A4

McKenzie Kozma

9/25/2019

1. Using %>% operator.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

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

```
library(readxl)

  2019 %>% sin()
```

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

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

```
## [1] -0.4817939
```

```
  2019 %>% log10() %>% tan() %>% cos() %>% sin()
```

```
## [1] 0.8340538
```

```
  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).

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

```
c2015 <- read_xlsx("c2015.xlsx")

c2015$SEX[c2015$SEX == "Unknown"] <- "Female"
c2015$AGE[c2015$AGE == "Less than 1"] <- "0"
c2015$AGE <- as.numeric(c2015$AGE)
```

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

```
c2015$AGE[is.na(c2015$AGE)] <- mean(c2015$AGE, na.rm = TRUE)

c2015$TRAV_SP <- as.numeric(str_remove(c2015$TRAV_SP, "MPH"))
```

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

```
c2015 <- c2015[!(is.na(c2015$TRAV_SP)), ]

c2015 %>%
  filter(SEX == "Female" & DAY_WEEK %in% c("Saturday", "Sunday")) %>%
  summarise_at(vars(AGE, TRAV_SP), mean, na.rm = TRUE)
```

```
## # A tibble: 1 x 2
##     AGE TRAV_SP
##   <dbl>   <dbl>
## 1  36.1    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

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

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

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

```
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 250204.   1.49   1.66   74.2  15.5  13.8   28.8  38.7  2015    49.9
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

```
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 250204.   1.49   1.66   74.2  15.5  13.8   28.8  38.7  2015    49.9
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

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

8. Use summarise_if to calculate the standard deviation of all numeric variables. (sd function for standard deviation)
9. Use summarise_if to calculate the number of missing values for each numeric variables. Hint: Use ~sum(is.na(.))
10. Calculate the log of the average for each numeric variable.
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, 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 220376.      1      1     67    15    15     30    36  2015      53
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

```
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 170029.   1.26   1.68   72.5  8.79  7.70   17.4  20.3     0    20.9
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

```
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     43     0     0       0
## # ... with 2 more variables: LATITUDE <int>, LONGITUD <int>
```

```
c2015 %>% summarise_if(is.numeric, ~log(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    12.4  0.397  0.507   4.31  2.74  2.63     NA  3.66  7.61    3.91
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

```
c2015 %>% summarise_if(is.numeric, ~log(abs(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    12.4  0.397  0.507   4.31  2.74  2.63   3.36  3.66  7.61    3.91
## # ... with 2 more variables: LATITUDE <dbl>, LONGITUD <dbl>
```

12. Calculate the number of missing values for each categorical variables using summarise_if
13. Calculate the number of missing values for each categorical variables using summarise_all
14. Calculate the number of states in the dataset. **Hint: You can use length(table())
15. Calculate the number of uniques values for each categorical variables using summarise_if
16. Calculate the number of uniques values for each categorical variables using summarise_all

```
c2015 %>% summarise_if(~!is.numeric(.), ~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        0     0
## # ... with 7 more variables: MOD_YEAR <int>, DEFORMED <int>,
## #   DAY_WEEK <int>, ROUTE <int>, HARM_EV <int>, LGT_COND <int>,
## #   WEATHER <int>
```

```
c2015 %>% select_if(~!is.numeric(.)) %>%
  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        0     0
## # ... with 7 more variables: MOD_YEAR <int>, DEFORMED <int>,
## #   DAY_WEEK <int>, ROUTE <int>, HARM_EV <int>, LGT_COND <int>,
## #   WEATHER <int>
```

```
c2015 %>% group_by(STATE) %>%
  summarise() %>%
  nrow()
```

```
## [1] 51
```

```
c2015 %>% summarise_if(~!is.numeric(.), ~length(unique(.)))
```

```
## # 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     3       3       8       26        4       10     8
## # ... with 7 more variables: MOD_YEAR <int>, DEFORMED <int>,
## #   DAY_WEEK <int>, ROUTE <int>, HARM_EV <int>, LGT_COND <int>,
## #   WEATHER <int>
```

```
c2015 %>% select_if(~!is.numeric(.)) %>%
  summarise_all(~length(unique(.)))
```

```
## # 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     3       3       8       26        4       10     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
18. Print out the names of all categorical variables that more than 30 distinct values
19. Print out the names of all numeric variables that has the maximum values greater than 30

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

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

```
c2015 %>% select_if(~length(unique(.)) > 30 & !is.numeric(.)) %>%
  names()
```

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

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

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

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

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

```
## # A tibble: 1 x 11
##   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 250204.   1.49   1.66   74.2  15.5  13.8   28.8  38.7  2015    49.9
## # ... with 1 more variable: LATITUDE <dbl>
```

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

```
## # A tibble: 1 x 11
##   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 250204.   1.49   1.66   74.2  15.5  13.8   28.8  38.7  2015    49.9
## # ... with 1 more variable: LATITUDE <dbl>
```

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

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.

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 <- c2015 %>%
  select_if(is.numeric) %>%
  select_if(~sd(., na.rm = TRUE) > 10)

d2 <- d1 %>%
  mutate_all(~(. - mean(., na.rm = TRUE)))
d2 %>% summarise_all(mean, na.rm = TRUE)
```

```
## # A tibble: 1 x 6
##    ST_CASE   COUNTY    MINUTE       AGE  TRAV_SP  LONGITUD
##      <dbl>    <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 1.91e-11 6.38e-15 -4.86e-16 -1.45e-15 3.25e-15 -1.66e-15
```

```
d3 <- d2 %>%
  mutate_all(~(./sd(., na.rm = TRUE)))
d3 %>% summarise_all(funs(mean, sd), 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 12
##   ST_CASE_mean COUNTY_mean MINUTE_mean  AGE_mean TRAV_SP_mean LONGITUD_mean
##          <dbl>       <dbl>       <dbl>     <dbl>        <dbl>         <dbl>
## 1    -3.27e-17    6.03e-17   -3.19e-17 -7.27e-17     1.57e-16     -7.66e-17
## # ... with 6 more variables: ST_CASE_sd <dbl>, COUNTY_sd <dbl>,
## #   MINUTE_sd <dbl>, AGE_sd <dbl>, TRAV_SP_sd <dbl>, LONGITUD_sd <dbl>
```