cm009 Exercises: tidy data

install.packages("dplyr")

```
library(tidyverse)
library(dplyr)
lotr  <- read_csv("https://raw.githubusercontent.com/jennybc/lotr-tidy/master/data/lotr_tidy.csv")
guest <- read_csv("https://raw.githubusercontent.com/STAT545-UBC/Classroom/master/data/wedding/attend.cs
email <- read_csv("https://raw.githubusercontent.com/STAT545-UBC/Classroom/master/data/wedding/emails.cs
```

Exercise 1: Univariate Pivoting

Consider the Lord of the Rings data:

```
lotr
```

```
## # A tibble: 18 x 4
##    Film                      Race   Gender Words
##    <chr>                     <chr>  <chr>  <dbl>
##  1 The Fellowship Of The Ring Elf    Female  1229
##  2 The Fellowship Of The Ring Hobbit Female    14
##  3 The Fellowship Of The Ring Man    Female     0
##  4 The Two Towers             Elf    Female   331
##  5 The Two Towers             Hobbit Female     0
##  6 The Two Towers             Man    Female   401
##  7 The Return Of The King     Elf    Female   183
##  8 The Return Of The King     Hobbit Female     2
##  9 The Return Of The King     Man    Female   268
## 10 The Fellowship Of The Ring Elf    Male     971
## 11 The Fellowship Of The Ring Hobbit Male    3644
## 12 The Fellowship Of The Ring Man    Male    1995
## 13 The Two Towers             Elf    Male     513
## 14 The Two Towers             Hobbit Male    2463
## 15 The Two Towers             Man    Male    3589
## 16 The Return Of The King     Elf    Male     510
## 17 The Return Of The King     Hobbit Male    2673
## 18 The Return Of The King     Man    Male    2459
```

1. Would you say this data is in tidy format?
2. Widen the data so that we see the words spoken by each race, by putting race as its own column.

```
(lotr_wide <- lotr %>%
  pivot_wider(id_cols = c(-Race, -Words),
              names_from = Race,
              values_from = Words))
```

```
## # A tibble: 6 x 5
##   Film                      Gender   Elf Hobbit   Man
##   <chr>                     <chr>  <dbl>  <dbl> <dbl>
## 1 The Fellowship Of The Ring Female  1229     14     0
## 2 The Two Towers             Female   331      0   401
## 3 The Return Of The King     Female   183      2   268
## 4 The Fellowship Of The Ring Male     971   3644  1995
```

```
## 5 The Two Towers             Male     513   2463  3589
## 6 The Return Of The King     Male     510   2673  2459
```

3. Re-lengthen the wide LOTR data from Question 2 above.

```
lotr_wide %>%
  pivot_longer(cols = c(-Film, -Gender),
               names_to  = "Race",
               values_to = "Words")
```

```
## # A tibble: 18 x 4
##    Film                      Gender Race   Words
##    <chr>                     <chr>  <chr>  <dbl>
##  1 The Fellowship Of The Ring Female Elf     1229
##  2 The Fellowship Of The Ring Female Hobbit    14
##  3 The Fellowship Of The Ring Female Man        0
##  4 The Two Towers             Female Elf      331
##  5 The Two Towers             Female Hobbit     0
##  6 The Two Towers             Female Man      401
##  7 The Return Of The King     Female Elf      183
##  8 The Return Of The King     Female Hobbit     2
##  9 The Return Of The King     Female Man      268
## 10 The Fellowship Of The Ring Male   Elf      971
## 11 The Fellowship Of The Ring Male   Hobbit  3644
## 12 The Fellowship Of The Ring Male   Man     1995
## 13 The Two Towers             Male   Elf      513
## 14 The Two Towers             Male   Hobbit  2463
## 15 The Two Towers             Male   Man     3589
## 16 The Return Of The King     Male   Elf      510
## 17 The Return Of The King     Male   Hobbit  2673
## 18 The Return Of The King     Male   Man     2459
```

Exercise 2: Multivariate Pivoting

Congratulations, you're getting married! In addition to the wedding, you've decided to hold two other events: a day-of brunch and a day-before round of golf. You've made a guestlist of attendance so far, along with food preference for the food events (wedding and brunch).

```
guest %>%
  DT::datatable(rownames = FALSE)
```

Show 10 entries Search:

party	name	meal_wedding	meal_brunch	attendance_wedding	attendance_brunch	attendance_golf
1	Sommer Medrano	PENDING	PENDING	PENDING	PENDING	PENDING
1	Phillip Medrano	vegetarian	Menu C	CONFIRMED	CONFIRMED	CONFIRMED
1	Blanka Medrano	chicken	Menu A	CONFIRMED	CONFIRMED	CONFIRMED
1	Emaan Medrano	PENDING	PENDING	PENDING	PENDING	PENDING
2	Blair Park	chicken	Menu C	CONFIRMED	CONFIRMED	CONFIRMED
2	Nigel Webb			CANCELLED	CANCELLED	CANCELLED
3	Sinead English	PENDING	PENDING	PENDING	PENDING	PENDING
4	Ayra Marks	vegetarian	Menu B	PENDING	PENDING	PENDING
5	Atlanta Connolly	PENDING	PENDING	PENDING	PENDING	PENDING
5	Denzel Connolly	fish	Menu B	CONFIRMED	CONFIRMED	CONFIRMED

Showing 1 to 10 of 30 entries Previous 1 2 3 Next

1. Put "meal" and "attendance" as their own columns, with the events living in a new column.

```
(guest_long <- guest %>%
  pivot_longer(cols      = c(-party, -name),
               names_to  = c(".value", "event"),
               names_sep = "_"))
```

```
## # A tibble: 90 x 5
##    party name            event   meal       attendance
##    <dbl> <chr>           <chr>   <chr>      <chr>
##  1     1 Sommer Medrano  wedding PENDING    PENDING
##  2     1 Sommer Medrano  brunch  PENDING    PENDING
##  3     1 Sommer Medrano  golf    <NA>       PENDING
##  4     1 Phillip Medrano wedding vegetarian CONFIRMED
##  5     1 Phillip Medrano brunch  Menu C     CONFIRMED
##  6     1 Phillip Medrano golf    <NA>       CONFIRMED
##  7     1 Blanka Medrano  wedding chicken    CONFIRMED
##  8     1 Blanka Medrano  brunch  Menu A     CONFIRMED
##  9     1 Blanka Medrano  golf    <NA>       CONFIRMED
## 10     1 Emaan Medrano   wedding PENDING    PENDING
## # ... with 80 more rows
```

2. Use `tidyr::separate()` to split the name into two columns: "first" and "last". Then, re-unite them with `tidyr::unite()`.

```
guest_long %>%
  separate(name, into = c("first", "last"), sep = " ") %>%
  unite(col = "name", first, last, sep = " ")
```

```
## # A tibble: 90 x 5
##    party name            event   meal       attendance
##    <dbl> <chr>           <chr>   <chr>      <chr>
```

```
##  1     1 Sommer Medrano  wedding PENDING    PENDING
##  2     1 Sommer Medrano  brunch  PENDING    PENDING
##  3     1 Sommer Medrano  golf    <NA>       PENDING
##  4     1 Phillip Medrano wedding vegetarian CONFIRMED
##  5     1 Phillip Medrano brunch  Menu C     CONFIRMED
##  6     1 Phillip Medrano golf    <NA>       CONFIRMED
##  7     1 Blanka Medrano  wedding chicken    CONFIRMED
##  8     1 Blanka Medrano  brunch  Menu A     CONFIRMED
##  9     1 Blanka Medrano  golf    <NA>       CONFIRMED
## 10     1 Emaan Medrano   wedding PENDING    PENDING
## # ... with 80 more rows
```

3. Which parties still have a "PENDING" status for all members and all events?

```
guest_long %>%
  group_by(party) %>%
  summarize(all_pending = all(attendance == "PENDING"))
```

```
## # A tibble: 15 x 2
##    party all_pending
##    <dbl> <lgl>
##  1     1 FALSE
##  2     2 FALSE
##  3     3 TRUE
##  4     4 TRUE
##  5     5 FALSE
##  6     6 FALSE
##  7     7 FALSE
##  8     8 TRUE
##  9     9 FALSE
## 10    10 TRUE
## 11    11 FALSE
## 12    12 FALSE
## 13    13 FALSE
## 14    14 FALSE
## 15    15 FALSE
```

4. Which parties still have a "PENDING" status for all members for the wedding?

```
guest %>%
  group_by(party) %>%
  summarize(pending_wedding = all(attendance_wedding == "PENDING"))
```

```
## # A tibble: 15 x 2
##    party pending_wedding
##    <dbl> <lgl>
##  1     1 FALSE
##  2     2 FALSE
##  3     3 TRUE
##  4     4 TRUE
##  5     5 FALSE
##  6     6 FALSE
##  7     7 FALSE
##  8     8 TRUE
##  9     9 FALSE
## 10    10 TRUE
```

```
## 11    11 FALSE
## 12    12 FALSE
## 13    13 FALSE
## 14    14 FALSE
## 15    15 FALSE
```

5. Put the data back to the way it was.

```
guest_long %>%
  pivot_wider(id_cols     = c(party, name),
              names_from  = c(event),
              names_sep   = "_",
              values_from = c(meal, attendance))
```

```
## # A tibble: 30 x 8
##    party name  meal_wedding meal_brunch meal_golf attendance_wedd~
##    <dbl> <chr> <chr>        <chr>       <chr>     <chr>
##  1     1 Somm~ PENDING      PENDING     <NA>      PENDING
##  2     1 Phil~ vegetarian   Menu C      <NA>      CONFIRMED
##  3     1 Blan~ chicken      Menu A      <NA>      CONFIRMED
##  4     1 Emaa~ PENDING      PENDING     <NA>      PENDING
##  5     2 Blai~ chicken      Menu C      <NA>      CONFIRMED
##  6     2 Nige~ <NA>         <NA>        <NA>      CANCELLED
##  7     3 Sine~ PENDING      PENDING     <NA>      PENDING
##  8     4 Ayra~ vegetarian   Menu B      <NA>      PENDING
##  9     5 Atla~ PENDING      PENDING     <NA>      PENDING
## 10     5 Denz~ fish         Menu B      <NA>      CONFIRMED
## # ... with 20 more rows, and 2 more variables: attendance_brunch <chr>,
## #   attendance_golf <chr>
```

6. You also have a list of emails for each party, in this worksheet under the variable `email`. Change this so that each person gets their own row. Use `tidyr::separate_rows()`

```
email %>%
  separate_rows(guest, sep = ", ")
```

```
## # A tibble: 28 x 2
##    guest           email
##    <chr>           <chr>
##  1 Sommer Medrano  sommm@gmail.com
##  2 Phillip Medrano sommm@gmail.com
##  3 Blanka Medrano  sommm@gmail.com
##  4 Emaan Medrano   sommm@gmail.com
##  5 Blair Park      bpark@gmail.com
##  6 Nigel Webb      bpark@gmail.com
##  7 Sinead English  singlish@hotmail.ca
##  8 Ayra Marks      marksa42@gmail.com
##  9 Jolene Welsh    jw1987@hotmail.com
## 10 Hayley Booker   jw1987@hotmail.com
## # ... with 18 more rows
```

Exercise 3: Making tibbles

1. Create a tibble that has the following columns:

- A `label` column with `"Sample A"` in its entries.

- 100 random observations drawn from the N(0,1) distribution in the column `x`
- `y` calculated as the `x` values + N(0,1) error.

```
n <- 100
  tibble(label = "Sample A",
              x = rnorm(n),
              y = x + rnorm(n))
```

```
## # A tibble: 100 x 3
##    label         x        y
##    <chr>     <dbl>    <dbl>
##  1 Sample A  1.02   1.88
##  2 Sample A -0.641  0.320
##  3 Sample A -0.268  1.16
##  4 Sample A -0.462 -0.803
##  5 Sample A  0.755  1.07
##  6 Sample A  0.218  1.24
##  7 Sample A -1.53  -0.00285
##  8 Sample A  0.221  1.69
##  9 Sample A -0.288  0.166
## 10 Sample A -0.346  0.0192
## # ... with 90 more rows
```

2. Generate a Gaussian sample of size 100 for each combination of the following means (`mu`) and standard deviations (`sd`).

```
n <- 100
mu <- c(-5, 0, 5)
sd <- c(1, 3, 10)
tibble(mu = mu, sd = sd) %>%
  group_by_all() %>%
  mutate(z = list(rnorm(n, mu, sd))) %>%
  unnest(z)
```

```
## # A tibble: 300 x 3
## # Groups:   mu, sd [3]
##       mu    sd     z
##    <dbl> <dbl> <dbl>
##  1    -5     1 -3.31
##  2    -5     1 -6.63
##  3    -5     1 -4.32
##  4    -5     1 -4.61
##  5    -5     1 -4.45
##  6    -5     1 -3.99
##  7    -5     1 -3.17
##  8    -5     1 -4.65
##  9    -5     1 -3.74
## 10    -5     1 -5.11
## # ... with 290 more rows
```

3. Fix the `experiment` tibble below (originally defined in the documentation of the `tidyr::expand()` function) so that all three repeats are displayed for each person, and the measurements are kept. The code is given, but needs one adjustment. What is it?

```
experiment <- tibble(
  name = rep(c("Alex", "Robert", "Sam"), c(3, 2, 1)),
  trt  = rep(c("a", "b", "a"), c(3, 2, 1)),
```

```
  rep = c(1, 2, 3, 1, 2, 1),
  measurement_1 = runif(6),
  measurement_2 = runif(6)
)
experiment %>% complete(nesting(name, trt), rep)
```

```
## # A tibble: 9 x 5
##   name   trt     rep measurement_1 measurement_2
##   <chr>  <chr> <dbl>         <dbl>         <dbl>
## 1 Alex   a         1         0.439         0.372
## 2 Alex   a         2         0.541         0.340
## 3 Alex   a         3         0.285         0.836
## 4 Robert b         1         0.648         0.603
## 5 Robert b         2         0.309         0.712
## 6 Robert b         3        NA            NA
## 7 Sam    a         1         0.171         0.602
## 8 Sam    a         2        NA            NA
## 9 Sam    a         3        NA            NA
```