Week 4 Exercises

Michael Durning

April 9, 2023

Please complete all exercises below. You may use any library that we have covered in class. The data we will be using comes from the tidyr package, so you must use that.

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 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) to a code for gender (f = female, m = male) to a code for 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(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## 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
```

```
# Load the data
data(who)
```

```r
# Reshape the data to long format
who_tidy_long <- who %>%
  select(country, iso2, iso3, year, new_sp_m014:newrel_f65) %>%
  pivot_longer(cols = new_sp_m014:newrel_f65,
               names_to = c("diagnosis", "gender", "age"),
               values_to = c("count"),
               names_pattern = ("new_?(.*)_(.)(.*)")
  )

head(who_tidy_long)
```

```
## # A tibble: 6 x 8
##   country     iso2  iso3   year diagnosis gender age   count
##   <chr>       <chr> <chr> <dbl> <chr>     <chr>  <chr> <dbl>
## 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
```

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.

```r
library(tidyr)
library(dplyr)

# Load the data
data(who)
data(population)

# Left join the two data sets
who_population_join <- left_join(who_tidy_long,
                                 population,
                                 by = c("country", "year")
                                 )

head(who_population_join)
```

```
## # A tibble: 6 x 9
##   country     iso2  iso3   year diagnosis gender age   count population
##   <chr>       <chr> <chr> <dbl> <chr>     <chr>  <chr> <dbl>      <dbl>
## 1 Afghanistan AF    AFG    1980 sp        m      014      NA         NA
## 2 Afghanistan AF    AFG    1980 sp        m      1524     NA         NA
## 3 Afghanistan AF    AFG    1980 sp        m      2534     NA         NA
## 4 Afghanistan AF    AFG    1980 sp        m      3544     NA         NA
## 5 Afghanistan AF    AFG    1980 sp        m      4554     NA         NA
## 6 Afghanistan AF    AFG    1980 sp        m      5564     NA         NA
```

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 prolem.

```r
library(tidyr)
library(dplyr)
```

```
who_population_age <- who_population_join

# Split from -2, start from end of string
age_column_split <- separate(who_population_age,
                             age,
                             into = c("min_age", "max_age"),
                             sep = -2,)
```

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(tidyr)
library(dplyr)

# If max_age = 65, change min age to max age
age_column_split <- mutate(age_column_split,
             min_age = ifelse(max_age == 65, max_age, min_age))

# If min_age = 65, change max_age to "Inf"
age_column_split <- mutate(age_column_split,
             max_age = ifelse(min_age == 65, "Inf", max_age))
```

5) Find the count per diagnosis for males and females.

See ?sum for a hint on resolving NA values.

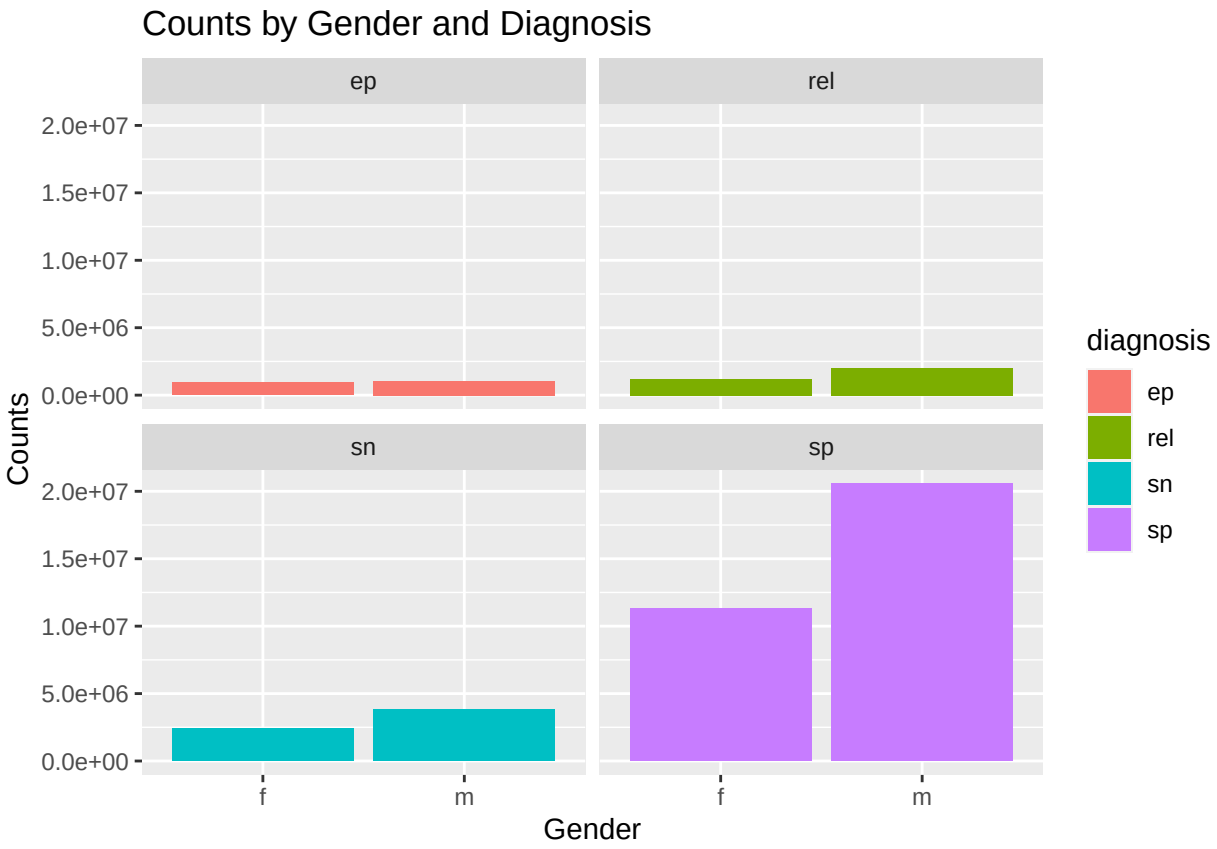
```
# Find count per m/f diagnosis
diagnosis_sum <- aggregate(count ~ diagnosis + gender,
                           data = age_column_split,
                           FUN = function(x) sum(x, na.rm = TRUE))
```

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)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
# create plot
ggplot(diagnosis_sum,
      aes(x = gender, y = count, fill = diagnosis)) +
      geom_col(position = "dodge") +
      facet_wrap(~ diagnosis) +
      ggtitle("Counts by Gender and Diagnosis") +
      xlab("Gender") +
      ylab("Counts")
```



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

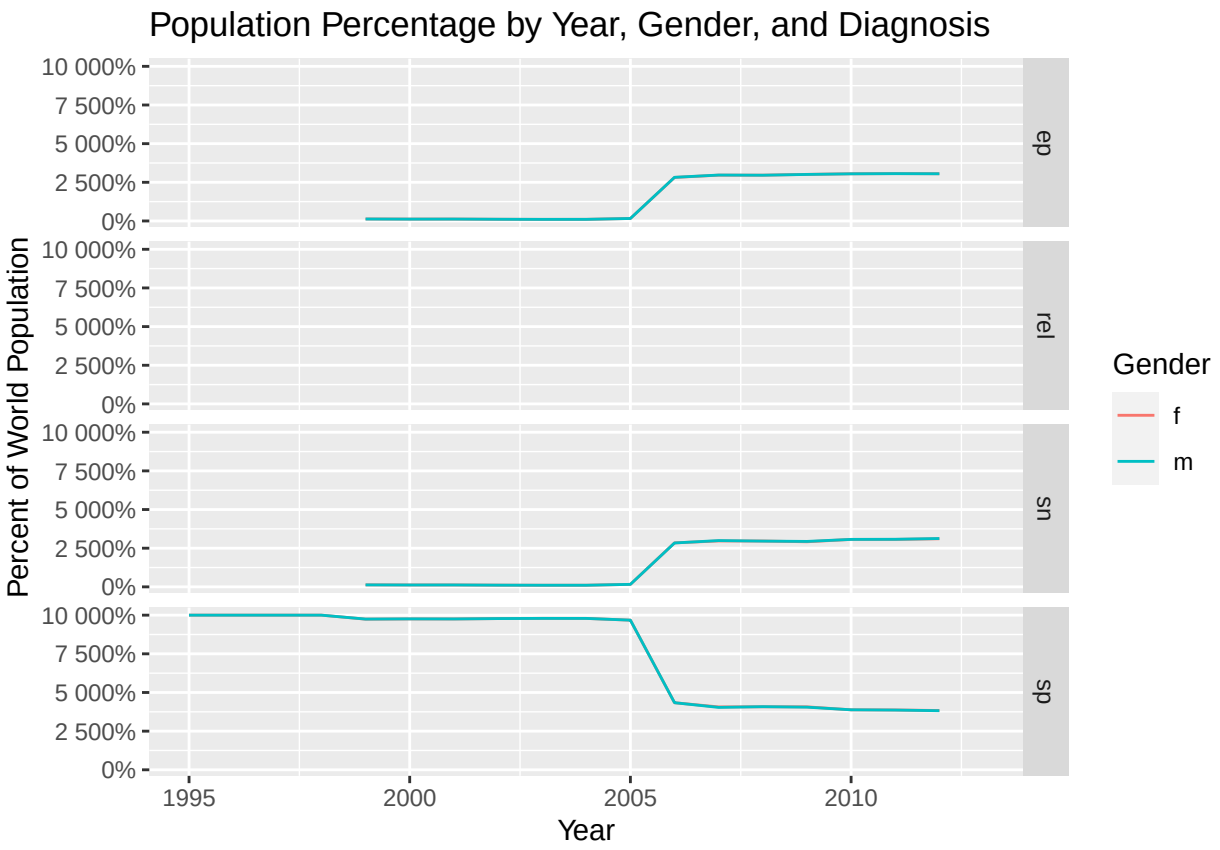
```
# Change dataframe name to allow for more readability
cleaned_data <- age_column_split


# Remove rows containing NA, then group and find percentages
percent_pop <- cleaned_data %>%
  na.omit() %>%
  group_by(year, gender, diagnosis) %>%
  summarize(population = n()) %>%
  mutate(percentage = population / sum(population) * 100)
```

```
## `summarise()` has grouped output by 'year', 'gender'. You can override using
## the `.groups` argument.
```

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.

```
# Create line plot
ggplot(percent_pop, aes(x = year, y = percentage, color = gender)) +
  geom_line() +
  facet_grid(rows = vars(diagnosis)) +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(x = "Year", y = "Percent of World Population", color = "Gender",
       title = "Population Percentage by Year, Gender, and Diagnosis")
```

```
## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



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

```r
# Combine the data min and max into age-range
cleaned_data <- cleaned_data %>%
  unite(age_range, c(min_age, max_age), sep = "-")
```

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
# count of all diagnoses
diag_count <- cleaned_data %>%
  count(diagnosis)

# count of age groups and their diagnosis
age_group_count <- cleaned_data %>%
  count(diagnosis, age_range)

# Join both above data frames
joined_data <- left_join(age_group_count, diag_count, by = "diagnosis") %>%
  rename(age_group_count = n.x, diag_count = n.y)

# Add a new column to calculate percentage contribution
```

```r
percentage_data <- joined_data %>%
  mutate(percent = age_group_count / diag_count * 100)

# Create a faceted bar plot
ggplot(percentage_data, aes(x = diagnosis, y = percent, fill = age_range)) +
  geom_col(position = "fill") +
  facet_grid(. ~ age_range) +
  labs(x = "Diagnosis", y = "Percent of Total", fill = "Age Range",
       title = "Percentage Contribution of Each Age Group by Diagnosis")
```