Lab 2 - Data wrangling

Ayden Frost

2026-02-12

```r
library(tidyverse)
```

Questions

Part 1

Question 1

```r
midwest |>
  count(state, sort = TRUE)
```

```
# A tibble: 5 x 2
  state     n
  <chr> <int>
1 IL      102
2 IN       92
3 OH       88
4 MI       83
5 WI       72
```

Illinois has the highest number of counties with 102, while Wisconsin has the lowest number of counties with only 72.

Question 2

```r
midwest |>
  count(county, state) |>
  count(county, name = "nstates") |>
  filter(nstates == 5)
```

```
# A tibble: 3 x 2
  county   nstates
  <chr>      <int>
1 CRAWFORD       5
2 JACKSON        5
3 MONROE         5
```

Question 3

```r
midwest |>
  filter(popdensity > 25000) |>
  select(county, state, popdensity, poptotal, area) |>
  arrange(desc(popdensity))
```

```
# A tibble: 9 x 5
  county    state popdensity poptotal  area
  <chr>     <chr>      <dbl>    <int> <dbl>
1 COOK      IL        88018.  5105067 0.058
2 MILWAUKEE WI        63952.   959275 0.015
3 WAYNE     MI        60334.  2111687 0.035
4 CUYAHOGA  OH        54313.  1412140 0.026
5 DU PAGE   IL        39083.   781666 0.02
6 MARION    IN        34659.   797159 0.023
7 HAMILTON  OH        34649.   866228 0.025
8 FRANKLIN  OH        28278.   961437 0.034
9 MACOMB    MI        25621.   717400 0.028
```

```r
midwest |>
  filter(popdensity == max(popdensity)) |>
  select(county, state, popdensity, poptotal, area)
```

```
# A tibble: 1 x 5
  county state popdensity poptotal  area
  <chr>  <chr>      <dbl>    <int> <dbl>
1 COOK   IL        88018.  5105067 0.058
```

Question 4

```r
midwest |>
  summarize(
    median(popdensity),
    q1 = quantile(popdensity, 0.25),
    q3 = quantile(popdensity, 0.75)
  )
```

```
# A tibble: 1 x 3
  `median(popdensity)`    q1    q3
                 <dbl> <dbl> <dbl>
1                1156.  622.  2330
```

The distribution of population density of counties is unimodal and extremely right-skewed. A typical Midwestern county has population density of 1156 people per unit area. The middle 50% of the counties have population densities between 622 to 2330 people per unit area.

Question 5

```r
midwest |>
  count(state, inmetro) |>
  group_by(state) |>
  mutate(prop = n / sum(n))
```

```
# A tibble: 10 x 4
# Groups:   state [5]
   state inmetro     n  prop
   <chr>   <int> <int> <dbl>
 1 IL          0    74 0.725
 2 IL          1    28 0.275
 3 IN          0    55 0.598
 4 IN          1    37 0.402
 5 MI          0    58 0.699
 6 MI          1    25 0.301
 7 OH          0    48 0.545
 8 OH          1    40 0.455
 9 WI          0    52 0.722
10 WI          1    20 0.278
```

Question 6

```
midwest |>
  filter(percbelowpoverty >= 40,
         percollege <= 10) |>
  select(county, state,
         percbelowpoverty,
         percollege)
```

```
# A tibble: 1 x 4
  county    state percbelowpoverty percollege
  <chr>     <chr>            <dbl>      <dbl>
1 MENOMINEE WI                48.7       7.34
```

```r
midwest |>
  filter(percollege >= 40,
         percbelowpoverty <= 20) |>
  select(county, state,
         percbelowpoverty,
         percollege)
```

```
# A tibble: 5 x 4
  county    state percbelowpoverty percollege
  <chr>     <chr>            <dbl>      <dbl>
1 CHAMPAIGN IL               15.6        41.3
2 DU PAGE   IL                2.71       42.8
3 HAMILTON  IN                3.59       42.1
4 WASHTENAW MI               12.2        48.1
5 DANE      WI               10.5        43.6
```

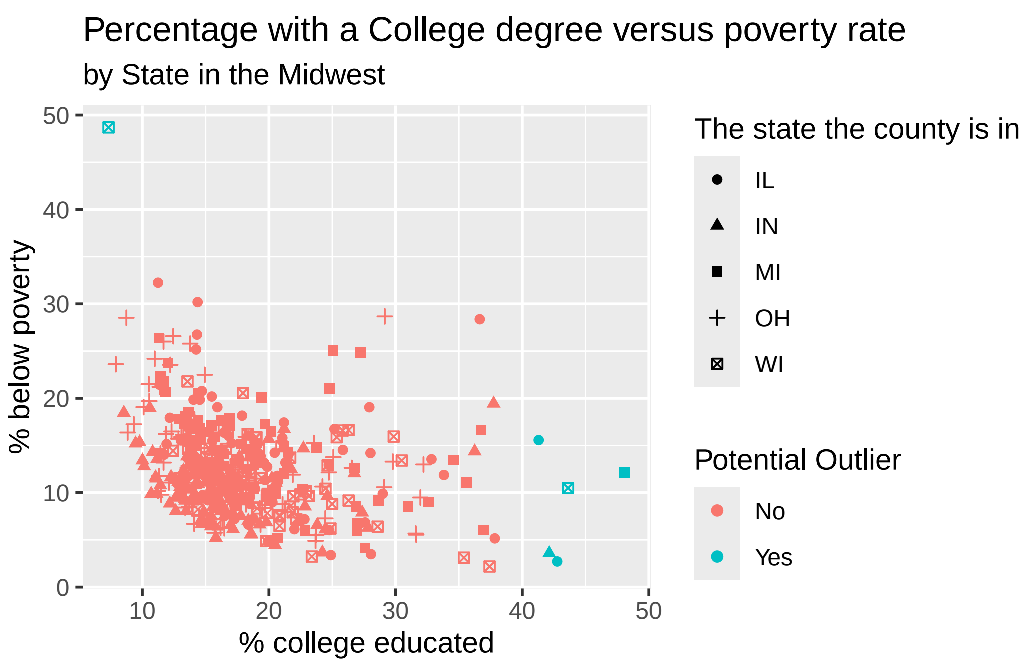
```r
midwest |>
  filter(
    (percbelowpoverty >= 40 & percollege <= 10) |
    (percbelowpoverty <= 20 & percollege >= 40)
  ) |>
  select(county, state,
         percbelowpoverty,
         percollege)
```

```
# A tibble: 6 x 4
  county    state percbelowpoverty percollege
  <chr>     <chr>            <dbl>      <dbl>
1 CHAMPAIGN IL               15.6       41.3
2 DU PAGE   IL                2.71      42.8
3 HAMILTON  IN                3.59      42.1
4 WASHTENAW MI               12.2       48.1
5 DANE      WI               10.5       43.6
6 MENOMINEE WI               48.7        7.34
```

```r
midwest |>
  mutate(
    potential_outlier = if_else(
    (percbelowpoverty >= 40 & percollege <= 10) |
    (percbelowpoverty <= 20 & percollege >= 40),
      "Yes",
      "No"
    )
  ) |>
  select(county, state,
         percbelowpoverty,
         percollege,
         potential_outlier) |>
  arrange(potential_outlier)
```

```
# A tibble: 437 x 5
   county    state percbelowpoverty percollege potential_outlier
   <chr>     <chr>            <dbl>      <dbl> <chr>
 1 ADAMS     IL               13.2        19.6 No
 2 ALEXANDER IL               32.2        11.2 No
 3 BOND      IL               12.1        17.0 No
 4 BOONE     IL                7.21       17.3 No
 5 BROWN     IL               13.5        14.5 No
 6 BUREAU    IL               10.4        18.9 No
 7 CALHOUN   IL               15.1        11.9 No
 8 CARROLL   IL               11.7        16.2 No
 9 CASS      IL               13.9        14.1 No
10 CHRISTIAN IL               11.7        13.6 No
# i 427 more rows
```

```
midwest |>
  mutate(
  potential_outlier = if_else (
(percbelowpoverty >= 40 & percollege <= 10) |
(percbelowpoverty <= 20 & percollege >= 40),
"Yes",
"No")) |>
ggplot (aes (x = percollege, y = percbelowpoverty, colour = potential_outlier, shape = state
geom_point() +
  labs (
  x= "% college educated",
  y = "% below poverty",
  colour = "Potential Outlier",
  shape = "The state the county is in",
  title = "Percentage with a College degree versus poverty rate",
  subtitle = "by State in the Midwest")
```



Question 7

```r
  state_population <- midwest |>
  group_by(state) |>
  summarize(total_population = sum(poptotal)) |>
  arrange(desc(total_population))
state_population
```

```
# A tibble: 5 x 2
  state total_population
  <chr>            <int>
1 IL            11430602
2 OH            10847115
3 MI             9295297
4 IN             5544159
5 WI             4891769
```

```r
state_population |>
  mutate(
    population_prop = total_population / sum(total_population)
  ) |>
  arrange(desc(population_prop))
```

```
# A tibble: 5 x 3
  state total_population population_prop
  <chr>            <int>           <dbl>
1 IL            11430602           0.272
2 OH            10847115           0.258
3 MI             9295297           0.221
4 IN             5544159           0.132
5 WI             4891769           0.116
```

Illinois is the most populated with 27% of the Midwestern population living there.

Question 8

```
state_poverty <- midwest |>
  group_by(state) |>
  summarise(mean_percbelowpoverty = mean(percbelowpoverty))
state_poverty
```

```
# A tibble: 5 x 2
  state mean_percbelowpoverty
  <chr>                 <dbl>
1 IL                     13.1
2 IN                     10.3
3 MI                     14.2
4 OH                     13.0
5 WI                     11.9
```

```r
state_poverty |>
  arrange(mean_percbelowpoverty)
```

```
# A tibble: 5 x 2
  state mean_percbelowpoverty
  <chr>                 <dbl>
1 IN                     10.3
2 WI                     11.9
3 OH                     13.0
4 IL                     13.1
5 MI                     14.2
```

Indiana has the lowest average percentage below poverty across its counties, while Michigan has the highest average percentage below poverty.

Part 2

Question 9

```r
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)

df
```

```
# A tibble: 5 x 3
  var_1 var_2  var_3
  <dbl> <chr>  <chr>
1    10 Pizza  Apple
2    20 Burger Apple
3    30 Pizza  Pear
4    40 Pizza  Pear
5    50 Burger Banana
```

a.)

```r
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)

df |>
  arrange(var_2)
```

```
# A tibble: 5 x 3
  var_1 var_2  var_3
  <dbl> <chr>  <chr>
1    20 Burger Apple
2    50 Burger Banana
3    10 Pizza  Apple
4    30 Pizza  Pear
5    40 Pizza  Pear
```

The code changed the order of the var_2 column from "Pizza, burger, pizza, pizza, burger" to "Burger, burger, pizza, pizza, pizza". In short, the arrange sorted the rows in variable 2 by ascending alphabetical or numeric order.

b.)

```r
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)

df |>
  group_by(var_2)
```

```
# A tibble: 5 x 3
# Groups:   var_2 [2]
  var_1 var_2  var_3
  <dbl> <chr>  <chr>
1    10 Pizza  Apple
2    20 Burger Apple
3    30 Pizza  Pear
4    40 Pizza  Pear
5    50 Burger Banana
```

The "group_by" didn't seem to have an effect on variable 2 as the values seemed to reflect those found in the original tibble table. The "group_by" grouped the data set based on the values within the variable 2 row, but left all the rows and other variables unchanged. When comparing "arrange" and "group_by", we can see that "group_by" only effects the data groupings while leaving everything else the same compared to the "arrange" function which simply rearranges the rows.

c.)

```r
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)

df |>
  group_by(var_2) |>
  summarize(mean_var_1 = mean(var_1))
```

```
# A tibble: 2 x 2
  var_2  mean_var_1
  <chr>       <dbl>
1 Burger       35
2 Pizza        26.7
```

The new code groups the data based on the values within the variable 2 row with "group_by", then forms 2 rows based on the mean values of variable 1 found within each new grouping using "summarize".

d.)

```
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)

df |>
  group_by(var_2, var_3) |>
  summarize(mean_var_1 = mean(var_1))
```

```
`summarise()` has grouped output by 'var_2'. You can override using the
`.groups` argument.
```

```
# A tibble: 4 x 3
# Groups:   var_2 [2]
  var_2  var_3  mean_var_1
  <chr>  <chr>       <dbl>
1 Burger Apple          20
2 Burger Banana         50
3 Pizza  Apple          10
4 Pizza  Pear           35
```

The code created groupings based on variables 2 and 3 using "group_by", then found the mean value of variable 1 found within each new grouping using "summarize". The message states that the summarize is based on the 2nd variable and to change that you can use the ".groups" argument.

e.)

```r
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)
df |>
  group_by(var_2, var_3) |>
  summarize(mean_var_1 = mean(var_1), .groups = "drop")
```

```
# A tibble: 4 x 3
  var_2  var_3  mean_var_1
  <chr>  <chr>       <dbl>
1 Burger Apple          20
2 Burger Banana         50
3 Pizza  Apple          10
4 Pizza  Pear           35
```

Everything in the code and output is the same as part d apart from the ".groups" which ungroups the data after the "summarize" has been completed.

f.)

```r
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)
df |>
  group_by(var_2, var_3) |>
  summarize(mean_var_1 = mean(var_1), .groups = "drop")
```

```
# A tibble: 4 x 3
  var_2  var_3  mean_var_1
  <chr>  <chr>       <dbl>
1 Burger Apple          20
2 Burger Banana         50
3 Pizza  Apple          10
4 Pizza  Pear           35
```

```r
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana")
)
df |>
  group_by(var_2, var_3) |>
  mutate(mean_var_1 = mean(var_1))
```

```
# A tibble: 5 x 4
# Groups:   var_2, var_3 [4]
  var_1 var_2  var_3  mean_var_1
  <dbl> <chr>  <chr>       <dbl>
1    10 Pizza  Apple          10
2    20 Burger Apple          20
3    30 Pizza  Pear           35
4    40 Pizza  Pear           35
5    50 Burger Banana         50
```

The first pipeline made groupings based on variables 2 and 3, then found the mean values of variable 1 within each new grouping using the "summarize" argument. The ".groups" made it so that there were no groupings after the "summarize" was complete. The second pipeline still grouped by variables 2 and 3 but instead made a new column that shows the mean values of variable 1 within each new grouping using the “mutate” argument. All groupings remained and a new column was created.