TidyVerse Create and Extend Vignette

Contents

Title: "CUNY SPS MDS DATA607_Tyderverse Create & Extend"

Author: "Charles Ugiagbe (create) and Mark Schmalfeld (extend)"

Date: "11/23/2021"

Introduction

Tidyverse is just a collection of R packages underlying same design philosophy, grammar, and data structure. There are currently 8 packages in the `tidyverse` package bundle including:

- `dplyr`: a set of tools for efficiently manipulating datasets;
- `forcats`: a package for manipulating categorical variables / factors;
- `ggplots2`: a classic package for data visualization;
- `purrr`: another set of tools for manipulating datasets, specially vecters, a complement to `dplyr`;
- `readr`: a set of faster and more user friendly functions to read data than R default functions;
- `stringr`: a package for common string operations;
- `tibble` a package for reimagining data.frames in a modern way;
- `tidyr`: a package for reshaping data, a complement to `dplyr`.

In this assignment, I will use some handy functions in tidyverse package to perform some Analysis

```
library(tidyverse)
```

Dataset

The dataset in this project is called "student performance" from https://www.kaggle.com/datasets; The dataset contains a sample of 1000 observations of 8 variables.

I use `read.csv` function to import the csv file to R.

```
url <- "https://raw.githubusercontent.com/omocharly/DATA607_PROJECTS/main/StudentsPerformance.csv"
data <- read.csv(url, header = TRUE)
```

```
head(data)
```

```
##   gender race.ethnicity parental.level.of.education        lunch
## 1 female        group B           bachelor's degree     standard
## 2 female        group C                some college     standard
## 3 female        group B              master's degree     standard
## 4   male        group A           associate's degree free/reduced
## 5   male        group C                some college     standard
## 6 female        group B           associate's degree     standard
##   test.preparation.course math.score reading.score writing.score
## 1                    none         72            72            74
## 2               completed         69            90            88
## 3                    none         90            95            93
## 4                    none         47            57            44
## 5                    none         76            78            75
## 6                    none         71            83            78
```

Glimpse help us to catch sight of the data to see the data structure.

```
glimpse(data)
```

```
## Rows: 1,000
## Columns: 8
## $ gender                      <chr> "female", "female", "female", "male", "mal~
## $ race.ethnicity              <chr> "group B", "group C", "group B", "group A"~
## $ parental.level.of.education <chr> "bachelor's degree", "some college", "mast~
## $ lunch                       <chr> "standard", "standard", "standard", "free/~
## $ test.preparation.course     <chr> "none", "completed", "none", "none", "none~
## $ math.score                  <int> 72, 69, 90, 47, 76, 71, 88, 40, 64, 38, 58~
## $ reading.score               <int> 72, 90, 95, 57, 78, 83, 95, 43, 64, 60, 54~
## $ writing.score               <int> 74, 88, 93, 44, 75, 78, 92, 39, 67, 50, 52~
```

dplyr::rename()

rename() changes the names of individual variables using in a column with a new one

```
data1 <- data %>% rename(race = race.ethnicity, parental_Educatn_level= parental.level.of.education, tes
head(data1)
```

```
##   gender    race parental_Educatn_level        lunch test.prep math.score
## 1 female group B      bachelor's degree     standard      none         72
## 2 female group C           some college     standard completed         69
## 3 female group B        master's degree     standard      none         90
## 4   male group A     associate's degree free/reduced      none         47
## 5   male group C           some college     standard      none         76
## 6 female group B     associate's degree     standard      none         71
##   reading.score writing.score
## 1            72            74
## 2            90            88
## 3            95            93
## 4            57            44
## 5            78            75
## 6            83            78
```

dplyr::select()

Select(): is use for selecting a range of consecutive variables or taking the complement of a set of variables

```
data2 <- data1 %>%
  select(gender, math.score, reading.score, writing.score)
head(data2)
```

```
##   gender math.score reading.score writing.score
## 1 female         72            72            74
## 2 female         69            90            88
## 3 female         90            95            93
## 4   male         47            57            44
## 5   male         76            78            75
## 6 female         71            83            78
```

```
data2b<- data1 %>%
  select( gender,test.prep,math.score,reading.score,writing.score)
head(data2b)
```

```
##   gender test.prep math.score reading.score writing.score
## 1 female      none         72            72            74
## 2 female completed         69            90            88
## 3 female      none         90            95            93
## 4   male      none         47            57            44
## 5   male      none         76            78            75
## 6 female      none         71            83            78
```

```
tail(data2b)
```

```
##      gender test.prep math.score reading.score writing.score
## 995    male      none         63            63            62
## 996  female completed         88            99            95
## 997    male      none         62            55            55
## 998  female completed         59            71            65
## 999  female completed         68            78            77
## 1000 female      none         77            86            86
```

dplyr::filter()

I use the filter() function to filter maths, writing and reading scores that are greater than 95

```
data3 <- data2 %>%
  filter(math.score == 100, writing.score > 95, reading.score > 95)
data3
```

```
##   gender math.score reading.score writing.score
## 1 female        100           100           100
## 2   male        100            97            99
## 3   male        100           100           100
## 4 female        100           100           100
```

dplyr::arrange()

arrange(): orders the rows of a data frame by the values of selected columns.

```
data4 <- data2 %>% arrange(desc(math.score))
head(data4)
```

```
##   gender math.score reading.score writing.score
## 1   male        100           100            93
## 2 female        100            92            97
## 3 female        100           100           100
## 4   male        100            96            86
## 5   male        100            97            99
## 6   male        100           100           100
```

dplyr::mutate()

mutate() adds new variables that are function of the existing ones to the table and also preserves existing ones.

```
data5 <- data4 %>%
  mutate(avg.score = (math.score + writing.score + reading.score) / 3)
head(data5)
```

```
##   gender math.score reading.score writing.score avg.score
## 1   male        100           100            93  97.66667
## 2 female        100            92            97  96.33333
## 3 female        100           100           100 100.00000
## 4   male        100            96            86  94.00000
## 5   male        100            97            99  98.66667
## 6   male        100           100           100 100.00000
```

```
data5<- data5 %>%
  mutate(avg.read_write_score =(writing.score+reading.score)/2)
head(data5)
```

```
##   gender math.score reading.score writing.score avg.score avg.read_write_score
## 1   male        100           100            93  97.66667                  96.5
## 2 female        100            92            97  96.33333                  94.5
## 3 female        100           100           100 100.00000                 100.0
## 4   male        100            96            86  94.00000                  91.0
## 5   male        100            97            99  98.66667                  98.0
## 6   male        100           100           100 100.00000                 100.0
```

dplyr::case_when()

Case_when: Function allows you to vectorize multiple if_else() statements. It is an R equivalent of a SQL CASE WHEN statement.

```
data6 <- data5 %>%
  mutate(pass_fail_grade = case_when(avg.score >= 85 ~ 'Pass'
                                     ,TRUE ~ 'Fail' )
         )
head(data6)
```

```
##   gender math.score reading.score writing.score avg.score avg.read_write_score
## 1   male        100           100            93  97.66667                  96.5
## 2 female        100            92            97  96.33333                  94.5
## 3 female        100           100           100 100.00000                 100.0
## 4   male        100            96            86  94.00000                  91.0
## 5   male        100            97            99  98.66667                  98.0
## 6   male        100           100           100 100.00000                 100.0
##   pass_fail_grade
## 1            Pass
## 2            Pass
## 3            Pass
## 4            Pass
## 5            Pass
## 6            Pass
```

dplyr::summarize()

```
data %>% group_by(gender) %>%
  summarize( math_score = sum (math.score)/ n())
```

```
## # A tibble: 2 x 2
##   gender math_score
##   <chr>       <dbl>
## 1 female       63.6
## 2 male         68.7
```

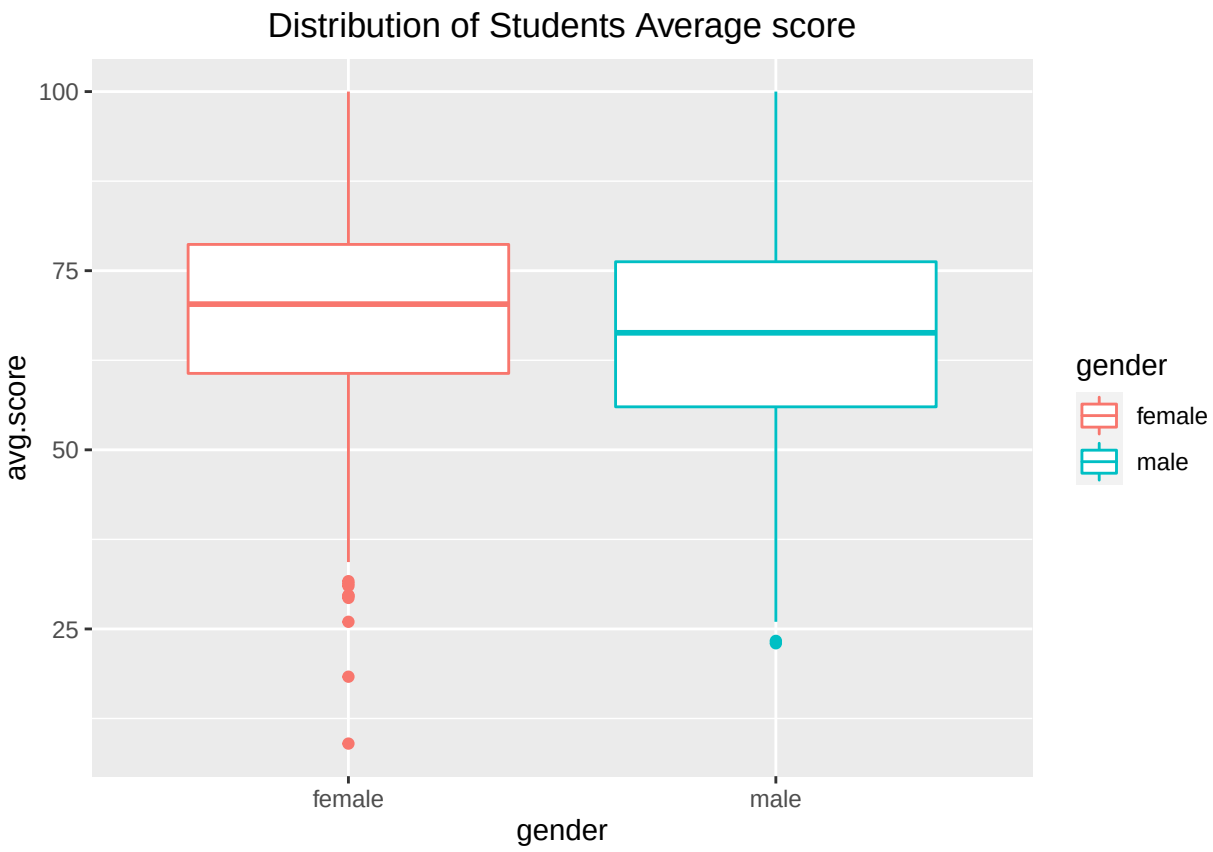
```
data %>% group_by(parental.level.of.education) %>%
  summarize(math_score= sum(math.score)/n())
```

```
## # A tibble: 6 x 2
##   parental.level.of.education math_score
##   <chr>                             <dbl>
## 1 associate's degree                 67.9
## 2 bachelor's degree                  69.4
## 3 high school                        62.1
## 4 master's degree                    69.7
## 5 some college                       67.1
## 6 some high school                   63.5
```

Tidyverse::ggplot2()

ggplot2 is a system for 'declaratively' creating graphics, based on "The Grammar of Graphics".

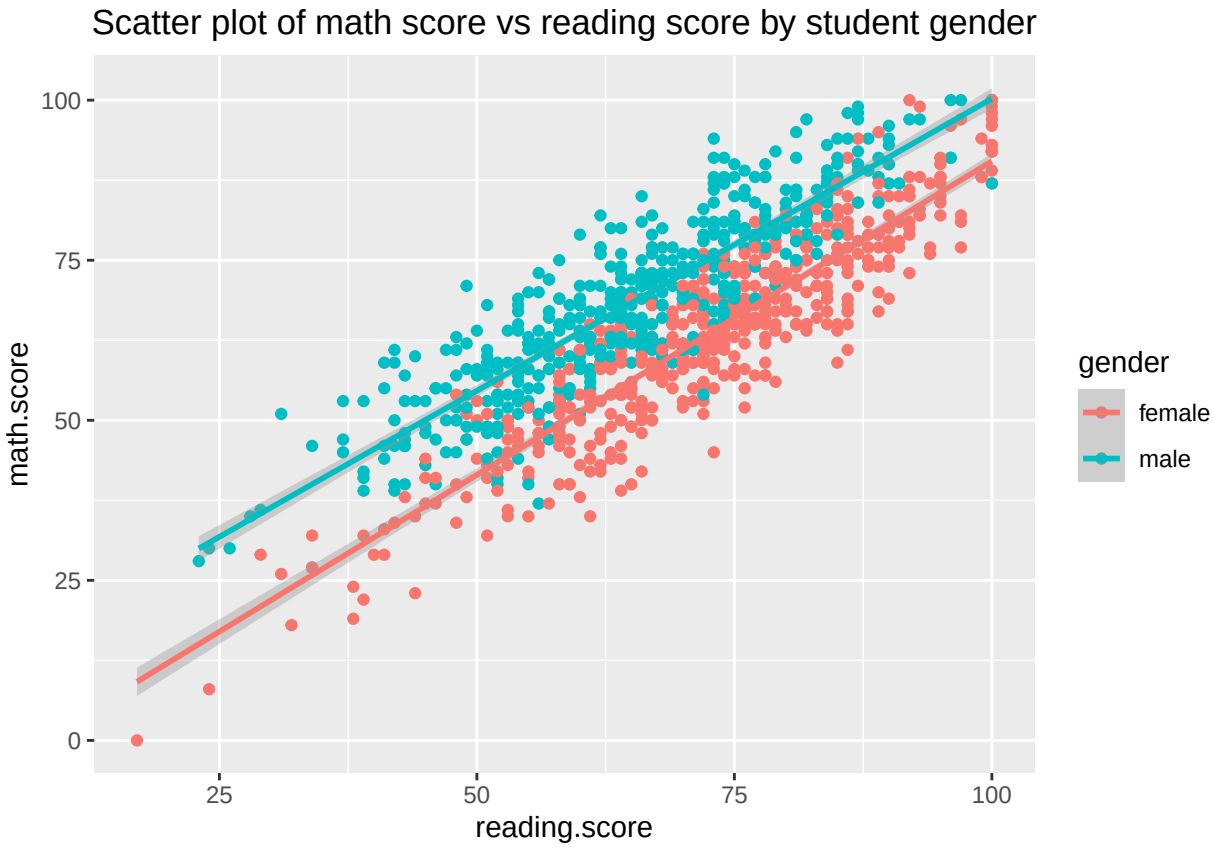
```
ggplot(data = data6, aes(x = gender, y = avg.score, col = gender), col = red) + geom_boxplot() + labs(t
```



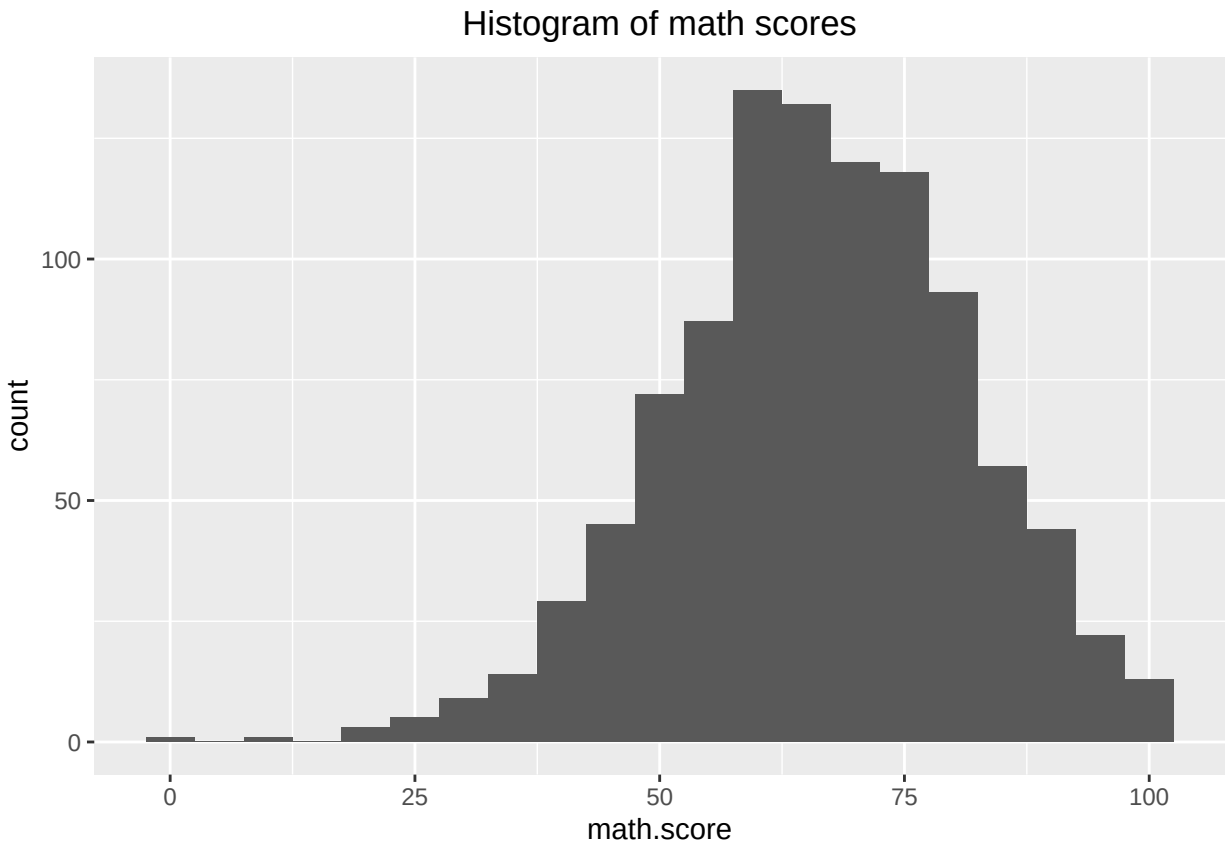
```
ggplot(data = data1, aes(x = reading.score, y= math.score, col = gender), col = red) + geom_point() + la
```

ggplot2 system has many different plot designs that you can create using the appropriate geom_type for the plot. Below are two simple examples to create a scatter plot evaluation the relationship between reading and math scores by gender (with linear correlation model line using geom_smooth) and a histogram plot of the math scores.

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(data = data1, aes(x = math.score), col = red) + geom_histogram(binwidth = 5) + labs(title="Histog
```



map() function use from purrr

Use of the purr Map function to calculate the mean vs a traditional approach.

```
data3a<-data2 %>%
  filter(gender== "female",math.score>0, writing.score >95, reading.score >95)
data3a
```

```
##    gender math.score reading.score writing.score
## 1  female         87           100           100
## 2  female         99           100           100
## 3  female         96           100           100
## 4  female         97           100           100
## 5  female         82            97            96
## 6  female         88            99           100
## 7  female        100           100           100
## 8  female         92           100            97
## 9  female         92           100           100
## 10 female         81            97            96
## 11 female         92           100            99
## 12 female         94            99           100
## 13 female         98           100            99
## 14 female         96            96            99
## 15 female         97            97            96
## 16 female         93           100           100
## 17 female         92           100           100
```

```
## 18 female        100           100           100
## 19 female         89           100           100
```

```
data3b<-data2 %>%
  filter(gender == "male", math.score>0, writing.score >95, reading.score >95)
data3b
```

```
##   gender math.score reading.score writing.score
## 1   male        100            97            99
## 2   male        100           100           100
```

```
paste0 (" The average female math score is ", (mean(data3a$math.score)))
```

```
## [1] " The average female math score is 92.8947368421053"
```

```
paste0 (" The average male math score is ", (mean(data3b$math.score)))
```

```
## [1] " The average male math score is 100"
```

```
data2a <- data1 %>%
  select(math.score, reading.score, writing.score)
head(data2a)
```

```
##   math.score reading.score writing.score
## 1         72            72            74
## 2         69            90            88
## 3         90            95            93
## 4         47            57            44
## 5         76            78            75
## 6         71            83            78
```

```
paste0 ("The mean math score, reading score and writing score")
```

```
## [1] "The mean math score, reading score and writing score"
```

```
map_dbl (data2a, mean)
```

```
##    math.score reading.score writing.score
##        66.089        69.169        68.054
```

```
paste0 ("The stardard deviaiton for math, reading and writing scores")
```

```
## [1] "The stardard deviaiton for math, reading and writing scores"
```

```
map_dbl(data2a, sd)
```

```
##    math.score reading.score writing.score
##      15.16308      14.60019      15.19566
```

purrr models short cut to evaluate a correlation

Use of Purr for models and to evaluate the model correlation

```
models<- data1 %>%
  split(.$gender) %>%
  map(~lm (math.score ~ reading.score, data= .))

models %>%
 map(summary) %>%
 map_dbl(~.$r.squared)
```

```
##    female      male
## 0.8267428 0.7840906
```

```
models2<- data %>%
  split(.$parental.level.of.education) %>%
  map(~lm (math.score ~ reading.score, data= .))

models2 %>%
 map(summary) %>%
 map_dbl(~.$r.squared)
```

```
## associate's degree  bachelor's degree        high school    master's degree
##          0.6396484          0.6675587          0.6427461          0.7315614
##       some college   some high school
##          0.6412086          0.6934304
```

dyplr unite() to merge different character columns

Use of unite() function to merge character columns. Merge - gender, level of education and race/ ethnicity.

```
data<- data %>%
  unite("Merged", parental.level.of.education:gender, remove=FALSE)
head (data)
```

```
##                              Merged gender race.ethnicity
## 1  bachelor's degree_group B_female female        group B
## 2       some college_group C_female female        group C
## 3    master's degree_group B_female female        group B
## 4   associate's degree_group A_male   male        group A
## 5         some college_group C_male   male        group C
## 6 associate's degree_group B_female female        group B
##   parental.level.of.education        lunch test.preparation.course math.score
## 1           bachelor's degree     standard                    none         72
## 2                some college     standard               completed         69
## 3             master's degree     standard                    none         90
## 4          associate's degree free/reduced                    none         47
## 5                some college     standard                    none         76
## 6          associate's degree     standard                    none         71
##   reading.score writing.score
```

```
## 1          72          74
## 2          90          88
## 3          95          93
## 4          57          44
## 5          78          75
## 6          83          78
```

Other usage of Tidyverse can be found in the textbook "R for Data Science" and other online resource.