Data Science Module 4 Exercises

Brady Lamson

2/23/2022

6: Tidy Data

6.2: Using pivot_longer() and pivot_wider()

Exercise 1

Write a command using pivot_longer() that converts the given data frame to narrow format. Name the columns Grp and Y.

```
xWide <- data.frame(
    GrpA = c(1, 4, 2, 3),
    GrpB = c(7, 5, 8, 6),
    GrpC = c(9, 9, 8, 7))

xWide %>%
    tidyr::pivot_longer(
        cols = c("GrpA", "GrpB", "GrpC"),
        names_to = "Grp",
        values_to = "Y"
    )
```

```
## # A tibble: 12 x 2
##    Grp       Y
##    <chr> <dbl>
##  1 GrpA      1
##  2 GrpB      7
##  3 GrpC      9
##  4 GrpA      4
##  5 GrpB      5
##  6 GrpC      9
##  7 GrpA      2
##  8 GrpB      8
##  9 GrpC      8
## 10 GrpA      3
## 11 GrpB      6
## 12 GrpC      7
```

Exercise 2

Here are data from a study in which a variable Y was recorded on each of five subjects before and after an intervention:

```
xNarrow <- data.frame(Subject = c(1:5, 1:5),
                      Period = c("Before", "Before", "Before", "Before",
                                 "Before", "After", "After", "After",
                                 "After", "After"),
                      Y = c(22, 45, 32, 45, 30, 60, 44, 24, 56, 59),
                      stringsAsFactors = FALSE)
```

a) Write a command involving pivot_wider() that converts xNarrow to a **wide** format.

```
xNarrow %>%
    tidyr::pivot_wider(
        names_from = Period,
        values_from = Y
    )
```

```
## # A tibble: 5 x 3
##   Subject Before After
##     <int>  <dbl> <dbl>
## 1       1     22    60
## 2       2     45    44
## 3       3     32    24
## 4       4     45    56
## 5       5     30    59
```

b)

```
data.frame(Period = c("Before", "Before", "Before",
                      "Before", "Before", "After", "After",
                      "After", "After", "After"),
           Y = c(22, 45, 32, 45, 30, 60, 44, 24, 56, 59),
           stringsAsFactors = FALSE) %>%
    tidyr::pivot_wider(
        names_from = Period,
        values_from = Y
    )
```

```
## Warning: Values from `Y` are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = {summary_fun}` to summarise duplicates.
## * Use the following dplyr code to identify duplicates.
##   {data} %>%
##     dplyr::group_by(Period) %>%
##     dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
##     dplyr::filter(n > 1L)
```

```
## # A tibble: 1 x 2
##   Before    After
##   <list>    <list>
## 1 <dbl [5]> <dbl [5]>
```

Exercise 3

Write a command involving pivot_longer() and the "helper" function num_range() that converts xWide to narrow format.

```
data.frame(Subject = c(1001, 1002, 1003),
                    t1 = c(22, 45, 32),
                    t2 = c(45, 30, 60),
                    t3 = c(44, 24, 56),
                    t4 = c(55, 27, 53)
                    ) %>%
    tidyr::pivot_longer(
        cols = num_range("t", 1:4),
        names_to = "Time",
        values_to = "Y"
    )
```

```
## # A tibble: 12 x 3
##    Subject Time      Y
##      <dbl> <chr> <dbl>
##  1    1001 t1       22
##  2    1001 t2       45
##  3    1001 t3       44
##  4    1001 t4       55
##  5    1002 t1       45
##  6    1002 t2       30
##  7    1002 t3       24
##  8    1002 t4       27
##  9    1003 t1       32
## 10    1003 t2       60
## 11    1003 t3       56
## 12    1003 t4       53
```

Exercise 4

```
xWide <- data.frame(Subject = c(1001, 1002, 1003),
                    Gender = c("m", "f", "f"),
                    t1 = c(22, 45, 32),
                    t2 = c(45, 30, 60),
                    t3 = c(44, 24, 56),
                    t4 = c(55, 27, 53))
```

What happens to the gender column when you convert xWide to narrow format? Try the below code.

```
xNarrow <- tidyr::pivot_longer(data = xWide,
                        cols = num_range("t", 1:4),
                        names_to = "Time",
                        values_to = "Y")
```

We end up with 8 rows of 'f' because we had two instances of 'f' that were quadrupled.

6.3: Separating and Uniting Columns Using separate() and unite()

Exercise 5

```
diseases <- data.frame(country = c("Afghanistan", "Afghanistan",
                                   "Brazil", "Brazil", "China", "China"),
                       year = c(1999, 2000, 1999, 2000, 1999, 2000),
                       rate = c("745/19987071", "2666/20595360",
                                "37737/172006362", "80488/174504898",
                                "212258/1272915272", "213766/1280428583"))
```

Write a command that separates the rate column into two columns.

```
diseases %>%
    tidyr::separate(
        col = rate,
        into = c("cases", "population"),
        sep = "/"
    )
```

```
##       country year  cases population
## 1 Afghanistan 1999    745   19987071
## 2 Afghanistan 2000   2666   20595360
## 3      Brazil 1999  37737  172006362
## 4      Brazil 2000  80488  174504898
## 5       China 1999 212258 1272915272
## 6       China 2000 213766 1280428583
```

Exercise 6

```
year <- c(2017, 2017, 2017, 2017, 2017, 2017, 2017, 2018, 2018, 2018, 2018, 2018,
          2018, 2018)
month <- c(6, 6, 7, 7, 7, 8, 8, 6, 6, 7, 7, 7, 8, 8)
day <- c(4, 18, 2, 16, 30, 13, 27, 3, 17, 1, 15, 29, 12, 26)
phosphate <- c(2.42, 3.50, 1.78, 2.46, 0.66, 1.16, 0.68, 0.90, 1.11, 1.25, 2.28,
               1.36, 0.43, 2.90)
nitrate <- c(3.38, 3.87, 1.28, 3.45, NA, 3.64, 1.88, 6.16, 2.55, 2.98, 3.90, 3.31,
             4.19, 5.35)

river <- data.frame(Year = year,
                    Month = month,
                    Day = day,
                    Phosphate = phosphate,
                    Nitrate = nitrate)
```

```
river %>%
    tidyr::unite(
        col = "Date",
        c(Month, Day, Year),
        sep = "/"
    ) %>%
    head()
```

```
##        Date Phosphate Nitrate
## 1  6/4/2017      2.42    3.38
## 2 6/18/2017      3.50    3.87
## 3  7/2/2017      1.78    1.28
## 4 7/16/2017      2.46    3.45
## 5 7/30/2017      0.66      NA
## 6 8/13/2017      1.16    3.64
```

6.4: Data Intake

Exercise 7

```
url <- "https://en.wikipedia.org/wiki/Mile_run_world_record_progression"
tables <- url %>% read_html() %>% html_nodes("table")
table4 <- rvest::html_table(tables[[4]])

table4 %>% head()
```

```
## # A tibble: 6 x 6
##   Time   Auto  Athlete          Nationality   Date              Venue
##   <chr>  <chr> <chr>            <chr>         <chr>             <chr>
## 1 4:14.4 ""    John Paul Jones  United States 31 May 1913[6]    Allston, Mass.
## 2 4:12.6 ""    Norman Taber     United States 16 July 1915[6]   Allston, Mass.
## 3 4:10.4 ""    Paavo Nurmi      Finland       23 August 1923[6] Stockholm
## 4 4:09.2 ""    Jules Ladoumègue France        4 October 1931[6] Paris
## 5 4:07.6 ""    Jack Lovelock    New Zealand   15 July 1933[6]   Princeton, N.J.
## 6 4:06.8 ""    Glenn Cunningham United States 16 June 1934[6]   Princeton, N.J.
```

Exercise 8

```
url <- "https://en.wikipedia.org/wiki/World_population"
tables <- url %>%
    rvest::read_html() %>%
    rvest::html_nodes("table")

rvest::html_table(tables[[5]]) %>%
    head()
```

```
## # A tibble: 6 x 6
##    Rank Country       Population    `% of world` Date        `Source(official ~`
##   <int> <chr>         <chr>         <chr>        <chr>       <chr>
## 1     1 China         1,412,121,560 17.8%        26 Feb 2022 National populatio~
## 2     2 India         1,388,517,013 17.5%        26 Feb 2022 National populatio~
## 3     3 United States 333,293,321   4.20%        26 Feb 2022 National populatio~
## 4     4 Indonesia     269,603,400   3.40%        1 Jul 2020  National annual pr~
## 5     5 Pakistan      220,892,331   2.78%        1 Jul 2020  UN Projection[95]
## 6     6 Brazil        214,403,748   2.70%        26 Feb 2022 National populatio~
```

6.5: Cleaning Data

Exercise 9

```
# Create houses_small data frame -----
houses_url <-
    "http://sites.msudenver.edu/ngrevsta/wp-content/uploads/sites/416/2021/02/houses-for-sale.txt"

houses <- read.csv(houses_url, header = TRUE, sep = "\t")
houses_small <- select(houses, fuel, heat, sewer, construction)

# Create codebook data frame -----
codebook_url <-
    "http://sites.msudenver.edu/ngrevsta/wp-content/uploads/sites/416/2021/02/house_codes.txt"

translations <- read.csv(codebook_url,
                         header = TRUE,
                         stringsAsFactors = FALSE,
                         sep = "\t")

codes <- translations %>%
    pivot_wider(names_from = system_type,
                values_from = meaning,
                values_fill = list(meaning = "invalid"))
```

```
dplyr::left_join(
    x = houses_small,
    y = dplyr::select(codes, code, heat_type),
    by = c(heat = "code")
) %>%
    head()
```

```
##   fuel heat sewer construction heat_type
## 1    3    4     2            0  electric
## 2    2    3     2            0 hot water
## 3    2    3     3            0 hot water
## 4    2    2     2            0   hot air
## 5    2    2     3            1   hot air
## 6    2    2     2            0   hot air
```

Exercise 10

```
data.frame(Name = c("Joe", "Lucy", "Tom", "Sally"),
                NumberChildren = c("2", "1", "0", "3"),
                stringsAsFactors = FALSE) %>%
    dplyr::mutate(
        NumberChildren = as.numeric(NumberChildren)
    ) %>%
    str()
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ Name          : chr  "Joe" "Lucy" "Tom" "Sally"
##  $ NumberChildren: num  2 1 0 3
```

Exercise 11

```
data.frame(Name = c("Joe", "Lucy", "Tom", "Sally"),
           NumberChildren = c("2", "Unknown", "0", "3"),
           stringsAsFactors = FALSE
          ) %>%
    dplyr::mutate(
        NumberChildren = as.numeric(NumberChildren)
    )
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
##    Name NumberChildren
## 1   Joe              2
## 2  Lucy             NA
## 3   Tom              0
## 4 Sally              3
```

The unknown variable in the number children column gets automatically converted to NA.

Exercise 12

Guess the output of the following code

Personal note after looking over all of them: I think all of these are equivalent to lubridate, that's my guess after running the first command.

a) lubridate::mdy("Dec 18, 1973")

- 12/18/1973

b) lubridate::mdy("December 18, 1973")

- "1973-12-18"

c) lubridate::mdy("12/18/1973")

- "1973-12-18"

d) lubridate::mdy("12/18/73")

- "1973-12-18"

e) lubridate::mdy("12-18-1973")

- "1973-12-18"

f) lubridate::mdy("12-18-73")

- "1973-12-18"

```
# Checking answers.

glue::glue("
    {lubridate::mdy(\"Dec 18, 1973\")}
    {lubridate::mdy(\"December 18, 1973\")}
    {lubridate::mdy(\"12/18/1973\")}
    {lubridate::mdy(\"12/18/73\")}
    {lubridate::mdy(\"12-18-1973\")}
    {lubridate::mdy(\"12-18-73\")}
")
```

```
## 1973-12-18
## 1973-12-18
## 1973-12-18
## 1973-12-18
## 1973-12-18
## 1973-12-18
```

Exercise 13

```
glue::glue("
    {lubridate::mdy(\"11/14/23\")}
")
```

```
## 2023-11-14
```

Exercise 14

How many elapsed days are there between January 15, 20017 and October 4, 2019?

```
day1 <- lubridate::mdy("1/15/07")
day2 <- lubridate::mdy("10/4/19")

glue::glue("
    {day2 - day1} days elapsed between January 15, 2007 and October 4, 2019
")
```

```
## 4645 days elapsed between January 15, 2007 and October 4, 2019
```

Exercise 15

Guess the results of the given commands and check your answers.

a) seq(from = mdy("12-20-1993"), to = mdy("01-15-2004"), by = "days")

- This will generate a vector of every day between the two provided dates.

b) seq(from = mdy("12-20-1993"), to = mdy("01-15-2004"), by = "weeks")

- This will do the same thing, but instead of every day there will be 7 day gaps.

c) seq(from = mdy("12-20-1993"), to = mdy("01-15-2004"), by = "years")

- This will do the same thing, but with yearly gaps instead of day gaps.

```
# A
seq(
    from = lubridate::mdy("12-20-1993"),
    to = lubridate::mdy("01-15-2004"),
    by = "days"
) %>%
    head()
```

```
## [1] "1993-12-20" "1993-12-21" "1993-12-22" "1993-12-23" "1993-12-24"
## [6] "1993-12-25"
```

```
# B

seq(
    from = lubridate::mdy("12-20-1993"),
    to = lubridate::mdy("01-15-2004"),
    by = "weeks"
) %>%
    head()
```

```
## [1] "1993-12-20" "1993-12-27" "1994-01-03" "1994-01-10" "1994-01-17"
## [6] "1994-01-24"
```

```
# C

seq(
    from = lubridate::mdy("12-20-1993"),
    to = lubridate::mdy("01-15-2004"),
    by = "years"
)
```

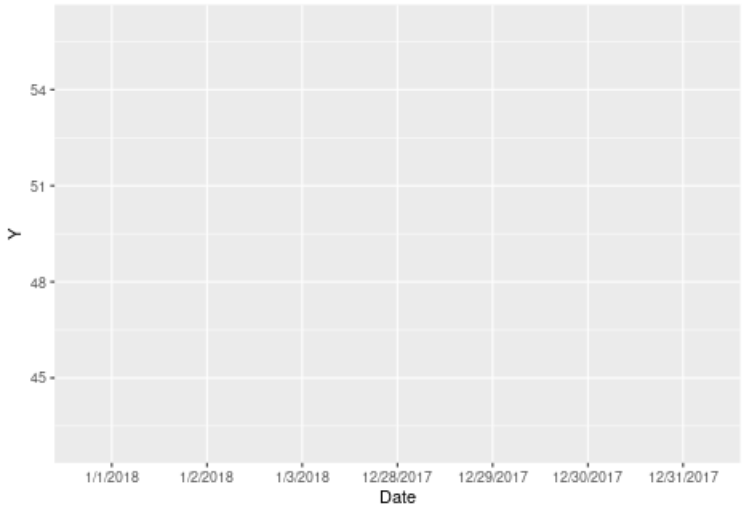
```
##  [1] "1993-12-20" "1994-12-20" "1995-12-20" "1996-12-20" "1997-12-20"
##  [6] "1998-12-20" "1999-12-20" "2000-12-20" "2001-12-20" "2002-12-20"
## [11] "2003-12-20"
```

Exercise 16

```
# A
my.data <- data.frame(Date = c("12/28/2017", "12/29/2017", "12/30/2017",
                               "12/31/2017", "1/1/2018", "1/2/2018", "1/3/2018"),
                      Y = c(44, 43, 47, 53, 53, 55, 56))

ggplot2::ggplot(
    data = my.data,
    mapping = ggplot2::aes(
        x = Date,
        y = Y
    )
) +
    ggplot2::geom_line()
```

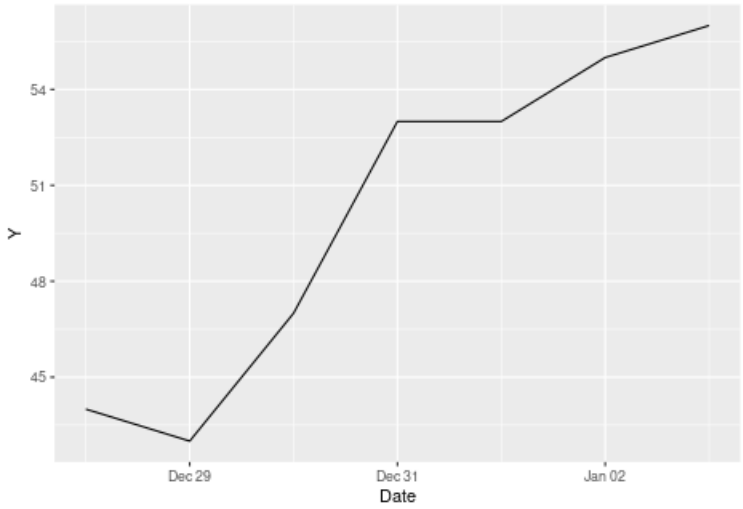
```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```



It's confused and doesn't know what to make of the date vector. This is unsurprising, they're all strings, not dates. We can easily fix that.

```
# B

my.data %>%
    dplyr::mutate(
        Date = lubridate::mdy(Date)
    ) %>%
    ggplot2::ggplot(
        mapping = ggplot2::aes(
            x = Date,
            y = Y
        )
    ) +
    ggplot2::geom_line()
```


54
51
48
45
Y
Dec 29
Dec 31
Jan 02
Date

7: Iteration

7.1: Iteration Using for() and best_by() with mutate()

Exercise 17

My guess is that the below code will print out "good sport" 5 times. I assume this because 1:5 is inclusive, not exclusive.

```
for (i in 1:5) {
    print("Good Sport")
}
```

```
## [1] "Good Sport"
## [1] "Good Sport"
## [1] "Good Sport"
## [1] "Good Sport"
## [1] "Good Sport"
```

Exercise 18

This loop will print each each of the integers in the vector squared. So 4, 16, 36, 64.

```
x <- c(2, 4, 6, 8)

for(i in x) {
    print(i^2)
}
```

```
## [1] 4
## [1] 16
## [1] 36
## [1] 64
```

Exercise 19

The sum of squares

$$\sum_{i=1}^{10} i^2 = 1^2 + 2^2 + ... + 10^2$$

can be computed using a for() loop.

```
sum.sq <- 0
for(i in 1:10) {
    sum.sq <- sum.sq + i^2
}
sum.sq
```

```
## [1] 385
```

a) Why is it necessary to make the assignment sum.sq <- 0 before entering the loop? What would happen if it wasn't there?

- Assigning it after entering the loop would get it reassigned to 0 at the start of every iteration. If it wasn't there the for loop simply wouldn't work, you'd need to assign something to actually get a meaningful return value, and that assignment would need to happen *outside* of the loop for it to not get immediately overridden.

b) Show what happens if the assignment happens inside the loop.

```
for(i in 1:10) {
    sum.sq <- 0
    sum.sq <- sum.sq + i^2
}
sum.sq
```

```
## [1] 100
```

Exercise 20

```
(1:10)^2
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

This squares everything in the vector 1:10. You could actually do the same thing as the above (correctly done) sum by getting the sum of this squared vector.

```
(1:10)^2 %>%
    sum()
```

```
## [1] 385
```

Exercise 21

```
by_subject <- nest_by(.data = lme4::sleepstudy, Subject)

models <- mutate(.data = by_subject, mod = list(lm(Reaction ~ Days, data = data)))

models %>% head()
```

```
## # A tibble: 6 x 3
## # Rowwise:  Subject
##   Subject               data mod
##   <fct>   <list<tibble[,2]>> <list>
## 1 308               [10 x 2] <lm>
## 2 309               [10 x 2] <lm>
## 3 310               [10 x 2] <lm>
## 4 330               [10 x 2] <lm>
## 5 331               [10 x 2] <lm>
## 6 332               [10 x 2] <lm>
```

```
models$mod[[
    which(models$Subject == 371)
]]
```

```
##
## Call:
## lm(formula = Reaction ~ Days, data = data)
##
## Coefficients:
## (Intercept)         Days
##     253.636        9.188
```

The equation of the fitted line for subject 371 is:

$$y = 253.636 + 9.188x$$