Week 4 Exercises

Sue Susman MEd, BSN, RN

April 9, 2023

Table of Contents

Exercise 1:

Examine the who and population data sets that come with the tidyr library. the who data is not tidy, you will need to reshape the new_sp_m014 to newrel_f65 columns to long format retaining country, iso2, iso3, and year. The data in the columns you are reshaping contains patterns described in the details section below. You will need to assign three columns: diagnosis, gender, and age to the patterns described in the details.

```
Your tidy data should look like the following:
  country     iso2  iso3   year diagnosis gender age   count
  <chr>       <chr> <chr> <int> <chr>     <chr>  <chr> <int>
1 Afghanistan AF    AFG    1980 sp        m      014      NA
2 Afghanistan AF    AFG    1980 sp        m      1524     NA
3 Afghanistan AF    AFG    1980 sp        m      2534     NA
4 Afghanistan AF    AFG    1980 sp        m      3544     NA
5 Afghanistan AF    AFG    1980 sp        m      4554     NA
6 Afghanistan AF    AFG    1980 sp        m      5564     NA
```

Details:
The data uses the original codes given by the World Health Organization. The column names for columns five through 60 are made by combining new_ to a code for method of:

- diagnosis (rel = relapse, sn = negative pulmonary smear, sp = positive pulmonary smear, ep = extrapulmonary)
- gender (f = female, m = male)
- age group (014 = 0-14 yrs of age, 1524 = 15-24 years of age, 2534 = 25 to 34 years of age, 3544 = 35 to 44 years of age, 4554 = 45 to 54 years of age, 5564 = 55 to 64 years of age, 65 = 65 years of age or older).

Note: use data(who) and data(population) to load the data into your environment. Use the arguments cols, names_to, names_pattern, and values_to. Your regex should be = ("new_?(.)_(.)(.)")
https://tidyr.tidyverse.org/reference/who.html

```
library(dplyr)
library(tidyr)
library(ggplot2)

# load the data
data(who)
data(population)

# reshape the data to long format
who_tidy <- who %>%
  pivot_longer(cols = starts_with("new_"),
            names_to = c("diagnosis", "gender", "age"),
            names_pattern = "new_?(.*)_(.)(.*)",
            values_to = "count") %>%
  select(country, iso2, iso3, year, diagnosis, gender, age, count)
```

Exercise 2:

There are two common keys between the data sets, with who as the left table, join the population data by country and year so that the population is available within the who dataset.

```
library(dplyr)

# load the data
data(who)
data(population)

# join the population data to the who data by country and year
who_population <- left_join(who, population, by = c("country", "year"))

# view the first six rows of the joined data
head(who_population)
## # A tibble: 6 × 61
##   country     iso2  iso3   year new_sp…¹ new_s…² new_s…³ new_s…⁴ new_s…⁵ new_s…⁶
##   <chr>       <chr> <chr> <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Afghanistan AF    AFG    1980       NA      NA      NA      NA      NA      NA
## 2 Afghanistan AF    AFG    1981       NA      NA      NA      NA      NA      NA
## 3 Afghanistan AF    AFG    1982       NA      NA      NA      NA      NA      NA
## 4 Afghanistan AF    AFG    1983       NA      NA      NA      NA      NA      NA
## 5 Afghanistan AF    AFG    1984       NA      NA      NA      NA      NA      NA
## 6 Afghanistan AF    AFG    1985       NA      NA      NA      NA      NA      NA
## # … with 51 more variables: new_sp_m65 <dbl>, new_sp_f014 <dbl>,
## #   new_sp_f1524 <dbl>, new_sp_f2534 <dbl>, new_sp_f3544 <dbl>,
## #   new_sp_f4554 <dbl>, new_sp_f5564 <dbl>, new_sp_f65 <dbl>,
## #   new_sn_m014 <dbl>, new_sn_m1524 <dbl>, new_sn_m2534 <dbl>,
## #   new_sn_m3544 <dbl>, new_sn_m4554 <dbl>, new_sn_m5564 <dbl>,
## #   new_sn_m65 <dbl>, new_sn_f014 <dbl>, new_sn_f1524 <dbl>,
## #   new_sn_f2534 <dbl>, new_sn_f3544 <dbl>, new_sn_f4554 <dbl>, …
```

Exercise 3:

Split the age column into two columns, min age and max age. Notice that there is no character separator. Check the documentation with ?separate to understand other ways to separate the age column. Keep in mind that 0 to 14 is coded as 014 (3 characters) and the other age groups are coded with 4 characters. 65 only has two characters, but we will ignore that until the next problem.

```
who_tidy <- who_tidy %>%
  mutate(min_age = case_when(
    age == "014" ~ 0,
    TRUE ~ as.numeric(substring(age, 1, 2))
  ),
  max_age = case_when(
    age == "014" ~ 14,
    age == "65" ~ 65,
    TRUE ~ as.numeric(substring(age, 3, 4))
  )) %>%
  select(-age)
```

Exercise 4:

Since we ignored the 65+ group in the previous problem we will fix it here. If you examine the data you will notice that 65 was placed into the max_age column and there is no value for min_age for those records. To fix this use mutate() in order to replace the blank value in the min_age column with the value from the max_age column and another mutate to replace the 65 in the max column with an Inf. Be sure to keep the variables as character vectors.

```
library(dplyr)

library(dplyr)

who_tidy <- who_tidy %>%
 mutate(max_age = ifelse(max_age == "65", "Inf", max_age))
```

Exercise 5:

Find the count per diagnosis for males and females.
See ?sum for a hint on resolving NA values.

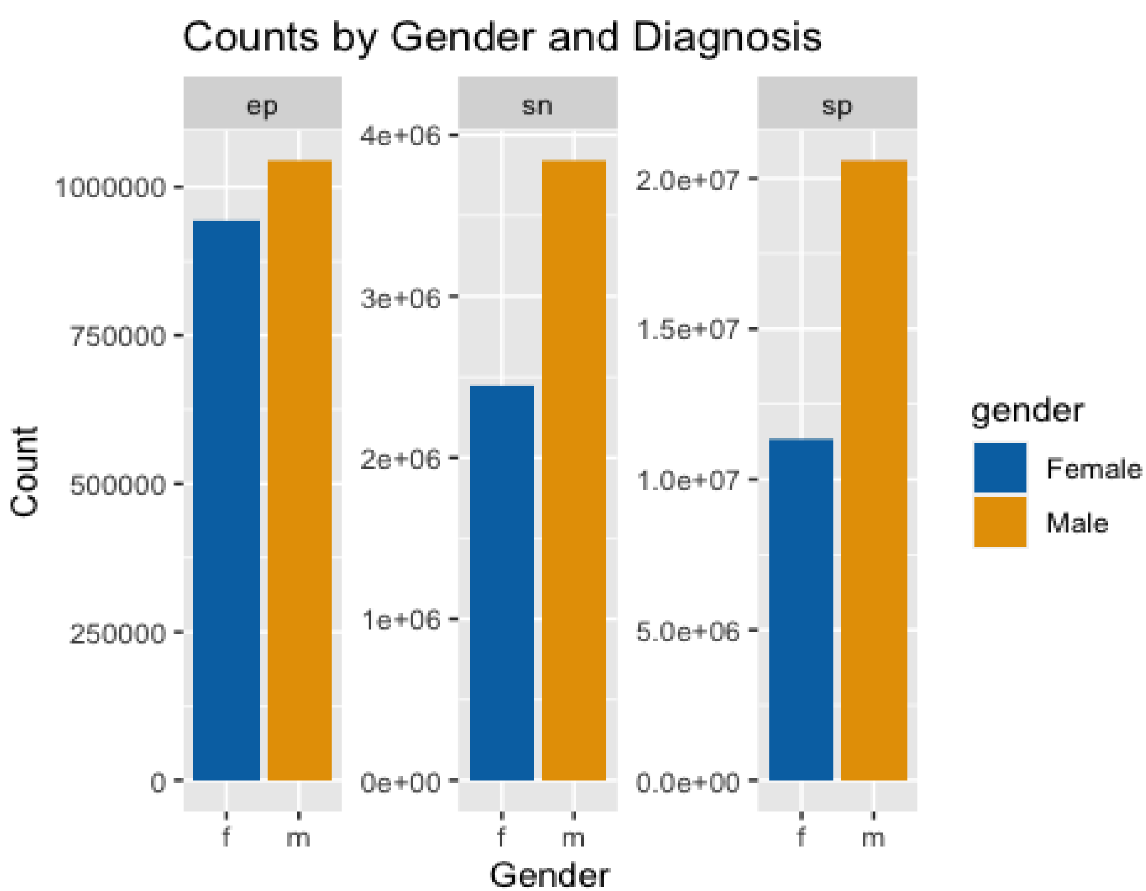
```
library(dplyr)

who_tidy %>%
 group_by(diagnosis, gender) %>%
 summarize(total_count = sum(count))
## `summarise()` has grouped output by 'diagnosis'. You can override using the
## `.groups` argument.
## # A tibble: 6 × 3
## # Groups:   diagnosis [3]
##   diagnosis gender total_count
##   <chr>     <chr>        <dbl>
## 1 ep        f               NA
## 2 ep        m               NA
## 3 sn        f               NA
## 4 sn        m               NA
## 5 sp        f               NA
## 6 sp        m               NA
 na.omit(who_tidy)
## # A tibble: 73,186 × 9
##    country     iso2  iso3   year diagnosis gender count min_age max_age
```

```
##    <chr>      <chr> <chr> <dbl> <chr>    <chr>  <dbl>   <dbl> <chr>
##  1 Afghanistan AF    AFG    1997 sp       m        0      0 14
##  2 Afghanistan AF    AFG    1997 sp       m       10      15 24
##  3 Afghanistan AF    AFG    1997 sp       m        6      25 34
##  4 Afghanistan AF    AFG    1997 sp       m        3      35 44
##  5 Afghanistan AF    AFG    1997 sp       m        5      45 54
##  6 Afghanistan AF    AFG    1997 sp       m        2      55 64
##  7 Afghanistan AF    AFG    1997 sp       m        0      65 Inf
##  8 Afghanistan AF    AFG    1997 sp       f        5      0 14
##  9 Afghanistan AF    AFG    1997 sp       f       38      15 24
## 10 Afghanistan AF    AFG    1997 sp       f       36      25 34
## # ... with 73,176 more rows
```

Exercise 6:

Now create a plot using ggplot and geom_col where your x axis is gender, your y axis represents the counts, and facet by diagnosis. Be sure to give your plot a title and resolve the axis labels.

```
library(ggplot2)

# create the plot
ggplot(who_tidy, aes(x = gender, y = count, fill = gender)) +
  geom_col() +
  facet_wrap(~ diagnosis, scales = "free_y") +
  labs(title = "Counts by Gender and Diagnosis", x = "Gender", y = "Count") +
  scale_fill_manual(values = c("#0072B2", "#E69F00"), labels = c("Female", "Male"))
options(scipen = 999)
```



Exercise 7:

Find the percentage of population by year, gender, and diagnosis. Be sure to remove rows containing NA values.

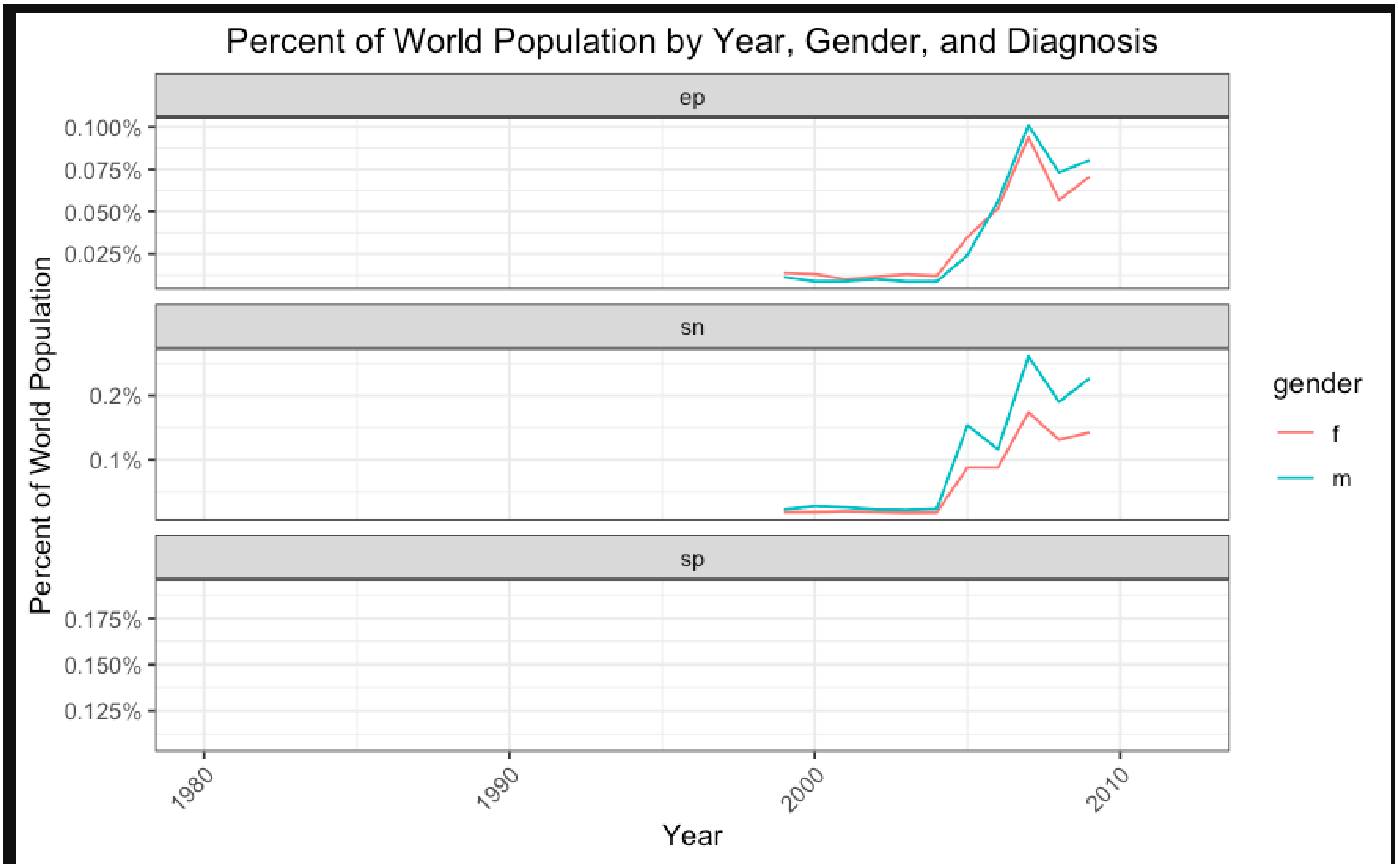
```
library(tidyr)
library(dplyr)

# join the who_tidy and population data sets
who_pop <- left_join(who_tidy, population, by = c("country", "year"))

# calculate the percentage of the population by year, gender, and diagnosis
who_pop %>%
  drop_na() %>%   # remove rows with NAs
  group_by(year, gender, diagnosis) %>%
  summarize(pct_pop = sum(count) / sum(population) * 100)
## `summarise()` has grouped output by 'year', 'gender'. You can override using
## the `.groups` argument.
## # A tibble: 92 × 4
## # Groups:   year, gender [36]
##     year gender diagnosis  pct_pop
##    <dbl> <chr>  <chr>        <dbl>
##  1  1995 f      sp        0.000574
##  2  1995 m      sp        0.000982
##  3  1996 f      sp        0.000663
##  4  1996 m      sp        0.00115
##  5  1997 f      sp        0.000737
##  6  1997 m      sp        0.00131
##  7  1998 f      sp        0.000807
##  8  1998 m      sp        0.00139
##  9  1999 f      ep        0.000138
## 10  1999 f      sn        0.000188
## # … with 82 more rows
```

Exercise 8:

Create a line plot in ggplot where your x axis contains the year and y axis contains the percent of world population. Facet this plot by diagnosis with each plot stacked vertically. You should have a line for each gender within each facet. Be sure to format your y axis and give your plot a title.

```
library(ggplot2)

# Filter out NA rows
who_tidy_no_na <- who_tidy[complete.cases(who_tidy),]

# Convert count and population to numeric
who_pop$count <- as.numeric(who_pop$count)
who_pop$population <- as.numeric(who_pop$population)

# Group by year, diagnosis, and gender and calculate the percentage of the population
pct_pop <- who_pop %>%
  group_by(year, diagnosis, gender) %>%
  summarize(pct_pop = sum(count) / sum(population) * 100, .groups = 'drop')

# Create the line plot
```

```
ggplot(pct_pop, aes(x = year, y = pct_pop, group = gender)) +
  geom_line(aes(color = gender)) +
  facet_wrap(~diagnosis, ncol = 1, scales = "free_y") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(title = "Percent of World Population by Year, Gender, and Diagnosis",
       x = "Year",
       y = "Percent of World Population") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1))
## Warning: Removed 204 rows containing missing values (`geom_line()`).
```



Exercise 9:

Now unite the min and max age variables into a new variable named age_range. Use a '-' as the separator.

```
library(tidyr)

who_tidy <- who_tidy %>%
  unite(age_range, min_age, max_age, sep = '-')
```

Exercise 10:

Find the percentage contribution of each age group by diagnosis. You will first need to find the count of all diagnoses then find the count of all diagnoses by age group. Join the former to the later and calculate the percent of each age group. Plot these as a geom_col where the x axis is the diagnosis, y axis is the percent of total, and faceted by age group.

```r
library(dplyr)
library(ggplot2)

library(dplyr)
library(ggplot2)

# Calculate count of all diagnoses by age group
nn <- who_tidy %>%
  count(age_range, diagnosis)

# Calculate count of diagnoses by age group and diagnosis
n_by_age <- who_tidy %>%
  count(age_range, diagnosis) %>%
  group_by(age_range) %>%
  mutate(pct_of_total = n / sum(n) * 100)

# Create plot
ggplot(n_by_age, aes(x = diagnosis, y = pct_of_total, fill = diagnosis)) +
  geom_col() +
  facet_wrap(~ age_range, scales = "free_y") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(title = "Percentage of Diagnoses by Age Range",
      x = "Diagnosis",
      y = "Percent of Total") +
  theme_bw()
```