Week 4 Exercises

James Clark

April 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)

#your code here
who_df<- pivot_longer(who,
    cols = !c(1:4), #keep these columns
    names_to = c("diagnosis", "gender", "age"), #names of new columns
    names_pattern = "new_?(.*)_(.)(.*)", #pattern to split old column names by to
     ↪  populate new columns, (.*)=match 0 or more times
    values_to = "count" #name of values column
)
```

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_pop_df<-who_df %>%
   left_join(population,by=c("country","year")) #left join to keep all who_df data but
    ↪  add population data
```

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_pop_df<-who_pop_df %>%
  separate(age,c("min_age","max_age"),sep=-2) #sep=-2 means to take last two digits and
   ↪  populate into max_age, put remaining in min_age
```

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_pop_df<-who_pop_df %>%

   ↪  mutate(min_age=replace(min_age,min_age=="",65),max_age=replace(max_age,max_age==65,Inf))
   ↪  #overwrites blanks in min_age to 65 and overwrites 65 in max_age to Inf
```

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

See ?sum for a hint on resolving NA values.

```
# your code here
who_pop_df %>% group_by(gender, diagnosis)%>% #group by gender and diagnosis
  summarize(frequency=sum(count,na.rm=TRUE)) #find total count of same diagnosis and same
   ↪  gender
```

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

```
## # A tibble: 8 x 3
## # Groups:   gender [2]
##   gender diagnosis frequency
##   <chr>  <chr>         <int>
## 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
```

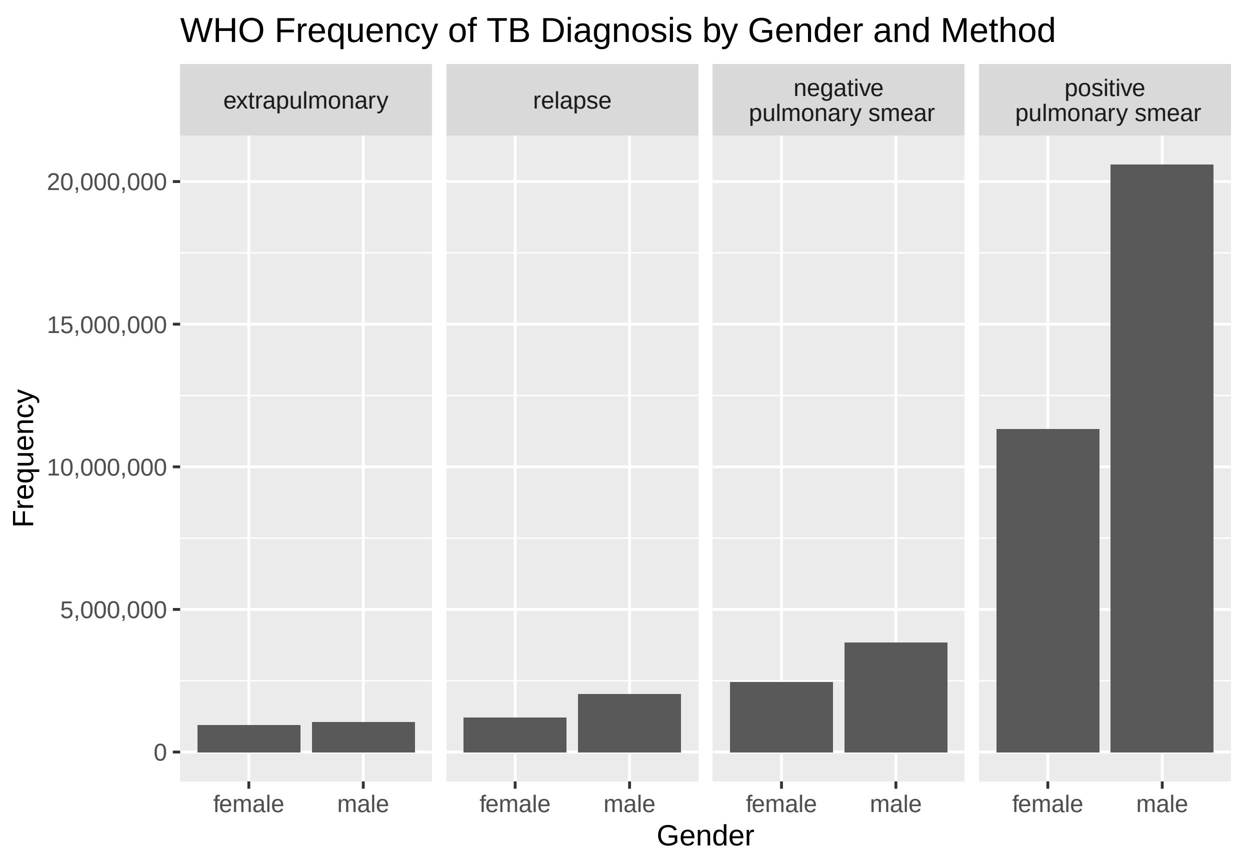
6) Now create a plot using ggplot and geom_col where your x axis is gender, your y axis represents the counts, and facet by diagnosis. Be sure to give your plot a title and resolve the axis labels.

```
#your code here

# ggplot(who_df)+
#   geom_col(aes(x=gender, y=count, fill=diagnosis) #bar chart by gender and diagnosis
#            ,position='dodge')+ #puts bars side-by-side instead of stacked
#   labs(title='WHO Frequency of TB Diagnosis by Gender and Method',
#        x="Gender",
#        y="Frequency"
#        )+ #label title and axes
#   scale_x_discrete(labels=c("m"="male","f"="female"))+ #change labels of m and f to
↪  what they stand for
#   scale_y_continuous(labels = scales::comma)+ #change y-axis from scientific notation
↪  to comma notation
#   scale_fill_discrete(labels = c("rel" = "relapse",
#                                  "sn" = "neg. pulmonary smear",
#                                  "sp" = "pos. pulmonary smear",
#                                  "ep" = "extrapulmonary")
#                       ) #change key values from symbols to name of what they stand for

ggplot(who_df)+
  geom_col(aes(x=gender, y=count))+ #bar chart of gender and count
  labs(title='WHO Frequency of TB Diagnosis by Gender and Method',
       x="Gender",
       y="Frequency"
       )+ #label title and axes
  scale_x_discrete(labels=c("m"="male","f"="female"))+ #change labels of m and f to what
  ↪  they stand for
  scale_y_continuous(labels = scales::comma)+ #change y-axis from scientific notation to
  ↪  comma notation
  facet_grid(.~diagnosis, #facet by diagnosis
             labeller = as_labeller(
               c("rel" = "relapse",
                 "sn" = "negative\n pulmonary smear",
                 "sp" = "positive\n pulmonary smear",
                 "ep" = "extrapulmonary")
               )#change key values from symbols to name of what they stand for
             )
```

```
## 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

percentage <- who_pop_df %>%
  drop_na()%>% #drop any rows with NA values
  group_by(year, gender, diagnosis) %>% #group data by year, gender, and diagnosis
  summarise(percentage=sum(count)/sum(population)) #find sum of count by year, gender,
  ↪  and diagnosis over all locations; divide result by population over all locations;
  ↪  summarise this in column called "percentage"
```

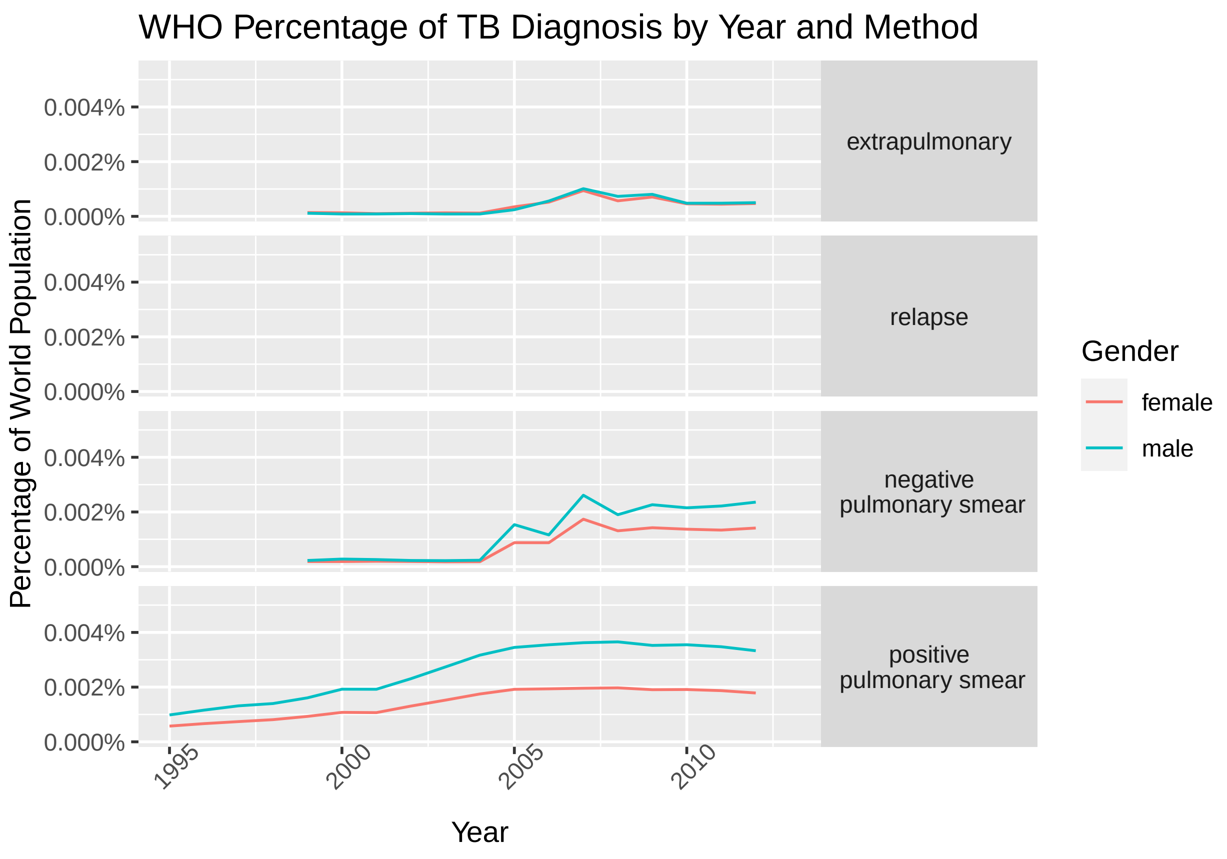
```
## `summarise()` has grouped output by 'year', 'gender'. You can override using
## the `.groups` argument.
```

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(percentage)+
  geom_line(aes(x=year, y=percentage,color=gender))+ #create a line plot of percentage by
  ↪  year, color individual lines by gender
  labs(title='WHO Percentage of TB Diagnosis by Year and Method',
       x="Year",
       y="Percentage of World Population",
       color="Gender"
       )+ #label title and axes and key
  scale_color_discrete(labels=c("m"="male", "f"="female"))+ #change labels of m and f to
  ↪  what they stand for
  scale_y_continuous(labels = scales::percent)+ #change y-axis from decimal to percentage
  facet_grid(rows=vars(diagnosis), #facet by diagnosis vertically
             labeller = as_labeller(c("rel" = "relapse",
                                      "sn" = "negative\n pulmonary smear",
                                      "sp" = "positive\n pulmonary smear",
                                      "ep" = "extrapulmonary")
                                    ) #change facet values from symbols to name of what
↪  they stand for
             )+
  theme(axis.text.x = element_text(angle = 45), #rotate x-axis values by 45 degrees
        strip.text.y.right = element_text(angle = 0) #make facet titles horizontal
        )
```

```
## `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.

```
# your code here
who_pop_df<- who_pop_df %>%
  unite(min_age,max_age,col="age_range",sep="-") #combine min_age column and max_age
  ↪  column, name new column age_range, separate values from two old columns by '-'
```

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.

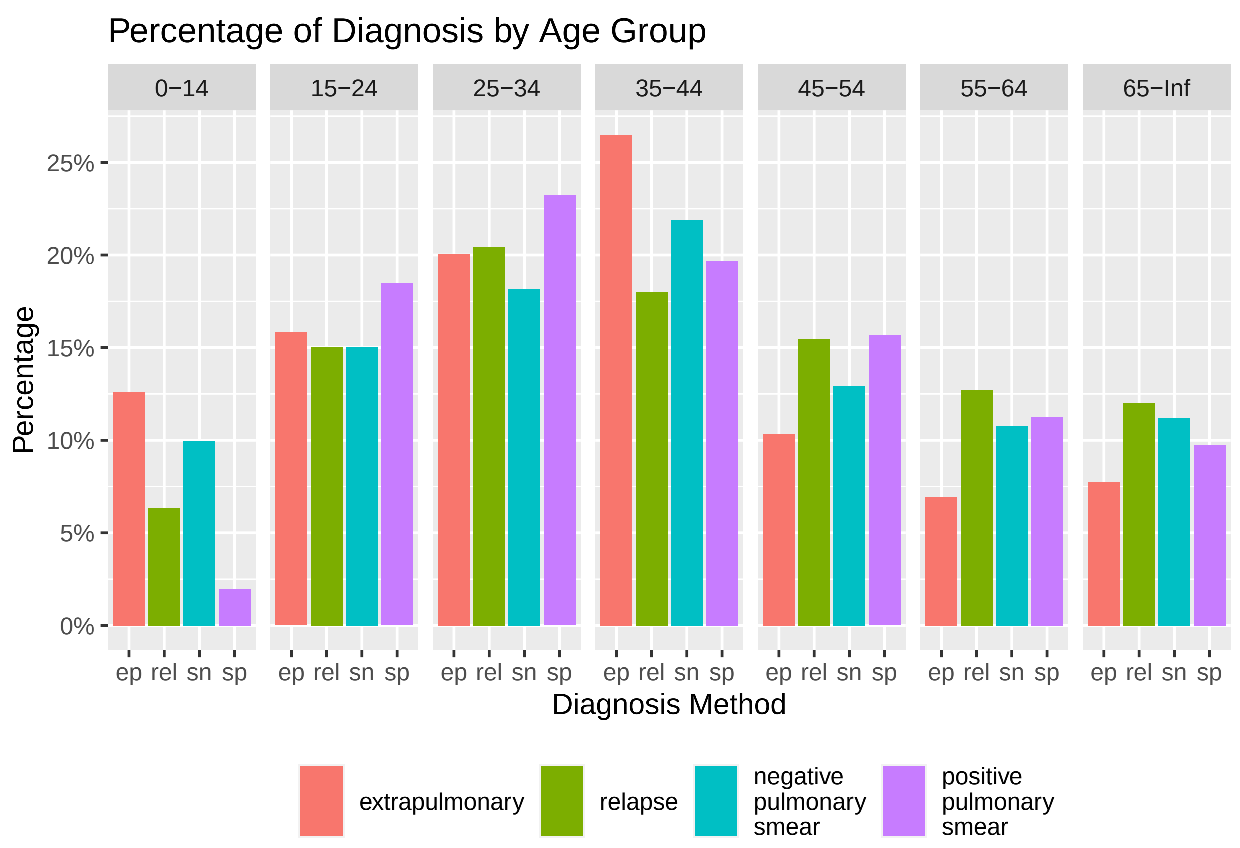
```r
# your code here

diagnosis_freq<-who_pop_df %>%
  drop_na() %>% #drop NA values
  group_by(diagnosis) %>% #group by diagnosis
  summarise(diag_freq=sum(count)) #find count of all diagnoses

age_diagnosis_data<- who_pop_df %>%
  drop_na() %>% #drop NA values
  group_by(diagnosis,age_range) %>% #group by age range and diagnosis
  summarise(age_diag_freq=sum(count)) %>% #find count of all diagnoses by age group
  left_join(diagnosis_freq) %>% # join former to the later
  mutate(percentage=age_diag_freq/diag_freq) # add column that calculates percentage
```

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

```r
ggplot(as.data.frame(age_diagnosis_data))+ #make data a dataframe
  geom_col(aes(x=diagnosis, y=percentage,fill=diagnosis))+ #bar chart of diagnosis and
  ↪  percentage
  #make diagnosis fill as well to create legend on what the abbreviations mean
  labs(title='Percentage of Diagnosis by Age Group',
       x="Diagnosis Method",
       y="Percentage",
       fill="",
       )+ #label title and axes, leave title for legend blank
  scale_fill_discrete(labels=c("rel" = "relapse",
                      "sn" = "negative\npulmonary\nsmear",
                      "sp" = "positive\npulmonary\nsmear",
                      "ep" = "extrapulmonary")
                    )+#make a key for fill values to name of what they stand for
  scale_y_continuous(labels = scales::percent, #change y-axis to percentages
                     breaks=seq(0,1,.05))+ #change y-axis ticks to appear every 5%
  facet_grid(.~age_range)+ #facet by age range
  #facet_grid(rows=vars(age_range))+ #facet by age range vertically, this did not look as
  ↪  nice
  theme(legend.direction = "horizontal", legend.position = "bottom") #move legend to
  ↪  bottom of graph and make horizontal
```


Percentage of Diagnosis by Age Group
0−14
15−24
25−34
35−44
45−54
55−64
65−Inf
Percentage
25%
20%
15%
10%
5%
0%
ep rel sn sp
Diagnosis Method
extrapulmonary
relapse
negative pulmonary smear
positive pulmonary smear