Week 4 Exercises

Joseph Annand

July 30, 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(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
```

```
library(tidyr)
library(ggplot2)
library(stringr)
library(scales)

#your code here
data(who)
data(population)

who <- who %>%
  pivot_longer(cols = matches("new_?(.*)_(.)(.*)"),
               names_to = c("diagnosis", "gender", "age"),
               names_pattern = "new_?(.*)_(.)(.*)",
               values_to = "count")
```

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.

```
# your code here
who <- left_join(who, population, by=c("country", "year"))
```

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.

```
# your code here
who <- who %>%
  mutate(age = str_replace(age, "014", "0014")) %>%
  separate(col = "age",
           into = c("min_age", "max_age"),
           sep = 2) %>%
  mutate(min_age = str_replace(min_age, "00", "0"))
```

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.

```
# your code here
who <- who %>%
  mutate(max_age = replace(max_age, max_age=="", as.character(Inf)))
```

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

See ?sum for a hint on resolving NA values.

```
# your code here
who %>%
  select(c("diagnosis", "gender", "count")) %>%
  group_by(gender,diagnosis) %>%
  summarise(diagnosis_count=sum(count, na.rm = TRUE))
```

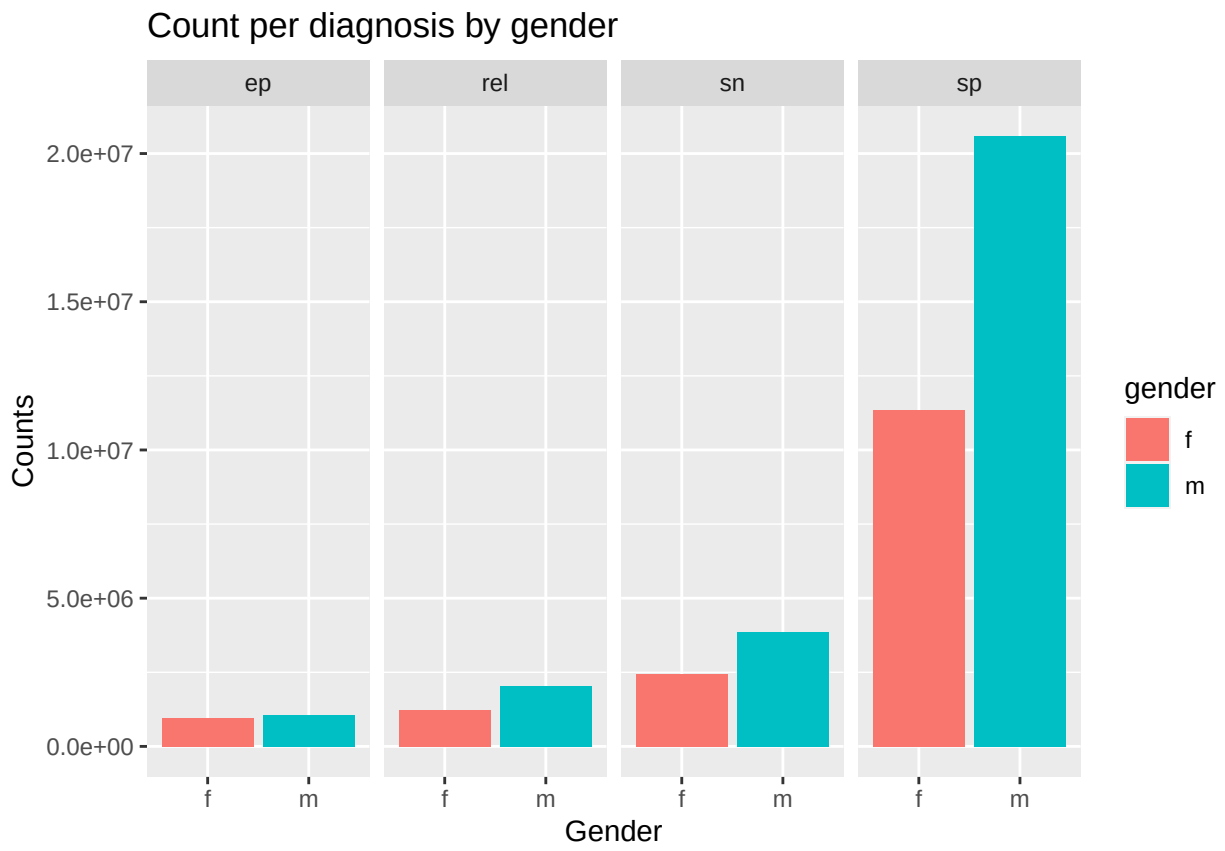
```
## `summarise()` has grouped output by 'gender'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 8 x 3
## # Groups:   gender [2]
##   gender diagnosis diagnosis_count
##   <chr>  <chr>               <dbl>
## 1 f      ep                941880
## 2 f      rel              1201596
## 3 f      sn               2439139
## 4 f      sp              11324409
## 5 m      ep               1044299
## 6 m      rel              2018976
## 7 m      sn               3840388
## 8 m      sp              20586831
```

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.

```
#your code here
ggplot(data = who,
       mapping = aes(x=gender, y=count)) +
  geom_col(aes(fill=gender)) +
  labs(x='Gender',
       y='Counts',
       title="Count per diagnosis by gender") +
  facet_grid(.~diagnosis)
```

```
## Warning: Removed 329394 rows containing missing values (`position_stack()`).
```



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

```
# your code here
pop_year_data <- who %>%
  select(c("population","year","gender","diagnosis","count")) %>%
  group_by(year,gender,diagnosis) %>%
  drop_na() %>%
  summarise(percent_pop = 100 * (sum(count) / mean(population)))
```

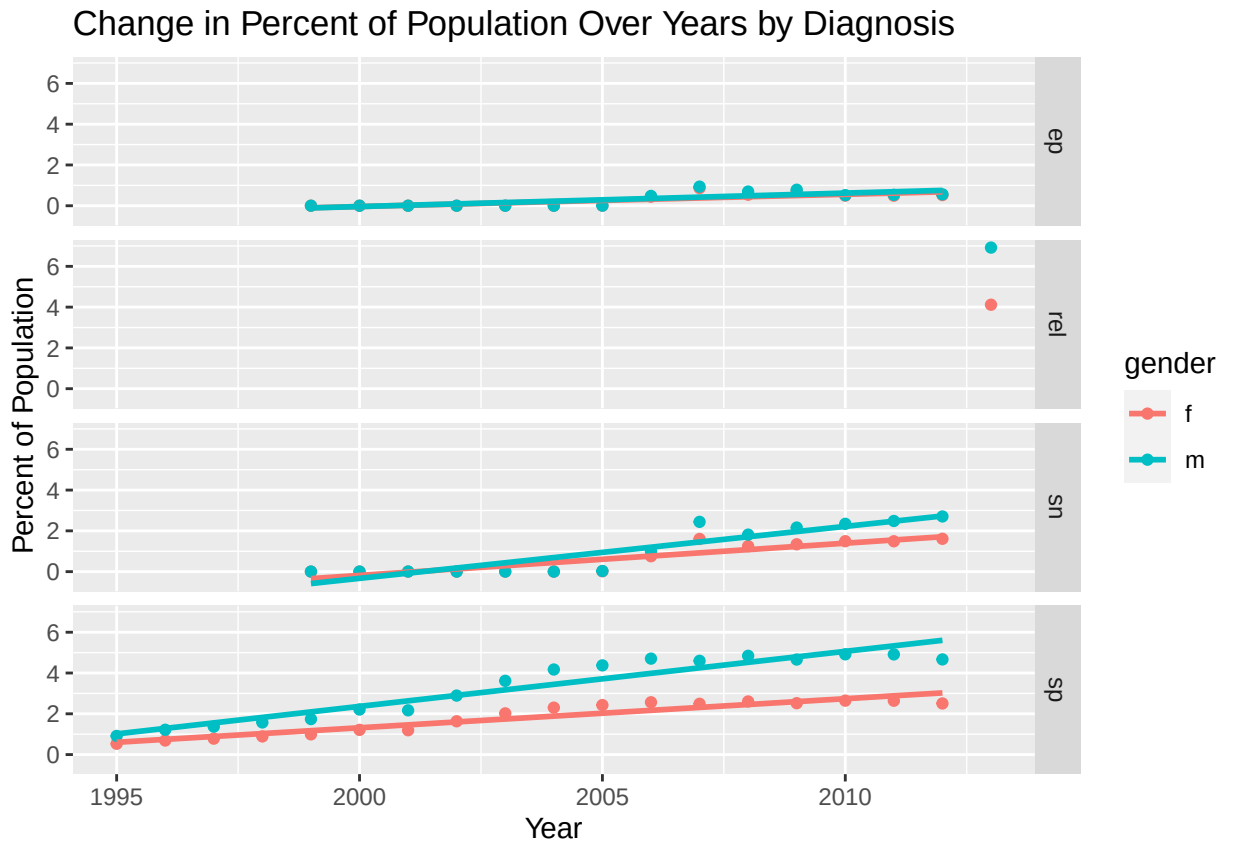
```
## `summarise()` has grouped output by 'year', 'gender'. You can override using
## the `.groups` argument.
```

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.

```
# your code here
ggplot(data = pop_year_data,
       mapping = aes(x=year, y=percent_pop, col=gender)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  facet_grid(diagnosis~.) +
  xlab("Year") +
```

```
  ylab("Percent of Population") +
  labs(title = "Change in Percent of Population Over Years by Diagnosis")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



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

```
# your code here
who <- who %>%
  unite(col = 'age_range', 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.

```
# your code here
all_diagnoses <- who %>%
  select(c("gender", "diagnosis", "age_range", "count")) %>%
  group_by(diagnosis) %>%
  summarise(total = sum(count, na.rm = TRUE))
```

```
diagnosis_by_age <- who %>%
  select(c("gender", "diagnosis", "age_range", "count")) %>%
  group_by(diagnosis, age_range) %>%
  summarise(total_by_age = sum(count, na.rm = TRUE))
```

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

```
age_diagnosis_data <- left_join(diagnosis_by_age, all_diagnoses)
```

```
## Joining with `by = join_by(diagnosis)`
```

```
age_diagnosis_data <- mutate(age_diagnosis_data, percent_of_total = 100 * (total_by_age / total))

ggplot(data = age_diagnosis_data,
       mapping = aes(x=diagnosis, y=percent_of_total)) +
  geom_col(aes(fill=diagnosis)) +
  labs(x='Diagnosis',
       y='Percent of Total',
       title="Percent of Total Diagnoses by Age Group") +
  facet_grid(.~age_range)
```