Homework #4

Answer the following questions in a single R script called `hw04.R`. Answers must be given by R commands. You cannot simply look at the data set and answer the question via direct inspection. Use comments (#) to indicate which portion of your code answers which question. Be sure that you obtain the correct solutions to each question when you execute your script one line at a time from top to bottom.

Each question will be graded out of 4 points according to the following criteria:

- 0 points: No attempts is made to answer the question.
- 1 point: An attempt is made that, although unsuccessful, revealed some understanding of what the question was asking.
- 2 points: Solution is incorrect, but with some modifications, could be corrected.
- 3 points: Solution is incorrect, but easily resolved with minor modifications **OR** solution is correct, but obtained via convoluted reasoning or by avoiding standard approaches.
- 4 points: Solution is correct and uses standard approaches.

For the following problems, you will use the data contained in `ws03_gun_violence.csv`. I will eventually include with this assignment a script called `hw04_start.R`. This script will essentially be the same as `hw03_start.R` except that it will give a completed solution to #9 on Homework #3. The tibble produced by this script will be called `df`. Copy `hw04_start.R` to `hw04.R` when starting your homework. You will need to run this portion of the script before starting the exercises below.

Recall that in #9 of Homework #3, we created a tibble where we have columns `participant_status` and `participant_type`. Incidents were replicated in this tibble according to the number participants recorded. For instance, if an incident involved 3 participants, then there would 3 rows with the same `incident_id` with the only difference in these rows being values in `participant_status` and `participant_type`. In the next two exercises, we will reconfigure this tibble so that each incident has only one row in the tibble.

#1) There are a variety of `participant_status` designators in `df`. Some of them are quite curious, such as "Killed, Unharmed". To make our analysis simpler, rename these values in `df` as follows:

"Unharmed, Arrested" $\mapsto$ "Unharmed"

"Killed, Unharmed" $\mapsto$ "Killed"

"Killed, Arrested" $\mapsto$ "Killed"

"Injured, Unharmed" $\mapsto$ "Injured"

"Injured, Unharmed" $\mapsto$ "Injured",

"Arrested" $\mapsto$ "Unharmed"

"Killed, Injured" $\mapsto$ "Killed",

"Injured, Unharmed, Arrested" $\mapsto$ "Injured",

"Killed, Unharmed, Arrested" $\mapsto$ "Killed"

#2) If we filter `df` to rows with `incident_id == 92734`, we will obtain three rows. In the first two rows, the `participant_status` and `participant_type` values are "Injured" and "Victim" (respectively). The other row has values of "Unharmed" and "Subject-Suspect". Consolidate `df` by adding a new column called `participant_count` that counts the number of times each pair of values within `participant_status` and `participant_type` occur for a given incident. This change will result in there only being two rows with `incident_id == 92734`. The first row will have values "Injured", "Victim" and 2 within the `participant_status`, `participant_type` and `participant_count` columns (respectively). The second row will have values 'Unharmed", "Subject-Suspect' and 1 within the `participant_status`, `participant_type` and `participant_count` columns (respectively). I recommend using the `group_by`, `across` and `summarize` commands to create `participant_count`.

#3) The `participant_type` and `participant_status` entries can be merged together into a single column using the `unite` function. Let's call this column `participants`. This column will contain the following values:

"Victim_Injured"

"Victim_Killed"

"Victim_Unharmed"

"Subject-Suspect_Injured"

"Subject-Suspect_Killed"

"Subject-Suspect_Unharmed"

Modify this tibble so that each of the values within `participants` becomes its own column (i.e. we create 6 new columns). For each row, the value in such a column should be the number of participants in that incident that fall within that category.

#4) Give a stacked bar chart by year with sub-bars colored for each of the six types of participants with with heights given by the number participants within that category during that year.

#5) Congressional district, in theory, should all have about the same population. The following code shows how to display a map of the contiguous United States with districts colored at random. You will likely need to install some packages. Included in this code is one-time installation commands for the packages that, I think, you will need. You may need to run additional installation commands; see error messages for instructions. Feel free to delete these installation commands once they have been executed.

```
# One-time use installation commands
install.packages("USAboundariesData")
install.packages("USAboundariesData",
  repos = "https://ropensci.r-universe.dev", type = "source")
install.packages("sf")
```

```
library(USAboundaries)
library(sf)
# Non-contiguous territories of the United States
non_cont <- c("Hawaii", "Alaska","Virgin Islands","American Samoa",
"Puerto Rico","Guam","Northern Mariana Islands")
# We get a map of the contiguous United States
us_map <- us_congressional(resolution = "high") %>%
  filter(!(state_name %in% non_cont))

# We create a column called my_random for displaying random values on the
# map. I use the dim command to find out how many rows are in us_map so
# that my_random has the correct number of entries.
us_map$my_random <-  runif(dim(us_map)[1])

ggplot(us_map) +
  geom_sf(aes(fill=my_random),color="black") +
  scale_fill_gradient(low="yellow",high="red") +
  theme_void()
```

Create a map that shows the number killed by congressional district. To do this, you should perform a `left_join` on `us_map` and another tibble generated from `df`. These tibbles should be joined on the state and congressional district columns of both tibbles. You will need to do some research to figure out how this works. Be aware that the congressional districts in `us_map` are encoded as strings while the congressional districts in `df` are encoded as numeric. I recommend converting strings to numeric using `as.numeric`. Also, if you use the `scale_fill_gradient` command above, the resulting map will appear mostly yellow. Feel free to play around with the parameters in this function to get a more distinctive output. In particular, I found that including `trans="log2"` helped a bit.

#6) Restrict the map in #5 to states in the north east (New York, Pennsylvania, New Jersey, Connecticut, Rhode Island, Massachusetts, New Hampshire, Vermont, Maine).

#7) The victim survival rate is given by:

$$1 - \frac{\text{number of victims killed}}{\text{number of victims}}.$$

Compute the victim survival rate by congressional district and display the results on the congressional district map of the contiguous United States.

#8) Compute the suspect survival rate during defensive incidents by congressional district. Display the results on the congressional district map of the contiguous United States.