Final Exam ISDS3105

Name:

The only electronic device allowed during the test is your laptop and **exclusively** for using RStudio. Before you start, make sure you create a **new project**. During the test, you are not allowed to access your notes, nor searching on the internet.

For each question, you shall debug the code to produce the **correct** output (the one under the bolded "correct:"). You can use the .Rmd copy of the exam to run and test your solutions, but you shall write your answers **on the the paper copy** since that is what you will submit. Each question contains only one mistake, although it may be repeated (e.g., using the wrong function multiple times). New solutions that still produce the correct output are also valid. If you decide to go for a new solution, I will consider the answer correct **only** if it retrieves the **same** output of the output under "correct." (no partial points).

The libraries to load for this test are:

```
library(tidyverse)
library(fivethirtyeight)
```

The datasets you need are either the built-in or from the package `fivethirtyeight`.

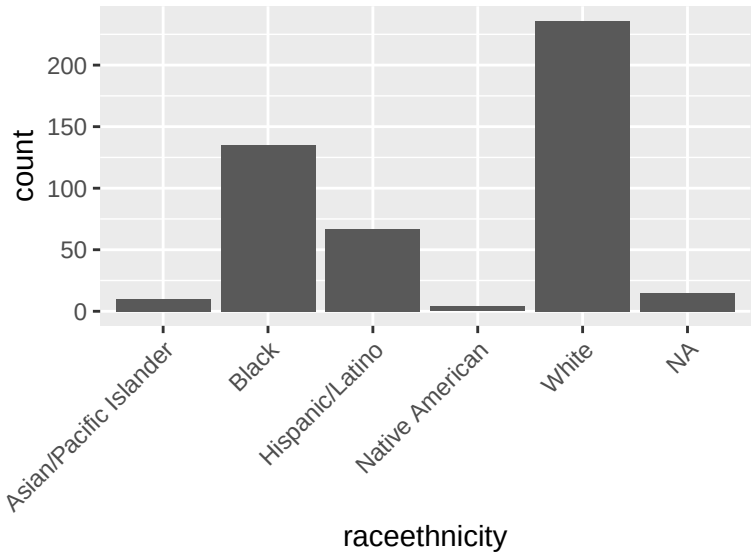
Data Visualization

1. Plot a barchart of the count of killings by race/ethnicity (5 points)

```
ggplot(data = police_killings) +
  geom_bar(aes(x = raceethnicity, y = n())) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Error: This function should not be called directly
```

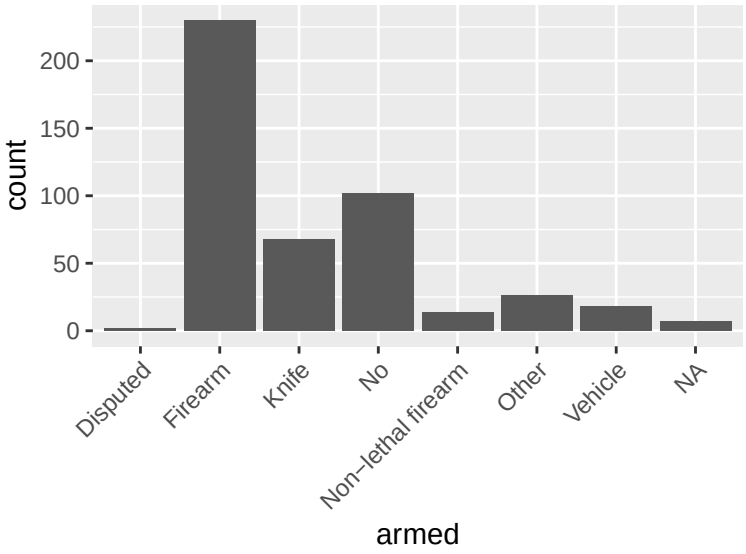
correct:



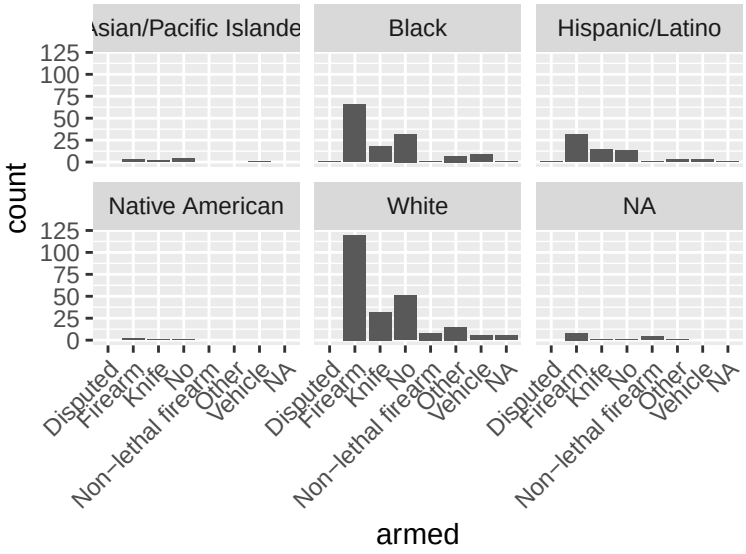
2. Plot the count of killings by `armed` (how/whether deceased was armed) facetting by `raceethnicity` (5 points)

```
ggplot(data = police_killings) +
  geom_bar(aes(x = armed, facet_wrap = raceethnicity)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Ignoring unknown aesthetics: facet_wrap
```



correct:



3. Plot a chart of the linear relationship between `age` and `urate` using `geom_smooth` and overlay a dot chart of each observation. (5 points)

```
ggplot(police_killings) +
  geom_smooth(aes(age, urate), method = 'lm') +
  geom_point(alpha = .5)
```

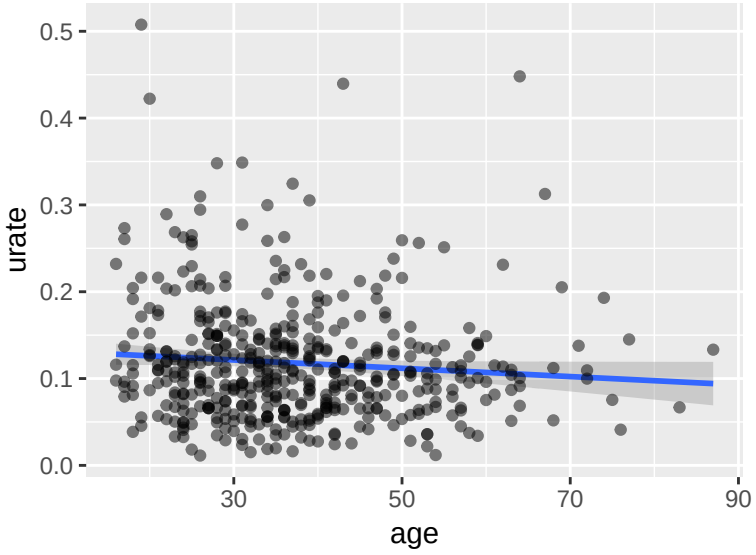
```
## Warning: Removed 6 rows containing non-finite values (stat_smooth).
```

```
## Error: geom_point requires the following missing aesthetics: x, y
```

correct:

```
## Warning: Removed 6 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



Data Analysis

4. Calculate the total count of killings by race/ethnicity and gender (5 points)

```
police_killings %>%
  group_by(raceethnicity, gender) %>%
  summarise(Total = sum(total))
```

```
## Error in summarise_impl(.data, dots): Evaluation error: object 'total' not found.
```

correct:

```
## # A tibble: 11 x 3
## # Groups:   raceethnicity [?]
##    raceethnicity          gender Total
##    <chr>                  <chr>  <int>
##  1 Asian/Pacific Islander Female     1
##  2 Asian/Pacific Islander Male       9
##  3 Black                  Female     7
##  4 Black                  Male     128
##  5 Hispanic/Latino        Female     1
##  6 Hispanic/Latino        Male      66
##  7 Native American        Male       4
##  8 White                  Female    11
##  9 White                  Male     225
## 10 <NA>                   Female     2
## 11 <NA>                   Male      13
```

5. Output a vector of killings' names with missing race/ethnicity (first 5 rows) (5 points)

```
nn <- police_killings %>%
       select(name == is.na(raceethnicity))
```

```
## Error in .f(.x[[i]], ...): object 'name' not found
```

```
nn$name[1:5]
```

```
## Error in eval(expr, envir, enclos): object 'nn' not found
```

correct:

```
## [1] "Brian Barbosa" "Ebin Proctor"  "Feras Morad"   "James Morris"
## [5] "Jessica Uribe"
```

6. Calculate the average share of White (`share_white`), Black (`share_black`) and Hispanic (`share_hispanic`) for the locations of the killings altogether (note that `share_*` refers to the percentage of a race/ethnicity in a killing's location) (6 points)

```
police_killings %>%
  gather('race', 'percent') %>%
  group_by(race) %>%
  summarise(avg  = mean(percent, na.rm = T))
```

```
## # A tibble: 34 x 2
##    race            avg
##    <chr>         <dbl>
##  1 age              NA
##  2 armed            NA
##  3 cause            NA
##  4 city             NA
##  5 college          NA
##  6 comp_income      NA
##  7 county_bucket    NA
##  8 county_fp        NA
##  9 county_id        NA
## 10 county_income    NA
## # ... with 24 more rows
```

correct:

```
## # A tibble: 3 x 2
##   race             avg
##   <chr>          <dbl>
## 1 share_black     17.9
## 2 share_hispanic  22.0
## 3 share_white     51.9
```

7. Count the number of killings by ethnicity for males under 25 (6 points)

```
police_killings %>%
  filter(gender == 'Male') %>%
  count(raceethnicity)
```

```
## # A tibble: 6 x 2
##   raceethnicity              n
##   <chr>                  <int>
## 1 Asian/Pacific Islander     9
## 2 Black                    128
## 3 Hispanic/Latino           66
## 4 Native American            4
## 5 White                    225
## 6 <NA>                      13
```

correct:

```
## # A tibble: 5 x 2
##   raceethnicity       n
##   <chr>           <int>
## 1 Black              25
## 2 Hispanic/Latino    18
## 3 Native American     1
## 4 White              23
## 5 <NA>                3
```

8. Calculate the month with the highest number of killings. Output a tibble of 1 row with the month that has the highest number of killings (6 points)

```
police_killings %>%
  count(month) %>%
  max(n)
```

```
## Error in FUN(X[[i]], ...): only defined on a data frame with all numeric variables
```

correct:

```
## # A tibble: 1 x 2
##   month     n
##   <chr> <int>
## 1 March   114
```

9. Use `purrr::map()` to render a separate ggplot bar chart of female/male count for each level of ethnicity. (7 points)

```
#output omitted for reasons of space
dt <- split(police_killings, police_killings$raceethnicity)
map(dt,  ~ ggplot(data = police_killings) + geom_bar(aes(x = gender)))
```

correct:

To make sure that the iteration works as expected, you want to check whether the last 5 plots rendered in the plots pane are correct (one for each ethnicity).

Functions

10. Use `purrr::map2` to calculate the mean for each element of `l`. Then round each mean by `r[i]` digits (check `?round()`). For instance, the first element is the result of `round(mean(c(2, 5, 9)), 0)`. (7 points)

```
l <- list( a = c(2, 5, 9), b = c(9, 29, 1, 4), c = c(6, 3, 1.5))
r <- c(a = 0, b = 2, b = 1)
map2(l, r, ~ mean() %>% round())
```

```
## Error in mean.default(): argument "x" is missing, with no default
```

correct:

```
## $a
## [1] 5
##
## $b
## [1] 10.75
##
## $c
## [1] 3.5
```

11. Use a function to calculate a percentage of killings by race/ethnicity given (1) an ethnicity name and (2) the column-name of the variable of interest. (7 points)

```
prc <- function(df, x, name, call. = FALSE) {
          if (!is.character(x) | is.character(name)) {stop('`x` must be a character')}
         sum(df[[x]] == name, na.rm = T) / nrow(df)

}
```

```
prc(police_killings, 'raceethnicity', 'Black')
```

```
## Error in prc(police_killings, "raceethnicity", "Black"): `x` must be a character
```

correct:

```
prc(police_killings, 'raceethnicity', 'Black')
```

```
## [1] 0.2890792
```

12. Convert the character string `c('5 Dec 2018')` to a vectir of class `Date` (3 points)

```
lubridate::ymd(c('5 Dec 2018'))
```

```
## Warning: All formats failed to parse. No formats found.
```

```
## [1] NA
```

correct:

```
## [1] "2018-12-05"
```

General R knowledge (TRUE/FALSE)

CORRECT ANSWER +3 points, MISSING ANSWER + 1 point, WRONG ANSWER -1 point

1. A `tibble` is also a `list`, but not every `list` is necessarily a `tibble`
2. When using logical operators (e.g., `&`, `>`, `|`) we always get a `NA` if an element on either the sides of the expression is `NA` (e.g. `1 == NA`, `NA & TRUE`, etc...)
3. If `df` is a tibble and `x` one of its variables, the two lines of code below retrieve the same output:

```
count(df, x)
df %>% group_by(x) %>% summarise( n = n())
```

4. The code snippets below retrieve the same output:

```
c(1, 4, 3) + 1            # 1.

c(1, 4, 3) + c(1, 1, 1)   # 2.
```

5. The code snippets below retrieve the same output when `x` is the same:

```
mean(is.na(x))              # 1.

x %>% is.na() %>% mean()    # 2.
```

6. The code snippets below retrieve the same output when `x` is the same:

```
police_killings %>% filter(age > x)      # 1.

police_killings %>% filter(!(age < x))   # 2.
```

7. The code snippets below both calculate the number of killings by gender and state

```
police_killings %>% count(gender, state)              # 1.

police_killings %>% group_by(gender, state) %>%       # 2.
  summarise(sum())
```

8. In ggplot2, we can overlay **at most** two geoms.
9. `filter` is for manipulating rows, while `slice` is for manipulating variables:
10. The output from `summarise()` gives as many rows as the grouping levels from `group_by()`
11. When using `gather()`, all the columns passed to `...` are collapsed into key-value pairs. Thus, the final output will always be a two-columns dataframe
12. In ggplot, we can map multiple variables to the same aesthetic