WHO dataset advanced visualization

Luke Geel

11/3/2020

```r
library(tidyverse)
```

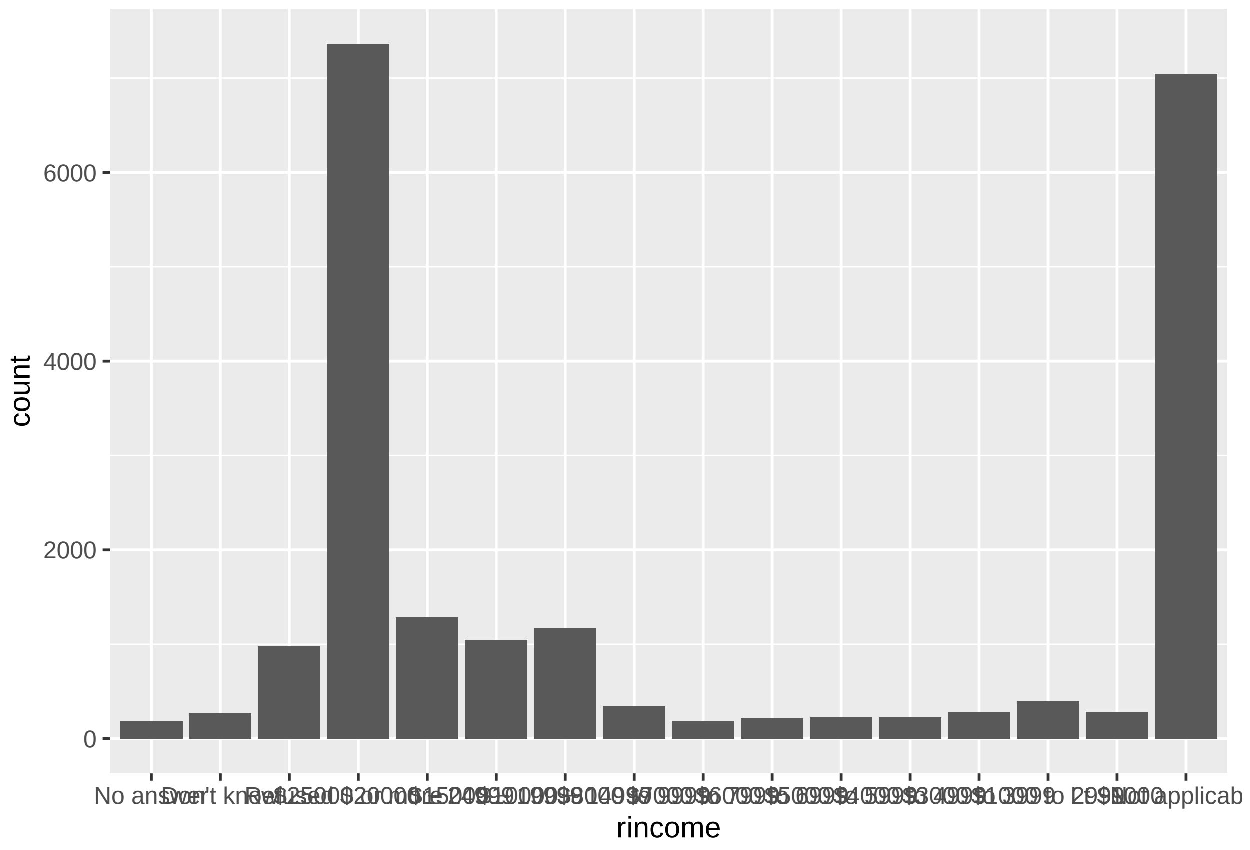
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.0
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
gss_cat
```

```
## # A tibble: 21,483 x 9
##     year marital         age race  rincome        partyid    relig denom tvhours
##    <int> <fct>         <int> <fct> <fct>          <fct>      <fct> <fct>   <int>
##  1  2000 Never married    26 White $8000 to 9999  Ind,near ~ Prot~ Sout~      12
##  2  2000 Divorced         48 White $8000 to 9999  Not str r~ Prot~ Bapt~      NA
##  3  2000 Widowed          67 White Not applicable Independe~ Prot~ No d~       2
##  4  2000 Never married    39 White Not applicable Ind,near ~ Orth~ Not ~       4
##  5  2000 Divorced         25 White Not applicable Not str d~ None  Not ~       1
##  6  2000 Married          25 White $20000 - 24999 Strong de~ Prot~ Sout~      NA
##  7  2000 Never married    36 White $25000 or more Not str r~ Chri~ Not ~       3
##  8  2000 Divorced         44 White $7000 to 7999  Ind,near ~ Prot~ Luth~      NA
##  9  2000 Married          44 White $25000 or more Not str d~ Prot~ Other       0
## 10  2000 Married          47 White $25000 or more Strong re~ Prot~ Sout~       3
## # ... with 21,473 more rows
```

Question 1 General Social Survey A. Make a bar chart for the rincome (reported income). What makes the default bar chart hard to understand?

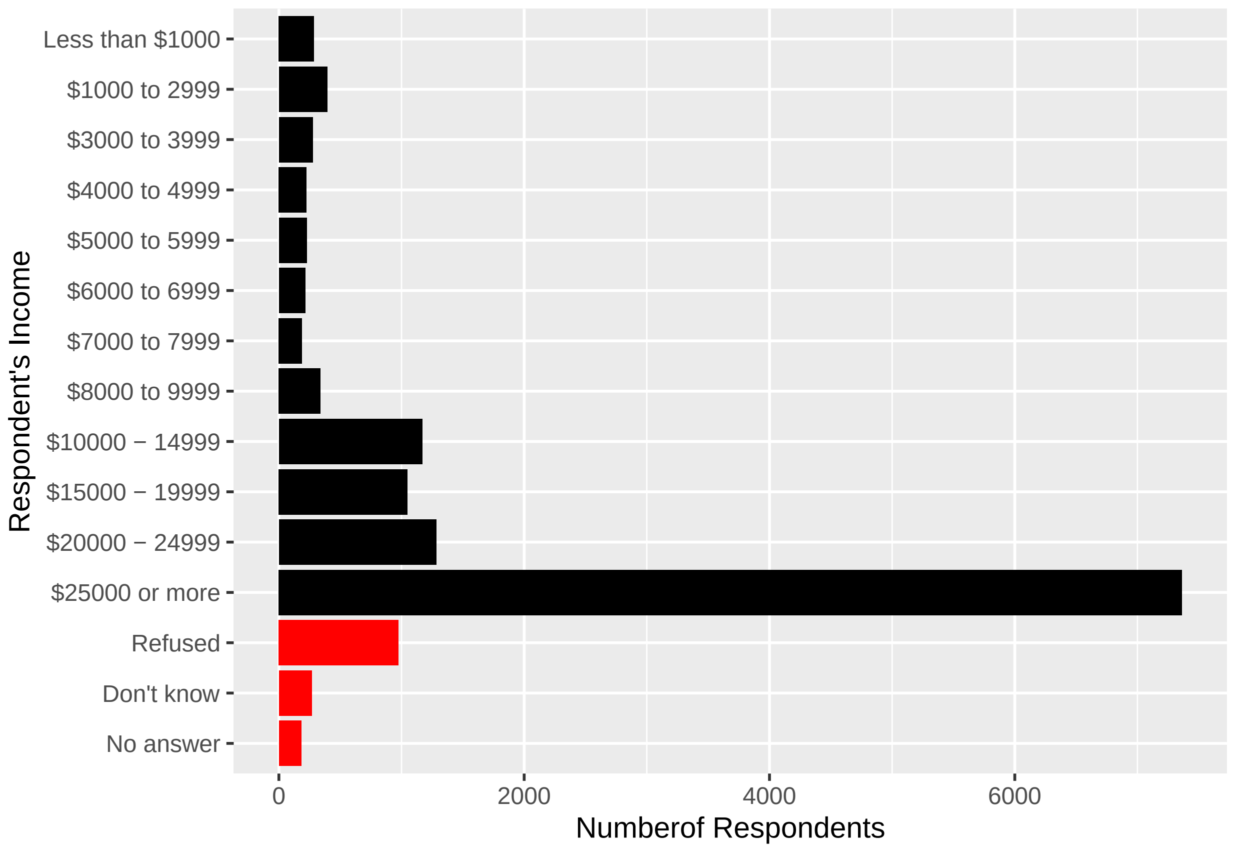
```r
ggplot(gss_cat)+
  geom_bar(aes(x=rincome))
```



The reason this default bar chart is so hard to understand is because the x-axis labels are so close together that you can't tell them apart.

B. Let's make this bar chart better: 1. remove the rows with value "Not applicable" 2. rename "Lt $1000" to "Less than $1000" 3. (optional) use color to distinguish non-response categories ("Refused", "Don't know", and "No answer") from income levels ("Lt $1000", . . .) 4. add meaningful y- and x-axis titles 5. flip the coordinate

```
new_gss_cat <- gss_cat
levels(new_gss_cat$rincome)[levels(new_gss_cat$rincome)=="Lt $1000"] <- "Less than $1000"
ggplot(new_gss_cat %>% filter(rincome != "Not applicable"))+
  geom_bar(aes(y=rincome, fill=rincome))+
  labs(x= "Numberof Respondents",y="Respondent's Income")+
  scale_fill_manual(values = c("red", "red","red","black","black","black","black","black","black","blac
  theme(legend.position = "none")
```



Question 2 who Data

```
names(who) <- str_replace(names(who), "newrel", "new_rel")
who2 <- who %>%
  gather("codes", "case", 5:60) %>%
  select(-iso2, -iso3) %>%
  separate(codes, c("new", "type", "sexage"), sep = "_") %>%
  select(-new) %>%
  separate(sexage, into = c("sex", "age"), sep = 1,convert=TRUE)
```

A. There are many missing values in the case variable. We need to think about how missing values are represented in this dataset. The main concern is whether a missing value means that there were no cases of Tuberculosis (TB) or whether it means that the WHO does not have data on the number of TB cases. Check the presence of zeroes in the case variable.

```
nrow(who2%>%filter(case=="0"))
```

```
## [1] 11080
```

```
nrow(who2%>%filter(is.na(case)))
```

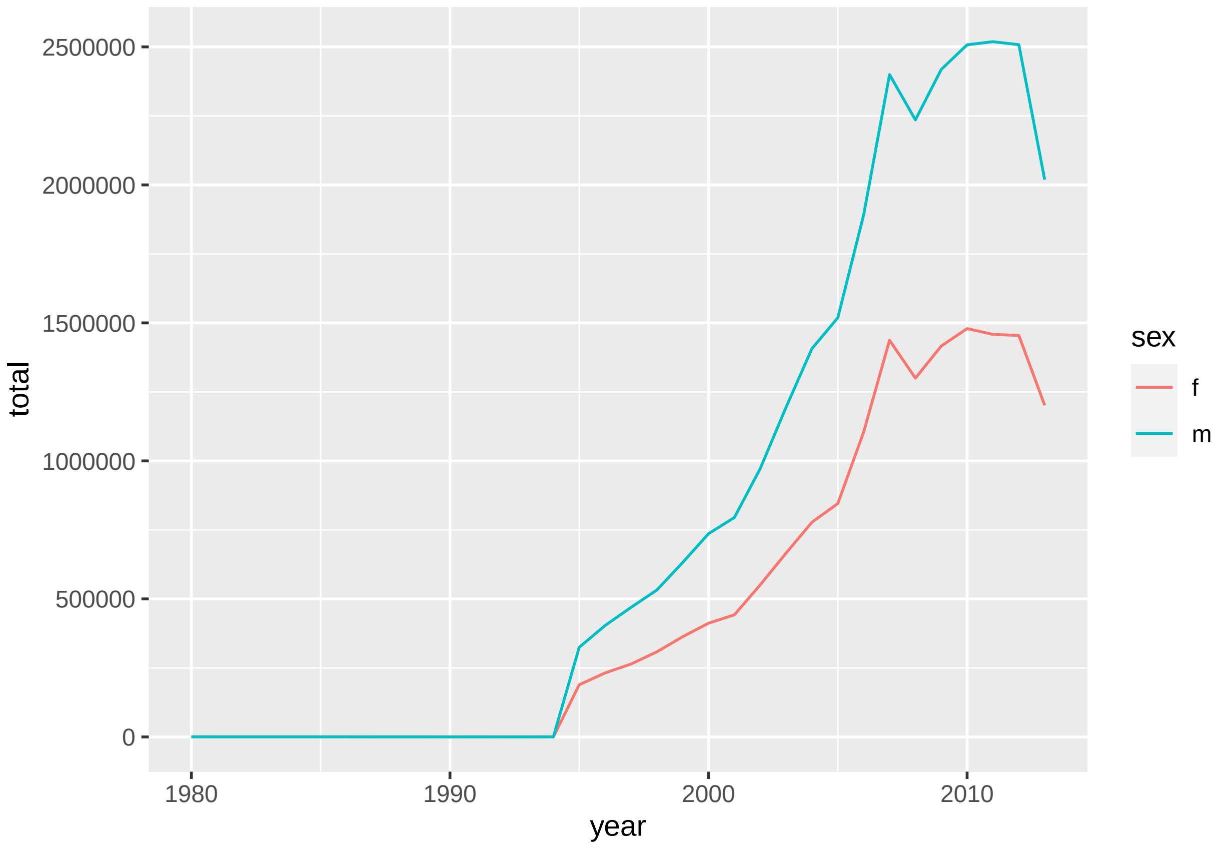
```
## [1] 329394
```

Based on these results I believe that a value of 0 means that there were 0 cases while an NA value means that the data is missing.

B. For each year and sex compute the total number of cases of TB (remove the missing values), and make the following time series plot of the total number of cases (Note the changes in the legend lables and the scientific notation in the y-axis). What do you find from the plot?

```r
who2 %>%
  filter(!is.na(case)) %>%
  group_by(year, sex) %>%
  summarize(total_cases = sum(case))%>%
  ggplot()+
  geom_line(mapping=aes(x=year,y=total_cases, color=sex))+
  labs(y="total")
```

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



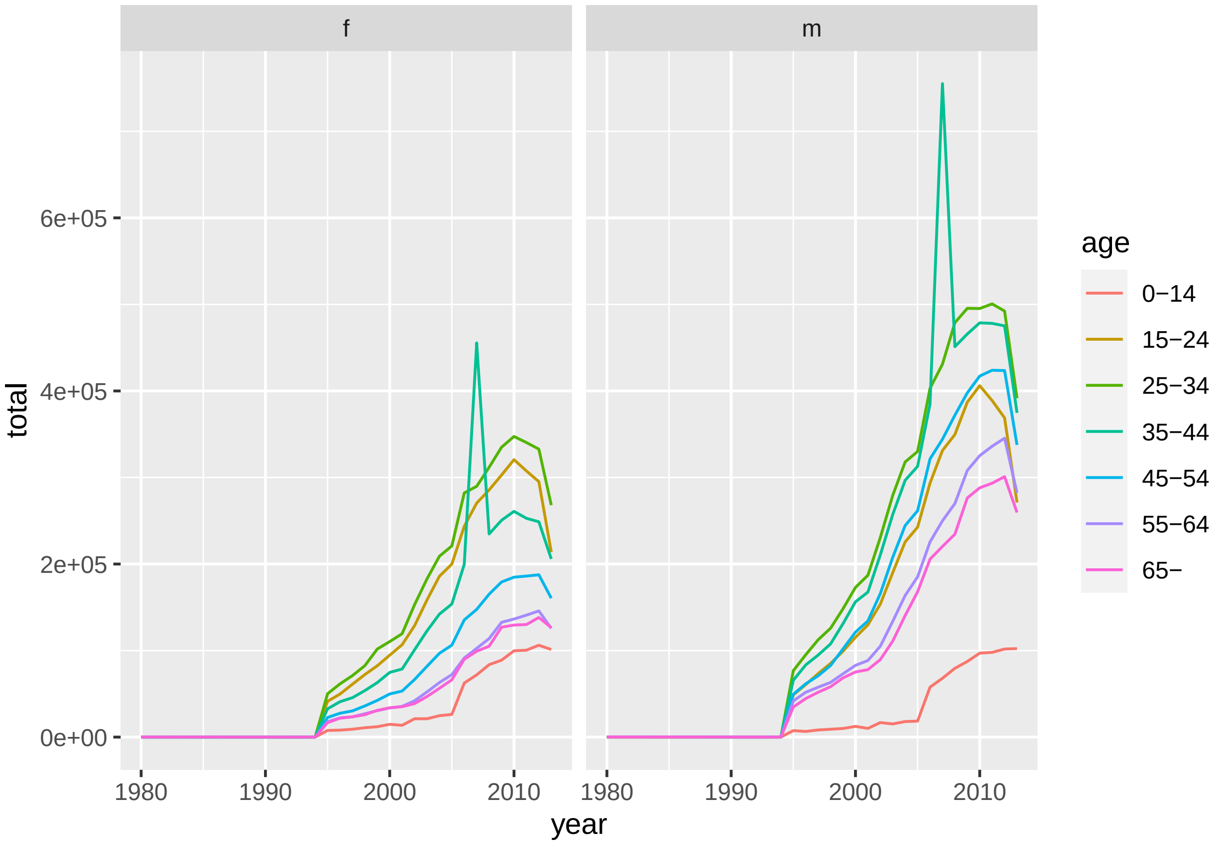
C. Let's plot the number of cases for each age group for males and females.

```r
who3 <- who2%>%filter(!is.na(case))%>%group_by(year,age)
attach(who3)
who3$agecat[age == 14 ] <- "0-14"
```

```
## Warning: Unknown or uninitialised column: `agecat`.
```

```r
who3$agecat[age == 1524 ] <- "15-24"
who3$agecat[age == 2534 ] <- "25-34"
who3$agecat[age == 3544 ] <- "35-44"
who3$agecat[age == 4554 ] <- "45-54"
who3$agecat[age == 5564 ] <- "55-64"
who3$agecat[age == 65 ] <- "65-"

who3 %>%group_by(year, agecat,sex)%>%
  summarize(total_cases = sum(case),sex=sex)%>%
  ggplot()+
  geom_line(mapping=aes(x=year, y=total_cases, color=agecat))+
  facet_grid(.~sex)+
  labs(y="total", color="age")
```

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



D. Collapse the age groups into five categories 0-14,15-24, 25-44, 45-64,65-. Below is an example.

```
who3 <- who2%>%filter(!is.na(case))%>%group_by(year,age)
attach(who3)
```

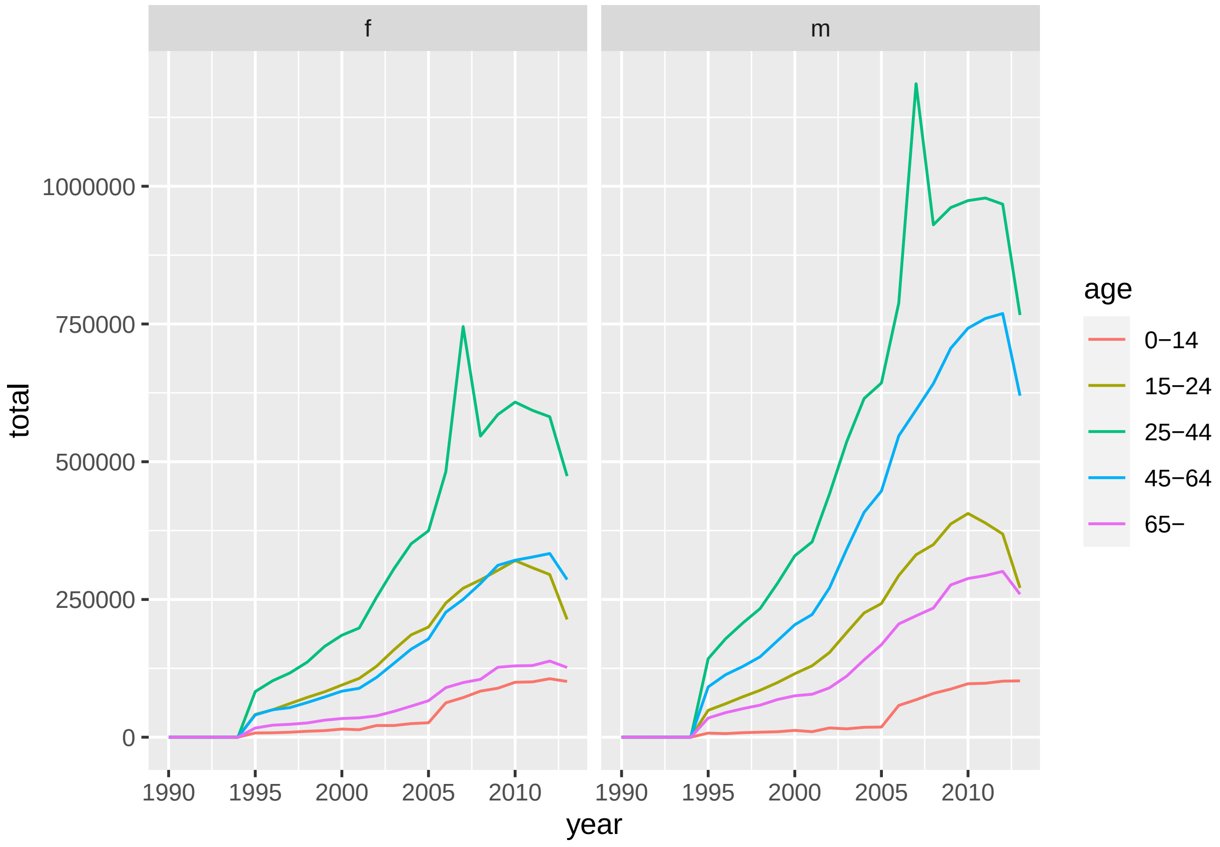
```
## The following objects are masked from who3 (pos = 3):
##
##     age, case, country, sex, type, year
```

```
who3$agecat[age == 14 ] <- "0-14"
```

```
## Warning: Unknown or uninitialised column: `agecat`.
```

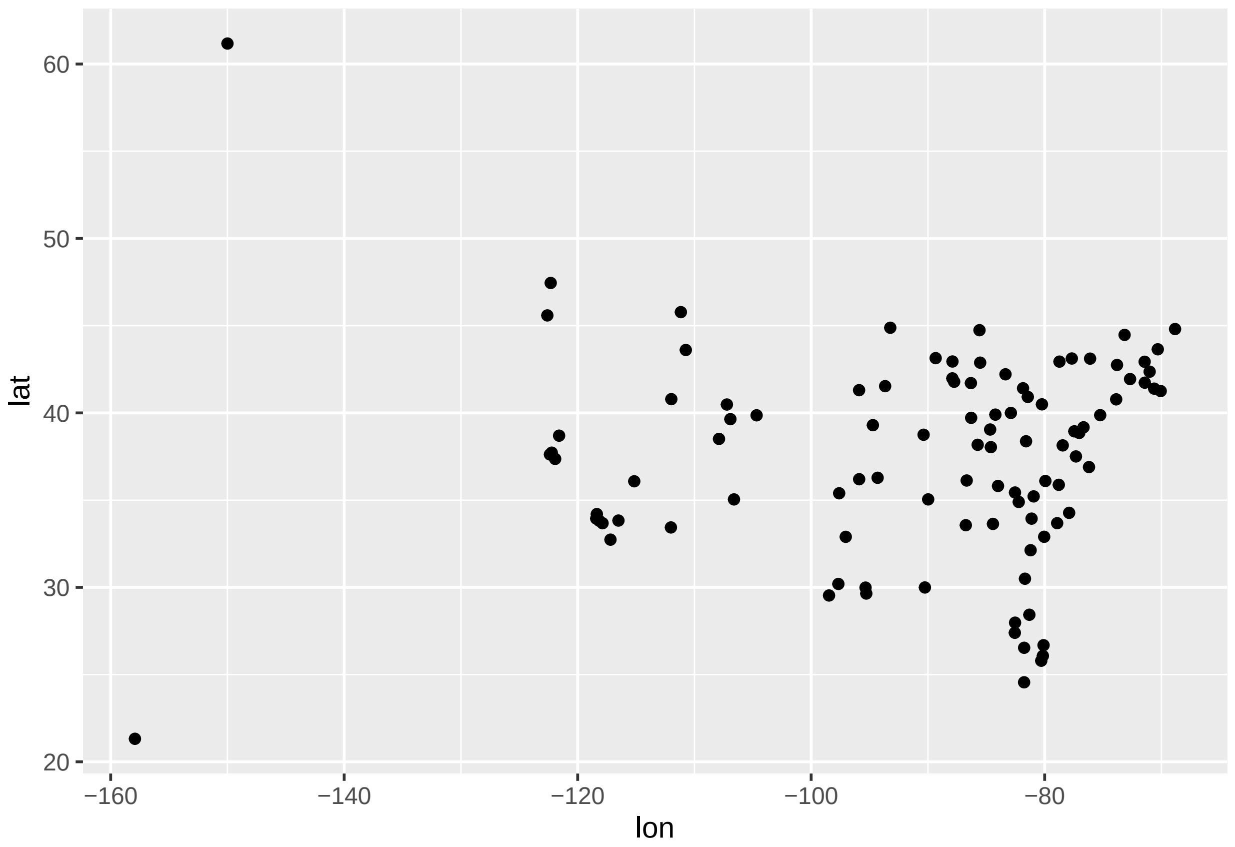
```
who3$agecat[age == 1524 ] <- "15-24"
who3$agecat[age == 2534 | age == 3544 ] <- "25-44"
who3$agecat[age == 4554 | age == 5564] <- "45-64"
who3$agecat[age == 65 ] <- "65-"

who3 %>%group_by(year, agecat,sex)%>%
  summarize(total_cases = sum(case),sex=sex)%>%
  ggplot()+
  geom_line(mapping=aes(x=year, y=total_cases, color=agecat))+
  facet_grid(.~sex)+
  labs(y="total", color="age")+
  xlim(1990,2013)
```

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

## Warning: Removed 133 rows containing missing values (`geom_line()`).
```



Question 3: More and more on flights dataset A. Explain what the following code does.

```
library(nycflights13)

airports %>%
  semi_join(flights, c("faa" = "dest")) %>%
  ggplot(aes(lon, lat))+
  geom_point()
```



What this code does is it joins the airports dataset with the flights dataset where the faa airport code from airports matches the dest variable from flights and plots each airport in both datasets.

B. Here's an easy way to draw a map of the United States. (You need to install and load the maps package.) airports %>% semi_join(flights, c("faa" = "dest")) %>% ggplot(aes(lon, lat)) + borders("state") + geom_point() + coord_quickmap()

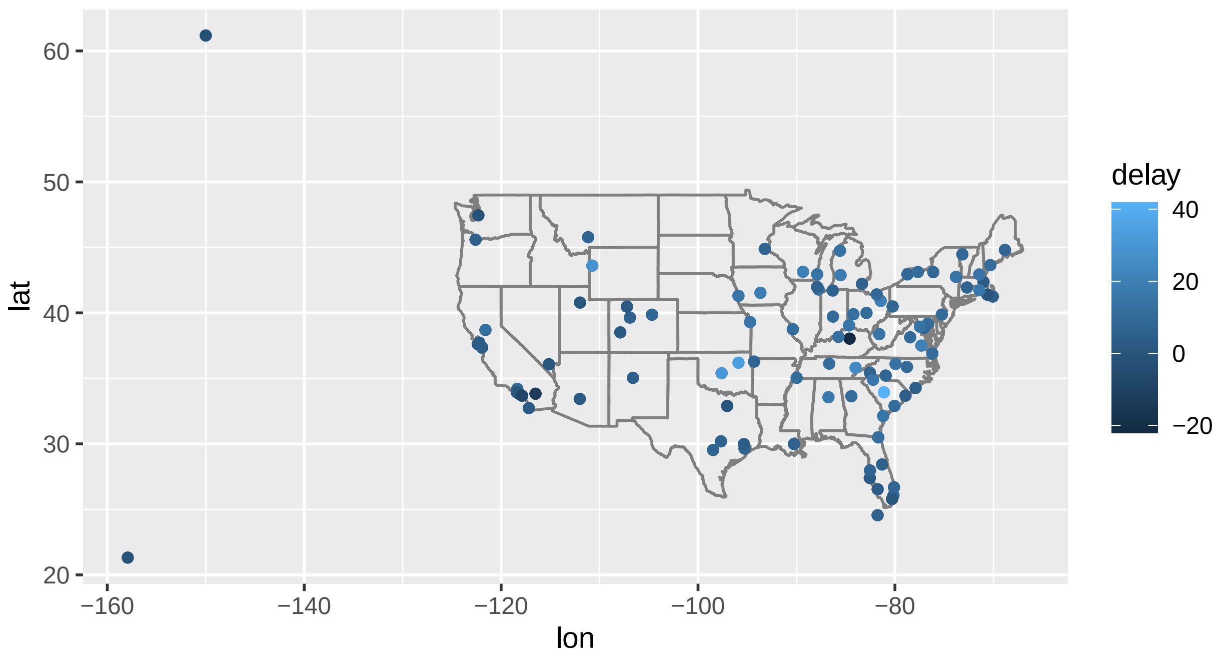
Modify the code to show the spatial distribution of delays. You need to 1. compute the average arrival delay by destination from the flights data 2. join on the airports data to get the longitude and latitude of each destination 3. plot each destination with the average arrival delay mapping to the size or color aesthetic (map a continuous variable to color or size will generate a bar in the legend) 4. add the borders of the states and change the coordinate using coord_quickmap()

```
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
#library(plyr)
#library(dplyr)
library(nycflights13)

flights1 <- flights%>%group_by(dest)%>%
  summarize(avg_arr_delay = mean(arr_delay, na.rm=TRUE))

flights1 %>%
  inner_join(airports, c("dest" = "faa")) %>%
  ggplot(aes(lon, lat)) +
  borders("state") +
  geom_point(aes(color = avg_arr_delay)) +
  coord_quickmap()+
  labs(color="delay")
```



C.

We need to first get the longitude and latitude of the orgin and destination of each flight. You need to 1. Join flights on airports by origin to get the longitude and latitude of the orgin 2. Join flights on airports by destination to get the longitude and latitude of the orgin

```
airports%>%semi_join(flights, c("faa"="origin"))
```

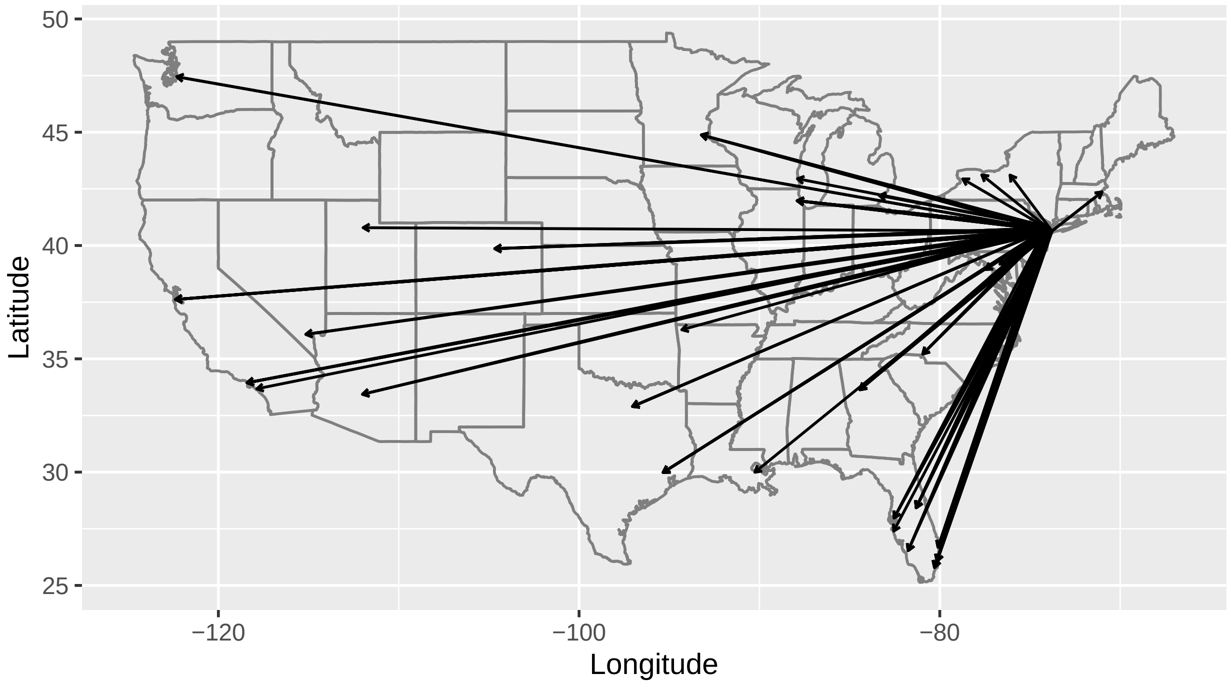
```
## # A tibble: 3 x 8
##   faa   name                  lat   lon   alt    tz dst   tzone
##   <chr> <chr>               <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 EWR   Newark Liberty Intl  40.7 -74.2    18    -5 A     America/New_York
## 2 JFK   John F Kennedy Intl  40.6 -73.8    13    -5 A     America/New_York
## 3 LGA   La Guardia           40.8 -73.9    22    -5 A     America/New_York
```

```
airports%>%semi_join(flights, c("faa"="dest"))
```

```
## # A tibble: 101 x 8
##    faa   name                                lat    lon   alt    tz dst   tzone
##    <chr> <chr>                             <dbl>  <dbl> <dbl> <dbl> <chr> <chr>
##  1 ABQ   Albuquerque International Sunport  35.0 -107.   5355    -7 A     Ameri~
##  2 ACK   Nantucket Mem                      41.3  -70.1    48    -5 A     Ameri~
##  3 ALB   Albany Intl                        42.7  -73.8   285    -5 A     Ameri~
##  4 ANC   Ted Stevens Anchorage Intl         61.2 -150.    152    -9 A     Ameri~
##  5 ATL   Hartsfield Jackson Atlanta Intl    33.6  -84.4  1026    -5 A     Ameri~
##  6 AUS   Austin Bergstrom Intl              30.2  -97.7   542    -6 A     Ameri~
##  7 AVL   Asheville Regional Airport         35.4  -82.5  2165    -5 A     Ameri~
##  8 BDL   Bradley Intl                       41.9  -72.7   173    -5 A     Ameri~
##  9 BGR   Bangor Intl                        44.8  -68.8   192    -5 A     Ameri~
## 10 BHM   Birmingham Intl                    33.6  -86.8   644    -6 A     Ameri~
## # ... with 91 more rows
```

D. We draw the route only for the first 100 flights: 1. use slice(1:100) to extract the first 100 rows from the data obtained in (c) 2. use geom_segment to draw the route for each flight. Check arrow argument to adjust the size of the arrow 3. add the borders of the states and change the coordinate using coord_quickmap()

```
flights %>%
  inner_join(select(airports, origin = faa, origin_lat = lat, origin_lon = lon), by = "origin") %>%
  inner_join(select(airports, dest = faa, dest_lat = lat, dest_lon = lon), by = "dest") %>%
  slice(1:100) %>%
  ggplot(aes(lon, lat)) +
  borders("state") +
  geom_segment(mapping = aes(x=origin_lon,y=origin_lat,xend=dest_lon,yend=dest_lat), arrow = arrow(leng
  coord_quickmap()+
  labs(x="Longitude", y="Latitude")
```



E. Let's explore the relationship between the age of a plane (each plane is represented by tailnum) and its delay. The year variable in the planes data shows the year manufactured. Following the steps to draw a figure (below is an exmaple): 1. join flights on planes (note the flights also has a variable year) 2. calculate the age of the plane 3. calculate the average departure delay and arrival delay by age 4. gather the average arr_delay and dep_delay columns into a key:value column pair names type and delay 5. recode the levels of type column to be "Departure" and "Arrival" 6. create the plot

```
#planes%>%
#left_join(flights,by="tailnum") %>%
# summarize(planeage = year.y-year.x)%>%
#  summarize(avg_dep_delay = mean(dep_delay), avg_arr_delay = #mean(arr_delay))%>%
#  ggplot()+
#  geom_line(aes(x=planeage, y=avg_delay))+
#  labs(x="age",y="delay")
```

F. Let's explore the relationship between some weather condition and the delay. We will focus on the departure delay and the weather at the orgin. Make the following plots and describe the relationship between precipitation (visibility) and the delay.

```
#full_join(flights,weather, by="origin")%>%
#  group_by(origin)%>%
#  mutate(avg_delay = mean(dep_delay), avg_precip = mean(precip))%>%
#  ggplot()+
#  geom_point(aes(x=avg_precip, y=dep_delay))
```