Lab 2 - Data wrangling

Sheila Yu

2026-02-04

```
library(tidyverse)
data <- midwest
```

Questions

Part 1

Question 1

```
counties_by_state <- data %>%
  count(state, name = "n_counties") %>%
  arrange(desc(n_counties))

counties_by_state
```

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

The code above displays calculation the number of counties in each state. Based on the calculation, Illinois (IL) has the highest number of counties, with a total of 102 counties. Wisconsin (WI) has the lowest number of counties, with a total of 72 counties.

Question 2

```r
data %>%
  count(county, name = "n_states") %>%
  filter(n_states == n_distinct(midwest$state))
```

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

Based on the result above, the county that fills in the blank of "Look at that, there is a county called _____ in each state in this dataset!" includes Crawford, Jackson and Monroe.

Question 3

Part a:

```
data %>%
  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
```

Based on the results above, there are a total of nine counties with a population density above 25,000. These counties include: Cook, Milwaukee, Wayne, Cuyahoga, Du Page, Marion, Hamilton, Franklin and Macomb.

Part b:

```
data %>%
  filter(popdensity == max(popdensity, na.rm = TRUE)) %>%
  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
```

Based on the result above, the county with the highest population density is Cook.

Question 4

```r
data %>%
  summarise(
    median_density = median(popdensity, na.rm = TRUE),
    q1 = quantile(popdensity, 0.25, na.rm = TRUE),
    q3 = quantile(popdensity, 0.75, na.rm = TRUE)
  )
```

```
# A tibble: 1 x 3
  median_density    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.

*Please note that the data are rounded to the nearest whole number.

Question 5

```
data %>%
  mutate(metro = if_else(inmetro == 1, "Yes", "No")) %>%
  count(state, metro) %>%
  group_by(state) %>%
  mutate(proportion = n / sum(n)) %>%
  ungroup()
```

```
# A tibble: 10 x 4
   state metro     n proportion
   <chr> <chr> <int>      <dbl>
 1 IL    No       74      0.725
 2 IL    Yes      28      0.275
 3 IN    No       55      0.598
 4 IN    Yes      37      0.402
 5 MI    No       58      0.699
 6 MI    Yes      25      0.301
 7 OH    No       48      0.545
 8 OH    Yes      40      0.455
 9 WI    No       52      0.722
10 WI    Yes      20      0.278
```

Here is a table with results from the calculation above displayed and rounded of to nearest percentage:

State	Metro or not	Proportion of population in the area
IL	No	73%
IL	Yes	27%
IN	No	60%
IN	Yes	40%
MI	No	70%
MI	Yes	30%
OH	No	55%
OH	Yes	45%
WI	No	72%
WI	Yes	28%

Question 6

Part a:

```
data %>%
  filter(
    percollege < 12,
    percbelowpoverty > 45
  ) %>%
  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
```

The outlier in the orange square is the county Menominee.

Part b:

```
data %>%
  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
```

The outliers in the red square are: Champaign, Du Page, Hamilton, Washtenaw, Dane.

Part c:

```r
data %>%
  filter(
    (percollege < 12 & percbelowpoverty > 45) |
    (percollege > 40 & percbelowpoverty < 20)
  ) %>%
  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
```

Above is a list of all the outliers in the orange and red square combined.

Part d:

```r
midwest <- midwest %>%
  mutate(
    potential_outlier = if_else(
      (percollege < 12 & percbelowpoverty > 45) |
      (percollege > 40 & percbelowpoverty < 20),
      "Yes",
      "No"
    )
  )

midwest %>%
  select(county, state, percbelowpoverty, percollege, potential_outlier) %>%
  arrange(potential_outlier)
```

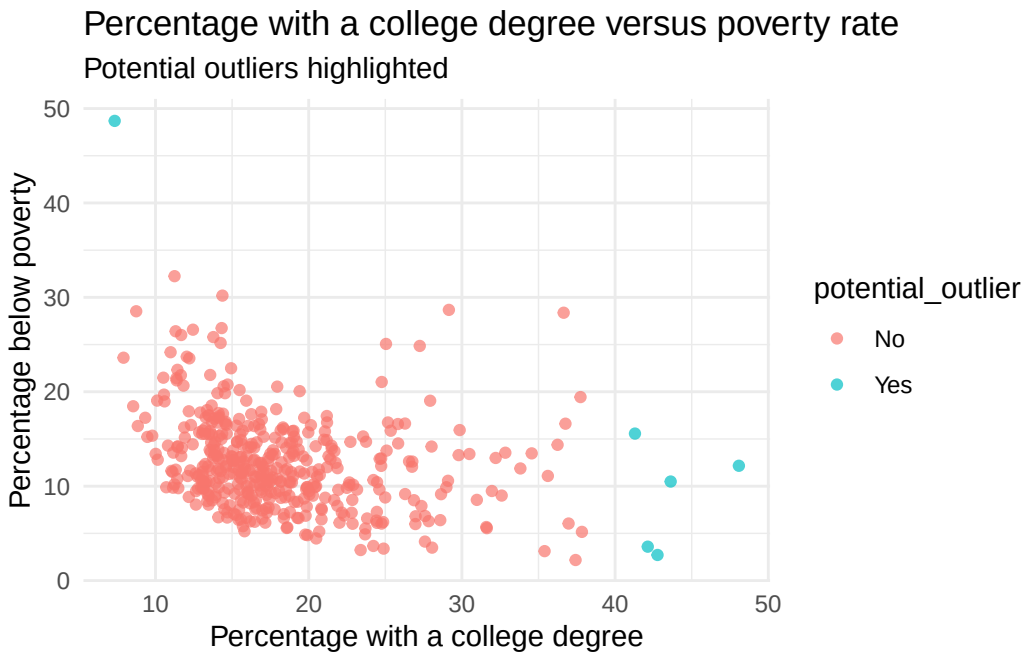
```
# A tibble: 437 x 5
   county    state percbelowpoverty percollege potential_outlier
   <chr>     <chr>            <dbl>      <dbl> <chr>
 1 ADAMS     IL               13.2        19.6 No
 2 ALEXANDER IL               32.2        11.2 No
 3 BOND      IL               12.1        17.0 No
```

```
 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
```

The updated dataset, *midwest*, now contains an additional column called potential_outlier, which indicates whether a county is included in the list of potential outliers identified in part c.

Part e:

```r
ggplot(midwest, aes(x = percollege, y = percbelowpoverty, color = potential_outlier)) +
  geom_point(alpha = 0.7) +
  labs(
    x = "Percentage with a college degree",
    y = "Percentage below poverty",
    title = "Percentage with a college degree versus poverty rate",
    subtitle = "Potential outliers highlighted"
  ) +
  theme_minimal()
```



Above is the visualization of the relationship between college education and poverty, with colour highlighting the potential outliers in blue.

Question 7

Part a:

```
state_population <- midwest %>%
  group_by(state) %>%
  summarise(total_population = sum(poptotal, na.rm = TRUE)) %>%
  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
```

Part b:

```
state_population %>%
  mutate(
    population_proportion = total_population / sum(total_population)
  ) %>%
  arrange(desc(population_proportion))
```

```
# A tibble: 5 x 3
  state total_population population_proportion
  <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
```

Part c: Illinois is the most populous Midwestern state, containing 27% of the total Midwest population. Wisconsin is the least populous state, accounting for only 12% of the Midwest population.

Question 8

```r
state_poverty <- midwest %>%
  group_by(state) %>%
  summarise(
    mean_percbelowpoverty = mean(percbelowpoverty, na.rm = TRUE)
  )

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
```

Based on the result above, Wisconsin has the lowest average percentage of population (10%) below the poverty line across its counties. Mississippi has the highest average percentage of population (14%) below the poverty line across its counties.

Part 2

Setting up data frame:

```
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
```

Question 9

Part a:

```
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 arrange () function reorders the rows of the data frame alphabetically by data in the column var_2.

Part b:

```r
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 () function groups the data frame by the variable var_2. It doesn't change the appearance of the data set, but it does createsgroups internally so that subsequent operations are performed separately for each group.

Part c:

```r
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 pipeline first groups the data by var_2, then calculates the mean of var_1 within each group that is created. Therefore it computes the mean of var_1 separately for each category in var_2.

Part d:

```r
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
```

This pipeline groups the data frame by both var_2 and var_3, and then calculates the mean of var_1 within each unique combination of var_2 and var_3. The message displays:

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

It indicates that after applying the summarize () function, the resulting data frame remains grouped by var_2. The message also notes that this behavior can be overridden by specifying the .groups argument, if an ungrouped output is desired.

Part e:

```
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
```

This pipeline repeats the same grouped mean calculation as part d, but the .groups = “drop” argument tells summarize () to remove all grouping from the output, thereby un-grouping the output. Compared to the output in part d, the output in part 3 results in tibbles that is not grouped. No message (as shown in part d) is showing, suggesting that the output is a regular ungrouped data frame. However, it is also worth to note that the rows and values are the same in both part d and e.

Part f:

```r
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 |>
  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
```

Pipeline 1 collapses the data and returns one row per group (var_2, var_3) with the mean of var_1. On the other hand, pipeline 2 does not collapse the data. Instead, it keeps all original rows, but adds a new column (mean_var_1) on top of existing rows where every row in the same group gets the same group mean value.