Assignment4

Liam

9/25/2019

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(readxl)
rmarkdown::github_document
```

```
## function (toc = FALSE, toc_depth = 3, fig_width = 7, fig_height = 5,
##     dev = "png", df_print = "default", includes = NULL, md_extensions = NULL,
##     hard_line_breaks = TRUE, pandoc_args = NULL, html_preview = TRUE)
## {
##     pandoc_args <- c(pandoc_args, "--template", pandoc_path_arg(rmarkdown_system_file("rmarkdown/temp
##     pandoc2 <- pandoc2.0()
##     variant <- if (pandoc2)
##         "gfm"
##     else "markdown_github"
##     if (!hard_line_breaks)
##         variant <- paste0(variant, "-hard_line_breaks")
##     variant <- paste0(variant, "-ascii_identifiers")
##     format <- md_document(variant = variant, toc = toc, toc_depth = toc_depth,
##         fig_width = fig_width, fig_height = fig_height, dev = dev,
##         df_print = df_print, includes = includes, md_extensions = md_extensions,
##         pandoc_args = pandoc_args)
##     format$pandoc$from <- gsub("+ascii_identifiers", "", format$pandoc$from,
##         fixed = TRUE)
##     if (html_preview) {
##         format$post_processor <- function(metadata, input_file,
##             output_file, clean, verbose) {
##             css <- pandoc_path_arg(rmarkdown_system_file("rmarkdown/templates/github_document/resour
##             args <- c("--standalone", "--self-contained", "--highlight-style",
##                 "pygments", "--template", pandoc_path_arg(rmarkdown_system_file("rmarkdown/templates/
##                 "--variable", paste0("github-markdown-css:",
##                   css), "--email-obfuscation", "none", if (pandoc2) c("--metadata",
##                   "pagetitle=PREVIEW"))
##             preview_file <- file_with_ext(output_file, "html")
```

```
##             pandoc_convert(input = output_file, to = "html",
##                 from = variant, output = preview_file, options = args,
##                 verbose = verbose)
##             preview_dir <- Sys.getenv("RMARKDOWN_PREVIEW_DIR",
##                 unset = NA)
##             if (!is.na(preview_dir)) {
##                 relocated_preview_file <- tempfile("preview-",
##                   preview_dir, ".html")
##                 file.copy(preview_file, relocated_preview_file)
##                 file.remove(preview_file)
##                 preview_file <- relocated_preview_file
##             }
##             if (verbose)
##                 message("\nPreview created: ", preview_file)
##             output_file
##         }
##     }
##     format
## }
## <bytecode: 0x000000001d6359c8>
## <environment: namespace:rmarkdown>
```

1. Compute the follows using `%>%` operator. **Notice that**

- `x %>%\ f = f(x)`,
- `x %>% f %>% g = g(f(x))` and
- `x %>% f(y) = f(x,y)`

a. sin(2019)

```
2019 %>%
  sin()
```

```
## [1] 0.8644605
```

b. sin(cos(2019))

```
2019 %>%
  sin() %>%
  cos()
```

```
## [1] 0.6490506
```

c. sin(cos(tan(log(2019))))

```
2019 %>%
  sin() %>%
  cos() %>%
  tan() %>%
  log()
```

```
## [1] -0.2761391
```

d. $\log_2(2019)$

```
2019 %>%
  log2()
```

```
## [1] 10.97943
```

2. Fixing the `SEX`, `AGE` and `TRAV_SP` following the steps in Assignment 2 (This time, do it on the entire dataset instead of the sample dataset).

```
c2015 <- read_excel('c2015.xlsx')
c2015 <- c2015 %>%
  mutate(TRAV_SP = as.numeric(substr(TRAV_SP, 1, 3))) %>%
  mutate(AGE = as.numeric(case_when(AGE=="Less than 1"~"0",
                                    TRUE~AGE))) %>%
  mutate(AGE=case_when(is.na(AGE) ~ mean(AGE,na.rm=TRUE),
                       TRUE ~ AGE)) %>%
  mutate(SEX=case_when(is.na(SEX) | SEX=="Not Rep" | SEX=="Unknown" ~ "Female",
                       TRUE~SEX))
```

```
## Warning: NAs introduced by coercion
```

```
## Warning: NAs introduced by coercion
```

3. Calculate the average age and average speed of female in the accident happened in the weekend.

Notice: These questions are to practice ***select_if*** *and* ***summarise_if, summarise_all...*** *functions in dplyr Check out the uses of these functions here and here.*

```
c2015 %>%
  mutate(dayType=case_when(DAY_WEEK=="Saturday" | DAY_WEEK=="Sunday" ~ "Weekend",
                           TRUE~"Weekday"))%>%
  filter(dayType=="Weekend") %>%
  filter(SEX=="Female") %>%
  summarise_at(c("TRAV_SP"), mean, na.rm=TRUE)
```

```
## # A tibble: 1 x 1
##   TRAV_SP
##     <dbl>
## 1    50.2
```

4. Use `select_if` and `is.numeric` functions to create a dataset with only numeric variables. Print out the names of all numeric variables

```
names(c2015 %>%
  select_if(is.numeric))
```

```
##  [1] "ST_CASE"   "VEH_NO"    "PER_NO"    "COUNTY"    "DAY"       "HOUR"
##  [7] "MINUTE"    "AGE"       "YEAR"      "TRAV_SP"   "LATITUDE" "LONGITUD"
```

5. Calculate the mean of all numeric variables using `select_if` and `summarise_all`

```
c2015 %>%
  select_if(is.numeric) %>%
  summarise_all(mean, na.rm=TRUE)
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 275607.   1.39   1.63   91.7  15.5  14.0   28.4  39.1  2015    49.9
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

6. We can shortcut 3 and 4 by using `summarise_if`: Use `summarise_if` to Calculate the mean of all numeric variables. (You may need to use `na.rm = TRUE` to ignore the NAs)

```
c2015 %>%
  summarise_if(is.numeric, mean, na.rm=TRUE)
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 275607.   1.39   1.63   91.7  15.5  14.0   28.4  39.1  2015    49.9
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

7. Use `summarise_if` to calculate the median of all numeric variables.

```
c2015 %>%
  summarise_if(is.numeric, median, na.rm=TRUE)
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1  270282      1      1     71    15    15     29    37  2015      53
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

8. Use `summarise_if` to calculate the standard deviation of all numeric variables. (`sd` function for standard deviation)

```
c2015 %>%
  summarise_if(is.numeric, sd, na.rm=TRUE)
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 163031.   1.45   1.84   95.0  8.78  9.06   17.3  20.1     0    20.9
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

9. Use `summarise_if` to calculate the number of missing values for each numeric variables. *Hint:* Use `~sum(is.na(.))`

```
c2015 %>%
  summarise_if(is.numeric, ~sum(is.na(.)))
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <int>  <int>  <int>  <int> <int> <int>  <int> <int> <int>   <int>
## 1       0      0      0      0     0     0    377     0     0   54549
## # ... with 2 more variables: LATITUDE <int>, LONGITUD <int>
```

10. Calculate the `log` of the average for each numeric variable.

```
c2015 %>%
  summarise_if(is.numeric, ~log(mean(.)))
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1    12.5  0.329  0.488   4.52  2.74  2.64     NA  3.67  7.61      NA
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

11. You will notice that there is one NA is produced in 10. Fix this by calculating the log of the absolute value average for each numeric variable.

```
c2015 %>%
  summarise_if(is.numeric, ~log(abs(mean(.))))
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1    12.5  0.329  0.488   4.52  2.74  2.64     NA  3.67  7.61      NA
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

12. Calculate the number of missing values for each categorical variables using `summarise_if`

```
c2015 %>%
  summarise_if(is.character, ~sum(is.na(.)))
```

```
## # A tibble: 1 x 16
##   STATE MONTH   SEX PER_TYP INJ_SEV SEAT_POS DRINKING MAN_COLL OWNER
##   <int> <int> <int>   <int>   <int>    <int>    <int>    <int> <int>
## 1     0     0     0       0       0        0        0     7197  7197
## # ... with 7 more variables: MOD_YEAR <int>, DEFORMED <int>,
## #   DAY_WEEK <int>, ROUTE <int>, HARM_EV <int>, LGT_COND <int>,
## #   WEATHER <int>
```

13. Calculate the number of missing values for each categorical variables using `summarise_all`

```
c2015 %>%
  select_if(is.character)%>%
  summarise_all(~sum(is.na(.)))
```

```
## # A tibble: 1 x 16
##   STATE MONTH   SEX PER_TYP INJ_SEV SEAT_POS DRINKING MAN_COLL OWNER
##   <int> <int> <int>   <int>   <int>    <int>    <int>    <int> <int>
## 1     0     0     0       0       0        0        0     7197  7197
## # ... with 7 more variables: MOD_YEAR <int>, DEFORMED <int>,
## #   DAY_WEEK <int>, ROUTE <int>, HARM_EV <int>, LGT_COND <int>,
## #   WEATHER <int>
```

14. Calculate the number of states in the dataset. **Hint: You can use `length(table())`

```
c2015 %>%
  summarise_at(c("STATE"),~length(table(STATE)))
```

```
## # A tibble: 1 x 1
##   STATE
##   <int>
## 1    51
```

15. Calculate the number of uniques values for each categorical variables using `summarise_if`.

```
c2015 %>%
  summarise_if(is.character, ~length(table(.)))
```

```
## # A tibble: 1 x 16
##   STATE MONTH   SEX PER_TYP INJ_SEV SEAT_POS DRINKING MAN_COLL OWNER
##   <int> <int> <int>   <int>   <int>    <int>    <int>    <int> <int>
## 1    51    12     2      11       8       29        4       11     8
## # ... with 7 more variables: MOD_YEAR <int>, DEFORMED <int>,
## #   DAY_WEEK <int>, ROUTE <int>, HARM_EV <int>, LGT_COND <int>,
## #   WEATHER <int>
```

16. Calculate the number of uniques values for each categorical variables using `summarise_all`.

```
c2015 %>%
  select_if(is.character)%>%
  summarise_all(~length(table(.)))
```

```
## # A tibble: 1 x 16
##   STATE MONTH   SEX PER_TYP INJ_SEV SEAT_POS DRINKING MAN_COLL OWNER
##   <int> <int> <int>   <int>   <int>    <int>    <int>    <int> <int>
## 1    51    12     2      11       8       29        4       11     8
## # ... with 7 more variables: MOD_YEAR <int>, DEFORMED <int>,
## #   DAY_WEEK <int>, ROUTE <int>, HARM_EV <int>, LGT_COND <int>,
## #   WEATHER <int>
```

17. Print out the names of all variables that have more than 30 distinct values

```
names(c2015 %>%
  select_if(~length(table(.))>30))
```

```
##  [1] "STATE"    "ST_CASE"  "VEH_NO"   "PER_NO"   "COUNTY"   "DAY"
##  [7] "MINUTE"   "AGE"      "MOD_YEAR" "TRAV_SP"  "LATITUDE" "LONGITUD"
## [13] "HARM_EV"
```

18. Print out the names of all categorical variables that more than 30 distinct values

```
names(c2015 %>%
  select_if(is.character)%>%
  select_if(~length(table(.))>30))
```

```
## [1] "STATE"    "MOD_YEAR" "HARM_EV"
```

19. Print out the names of all numeric variables that has the maximum values greater than 30

```
names(c2015 %>%
  select_if(is.numeric) %>%
  select_if(~max(table(.))>30))
```

```
##  [1] "ST_CASE"  "VEH_NO"   "PER_NO"   "COUNTY"   "DAY"      "HOUR"
##  [7] "MINUTE"   "AGE"      "YEAR"     "TRAV_SP"  "LATITUDE" "LONGITUD"
```

20. Calculate the mean of all numeric variables that has the maximum values greater than 30 using 'summarise_if'

```
c2015 %>%
  select_if(is.numeric) %>%
  summarize_if(~max(table(.))>30, mean, na.rm=TRUE)
```

```
## # A tibble: 1 x 12
##   ST_CASE VEH_NO PER_NO COUNTY   DAY  HOUR MINUTE   AGE  YEAR TRAV_SP
##     <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 275607.   1.39   1.63   91.7  15.5  14.0   28.4  39.1  2015    49.9
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

21. Calculate the mean of all numeric variables that has the maximum values greater than 30 using 'summarise_all'

```
names(c2015 %>%
  select_if(is.numeric) %>%
  select_if(~max(table(.))>30) %>%
  summarise_all(mean, na.rm=TRUE))
```

```
##  [1] "ST_CASE"  "VEH_NO"   "PER_NO"   "COUNTY"   "DAY"      "HOUR"
##  [7] "MINUTE"   "AGE"      "YEAR"     "TRAV_SP"  "LATITUDE" "LONGITUD"
```

22. Create a dataset containing variables with standard deviation greater than 10. Call this data `d1`

```
temp <- names(c2015 %>%
                select_if(is.numeric))
temp <- temp[c2015 %>%
  select_if(is.numeric) %>%
  summarise_all(sd, na.rm=TRUE) > 10]
d1<-c2015 %>%
  select(temp)
```

23. Centralizing a variable is subtract it by its mean. Centralize the variables of `d1` using `mutate_all`. Check the means of all centralized variables to confirm that they are all zeros.

```
d1 %>%
  mutate_all(funs(.- mean(., na.rm=TRUE))) %>%
  summarize_all(mean, na.rm=TRUE)
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
## # A tibble: 1 x 6
##    ST_CASE   COUNTY    MINUTE      AGE  TRAV_SP  LONGITUD
##      <dbl>    <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 4.73e-11 1.32e-14 -1.25e-15 1.58e-15 3.25e-15 -6.92e-15
```

24. Standarizing a variable is to subtract it to its mean and then divide by its standard deviation. Standardize the variables of `d1` using `mutate_all`. Check the means and standard deviation of all centralized variables to confirm that they are all zeros (for the means) and ones (for standard deviation).

```
d1 %>%
  mutate_all(funs(.- mean(., na.rm=TRUE))) %>%
  mutate_all(funs(./sd(.))) %>%
  summarize_all(c(mean, sd), na.rm = TRUE)
```

```
## # A tibble: 1 x 12
##   ST_CASE_fn1 COUNTY_fn1 MINUTE_fn1  AGE_fn1 TRAV_SP_fn1 LONGITUD_fn1
##         <dbl>      <dbl>      <dbl>    <dbl>       <dbl>        <dbl>
## 1   -9.97e-17   1.15e-16        NaN 8.49e-17         NaN          NaN
## # ... with 6 more variables: ST_CASE_fn2 <dbl>, COUNTY_fn2 <dbl>,
## #   MINUTE_fn2 <dbl>, AGE_fn2 <dbl>, TRAV_SP_fn2 <dbl>, LONGITUD_fn2 <dbl>
```