Assignment-3

Liam

9/18/2019

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

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

## v ggplot2 3.2.1     v readr   1.3.1
## v tibble  2.1.3     v purrr   0.3.2
## v tidyr   0.8.3     v stringr 1.4.0
## v ggplot2 3.2.1     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: 0x000000001da05650>
## <environment: namespace:rmarkdown>
```

```r
titanic<-read_csv("C:\\Users\\student\\Desktop\\Fall2019\\R\\titanic.csv")
```

```
## Parsed with column specification:
## cols(
##   PassengerId = col_double(),
##   Survived = col_double(),
##   Pclass = col_double(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_double(),
##   Parch = col_double(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

1. Read the titanic data set as a tibble. Redo questions 13 to 23 in the Assignment 1 using dplyr. Notice: you may want to use logical operators such as:

Operators	Discription
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y

1.13. Calculate the mean age of female passengers

```
titanic %>%
  filter(Sex == 'female') %>%
  summarise(meanAge=mean(Age, na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   meanAge
##     <dbl>
## 1    27.9
```

1.14. Calculate the median fare of the passengers in Class 1

```
titanic %>%
  filter(Pclass=='1') %>%
  summarise(medFare=median(Fare, na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   medFare
##     <dbl>
## 1    60.3
```

1.15. Calculate the median fare of the female passengers that are not in Class 1

```
titanic %>%
  filter(Pclass!=1 & Sex=='female') %>%
  summarise(medFare=median(Fare, na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   medFare
##     <dbl>
## 1    14.5
```

1.16. Calculate the median age of survived passengers who are female and Class 1 or Class 2,

```
titanic %>%
  filter(Survived==1 & Sex=='female' & (Pclass==1 | Pclass==2)) %>%
  summarise(medAge=median(Age, na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   medAge
##    <dbl>
## 1     31
```

1.17. Calculate the mean fare of female teenagers survived passengers

```
titanic %>%
  filter(Sex=='female' & Age > 12 & Age < 20 & Survived==1) %>%
  summarise(meanFare= mean(Fare, na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   meanFare
##      <dbl>
## 1     49.2
```

1.18. Calculate the mean fare of female teenagers survived passengers for each class

```
titanic %>%
  filter(Sex=='female' & Age > 12 & Age < 20 & Survived==1) %>%
  group_by(Pclass) %>%
  summarise(meanFare=mean(Fare, na.rm=TRUE))
```

```
## # A tibble: 3 x 2
##   Pclass meanFare
##    <dbl>    <dbl>
## 1      1   108.
## 2      2    20.0
## 3      3     8.77
```

1.19. Calculate the ratio of Survived and not Survived for passengers who are who pays more than the average fare

```
titanic %>%
  filter(Fare > mean(Fare)) %>%
  count(Survived) %>%
  mutate(freq=n/sum(n))%>%
  select(Survived, freq)
```

```
## # A tibble: 2 x 2
##   Survived  freq
##      <dbl> <dbl>
## 1        0 0.403
## 2        1 0.597
```

1.20. Add column that standardizes the fare (subtract the mean and divide by standard deviation) and name it sfare

```
titanic <- titanic %>%
  mutate(sfare=(Fare-mean(Fare))/sd(Fare))
```

1.21. Add categorical variable named cfare that takes value cheap for passengers paying less the average fare and takes value expensive for passengers paying more than the average fare.

```
titanic <- titanic %>%
  mutate(cfare=case_when(Fare<=mean(Fare)~'cheap',
                         Fare>mean(Fare)~'Expensive'))
```

1.22. Add categorical variable named cage that takes value 0 for age 0-10, 1 for age 10-20, 2 for age 20-30, and so on

```
titanic<- titanic %>%
  mutate(cage=case_when(Age<10~'0',
                        Age<20~'1',
                        Age<30~'2',
                        Age<40~'3',
                        Age<50~'4',
                        Age<60~'5',
                        Age<70~'6',
                        Age<80~'7',
                        Age<90~'8',
                        Age>=90~'9'))
  #cut
```

1.23. Show the frequency of Ports of Embarkation. It appears that there are two missing values in the Embarked variable. Assign the most frequent port to the missing ports. Hint: Use the levels function to modify the categories of categorical variables.

```
titanic%>%
  count(Embarked)
```

```
## # A tibble: 4 x 2
##   Embarked     n
##   <chr>    <int>
## 1 C          168
## 2 Q           77
## 3 S          644
## 4 <NA>         2
```

```
titanic %>%
  mutate(Embarked = case_when(is.na(Embarked)~'S',
                              TRUE~Embarked)) %>%
  count(Embarked)
```

```
## # A tibble: 3 x 2
##   Embarked     n
##   <chr>    <int>
## 1 C          168
## 2 Q           77
## 3 S          646
```

2. Using Dplyr and in Assignment 2, redo 4 using `sample_n` function, redo 5 using `glimpse`, redo 11, 12 and 13. For 11, 12 and 13, you may want to use the combo `group_by` and `summarise`

```
c2015 <- read_excel('c2015.xlsx')
```

2.4. Use `dim` function to check the dimension of the data. Since this data is quite big, a common practice is to randomly subset the data to analyze. Use `sample` function to create a new dataset that has a random 1000 observations from the original data. Use `set.seed(2019)` before using the `sample` function to set the seed for the randomness so that everyone in class is working with the same random subset of the data.

```
dim(c2015)
```

```
## [1] 80587    28
```

```
set.seed(2019)
c2015Sample <-c2015[sample(nrow(c2015), 1000),]
```

2.5. Use `summary` function to have a quick look at the data. You will notice there is one variable is actually a constant. Remove that variable from the data.

```
summary(c2015Sample)
```

```
##     STATE              ST_CASE           VEH_NO           PER_NO
##  Length:1000        Min.   : 10020   Min.   : 0.000   Min.   : 1.000
##  Class :character   1st Qu.:122408   1st Qu.: 1.000   1st Qu.: 1.000
##  Mode  :character   Median :270249   Median : 1.000   Median : 1.000
##                     Mean   :276444   Mean   : 1.385   Mean   : 1.697
##                     3rd Qu.:420726   3rd Qu.: 2.000   3rd Qu.: 2.000
##                     Max.   :560071   Max.   :13.000   Max.   :48.000
##
##      COUNTY             DAY           MONTH                HOUR
##  Min.   :  1.00   Min.   : 1.00   Length:1000        Min.   : 0.00
##  1st Qu.: 32.50   1st Qu.: 8.00   Class :character   1st Qu.: 8.00
##  Median : 71.00   Median :16.00   Mode  :character   Median :16.00
##  Mean   : 93.05   Mean   :15.89                      Mean   :14.26
##  3rd Qu.:117.00   3rd Qu.:24.00                      3rd Qu.:20.00
##  Max.   :810.00   Max.   :31.00                      Max.   :99.00
##
##      MINUTE          AGE                SEX              PER_TYP
##  Min.   : 0.00   Length:1000        Length:1000        Length:1000
##  1st Qu.:14.00   Class :character   Class :character   Class :character
##  Median :27.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :27.76
##  3rd Qu.:43.00
##  Max.   :59.00
##  NA's   :5
##    INJ_SEV            SEAT_POS           DRINKING              YEAR
##  Length:1000        Length:1000        Length:1000        Min.   :2015
##  Class :character   Class :character   Class :character   1st Qu.:2015
##  Mode  :character   Mode  :character   Mode  :character   Median :2015
##                                                           Mean   :2015
##                                                           3rd Qu.:2015
##                                                           Max.   :2015
##
##    MAN_COLL            OWNER            MOD_YEAR
##  Length:1000        Length:1000        Length:1000
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    TRAV_SP            DEFORMED           DAY_WEEK
##  Length:1000        Length:1000        Length:1000
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##     ROUTE              LATITUDE        LONGITUD         HARM_EV
##  Length:1000        Min.   :21.30   Min.   :-160.34   Length:1000
##  Class :character   1st Qu.:33.48   1st Qu.: -97.59   Class :character
##  Mode  :character   Median :36.42   Median : -87.43   Mode  :character
##                     Mean   :36.72   Mean   : -91.83
##                     3rd Qu.:40.40   3rd Qu.: -81.41
##                      Max.   :61.54   Max.   : -67.72
##                      NA's   :7       NA's   :7
##    LGT_COND           WEATHER
##  Length:1000        Length:1000
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
c2015Sample<- c2015Sample %>%
  select(-c(YEAR))
```

2.11. Compare the average speed of those who had `"No Apprent Injury"` and the rest. What do you observe?

```
c2015Sample %>%
  mutate(inj=case_when(INJ_SEV=='No Apparent Injury (O)' ~ 'No Apparent Injury',
                       TRUE ~ 'Injury'))%>%
  select(inj, TRAV_SP) %>%
  group_by(inj) %>%
  mutate(trav_sp = as.numeric(substr(TRAV_SP, 1, 3))) %>%
  summarise(avgSpd = mean(trav_sp, na.rm=TRUE))
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## # A tibble: 2 x 2
##   inj                avgSpd
##   <chr>               <dbl>
## 1 Injury               53.1
## 2 No Apparent Injury   44.6
```

```
c2015Sample <- c2015Sample %>%
    mutate(TRAV_SP = as.numeric(substr(TRAV_SP, 1, 3)))
```

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

2.12. Use the `SEAT_POS` variable to filter the data so that there is only drivers in the dataset. Compare the average speed of man drivers and woman drivers. Comment on the results.

```
c2015Sample %>%
  filter(SEAT_POS=='Front Seat, Left Side') %>%
  filter(SEX != 'Unknown' & is.na(SEX)==FALSE) %>%
  group_by(SEX) %>%
  summarise(avgSpd=mean(TRAV_SP, na.rm=TRUE))
```

```
## # A tibble: 2 x 2
##   SEX    avgSpd
##   <chr>   <dbl>
## 1 Female   46.1
## 2 Male     51.7
```

##2.13. Compare the average speed of drivers who drink and those who do not. Comment on the results. **Hint:** This calculation can be done manually or by using the `aggregate` function or `by` function in base R. For example:

```
c2015Sample %>%
  mutate(DRINKING=case_when(DRINKING=='Yes (Alcohol Involved)'~ 'Y',
                            TRUE~'N')) %>%
  group_by(DRINKING) %>%
  summarise(avgSpd=mean(TRAV_SP, na.rm=TRUE))
```

```
## # A tibble: 2 x 2
##   DRINKING avgSpd
##   <chr>     <dbl>
## 1 N          48.9
## 2 Y          68.6
```

3. Calculate the travel speed (`TRAV_SP` variable) by day. Compare the travel speed of the first 5 days and the last 5 days of months. (Day 1-5 vs Day 26-30)

```
c2015Sample %>%
  mutate(DAY=case_when(DAY<=5~'DAY 1-5',
                       DAY>=26~'DAY 26-30',
                       TRUE~'OTHER'))%>%
  group_by(DAY) %>%
  summarise(avgSpd=mean(TRAV_SP, na.rm=TRUE))
```

```
## # A tibble: 3 x 2
##   DAY       avgSpd
##   <chr>      <dbl>
## 1 DAY 1-5     50.7
## 2 DAY 26-30   53.4
## 3 OTHER       50.2
```

4. Calculate the travel speed (`TRAV_SP` variable) by day of the week. Compare the travel speed of the weekdays and weekends.

```
c2015Sample %>%
  group_by(DAY_WEEK) %>%
  summarise(aveTRAV_SP=mean(TRAV_SP, na.rm=TRUE))
```

```
## # A tibble: 7 x 2
##   DAY_WEEK  aveTRAV_SP
##   <chr>          <dbl>
## 1 Friday          50.7
## 2 Monday          48.6
## 3 Saturday        53.3
## 4 Sunday          55.8
## 5 Thursday        50.8
## 6 Tuesday         47.2
## 7 Wednesday       44.7
```

```
c2015Sample %>%
  mutate(dayType=case_when(DAY_WEEK=="Saturday" | DAY_WEEK=="Sunday" ~ "Weekend",
                           TRUE~"Weekday"))%>%
  group_by(dayType) %>%
  summarise(aveTRAV_SPD = mean(TRAV_SP, na.rm=TRUE))
```

```
## # A tibble: 2 x 2
##   dayType aveTRAV_SPD
##   <chr>         <dbl>
## 1 Weekday        48.7
## 2 Weekend        54.4
```

5. Find the top 5 states with greatest travel speed.

```
c2015Sample %>%
  select(STATE, TRAV_SP) %>%
  group_by(STATE) %>%
  summarize(aveTRAV_SP=mean(TRAV_SP, na.rm=TRUE)) %>%
  arrange(desc(aveTRAV_SP)) %>%
  top_n(5, aveTRAV_SP)
```

```
## # A tibble: 5 x 2
##   STATE        aveTRAV_SP
##   <chr>             <dbl>
## 1 South Dakota      107
## 2 North Dakota       85
## 3 Nevada             73.5
## 4 Wyoming            66.5
## 5 Kentucky           65.4
```

6. Rank the travel speed by MONTH.

```
c2015Sample %>%
  select(MONTH, TRAV_SP) %>%
  group_by(MONTH) %>%
  summarize(aveTRAV_SP=mean(TRAV_SP, na.rm=TRUE)) %>%
  arrange(desc(aveTRAV_SP))
```

```
## # A tibble: 12 x 2
##    MONTH     aveTRAV_SP
##    <chr>          <dbl>
##  1 April           59.3
##  2 December        59.0
##  3 September       54.7
##  4 June            53.4
##  5 October         52.5
##  6 November        52.5
##  7 August          48.9
##  8 May             48.3
##  9 February        46.4
## 10 March           45.4
## 11 January         45.2
## 12 July            44.9
```

7. Find the average speed of teenagers in December.

```
c2015Sample %>%
  filter(AGE>12 & AGE <20) %>%
  select(MONTH, TRAV_SP) %>%
  group_by(MONTH) %>%
  summarize(aveTRAV_SP=mean(TRAV_SP, na.rm=TRUE)) %>%
  filter(MONTH=="December") %>%
  arrange(desc(aveTRAV_SP))
```

```
## # A tibble: 1 x 2
##   MONTH    aveTRAV_SP
##   <chr>         <dbl>
## 1 December         80
```

8. Find the month that female drivers drive fastest on average.

```
c2015Sample %>%
  filter(SEX=="Female") %>%
  select(MONTH, TRAV_SP) %>%
  group_by(MONTH) %>%
  summarize(aveTRAV_SP=mean(TRAV_SP, na.rm=TRUE)) %>%
  top_n(1, aveTRAV_SP)
```

```
## # A tibble: 1 x 2
##   MONTH    aveTRAV_SP
##   <chr>         <dbl>
## 1 December       60.3
```

9. Find the month that male driver drive slowest on average.

```
c2015Sample %>%
  filter(SEX=="Male") %>%
  select(MONTH, TRAV_SP) %>%
  group_by(MONTH) %>%
  summarize(aveTRAV_SP=mean(TRAV_SP, na.rm=TRUE)) %>%
  top_n(-1, aveTRAV_SP)
```

```
## # A tibble: 1 x 2
##   MONTH    aveTRAV_SP
##   <chr>         <dbl>
## 1 February         38
```

10. Create a new column containing information about the season of the accidents. Compare the percentage of Fatal Injury by seasons.

```
c2015Sample %>%
  mutate(SEASON=case_when(MONTH=="March" | MONTH=="April" | MONTH=="May" ~ "Spring",
                          MONTH=="June" | MONTH=="July" | MONTH=="August" ~ "Summer",
                          MONTH=="September" | MONTH=="October" | MONTH=="November" ~ "Fall",
                          TRUE~"Winter")) %>%
  group_by(SEASON) %>%
  summarise(prop.table(table(INJ_SEV))[4])
```

```
## # A tibble: 4 x 2
##   SEASON `prop.table(table(INJ_SEV))[4]`
##   <chr>                            <dbl>
## 1 Fall                            0.0845
## 2 Spring                          0.101
## 3 Summer                          0.0890
## 4 Winter                          0.116
```

11. Compare the percentage of fatal injuries for different type of deformations (DEFORMED variable)

```
c2015Sample %>%
  group_by(DEFORMED) %>%
  summarise(prop.table(table(INJ_SEV))[4])
```

```
## # A tibble: 7 x 2
##   DEFORMED          `prop.table(table(INJ_SEV))[4]`
##   <chr>                                       <dbl>
## 1 Disabling Damage                           0.1
## 2 Functional Damage                          0.0690
## 3 Minor Damage                               0.0128
## 4 No Damage                                 NA
## 5 Not Reported                               0.0455
## 6 Unknown                                    0.1
## 7 <NA>                                       0.0316
```