   d18 = col_logical(),
```

```
##   d19 = col_logical(),
##   d20 = col_logical(),
##   d21 = col_logical(),
##   d22 = col_logical(),
##   d24 = col_logical()
## )

## See spec(...) for full column specifications.
```

```
weather_tidy <- weather %>% pivot_longer(cols = starts_with("d"), names_to = "day", names_pattern = "d(

weather_tidy
```

```
## # A tibble: 66 x 6
##    id       year month element day   value
##    <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
```

```
weather4 <- weather_tidy %>% pivot_wider(names_from = element, values_from = value)

summary(weather4)
```

```
##       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
```

Exercise 11 - Tidy the billboard dataset (built-in)

- First gather up all the week entries into a row for each week for each song (where there is an entry)

- Then, convert the week variable to a number and figure out the date corresponding to each week on the chart
- Sort the data by artist, track and week.

```
billboard %>% pivot_longer(cols = starts_with("wk"), names_to = "week", names_pattern = "wk(.*)", value
```

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

Exercise 12 - Do the same with the built-in dataset anscombe.

```
anscombe %>%
   pivot_longer(everything(),
                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
## # ... with 34 more rows
```

Exercise 13 -

```
world_bank_pop_tidy <- world_bank_pop %>% pivot_longer(cols = c(`2000`:`2017`), names_to = "year", valu

world_bank_pop_tidy
```

```
## # A tibble: 19,008 x 4
##    country indicator   year  value
##    <chr>   <chr>       <chr> <dbl>
##  1 ABW     SP.URB.TOTL 2000  42444
##  2 ABW     SP.URB.TOTL 2001  43048
##  3 ABW     SP.URB.TOTL 2002  43670
##  4 ABW     SP.URB.TOTL 2003  44246
##  5 ABW     SP.URB.TOTL 2004  44669
##  6 ABW     SP.URB.TOTL 2005  44889
##  7 ABW     SP.URB.TOTL 2006  44881
##  8 ABW     SP.URB.TOTL 2007  44686
##  9 ABW     SP.URB.TOTL 2008  44375
## 10 ABW     SP.URB.TOTL 2009  44052
## # ... with 18,998 more rows
```

```
world_bank_pop_tidy %>% group_by(indicator) %>% count()
```

```
## # A tibble: 4 x 2
## # Groups:   indicator [4]
##   indicator       n
##   <chr>       <int>
## 1 SP.POP.GROW  4752
## 2 SP.POP.TOTL  4752
## 3 SP.URB.GROW  4752
## 4 SP.URB.TOTL  4752
```