Skills PS4

Meredith Gavin

4/30/2022

```
library(tidyverse)
```

Front matter This submission is my work alone and complies with the 30535 integrity policy.

Add your initials to indicate your agreement: **MG**

Add your collaborators: **___**

Late coins used this pset: 1. Late coins left: 4.

Problems

You change section heading goes here

1: Tidy Data with pivot_longer() and pivot_wider()

1.1

Load the tables:

```
library(tidyverse)

?table1

table1 <- table1
head(table1)
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

```
table2 <- table2
head(table2)
```

```
## # A tibble: 6 x 4
##   country      year type            count
##   <chr>       <int> <chr>           <int>
## 1 Afghanistan  1999 cases             745
## 2 Afghanistan  1999 population   19987071
## 3 Afghanistan  2000 cases            2666
## 4 Afghanistan  2000 population   20595360
## 5 Brazil       1999 cases           37737
## 6 Brazil       1999 population  172006362
```

```
table4a <- table4a
head(table4a)
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
##   <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b <- table4b
head(table4b)
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
##   <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

Calculate the TB rates:

Using table1:

```
table1 %>%
  mutate(tb_rate = (cases/population)*10000)
```

```
## # A tibble: 6 x 5
##   country      year  cases population tb_rate
##   <chr>       <int>  <int>      <int>   <dbl>
## 1 Afghanistan  1999    745   19987071   0.373
## 2 Afghanistan  2000   2666   20595360   1.29
## 3 Brazil       1999  37737  172006362   2.19
## 4 Brazil       2000  80488  174504898   4.61
## 5 China        1999 212258 1272915272   1.67
## 6 China        2000 213766 1280428583   1.67
```

Using table2:

```
table2_with_rates <- table2 %>%
  pivot_wider(
    names_from = type,
```

```r
    values_from = count
  ) %>%
  mutate(tb_rate = (cases/population)*10000)

head(table2_with_rates)
```

```
## # A tibble: 6 x 5
##   country      year  cases population tb_rate
##   <chr>       <int>  <int>      <int>   <dbl>
## 1 Afghanistan  1999    745   19987071   0.373
## 2 Afghanistan  2000   2666   20595360   1.29
## 3 Brazil       1999  37737  172006362   2.19
## 4 Brazil       2000  80488  174504898   4.61
## 5 China        1999 212258 1272915272   1.67
## 6 China        2000 213766 1280428583   1.67
```

Using table4a and table4b:

First, we needed to adjust the tables so that they could be compatible for a join. Once the tables are compatible, we can join table4a and table4b to create a table4. Table4 includes all the information we need to compute infection rates.

```r
table4a_cleaned <- pivot_longer(table4a,
             cols = `1999`:`2000`,
             names_to = "year",
             values_to = "cases"
             )

table4b_cleaned <- pivot_longer(table4b,
             cols = `1999`:`2000`,
             names_to = "year",
             values_to = "population")

table4 <- table4a_cleaned %>%
  left_join(table4b_cleaned,
            by = c("year", "country"))

table4 %>%
  mutate(tb_rate = (cases/population)*10000)
```

```
## # A tibble: 6 x 5
##   country     year   cases population tb_rate
##   <chr>       <chr>  <int>      <int>   <dbl>
## 1 Afghanistan 1999     745   19987071   0.373
## 2 Afghanistan 2000    2666   20595360   1.29
## 3 Brazil      1999   37737  172006362   2.19
## 4 Brazil      2000   80488  174504898   4.61
## 5 China       1999  212258 1272915272   1.67
## 6 China       2000  213766 1280428583   1.67
```

Table 1 was the easiest to work with because the table didn't need to be adjusted at all in order to create the tuberculosis rate variable. Tables 4a and 4b were the most difficult to work with because neither had the full information necessary. As a result, we had to adjust the tables to make them compatible and then join the two before creating the tuberculosis rate variable.

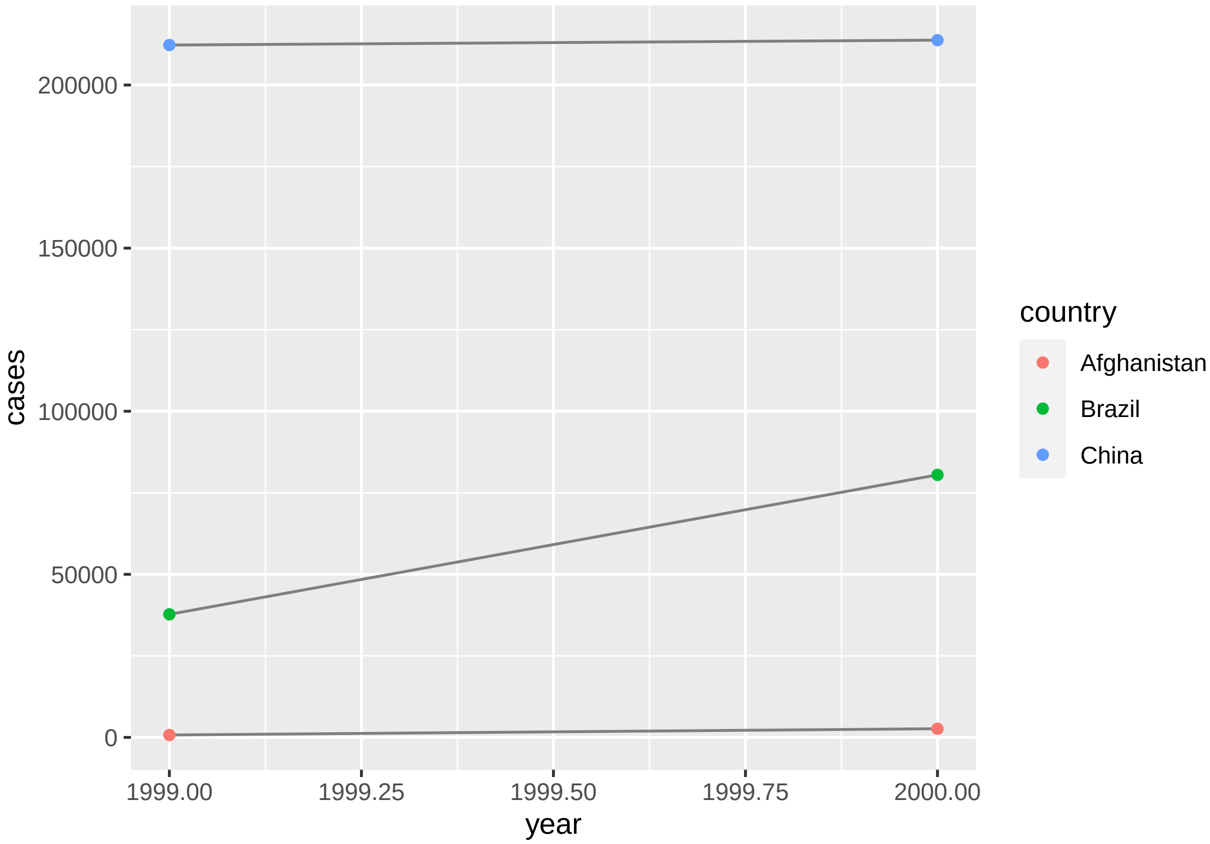
1.2

To recreate the plot from section 12.2 using table2 instead of table1, we can use table2_with_rates, which is is table2 with TB rates. From there, we can use the table to calculate the number of cases each year using count(). Then we can use the code for plotting table1 to plot table2_with_rates.

```
head(table2_with_rates)
```

```
## # A tibble: 6 x 5
##   country      year  cases population tb_rate
##   <chr>       <int>  <int>      <int>   <dbl>
## 1 Afghanistan  1999    745   19987071   0.373
## 2 Afghanistan  2000   2666   20595360   1.29
## 3 Brazil       1999  37737  172006362   2.19
## 4 Brazil       2000  80488  174504898   4.61
## 5 China        1999 212258 1272915272   1.67
## 6 China        2000 213766 1280428583   1.67
```

```
table2_with_rates %>%
  count(cases,
        wt = year) %>%
  ggplot(data = table2_with_rates,
         mapping = aes(year, cases)) +
  geom_line(aes(group = country),
            colour = "grey50") +
  geom_point(aes(colour = country))
```



1.3

```r
stocks <- tibble(
  year   = c(2015, 2015, 2016, 2016),
  half  = c(   1,    2,     1,    2),
  return = c(1.88, 0.59, 0.92, 0.17)
)

stocks
```

```
## # A tibble: 4 x 3
##    year  half return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2016     1   0.92
## 4  2016     2   0.17
```

```r
 stocks %>%
  pivot_wider(names_from = year,
              values_from = return) %>%
  pivot_longer(`2015`:`2016`,
               names_to = "year",
               values_to = "return")
```

```
## # A tibble: 4 x 3
##    half year  return
##   <dbl> <chr>  <dbl>
## 1     1 2015    1.88
## 2     1 2016    0.92
## 3     2 2015    0.59
## 4     2 2016    0.17
```

The "names_from =" and "values_from =" arguments do not need quotation marks because they are referring to the integer values for year and return. The "names_to =" and "values_to =" arguments need quotations because we want R to recognize them as header names, not integer values.

1.4

```r
library(reprex)

reprex({
  library(tidyverse)
  table4a %>% pivot_longer(1999:2000,
                          names_to = "year",
                          values_to = "cases")})
```

```
## i Non-interactive session, setting `html_preview = FALSE`.
```

```
## i Rendering reprex...
```

```
## CLIPR_ALLOW has not been set, so clipr will not run interactively
```

Reprex Output:

library(tidyverse) table4a %>% pivot_longer(1999:2000, names_to = "year", values_to = "cases") Error in `loc_validate()`: Can't subset columns past the end. Locations 1999 and 2000 don't exist. There are only 3 columns.

1.5

The reprex output indicates that R is having difficulty identifying the column headers "1999" and "2000." As a result, the function cannot create a data table as requested. The code is identifying the years 1999 and 2000 as integers instead of as character values. We can fix this by writing the years in single quotation marks.

```
table4a %>% pivot_longer(`1999`:`2000`,
                         names_to = "year",
                         values_to = "cases")
```

```
## # A tibble: 6 x 3
##   country     year   cases
##   <chr>       <chr>  <int>
## 1 Afghanistan 1999     745
## 2 Afghanistan 2000    2666
## 3 Brazil      1999   37737
## 4 Brazil      2000   80488
## 5 China       1999  212258
## 6 China       2000  213766
```

1.6

```
people <- tribble(
~name, ~key, ~value,
#-----------------/--------/------

"Phillip Woods",    "age",      45,
"Phillip Woods",    "height",   186,
"Phillip Woods",    "age",      50,
"Phillip Woods",    "height",   185,
"Jessica Cordero",  "age",      37,
"Jessica Cordero",  "height",   156
)

people %>%
  mutate(unique_id = row_number()) %>%
  pivot_wider(names_from = key,
              values_from = value)
```

```
## # A tibble: 6 x 4
##   name            unique_id   age height
##   <chr>               <int> <dbl>  <dbl>
## 1 Phillip Woods           1    45     NA
## 2 Phillip Woods           2    NA    186
## 3 Phillip Woods           3    50     NA
## 4 Phillip Woods           4    NA    185
## 5 Jessica Cordero         5    37     NA
## 6 Jessica Cordero         6    NA    156
```

The original problem with the tribble is that we have four observations of the name "Phillip Woods" and 2 observations of each age and height. When we pivot_wider(), R understands all 4 "Phillip Woods" as the same observation, however that doesn't work because one person cannot have two heights and two ages. To fix this, we add a unique identifier column. This allows R to read each name in the tribble as an individual observation. We could have multiple people with the same name, so we can't assume that any height or age is paired. R then fills in unknown observations with NA.

1.7

```
preg <- tribble(
  ~pregnant, ~male, ~female,
  "yes", NA, 10,
  "no", 20, 12
)

preg
```

```
## # A tibble: 2 x 3
##   pregnant  male female
##   <chr>    <dbl>  <dbl>
## 1 yes         NA     10
## 2 no          20     12
```

```
preg %>%
  pivot_longer(male:female,
               names_to = "sex",
               values_to = "count") %>%
  pivot_wider(names_from = pregnant,
              values_from = count) %>%
  rename(pregnant = yes,
         not_pregnant = no)
```

```
## # A tibble: 2 x 3
##   sex    pregnant not_pregnant
##   <chr>     <dbl>        <dbl>
## 1 male         NA           20
## 2 female       10           12
```

```
#Resource: https://www.datasciencemadesimple.com/rename-the-column-name-in-r-using-dplyr/
```

I used both pivot_longer() and pivot_wider(). Using pivot_longer(), I created a tibble where "pregnant" was still a column with two observations of yes and two observations of no (to correspond with two observations each of male and female). From there, I used pivot_wider() to create columns based on the values yes and no in the "pregnant" column. For ease of understanding, I renamed the "yes" column to pregnant and the "no" column to not_pregnant. The variables are now sex, pregnant, and not pregnant.

1.8

```
tibble(x = c("a,b,c", "d,e,f,g", "h,i,j")) %>%
  separate(x, c("one", "two", "three"),
           extra = "merge")
```

```
## # A tibble: 3 x 3
##   one   two   three
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     f,g
## 3 h     i     j
```

```
tibble(x = c("a,b,c", "d,e", "f,g,i")) %>%
  separate(x, c("one", "two", "three"),
           extra = "merge")
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 1 rows [2].
```

```
## # A tibble: 3 x 3
##   one   two   three
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     <NA>
## 3 f     g     i
```

There are three options for "extra =", warn, drop, and merge. If there are too many values in the tibble, extra tells R what to do with it when separating the observations in columns. Warn, which is the default, tells R to warn us when there is an extra value. Drop gets rid of extra values without warning. Merge adds the value into the table by creating a cell with more than one observation.

Fill tells R what to do with a missing value when converting a tibble into a table. In this case there is one less value in column two than in columns one and three. The default, fill = "warn", creates a warning when there is a missing value. Right fills the empty spot with the observation to the right, and left fills the empty value with the observation from the left.

1.9

```
who <- tidyr::who

who %>%
  select(country, iso2, iso3) %>%
  group_by(country) %>%
  distinct() %>%
  filter(n() > 1)
```

```
## # A tibble: 0 x 3
## # Groups:   country [0]
## # ... with 3 variables: country <chr>, iso2 <chr>, iso3 <chr>
```

By grouping isolating the country, iso2, and iso3 columns and then grouping by country, we create what should be a list of unique rows. Using filter(), we can check for any repeated rows. Because there are no repeated rows, we can infer that country, iso2, and iso3 are individual country names and codes.

Also, the data is pulled from the World Health Organization (WHO). The organization provides a data dictionary dataset which confirms that iso2 and iso3 are both country identifiers.

2 Tidying Case Study

Load the data from R4DS

```
#The original dataset:

who <- who
head(who)
```

```
## # A tibble: 6 x 60
##   country   iso2  iso3   year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544
##   <chr>     <chr> <chr> <dbl>       <dbl>        <dbl>        <dbl>        <dbl>
## 1 Afghanis~ AF    AFG    1980          NA           NA           NA           NA
## 2 Afghanis~ AF    AFG    1981          NA           NA           NA           NA
## 3 Afghanis~ AF    AFG    1982          NA           NA           NA           NA
## 4 Afghanis~ AF    AFG    1983          NA           NA           NA           NA
## 5 Afghanis~ AF    AFG    1984          NA           NA           NA           NA
## 6 Afghanis~ AF    AFG    1985          NA           NA           NA           NA
## # ... with 52 more variables: new_sp_m4554 <dbl>, new_sp_m5564 <dbl>,
## #   new_sp_m65 <dbl>, new_sp_f014 <dbl>, new_sp_f1524 <dbl>,
## #   new_sp_f2534 <dbl>, new_sp_f3544 <dbl>, new_sp_f4554 <dbl>,
## #   new_sp_f5564 <dbl>, new_sp_f65 <dbl>, new_sn_m014 <dbl>,
## #   new_sn_m1524 <dbl>, new_sn_m2534 <dbl>, new_sn_m3544 <dbl>,
## #   new_sn_m4554 <dbl>, new_sn_m5564 <dbl>, new_sn_m65 <dbl>,
## #   new_sn_f014 <dbl>, new_sn_f1524 <dbl>, new_sn_f2534 <dbl>, ...
```

```
#The tidied dataset:
who_tidy <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
  mutate(
    key = stringr::str_replace(key, "newrel",
                               "new_rel")
  ) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

```r
head(who_tidy)
```

```
## # A tibble: 6 x 6
##   country      year var   sex   age   cases
##   <chr>       <dbl> <chr> <chr> <chr> <dbl>
## 1 Afghanistan  1997 sp    m     014       0
## 2 Afghanistan  1997 sp    m     1524     10
## 3 Afghanistan  1997 sp    m     2534      6
## 4 Afghanistan  1997 sp    m     3544      3
## 5 Afghanistan  1997 sp    m     4554      5
## 6 Afghanistan  1997 sp    m     5564      2
```

2.1

2.1.a

The textbook sets values_drop_na = TRUE to get rid of NA values in the data. If we dropped that part of the code, the data would look like this:

```r
who_keep_na <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases"
  ) %>%
  mutate(
    key = stringr::str_replace(key, "newrel", "new_rel")
  ) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)

head(who_keep_na)
```

```
## # A tibble: 6 x 6
##   country      year var   sex   age   cases
##   <chr>       <dbl> <chr> <chr> <chr> <dbl>
## 1 Afghanistan  1980 sp    m     014      NA
## 2 Afghanistan  1980 sp    m     1524     NA
## 3 Afghanistan  1980 sp    m     2534     NA
## 4 Afghanistan  1980 sp    m     3544     NA
## 5 Afghanistan  1980 sp    m     4554     NA
## 6 Afghanistan  1980 sp    m     5564     NA
```

```r
head(who_tidy)
```

```
## # A tibble: 6 x 6
##   country      year var   sex   age   cases
##   <chr>       <dbl> <chr> <chr> <chr> <dbl>
## 1 Afghanistan  1997 sp    m     014       0
```

```
## 2 Afghanistan  1997 sp    m     1524     10
## 3 Afghanistan  1997 sp    m     2534      6
## 4 Afghanistan  1997 sp    m     3544      3
## 5 Afghanistan  1997 sp    m     4554      5
## 6 Afghanistan  1997 sp    m     5564      2
```

We can use complete() to identify missing values:

```
who_tidy %>%
  nrow()
```

```
## [1] 76046
```

```
who_tidy %>%
  complete(country, year, sex, age) %>%
  nrow()
```

```
## [1] 132653
```

```
132653 - 76046
```

```
## [1] 56607
```

Using complete(), we completed the dataset with implicitly missing observations. For example, we simply didn't have an observation of Afghanistan's tuberculosis cases in 1987. Once we have that information, we can count the total number of rows and compare it to the number of rows in who_tidy. In total, we have 5,6607 missing observations when compared with who_tidy in which NA values have been removed.

2.1.b

```
who_keep_na %>%
  filter(is.na(country) | is.na(year))
```

```
## # A tibble: 0 x 6
## # ... with 6 variables: country <chr>, year <dbl>, var <chr>, sex <chr>,
## #   age <chr>, cases <dbl>
```

The tidied who data with the NA values kept is not explicitly missing any country-year pairs.

2.2

In the WHO case study, an NA refers to a value which they simply have no data on. For example, if there is no information on tuberculosis cases in Afghanistan in 1983, the cases would be coded as NA. Zero refers to an observation of zero. If they do have data on tuberculosis in Afghanistan in 1983, but there were no cases, then it would be coded as '0'.

2.3

```r
#The tidied dataset removing the mutate() step:
who_tidy_no_mutate <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 2580 rows [243,
## 244, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 903,
## 904, 905, 906, ...].
```

```r
head(who_tidy_no_mutate)
```

```
## # A tibble: 6 x 6
##   country      year var   sex   age   cases
##   <chr>       <dbl> <chr> <chr> <chr> <dbl>
## 1 Afghanistan  1997 sp    m     014       0
## 2 Afghanistan  1997 sp    m     1524     10
## 3 Afghanistan  1997 sp    m     2534      6
## 4 Afghanistan  1997 sp    m     3544      3
## 5 Afghanistan  1997 sp    m     4554      5
## 6 Afghanistan  1997 sp    m     5564      2
```

```r
head(who_tidy)
```

```
## # A tibble: 6 x 6
##   country      year var   sex   age   cases
##   <chr>       <dbl> <chr> <chr> <chr> <dbl>
## 1 Afghanistan  1997 sp    m     014       0
## 2 Afghanistan  1997 sp    m     1524     10
## 3 Afghanistan  1997 sp    m     2534      6
## 4 Afghanistan  1997 sp    m     3544      3
## 5 Afghanistan  1997 sp    m     4554      5
## 6 Afghanistan  1997 sp    m     5564      2
```

The mutate() code tells R to replace the name "newrel" with "new_rel". When we don't do that, we get an error in which R identifies missing values in 2580 rows and replaces them with "NA". In reality, the values should (mostly) be available, its just an issue of technical differences in the names.

2.4

2.4.a

The total number of cases (n_cases) by sex for each country-year pair:

```
who_tidy_grouped <-
  who_tidy %>%
  group_by(year, country, sex) %>%
  summarise(n_cases = sum(cases, na.rm = TRUE))
```

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

```
head(who_tidy_grouped)
```

```
## # A tibble: 6 x 4
## # Groups:   year, country [3]
##    year country      sex   n_cases
##   <dbl> <chr>        <chr>   <dbl>
## 1  1980 Canada       f         333
## 2  1980 Canada       m         618
## 3  1980 Cook Islands f           4
## 4  1980 Cook Islands m           4
## 5  1981 Canada       f         290
## 6  1981 Canada       m         513
```

2.4.b

Raw values likely won't provide clear evidence because we would need to individually compare case counts between men and women in each country over each year of observation. Some countries only have a small number of cases each year, which is not enough to create solid evidence of significant differences in the number of cases between men and women. We need a separate variable that allows us to compare differences across countries and years.

2.4.c

The ratio of male to female patients with tuberculosis (m_to_f_ratio) for each country-year pair:

```
who_tidy_ratios <-
  who_tidy %>%
  group_by(year, country, sex) %>%
  summarise(n_cases = sum(cases, na.rm = TRUE)) %>%
  pivot_wider(names_from = "sex", values_from = "n_cases") %>%
  mutate(m_to_f_ratio = m/f,
         n_cases = f + m)
```

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

```
head(who_tidy_ratios)
```

```
## # A tibble: 6 x 6
## # Groups:   year, country [6]
##    year country          f     m m_to_f_ratio n_cases
##   <dbl> <chr>        <dbl> <dbl>        <dbl>   <dbl>
```

```
## 1  1980 Canada         333   618          1.86      951
## 2  1980 Cook Islands     4     4          1            8
## 3  1981 Canada         290   513          1.77      803
## 4  1981 Cook Islands     1     1          1            2
## 5  1982 Canada         288   524          1.82      812
## 6  1982 Cook Islands     5     7          1.4        12
```

2.4.d

Grouping the table only by year would likely skew the data. Some countries have very few cases while others have a very high number, certain countries may be hot spots in certain years at the same time that others are seeing a lull in case counts, and country cultures may impact the ratio. For example, in countries where women are more likely to be caring for the sick, they may see higher rates of women being diagnosed with tuberculosis.

2.4.e

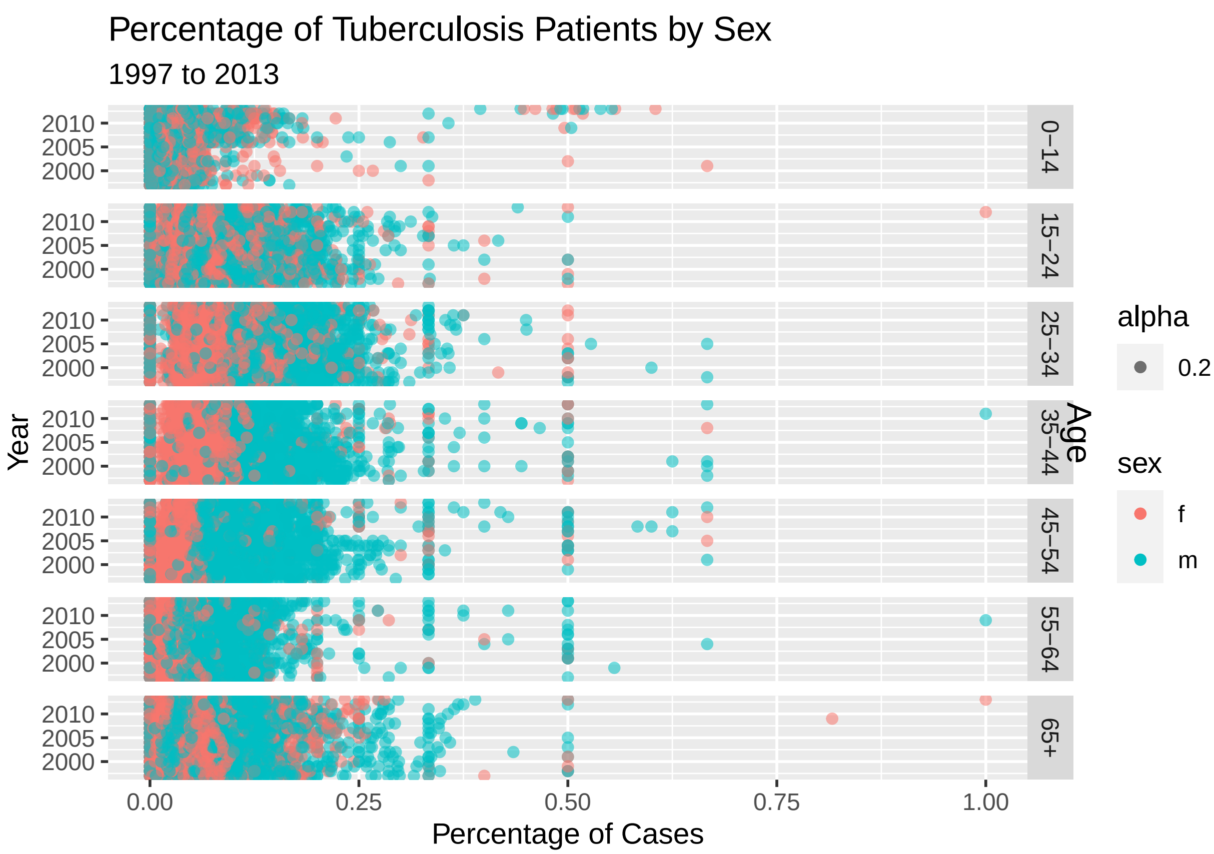
```
age_labels <- c(
  '014' = "0-14",
  '1524' = "15-24",
  '2534' = "25-34",
  '3544' = "35-44",
  '4554' = "45-54",
  '5564' = "55-64",
  '65' = "65+"
)

head(who_tidy)
```

```
## # A tibble: 6 x 6
##   country      year var   sex   age   cases
##   <chr>       <dbl> <chr> <chr> <chr> <dbl>
## 1 Afghanistan  1997 sp    m     014       0
## 2 Afghanistan  1997 sp    m     1524     10
## 3 Afghanistan  1997 sp    m     2534      6
## 4 Afghanistan  1997 sp    m     3544      3
## 5 Afghanistan  1997 sp    m     4554      5
## 6 Afghanistan  1997 sp    m     5564      2
```

```
who_tidy %>%
  group_by(country, year, sex, age) %>%
  summarise(n_cases = sum(cases, na.rm = TRUE)) %>%
  group_by(country, year) %>%
  mutate(total_cases = sum(n_cases)) %>%
  ungroup() %>%
  mutate(percent_by_sex = n_cases/total_cases) %>%
  filter(total_cases > 1,
         year >= 1997) %>%
  ggplot(mapping = aes(y = year,
                       x = percent_by_sex,
                       color = sex)) +
```

```
  geom_point(aes(alpha = 0.2)) +
 facet_grid(age ~., labeller = as_labeller(age_labels)) +
  #Resource: https://stackoverflow.com/questions/3472980/how-to-change-facet-labels
  labs(y = "Year",
       x = "Percentage of Cases",
       color = "sex",
       title = "Percentage of Tuberculosis Patients by Sex",
       subtitle = "1997 to 2013",
       tag = "Age") +
  theme(plot.tag.position = c(.88, 0.5),
        plot.tag = element_text(angle = -90))
```

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




```
# Resource: https://stackoverflow.com/questions/59632276/how-to-add-vertical-label-on-the-right-side-o
```

2.4.f

The graph includes information on percentage of cases by sex each year, grouped by age. I filtered the data to include only observations where cases are greater than 1, to eliminate scenarios in which percentage is 100% one sex only because there is only on tuberculosis case. The plot focuses only on cases 1997 onward, and it indicates a difference in the percentages of male and female cases. There is no clear grouping of male and female patients in the 0 to 14 age bracket. This could be because there is a relatively low number of cases in this age group. In the 25-34 and 35-44 age brackets, the percentage of male cases clearly outweigh the percentage of female patients. By the time we reach the 65+ age bracket, the pattern is less defined, but it does appear that male patients still make up a larger proportion of total cases. It is less defined, however, perhaps because there is a greater number of cases in elderly populations and/or because women tend to outlive men, so women make up a larger portion of that age bracket to begin with. It was difficult to plot this data and to present it in a way that is representative of the data. In this scenario, I omitted country distinctions in the plot. There were simply too many countries to include in the plot while keeping the data understandable. It would potentially make sense to plot country data individually or by region.

3 Joins

Load dataset "flights" and "airports":

```
flights <- nycflights13::flights
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
flights %>% nrow()
```

```
## [1] 336776
```

```
airports <- nycflights13::airports
head(airports)
```

```
## # A tibble: 6 x 8
##   faa   name                             lat   lon   alt    tz dst   tzone
##   <chr> <chr>                          <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport               41.1 -80.6  1044    -5 A     America/Ne~
## 2 06A   Moton Field Municipal Airport   32.5 -85.7   264    -6 A     America/Ch~
## 3 06C   Schaumburg Regional             42.0 -88.1   801    -6 A     America/Ch~
## 4 06N   Randall Airport                 41.4 -74.4   523    -5 A     America/Ne~
## 5 09J   Jekyll Island Airport           31.1 -81.4    11    -5 A     America/Ne~
## 6 0A9   Elizabethton Municipal Airport  36.4 -82.2  1593    -5 A     America/Ne~
```

```
airports %>% nrow()
```

```
## [1] 1458
```

3.1

The following code calculates the average flight delays by destination:

```
avg_delays_by_dest <- flights %>%
  group_by(dest) %>%
  summarize(avg_delay = mean(arr_delay, na.rm = TRUE))

head(avg_delays_by_dest)
```

```
## # A tibble: 6 x 2
##   dest  avg_delay
##   <chr>     <dbl>
## 1 ABQ        4.38
## 2 ACK        4.85
## 3 ALB       14.4
## 4 ANC       -2.5
## 5 ATL       11.3
## 6 AUS        6.02
```

```
avg_delays_by_dest %>% nrow()
```

```
## [1] 105
```

Join avg_delays_by_dest with airports:

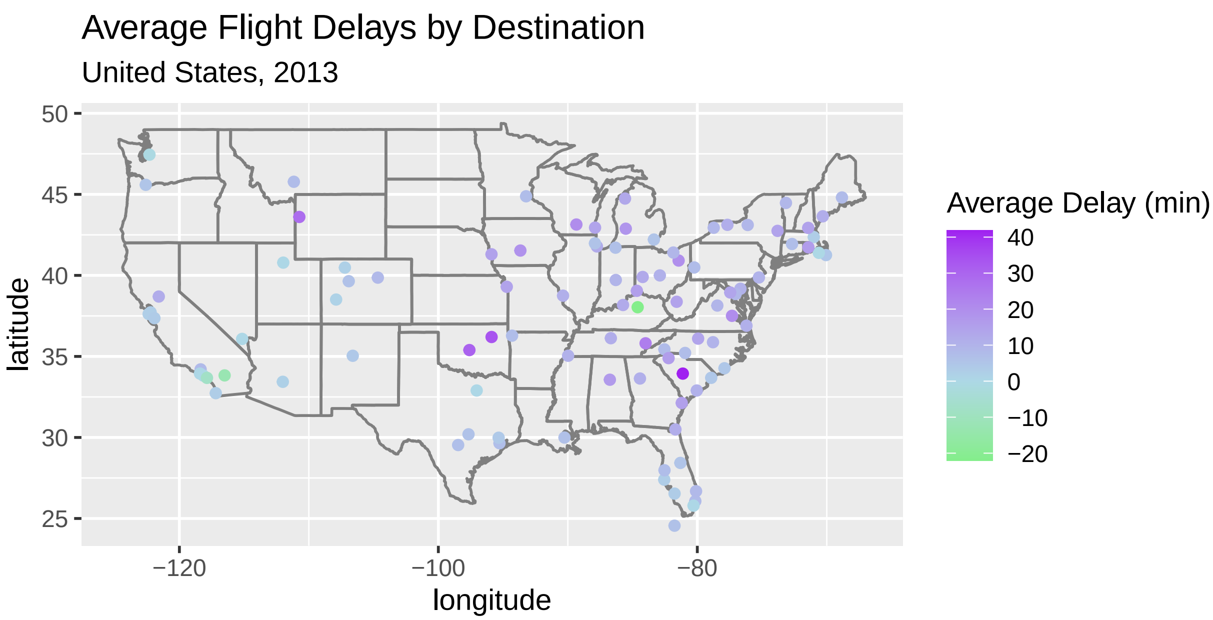
```
avg_delays_airports_join <-
  avg_delays_by_dest %>%
left_join(airports,
          c("dest" = "faa")) %>%
  filter(dest != "HNL" & dest != "ANC")

head(avg_delays_airports_join)
```

```
## # A tibble: 6 x 9
##   dest  avg_delay name                          lat    lon   alt    tz dst   tzone
##   <chr>     <dbl> <chr>                       <dbl>  <dbl> <dbl> <dbl> <chr> <chr>
## 1 ABQ        4.38 Albuquerque Internationa~  35.0 -107.   5355    -7 A     Amer~
## 2 ACK        4.85 Nantucket Mem               41.3  -70.1    48    -5 A     Amer~
## 3 ALB       14.4  Albany Intl                 42.7  -73.8   285    -5 A     Amer~
## 4 ATL       11.3  Hartsfield Jackson Atlan~  33.6  -84.4  1026    -5 A     Amer~
## 5 AUS        6.02 Austin Bergstrom Intl       30.2  -97.7   542    -6 A     Amer~
## 6 AVL        8.00 Asheville Regional Airpo~  35.4  -82.5  2165    -5 A     Amer~
```

```
ggplot(data = avg_delays_airports_join,
       mapping = aes(lon, lat)) +
    borders("state") +
    geom_point(aes(color = avg_delay)) +
    coord_quickmap() +
  scale_color_gradient2(low = "green",
                        mid = "light blue",
```

```r
                        high = "purple",
                        breaks = c(-20, -10, 0, 10, 20, 30, 40)) +
  labs(title = "Average Flight Delays by Destination",
       subtitle = "United States, 2013",
       x = "longitude",
       y = "latitude",
       color = "Average Delay (min)")
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



3.2

```r
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```r
head(airports)
```

```
## # A tibble: 6 x 8
##   faa   name                             lat   lon   alt    tz dst   tzone
##   <chr> <chr>                          <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport               41.1 -80.6  1044    -5 A     America/Ne~
## 2 06A   Moton Field Municipal Airport   32.5 -85.7   264    -6 A     America/Ch~
## 3 06C   Schaumburg Regional             42.0 -88.1   801    -6 A     America/Ch~
## 4 06N   Randall Airport                 41.4 -74.4   523    -5 A     America/Ne~
## 5 09J   Jekyll Island Airport           31.1 -81.4    11    -5 A     America/Ne~
## 6 0A9   Elizabethton Municipal Airport  36.4 -82.2  1593    -5 A     America/Ne~
```

```
flights_airports_lat_lon <- flights %>%
  left_join(airports,  c("dest" = "faa")) %>%
  left_join(airports, c("origin" = "faa")) %>%
  rename(lat.dest = lat.x,
         lon.dest = lon.x,
         lat.origin = lat.y,
         lon.origin = lon.y)

head(flights_airports_lat_lon)
```

```
## # A tibble: 6 x 33
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # ... with 25 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, name.x <chr>, lat.dest <dbl>,
## #   lon.dest <dbl>, alt.x <dbl>, tz.x <dbl>, dst.x <chr>, tzone.x <chr>,
## #   name.y <chr>, lat.origin <dbl>, lon.origin <dbl>, alt.y <dbl>, tz.y <dbl>,
## #   dst.y <chr>, tzone.y <chr>
```

3.3

```
flights_airports_lat_lon  %>%
  group_by(tailnum) %>%
  summarise(n_carriers = n_distinct(carrier)) %>%
  filter(n_carriers > 1) %>%
  nrow()
```

```
## [1] 18
```

There are 11 planes (identified by tailnumber) that are flown by more than one carrier. There are seven instances with missing tailnumbers. It is possible that those missing values would be included in this count.

3.4

Question: Is there a relationship between the age of a plane and its delays?

Join flights and planes data sets to create a dataframe with information on plane metadata and delays:

```
planes <- nycflights13::planes
head(planes)
```

```
## # A tibble: 6 x 9
##   tailnum  year type              manufacturer model engines seats speed engine
##   <chr>   <int> <chr>             <chr>        <chr>   <int> <int> <int> <chr>
## 1 N10156   2004 Fixed wing multi ~ EMBRAER      EMB-~       2    55    NA Turbo~
## 2 N102UW   1998 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
## 3 N103US   1999 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
## 4 N104UW   1999 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
## 5 N10575   2002 Fixed wing multi ~ EMBRAER      EMB-~       2    55    NA Turbo~
## 6 N105UW   1999 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
```

```
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
flights_planes_join <-
  left_join(flights,
          planes,
          by = "tailnum")

head(flights_planes_join)
```

```
## # A tibble: 6 x 27
##   year.x month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1   2013     1     1      517            515         2      830            819
## 2   2013     1     1      533            529         4      850            830
## 3   2013     1     1      542            540         2      923            850
## 4   2013     1     1      544            545        -1     1004           1022
## 5   2013     1     1      554            600        -6      812            837
## 6   2013     1     1      554            558        -4      740            728
## # ... with 19 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
```

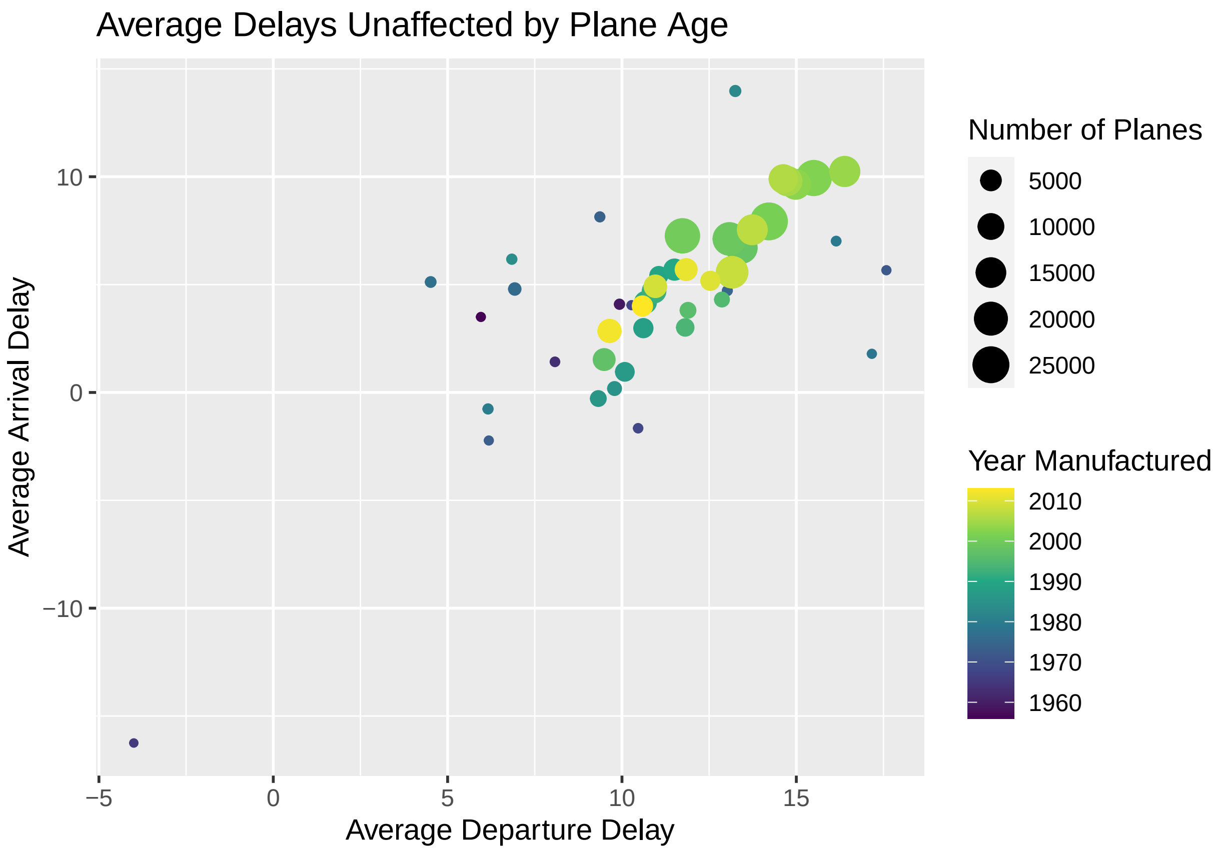
```
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, year.y <int>, type <chr>,
## #   manufacturer <chr>, model <chr>, engines <int>, seats <int>, speed <int>,
## #   engine <chr>
```

Plot the relationship between the plane's manufacturing year and arrival and departure delays:

```
flight_planes_join_avg_delays <-
  flights_planes_join %>%
  group_by(year.y) %>%
  summarise(avg_arr_delay = mean(arr_delay,
                                 na.rm = TRUE),
            avg_dep_delay = mean(dep_delay,
                                 na.rm = TRUE),
            n_planes = n()) %>%
  filter(year.y != "NA")

flight_planes_join_avg_delays
```

```
## # A tibble: 46 x 4
##    year.y avg_arr_delay avg_dep_delay n_planes
##     <int>         <dbl>         <dbl>    <int>
##  1   1956          3.5           5.95       22
##  2   1959          4.09          9.93      117
##  3   1963          1.42          8.08       52
##  4   1965        -16.2          -4           4
##  5   1967          4.05         10.3        22
##  6   1968         -1.66         10.5        43
##  7   1972          5.67         17.6        25
##  8   1973         -2.23          6.18       22
##  9   1974          8.14          9.37      103
## 10   1975          4.72         13.0        92
## # ... with 36 more rows
```

```
ggplot(data = flight_planes_join_avg_delays,
       mapping = aes(y = avg_arr_delay,
                     x = avg_dep_delay)) +
  geom_point(aes(color = year.y,
                 size = n_planes)) +
  scale_color_continuous(type = "viridis") +
  labs(title = "Average Delays Unaffected by Plane Age",
       x = "Average Departure Delay",
       y = "Average Arrival Delay",
       size = "Number of Planes",
       color = "Year Manufactured")
```



There does not appear to be a relationship between plane age and delays. There is, as expected a clear positive relationship between departure and arrival delays. There is also a greater number of planes that have been manufactured more recently than there are older planes in the dataset. However, all the planes, regardless of manufacturing year, are grouped in the same area of the plot. If anything, the newer planes may have slightly higher average delays, but that is likely skewed by there simply being more newer planes than older planes.