Class_Exercises_ClassNotes_4

Tobias Boggess

2/21/2022

Section 6.2 Exercises

Exercise 1: Write a command involving pivot_longer() that converts xWide to narrow format. Name the columns Grp and Y. Report your R command.

Code:

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

```
##   GrpA GrpB GrpC
## 1    1    7    9
## 2    4    5    9
## 3    2    8    8
## 4    3    6    7
```

```
pivot_longer(data = xWide, cols = c("GrpA", "GrpB", "GrpC"))
```

```
## # A tibble: 12 x 2
##    name  value
##    <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: Do the following.

Code:

```
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
)
xNarrow
```

```
##    Subject Period  Y
## 1        1 Before 22
## 2        2 Before 45
## 3        3 Before 32
## 4        4 Before 45
## 5        5 Before 30
## 6        1  After 60
## 7        2  After 44
## 8        3  After 24
## 9        4  After 56
## 10       5  After 59
```

a) Write a command involving pivot_wider() that converts xNarrow to a wide format. Report your R command.
Code:

```
pivot_wider(
  data = xNarrow,
  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) What would happen if the Subject variable was missing? Try it by running the command you wrote for part a on the following data frame:
Code:

```
xNarrowNoSubject <-
  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
  )

pivot_wider(
  data = xNarrowNoSubject,
  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]>
```

This provides a warning "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)" and only prints a single line.

Exercise 3: Write a command involving pivot_longer() and the "helper" function num_range() that converts xWide to narrow format. Report your R command

Code:

```
xWide <- 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)
)

pivot_longer(data = xWide, cols = num_range("t", 0:100))
```

```
## # A tibble: 12 x 3
##    Subject name  value
##      <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: What happens to the Gender column when you convert xWide to narrow format? Try it, by typing:

Code:

```
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)
)
xWide
```

```
##   Subject Gender t1 t2 t3 t4
## 1    1001      m 22 45 44 55
## 2    1002      f 45 30 24 27
## 3    1003      f 32 60 56 53
```

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

xNarrow
```

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

This printed the Gender column in accordance to the Time column.

Section 6.3 Exercises

Exercise 5: Write a command involving separate() that separates the rate column into two columns named cases and population. Report your R command.

Code:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
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"
  )
)
diseases
```

```
##       country year              rate
## 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
```

```r
separate(data = diseases, 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: Write a command involving unite() that combines the Month, Day, and Year columns into a single column named Date having the form month/day/year. Report your R command.

Code:

```
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)
```

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

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

```r
new_river <-
  unite(data = river,
        col = "Date",
        c(Month, Day, Year),
        sep = "/")

head(new_river)
```

```
##        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
```

Section 6.4 Exercises

Exercise 7: create an R data frame containing the data from the fourth table of world record times for the mile run. Report your R commands.

Code:

```
library(rvest)
url <-
  "https://en.wikipedia.org/wiki/Mile_run_world_record_progression"
tables <- url %>% read_html() %>% html_nodes("table")
Table4 <- html_table(tables[[4]])
head(Table4)
```

```
## # 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: Using the approach described above, create an R data frame containing the data from the fifth table of populations. Report your R commands.

Code:

```
url1 <- "https://en.wikipedia.org/wiki/World_population"
tables <- url1 %>% read_html() %>% html_nodes("table")
Table5 <- html_table(tables[[5]])
head(Table5)
```

```
## # A tibble: 6 x 6
##    Rank Country       Population    `% of world` Date        `Source(official ~`
##   <int> <chr>         <chr>         <chr>        <chr>       <chr>
## 1     1 China         1,412,078,000 17.8%        23 Feb 2022 National populatio~
## 2     2 India         1,388,396,236 17.5%        23 Feb 2022 National populatio~
## 3     3 United States 333,276,387   4.20%        23 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        214,390,509   2.70%        23 Feb 2022 National populatio~
```

Section 6.5 Exercises

Exercise 9: Now recode heat from integers to "character" values ("hot air", "hot water", and "electric") by using left_join() to merge Houses_small with Codes, matching rows in Codes and Houses_small by the (integer) variables code and heat. Report your R command(s).

Code:

```
myURL <-
  "http://sites.msudenver.edu/ngrevsta/wp-content/uploads/sites/416/2021/02/houses-for-sale.txt"
Houses <- read.csv(myURL, header = TRUE, sep = "\t")
Houses_small <- select(Houses, fuel, heat, sewer, construction)
myURL <-
  "http://sites.msudenver.edu/ngrevsta/wp-content/uploads/sites/416/2021/02/house_codes.txt"
Translations <- read.csv(myURL,
                         header = TRUE,
                         stringsAsFactors = FALSE,
                         sep = "\t")
Codes <- Translations %>% pivot_wider(
  names_from = system_type,
  values_from = meaning,
  values_fill = list(meaning = "invalid")
)

Houses_new_small <-
                   Houses_small %>%
                   left_join(Codes %>%
                   select(code, heat_type),
                   by = c(heat = "code"))
head(Houses_new_small)
```

```
##   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: Write one or more commands using mutate() and either as.numeric() or parse_number() that convert the NumberChildren column of x to numeric. Check your answer using str(). Report your R command(s).

Code:

```
x <- data.frame(
  Name = c("Joe", "Lucy", "Tom", "Sally"),
  NumberChildren = c("2", "1", "0", "3"),
  stringsAsFactors = FALSE
)
str(x)
```

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

```
x <- mutate(.data = x, y = as.numeric(NumberChildren))
x
```

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

Exercise 11: What happens to the value in the 2nd position of NumberChildren when you type the following:

Code:

```
x <- data.frame(
  Name = c("Joe", "Lucy", "Tom", "Sally"),
  NumberChildren = c("2", "Unknown", "0", "3"),
  stringsAsFactors = FALSE
)

x <- mutate(x, NumberChildren = as.numeric(NumberChildren))
```

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

```
x
```

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

The value in the second position will trn into an NA value.

Exercise 12: Guess what each of the following commands returns, then check your answers.

Code:

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

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

```
mdy("Dec 18, 1973")
```

```
## [1] "1973-12-18"
```

This will print in the format year-month-day as a numeric version.

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

```
mdy("December 18, 1973")
```

```
## [1] "1973-12-18"
```

This will print the same thing as *part a.*

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

```
mdy("12/18/1973")
```

```
## [1] "1973-12-18"
```

This will do the same as *part a.*

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

```
mdy("12/18/73")
```

```
## [1] "1973-12-18"
```

This will do the same as *part a.*

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

```
mdy("12-18-1973")
```

```
## [1] "1973-12-18"
```

This will do the same as *part a*.

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

```
mdy("12-18-73")
```

```
## [1] "1973-12-18"
```

This will do the same as *part a*.

Exercise 13: Does mdy() interpret "11/14/23" as referring to the year 2023 or 1923? Try it.

Code:

```
mdy("11/14/23")
```

```
## [1] "2023-11-14"
```

This will assume 2023 is the year.

Exercise 14: How many elapsed days are there between January 15, 2007 ("1/15/07") and October 4, 2019 ("10/4/19")?

Code:

```
days1 <-
  seq(
    from = mdy("1/15/2007"),
    to = mdy("10/4/2019"),
    by = "days"
  )
length(days1)
```

```
## [1] 4646
```

There should be 4646 days between the days mentioned above.

Exercise 15: Guess what the following do.

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

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

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

This will display all the days between 12-20-1993 and 01-15-2004.

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

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

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

This will show every 7 days starting 12-20-1993 til 01-15-2004.

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

```
seq(
  from = mdy("12-20-1993"),
  to = 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"
```

This will show the yearly date starting at 12-20-1993.

Exercise 16: Do the following.

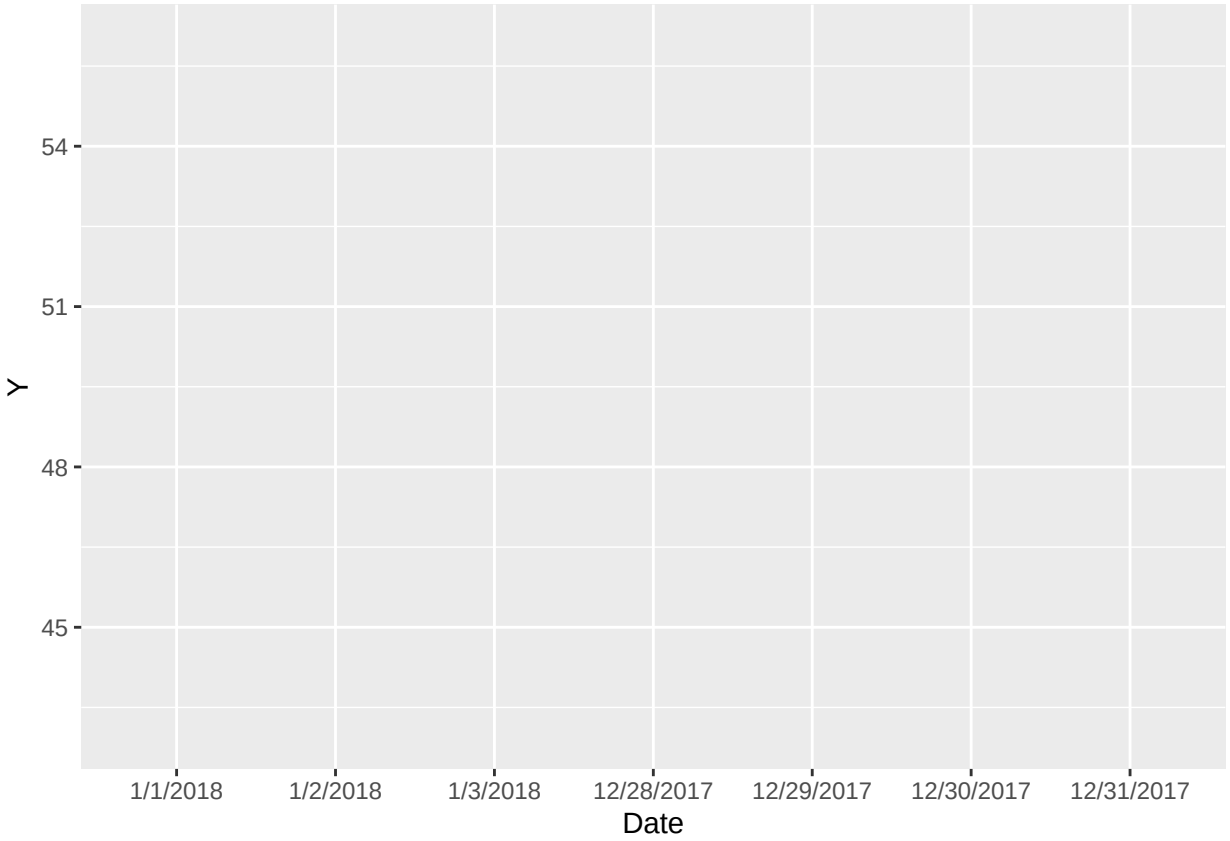
Code:

```
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)
  )
```

a) Why doesn't the code below work?
Code:

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

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

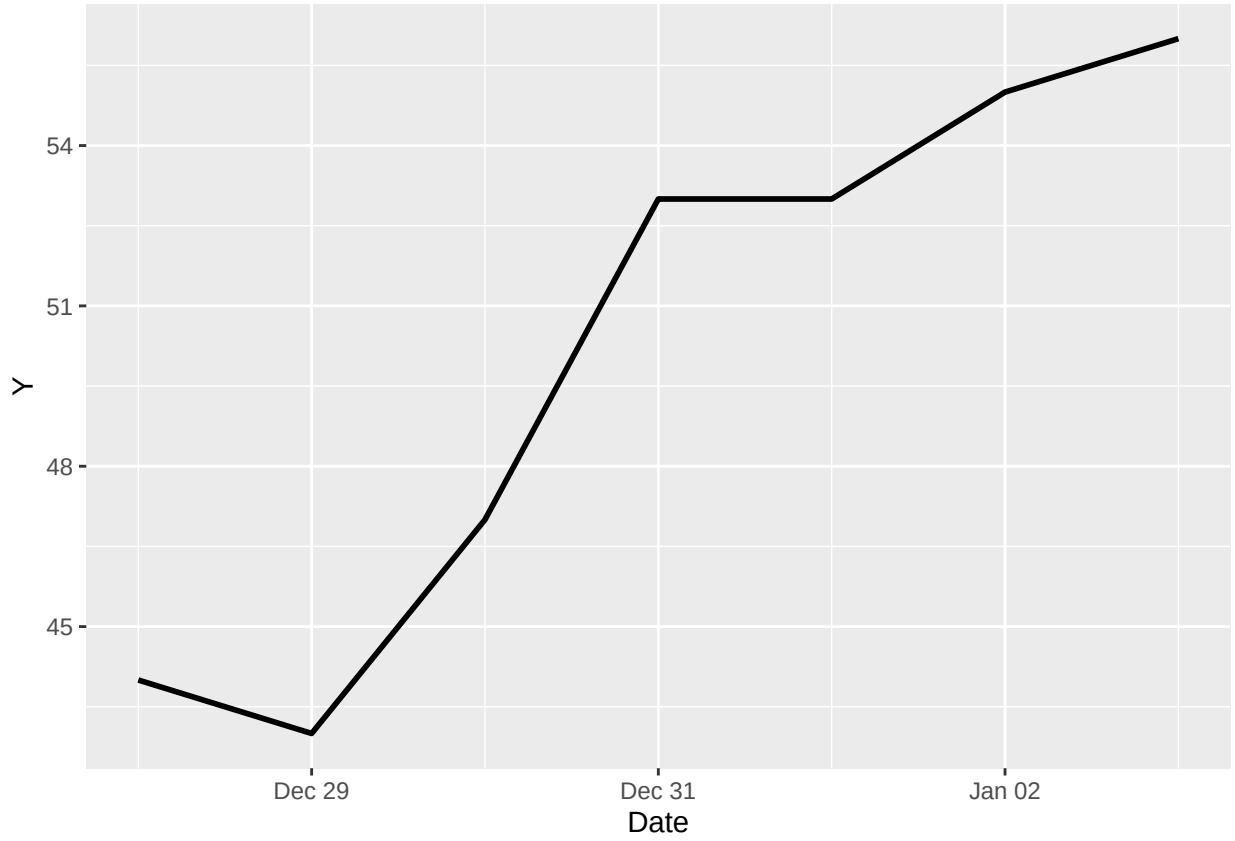


This code doesn't work because geom_line() only registers having one observation, the Y variable. The Date object doesn't work because it isn't considered something other than a character or numeric value.

b) How can you use mutate() (from the "dplyr" package) and mdy() to fix the problem? Do it and report your R commands.
Code:

```
my.data$Date <- mdy(my.data$Date)

ggplot(data = my.data, mapping = aes(Date, Y)) +
  geom_line(size = 1)
```



Section 7.1 Exercises

Exercise 17: Guess how many times "Good Sport" will be printed to the screen in the following set of commands. Then check your answer.

Code:

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

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

The above code will print "Good Sport" 5 times.

Exercise 18: The sequence of values we iterate over doesn't have to be of the form 1:n. Guess what will be printed to the screen in the following set of commands. Then check your answer.

Code:

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

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

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

This is what will show on the console, "4, 16, 36, 64".

Exercise 19: Do the following.

Code:

```
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 sum.sq <- 0 wasn't there? Try it (after removing sum.sq from your Workspace if it's there).
The statement sum.sq needs to be initialized because it is on the right side of the statement which is evaluated first so if its not there the computer doesn't know the variable sum.sq.
Code:

```
#rm(sum.sq)

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

#sum.sq
```

The statement return an error, "Error in sum.sq : object 'sum.sq' not found".

b) What would happen if sum.sq <- 0 was mistakenly placed inside the loop? Try it:
Code:

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

sum.sq
```

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

The following will print out 10^2 which is 100.

Exercise 20: Do the following.

a) What does the following loop do? Code:

```
num.sq <- rep(NA, 10)

for (i in 1:10) {
  num.sq[i] <- i^2
}

num.sq
```

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

The above will show a vector containing the squares of each value for i.

b) What does the following command do?
Code:

```
num.sq <- (1:10)^2
num.sq
```

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

The above code will print a vector containing the squares of numbers 1:10.

Exercise 21: Using the sleepstudy data (from the "lme4" package), use mutate() with nest_by() and lm() to fit lines separately to each Subject, with Days as the x variable and Response as the y variable by typing:

Code:

```
library(lme4) # Contains the sleepstudy data set.
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
by_subject <- nest_by(.data = sleepstudy, Subject)
models <-
  mutate(.data = by_subject, mod = list(lm(Reaction ~ Days, data = data)))
head(models)
```

```
## # 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>
```

What's the equation of the fitted line for Subject 371 (the 17th subject in the study)?

```
models$mod[[17]]
```

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

The equation is $\mathbf{y = 9.188X + 253.636}$.