Week 4 Exercises

Wanida Ruangsiriluk

November 17, 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)

data(who)
data(population)

who <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = c('diagnostic', 'gender', 'age'),
    names_pattern = "new_?(.*)_(.)(.*)",
    values_to = 'count'
  )

tail(who)
```

```
## # A tibble: 6 x 8
##   country  iso2  iso3   year diagnostic gender age   count
##   <chr>    <chr> <chr> <dbl> <chr>      <chr>  <chr> <dbl>
## 1 Zimbabwe ZW    ZWE    2013 rel        f      1524   2069
## 2 Zimbabwe ZW    ZWE    2013 rel        f      2534   4649
## 3 Zimbabwe ZW    ZWE    2013 rel        f      3544   3526
## 4 Zimbabwe ZW    ZWE    2013 rel        f      4554   1453
## 5 Zimbabwe ZW    ZWE    2013 rel        f      5564    811
## 6 Zimbabwe ZW    ZWE    2013 rel        f      65      725
```

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.

```
who <- who %>%
  left_join(population, by= c('country', 'year'))

tail(who)
```

```
## # A tibble: 6 x 9
##   country  iso2  iso3   year diagnostic gender age   count population
##   <chr>    <chr> <chr> <dbl> <chr>      <chr>  <chr> <dbl>      <dbl>
## 1 Zimbabwe ZW    ZWE    2013 rel        f      1524   2069   14149648
## 2 Zimbabwe ZW    ZWE    2013 rel        f      2534   4649   14149648
## 3 Zimbabwe ZW    ZWE    2013 rel        f      3544   3526   14149648
## 4 Zimbabwe ZW    ZWE    2013 rel        f      4554   1453   14149648
## 5 Zimbabwe ZW    ZWE    2013 rel        f      5564    811   14149648
## 6 Zimbabwe ZW    ZWE    2013 rel        f      65      725   14149648
```

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.

```
who <- separate(who, col=age, into = c('min_age', 'max_age'),
         sep = "(?<=\\d)(?=\\d{2}$)", remove = TRUE,
         convert = FALSE, fill = "warn")
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 57920 rows [7, 14, 21,
## 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112, 119, 126, 133, 140, ...].
```

```
tail(who, n=10)
```

```
## # A tibble: 10 x 10
##    country  iso2  iso3   year diagnostic gender min_age max_age count population
##    <chr>    <chr> <chr> <dbl> <chr>      <chr>  <chr>   <chr>   <dbl>      <dbl>
##  1 Zimbabwe ZW    ZWE    2013 rel        m      45      54       2349   14149648
##  2 Zimbabwe ZW    ZWE    2013 rel        m      55      64       1206   14149648
##  3 Zimbabwe ZW    ZWE    2013 rel        m      65      <NA>     1208   14149648
##  4 Zimbabwe ZW    ZWE    2013 rel        f      0       14       1252   14149648
##  5 Zimbabwe ZW    ZWE    2013 rel        f      15      24       2069   14149648
##  6 Zimbabwe ZW    ZWE    2013 rel        f      25      34       4649   14149648
##  7 Zimbabwe ZW    ZWE    2013 rel        f      35      44       3526   14149648
##  8 Zimbabwe ZW    ZWE    2013 rel        f      45      54       1453   14149648
##  9 Zimbabwe ZW    ZWE    2013 rel        f      55      64        811   14149648
## 10 Zimbabwe ZW    ZWE    2013 rel        f      65      <NA>      725   14149648
```

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.

```
#Note that, the 'looking forward' syntax in exercise 3 already put 65+ age
#in the min_age column, and max as NA.
#Below only mutate to replace max_age from NA to Inf

who <- who %>%
  mutate_at(8, ~replace_na(., 'Inf'))
tail(who, n=10)
```

```
## # A tibble: 10 x 10
##    country  iso2  iso3   year diagnostic gender min_age max_age count population
##    <chr>    <chr> <chr> <dbl> <chr>      <chr>  <chr>   <chr>   <dbl>      <dbl>
##  1 Zimbabwe ZW    ZWE    2013 rel        m      45      54       2349   14149648
##  2 Zimbabwe ZW    ZWE    2013 rel        m      55      64       1206   14149648
##  3 Zimbabwe ZW    ZWE    2013 rel        m      65      Inf      1208   14149648
##  4 Zimbabwe ZW    ZWE    2013 rel        f      0       14       1252   14149648
##  5 Zimbabwe ZW    ZWE    2013 rel        f      15      24       2069   14149648
##  6 Zimbabwe ZW    ZWE    2013 rel        f      25      34       4649   14149648
##  7 Zimbabwe ZW    ZWE    2013 rel        f      35      44       3526   14149648
##  8 Zimbabwe ZW    ZWE    2013 rel        f      45      54       1453   14149648
##  9 Zimbabwe ZW    ZWE    2013 rel        f      55      64        811   14149648
## 10 Zimbabwe ZW    ZWE    2013 rel        f      65      Inf       725   14149648
```

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

See ?sum for a hint on resolving NA values.

```
summary_counts <- who %>%
  group_by(diagnostic, gender) %>%
  summarise(events = sum(count, na.rm = TRUE)
            )
```

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

```
print(summary_counts)
```

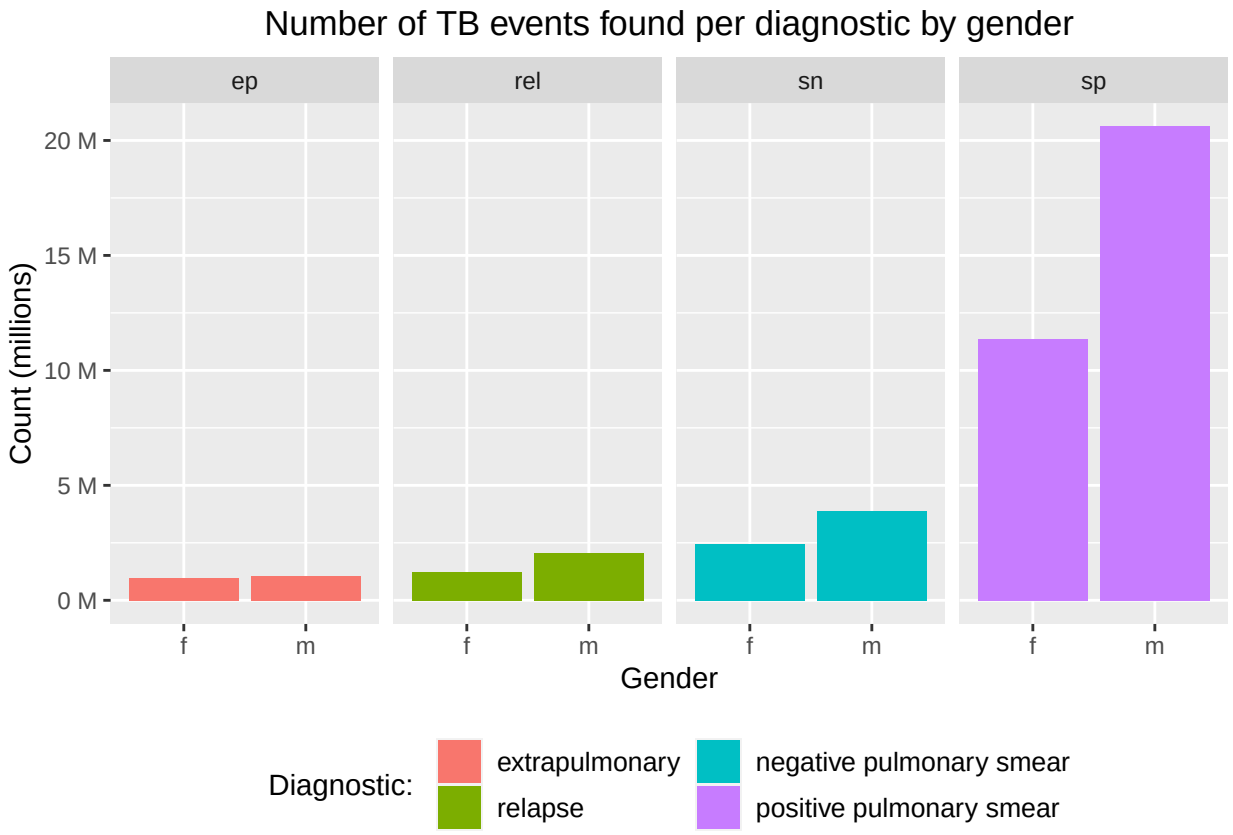
```
## # A tibble: 8 x 3
## # Groups:   diagnostic [4]
##   diagnostic gender   events
##   <chr>      <chr>     <dbl>
## 1 ep         f        941880
## 2 ep         m       1044299
## 3 rel        f       1201596
## 4 rel        m       2018976
## 5 sn         f       2439139
## 6 sn         m    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

```
library(scales)


ggplot(who) +
  geom_col(aes(x=gender,y = count, fill=diagnostic)) +
  labs(x='Gender',
       y='Count (millions)',
       title='Number of TB events found per diagnostic by gender') +
       scale_fill_discrete(name = 'Diagnostic: ',
                    labels = c('extrapulmonary', 'relapse',
                      'negative pulmonary smear', 'positive pulmonary smear')) +
      scale_y_continuous(labels = unit_format(unit = 'M', scale = 1e-6)) +
      facet_grid(.~diagnostic) +
      theme(plot.title = element_text(hjust=0.5), legend.position = 'bottom',
             legend.text = element_text(size = 10)) +
      guides(fill = guide_legend(nrow = 2))
```

```
## 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.

```
# remove NA row, give answer in %

#First, remove NA rows for count and population in a new df called "filter_df"
filter_df <- filter(who, population !=is.na(population), count !=is.na(count))
print(filter_df)
```

```
## # A tibble: 64,420 x 10
##    country  iso2  iso3   year diagnostic gender min_age max_age count population
##    <chr>    <chr> <chr> <dbl> <chr>      <chr>  <chr>   <chr>   <dbl>      <dbl>
##  1 Afghani~ AF    AFG    1997 sp         m      15      24         10   19021226
##  2 Afghani~ AF    AFG    1997 sp         m      25      34          6   19021226
##  3 Afghani~ AF    AFG    1997 sp         m      35      44          3   19021226
##  4 Afghani~ AF    AFG    1997 sp         m      45      54          5   19021226
##  5 Afghani~ AF    AFG    1997 sp         m      55      64          2   19021226
##  6 Afghani~ AF    AFG    1997 sp         f      0       14          5   19021226
##  7 Afghani~ AF    AFG    1997 sp         f      15      24         38   19021226
##  8 Afghani~ AF    AFG    1997 sp         f      25      34         36   19021226
##  9 Afghani~ AF    AFG    1997 sp         f      35      44         14   19021226
## 10 Afghani~ AF    AFG    1997 sp         f      45      54          8   19021226
## # i 64,410 more rows
```

```r
pct_df <- filter_df %>%
  group_by(year, gender, diagnostic) %>%
  summarise(ratio_pop = sum(count / population)) %>%
  mutate(pct_pop_format = percent(ratio_pop, scale = 100, accuracy = 0.001))
```

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

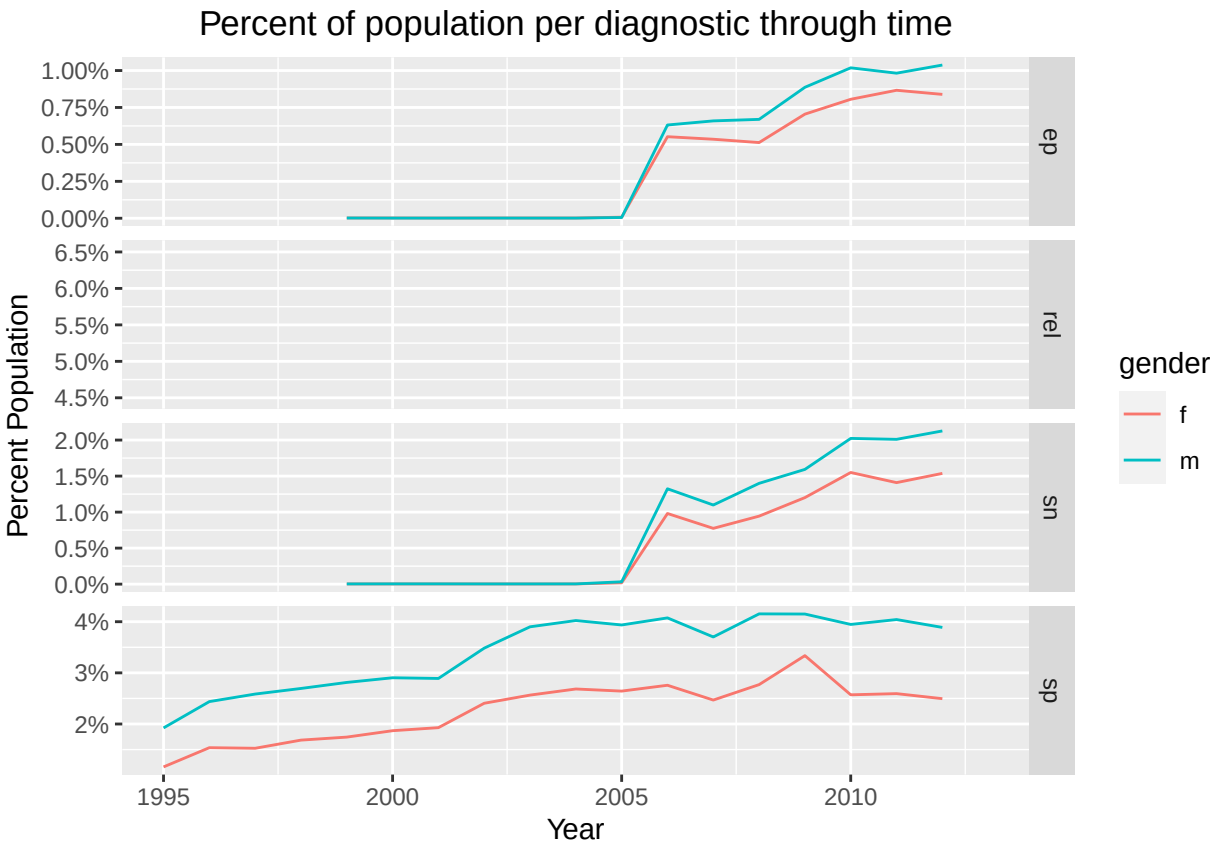
```r
print(pct_df)
```

```
## # A tibble: 94 x 5
## # Groups:   year, gender [38]
##     year gender diagnostic ratio_pop pct_pop_format
##    <dbl> <chr>  <chr>          <dbl> <chr>
##  1  1995 f      sp         0.0116    1.161%
##  2  1995 m      sp         0.0192    1.923%
##  3  1996 f      sp         0.0154    1.537%
##  4  1996 m      sp         0.0244    2.438%
##  5  1997 f      sp         0.0153    1.526%
##  6  1997 m      sp         0.0259    2.585%
##  7  1998 f      sp         0.0169    1.685%
##  8  1998 m      sp         0.0270    2.695%
##  9  1999 f      ep         0.0000197 0.002%
## 10  1999 f      sn         0.0000263 0.003%
## # i 84 more rows
```

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.

```r
ggplot(pct_df) +
  geom_line(aes(x = year, y = ratio_pop, group = gender, color = gender)) +
  labs(x='Year',
       y='Percent Population',
       title='Percent of population per diagnostic through time') +
  scale_y_continuous(labels = label_percent()) +
  facet_grid(rows = vars(diagnostic), scales = "free_y") +
  theme(plot.title = element_text(hjust=0.5))
```

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




```
## Notice the graph of 'rel' diagnostic group has only 1 data point,
# so, there is no line on the graph.  Free Y scale for easy visualization.
```

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

```
filter_df <- filter_df %>%
  unite(col = 'age_range', 'min_age':'max_age', sep ='-')

tail(filter_df)
```

```
## # A tibble: 6 x 9
##   country  iso2  iso3   year diagnostic gender age_range count population
##   <chr>    <chr> <chr> <dbl> <chr>      <chr>  <chr>     <dbl>      <dbl>
## 1 Zimbabwe ZW    ZWE    2013 rel        f      15-24      2069   14149648
## 2 Zimbabwe ZW    ZWE    2013 rel        f      25-34      4649   14149648
## 3 Zimbabwe ZW    ZWE    2013 rel        f      35-44      3526   14149648
## 4 Zimbabwe ZW    ZWE    2013 rel        f      45-54      1453   14149648
## 5 Zimbabwe ZW    ZWE    2013 rel        f      55-64       811   14149648
## 6 Zimbabwe ZW    ZWE    2013 rel        f      65-Inf      725   14149648
```

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
# Total count per diagnostic

ttcount_diag <- filter_df %>%
  group_by(diagnostic) %>%
  summarise(ttcount_diag = sum(count))

pct_age <- filter_df %>%
  left_join(ttcount_diag, by = 'diagnostic') %>%
  group_by(age_range, diagnostic) %>%
  summarise(pct_age_diag = sum(count / ttcount_diag))
```

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

```r
print(pct_age)
```

```
## # A tibble: 28 x 3
## # Groups:   age_range [7]
##    age_range diagnostic pct_age_diag
##    <chr>     <chr>             <dbl>
##  1 0-14      ep               0.126
##  2 0-14      rel              0.0634
##  3 0-14      sn               0.0997
##  4 0-14      sp               0.0197
##  5 15-24     ep               0.158
##  6 15-24     rel              0.150
##  7 15-24     sn               0.151
##  8 15-24     sp               0.185
##  9 25-34     ep               0.201
## 10 25-34     rel              0.204
## # i 18 more rows
```

```r
#### Replace '65-Inf' with '65+' for graph label
library(stringr)
pct_age$age_range <- str_replace(pct_age$age_range,"65-Inf", "65+")



#### Load viridis color palettes and change faucet label of '65-Inf' to '65+'
library(viridis)
```

```
## Loading required package: viridisLite
##
## Attaching package: 'viridis'
##
## The following object is masked from 'package:scales':
##
##     viridis_pal
```

```r
ggplot(pct_age) +
  geom_col(aes(x = diagnostic, y = pct_age_diag, fill = age_range)) +
```

```
labs(x='Diagnostic',
     y='Percent of Total Diagnostic',
     title='Percent of each age group contributed to each diagnostic',
     fill = 'Age Range (years old)') +
scale_y_continuous(labels = label_percent(scale = 100)) +
facet_grid(.~age_range) +
theme(plot.title = element_text(hjust=0.5), legend.position = 'bottom',
     legend.text = element_text(size = 12)) +
scale_fill_viridis(discrete = TRUE)
```