Final_Reflection

Beatrice Kemboi

4/28/2022

[Project URL] knitr::include_url(https://github.com/bkemboi394/Ranking-US-States-Data-Analysis/blob/main/Ranking_USA_States.md)

Project partner : Moreen Owino

Introduction:

Reading through the first draft of my reflection, I realized just how verbose it was. So, for this one, I'll try to be more succinct and to the point. I will use some code chunks from the project and activities to demonstrate my grasp of the course objectives, and will explain them as follows.

Loading tidyverse as this course is based mainly on using tidyverse and its packages to achieve the course objectives

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.3.4     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Objective 1 : Import, manage, and clean data

I will use a portion of my project, wrangling US states median income from 2011 to 2020, to show how I achieved the processes of importing, managing, and cleaning data.

First, I used read_csv to load the data as a tibble into R as I want the first few rows and columns to nicely print out when I finally knit.

In addition, I used here package as I am loading from a data folder, Statesmedianincome, that I created to store all the states median income data together.

```
statesmedianincome_to2018 <- read_csv(here::here("Statesmedianincome", "medianstateincome_to2018.csv"))
```

```
## Warning: Missing column names filled in: 'X2' [2], 'X3' [3], 'X4' [4], 'X5' [5],
## 'X6' [6], 'X7' [7], 'X8' [8], 'X9' [9], 'X10' [10], 'X11' [11], 'X12' [12],
## 'X13' [13], 'X14' [14], 'X15' [15], 'X16' [16], 'X17' [17], 'X18' [18],
```

```
## 'X19' [19], 'X20' [20], 'X21' [21], 'X22' [22], 'X23' [23], 'X24' [24],
## 'X25' [25], 'X26' [26], 'X27' [27], 'X28' [28], 'X29' [29], 'X30' [30],
## 'X31' [31], 'X32' [32], 'X33' [33], 'X34' [34], 'X35' [35], 'X36' [36],
## 'X37' [37], 'X38' [38], 'X39' [39], 'X40' [40], 'X41' [41], 'X42' [42],
## 'X43' [43], 'X44' [44], 'X45' [45], 'X46' [46], 'X47' [47], 'X48' [48],
## 'X49' [49], 'X50' [50], 'X51' [51], 'X52' [52], 'X53' [53], 'X54' [54],
## 'X55' [55], 'X56' [56], 'X57' [57], 'X58' [58], 'X59' [59], 'X60' [60],
## 'X61' [61], 'X62' [62], 'X63' [63], 'X64' [64], 'X65' [65], 'X66' [66],
## 'X67' [67], 'X68' [68], 'X69' [69], 'X70' [70], 'X71' [71], 'X72' [72],
## 'X73' [73], 'X74' [74], 'X75' [75]
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
statesmedianincome_to2018
```

```
## # A tibble: 115 x 75
##    `Table with row ~ X2    X3    X4    X5    X6    X7    X8    X9    X10   X11
##    <chr>             <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
##  1 Table H-8.  Medi~  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
##  2 (Households as o~  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
##  3 CURRENT DOLLARS    <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
##  4 State             "201~  <NA> "201~  <NA> "201~  <NA> "201~  <NA> "201~  <NA>
##  5 <NA>              "Med~ "Sta~ "Med~ "Sta~ "Med~ "Sta~ "Med~ "Sta~ "Med~ "Sta~
##  6 United States     "63,~ "420" "61,~ "322" "61,~ "335" "59,~ "436" "56,~ "321"
##  7 Alabama           "49,~ "2,4~ "50,~ "1,0~ "51,~ "845" "47,~ "2,3~ "44,~ "3,4~
##  8 Alaska            "68,~ "3,3~ "77,~ "3,7~ "72,~ "2,7~ "75,~ "4,0~ "75,~ "3,4~
##  9 Arizona           "62,~ "2,2~ "59,~ "2,6~ "61,~ "2,6~ "57,~ "1,9~ "52,~ "2,0~
## 10 Arkansas          "49,~ "2,1~ "49,~ "2,4~ "48,~ "2,6~ "45,~ "2,1~ "42,~ "1,5~
## # ... with 105 more rows, and 64 more variables: X12 <chr>, X13 <chr>,
## #   X14 <chr>, X15 <chr>, X16 <chr>, X17 <chr>, X18 <chr>, X19 <chr>,
## #   X20 <chr>, X21 <chr>, X22 <chr>, X23 <chr>, X24 <chr>, X25 <chr>,
## #   X26 <chr>, X27 <chr>, X28 <chr>, X29 <chr>, X30 <chr>, X31 <chr>,
## #   X32 <chr>, X33 <chr>, X34 <chr>, X35 <chr>, X36 <chr>, X37 <chr>,
## #   X38 <chr>, X39 <chr>, X40 <chr>, X41 <chr>, X42 <chr>, X43 <chr>,
## #   X44 <chr>, X45 <chr>, X46 <chr>, X47 <chr>, X48 <chr>, X49 <chr>,
## #   X50 <chr>, X51 <chr>, X52 <chr>, X53 <chr>, X54 <chr>, X55 <chr>,
## #   X56 <chr>, X57 <chr>, X58 <chr>, X59 <chr>, X60 <chr>, X61 <chr>,
## #   X62 <chr>, X63 <chr>, X64 <chr>, X65 <chr>, X66 <chr>, X67 <chr>,
## #   X68 <chr>, X69 <chr>, X70 <chr>, X71 <chr>, X72 <chr>, X73 <chr>,
## #   X74 <chr>, X75 <chr>
```

As you can see, the loaded data is really messy, so I managed and tidied it step-wise through the following steps:

i.) Used the slice command to remove the first three unnecessary rows

```
states_medianincome_to2018 <- statesmedianincome_to2018 %>%
slice(-(1:3))
states_medianincome_to2018
```

```
## # A tibble: 112 x 75
##    `Table with row ~ X2    X3    X4    X5    X6    X7    X8    X9    X10   X11
##    <chr>             <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
##  1 State             "201~  <NA> "201~  <NA> "201~  <NA> "201~  <NA> "201~  <NA>
##  2 <NA>              "Med~ "Sta~ "Med~ "Sta~ "Med~ "Sta~ "Med~ "Sta~ "Med~ "Sta~
##  3 United States     "63,~ "420" "61,~ "322" "61,~ "335" "59,~ "436" "56,~ "321"
##  4 Alabama           "49,~ "2,4~ "50,~ "1,0~ "51,~ "845" "47,~ "2,3~ "44,~ "3,4~
##  5 Alaska            "68,~ "3,3~ "77,~ "3,7~ "72,~ "2,7~ "75,~ "4,0~ "75,~ "3,4~
##  6 Arizona           "62,~ "2,2~ "59,~ "2,6~ "61,~ "2,6~ "57,~ "1,9~ "52,~ "2,0~
##  7 Arkansas          "49,~ "2,1~ "49,~ "2,4~ "48,~ "2,6~ "45,~ "2,1~ "42,~ "1,5~
##  8 California        "70,~ "1,2~ "70,~ "1,5~ "69,~ "1,4~ "66,~ "1,0~ "63,~ "1,7~
##  9 Colorado          "73,~ "3,5~ "74,~ "2,6~ "74,~ "2,9~ "70,~ "4,1~ "66,~ "3,6~
## 10 Connecticut       "72,~ "5,1~ "74,~ "3,4~ "72,~ "3,5~ "75,~ "3,4~ "72,~ "4,4~
## # ... with 102 more rows, and 64 more variables: X12 <chr>, X13 <chr>,
## #   X14 <chr>, X15 <chr>, X16 <chr>, X17 <chr>, X18 <chr>, X19 <chr>,
## #   X20 <chr>, X21 <chr>, X22 <chr>, X23 <chr>, X24 <chr>, X25 <chr>,
## #   X26 <chr>, X27 <chr>, X28 <chr>, X29 <chr>, X30 <chr>, X31 <chr>,
## #   X32 <chr>, X33 <chr>, X34 <chr>, X35 <chr>, X36 <chr>, X37 <chr>,
## #   X38 <chr>, X39 <chr>, X40 <chr>, X41 <chr>, X42 <chr>, X43 <chr>,
## #   X44 <chr>, X45 <chr>, X46 <chr>, X47 <chr>, X48 <chr>, X49 <chr>,
## #   X50 <chr>, X51 <chr>, X52 <chr>, X53 <chr>, X54 <chr>, X55 <chr>,
## #   X56 <chr>, X57 <chr>, X58 <chr>, X59 <chr>, X60 <chr>, X61 <chr>,
## #   X62 <chr>, X63 <chr>, X64 <chr>, X65 <chr>, X66 <chr>, X67 <chr>,
## #   X68 <chr>, X69 <chr>, X70 <chr>, X71 <chr>, X72 <chr>, X73 <chr>,
## #   X74 <chr>, X75 <chr>
```

ii.) Removed the existing column headers as they are not my desired column headers

```
names(states_medianincome_to2018) <- NULL
names(states_medianincome_to2018) <- states_medianincome_to2018[1,]
```

```
## Warning: The `value` argument of `names<-` can't be empty as of tibble 3.0.0.
## Columns 3, 5, 7, 9, 11, and 32 more must be named.
```

```
## Warning: The `value` argument of `names<-` must be a character vector as of
## tibble 3.0.0.
```

```
states_medianincome_to2018
```

```
## # A tibble: 112 x 75
##    State  `2018`  ``    `2017 (40)` ``    `2017` ``    `2016` ``    `2015` ``
##    <chr>  <chr>   <chr> <chr>       <chr> <chr>  <chr> <chr>  <chr> <chr>  <chr>
##  1 State  "2018"   <NA> "2017 (40)"  <NA> "2017"  <NA> "2016"  <NA> "2015"  <NA>
##  2 <NA>   "Media~ "Sta~ "Median\ni~ "Sta~ "Medi~ "Sta~ "Medi~ "Sta~ "Medi~ "Sta~
##  3 Unite~ "63,17~ "420" "61,136"    "322" "61,3~ "335" "59,0~ "436" "56,5~ "321"
##  4 Alaba~ "49,93~ "2,4~ "50,865"    "1,0~ "51,1~ "845" "47,2~ "2,3~ "44,5~ "3,4~
##  5 Alaska "68,73~ "3,3~ "77,987"    "3,7~ "72,2~ "2,7~ "75,7~ "4,0~ "75,1~ "3,4~
##  6 Arizo~ "62,28~ "2,2~ "59,700"    "2,6~ "61,1~ "2,6~ "57,1~ "1,9~ "52,2~ "2,0~
##  7 Arkan~ "49,78~ "2,1~ "49,751"    "2,4~ "48,8~ "2,6~ "45,9~ "2,1~ "42,7~ "1,5~
##  8 Calif~ "70,48~ "1,2~ "70,038"    "1,5~ "69,7~ "1,4~ "66,6~ "1,0~ "63,6~ "1,7~
##  9 Color~ "73,03~ "3,5~ "74,984"    "2,6~ "74,1~ "2,9~ "70,5~ "4,1~ "66,5~ "3,6~
## 10 Conne~ "72,81~ "5,1~ "74,304"    "3,4~ "72,7~ "3,5~ "75,9~ "3,4~ "72,8~ "4,4~
## # ... with 102 more rows, and 64 more variables: 2014 <chr>,  <chr>,
## #   2013 (39) <chr>, .1 <chr>, 2013 (38) <chr>, .2 <chr>, 2012 <chr>, .3 <chr>,
```

```
## #   2011 <chr>, .4 <chr>, 2010 (37) <chr>, .5 <chr>, 2009 (36) <chr>, .6 <chr>,
## #   2008 <chr>, .7 <chr>, 2007 <chr>, .8 <chr>, 2006 <chr>, .9 <chr>,
## #   2005 <chr>, .10 <chr>, 2004(revised) <chr>, .11 <chr>, 2003 <chr>,
## #   .12 <chr>, 2002 <chr>, .13 <chr>, 2001 <chr>, .14 <chr>, 2000 (30) <chr>,
## #   .15 <chr>, 1999 (29) <chr>, .16 <chr>, 1998 <chr>, .17 <chr>, 1997 <chr>,
## #   .18 <chr>, 1996 <chr>, .19 <chr>, 1995 (25) <chr>, .20 <chr>,
## #   1994 (24) <chr>, .21 <chr>, 1993 (23) <chr>, .22 <chr>, 1992 (22) <chr>,
## #   .23 <chr>, 1991 <chr>, .24 <chr>, 1990 <chr>, .25 <chr>, 1989 <chr>,
## #   .26 <chr>, 1988 <chr>, .27 <chr>, 1987 (21) <chr>, .28 <chr>, 1986 <chr>,
## #   .29 <chr>, 1985 (20) <chr>, .30 <chr>, 1984 (19) <chr>, .31 <chr>
```

iii.) Used a combination of various versions of select, rename, slice and na.omit functions to clean and subset the data further.

The loaded data contained median income from 1984 to 2018, so had to subset to get just those of 2011 to 2018.

Added comments to explain where it is not so obvious.

```
states_medianincome_2011to2018 <- states_medianincome_to2018 %>%
 select(State,starts_with(c("2"))) %>% #selects columns from 2000 on wards
  select(-c("2010 (37)":"2000 (30)")) %>%  #removes columns 2000 to 2010
     rename(twentysvtn="2017",twentysvtnt="2017 (40)") %>% #solving ambiguity
       slice(-c(1:3)) %>% #further cutting down more unnecessary rows
          na.omit %>%  ##removing some rows that had just nas
             slice(-c(52:104)) #removing unwanted rows, i.e,beyond the 51 states


states_medianincome_2011to2018
```

```
## # A tibble: 51 x 11
##    State       `2018` twentysvtnt twentysvtn `2016` `2015` `2014` `2013 (39)`
##    <chr>       <chr>  <chr>       <chr>      <chr>  <chr>  <chr>  <chr>
##  1 Alabama     49,936 50,865      51,113     47,221 44,509 42,278 47,320
##  2 Alaska      68,734 77,987      72,231     75,723 75,112 67,629 72,472
##  3 Arizona     62,283 59,700      61,125     57,100 52,248 49,254 52,611
##  4 Arkansas    49,781 49,751      48,829     45,907 42,798 44,922 39,376
##  5 California  70,489 70,038      69,759     66,637 63,636 60,487 60,794
##  6 Colorado    73,034 74,984      74,172     70,566 66,596 60,940 67,912
##  7 Connecticut 72,812 74,304      72,780     75,923 72,889 70,161 69,291
##  8 Delaware    65,012 64,961      62,318     58,046 57,756 57,522 54,091
##  9 D.C.        85,750 81,282      83,382     70,982 70,071 68,277 60,057
## 10 Florida     54,644 53,086      53,681     51,176 48,825 46,140 48,532
## # ... with 41 more rows, and 3 more variables: 2013 (38) <chr>, 2012 <chr>,
## #   2011 <chr>
```

iv.) Wrote a function that lets me iterate through all the year columns and utilize lapply to remove the a thousand comma separator from them as well as make them numeric

Needed to apply some numeric operations to my 'numeric' columns, but couldn't because of the a thousand coma separator that was preventing me from converting them to numeric from character.

```
col_conv <- c(2:11)  #sub setting the appropriate columns
states_medianincome_2011to2018[ , col_conv] <- lapply(states_medianincome_2011to2018[ , col_conv],functi

states_medianincome_2011to2018
```

```
## # A tibble: 51 x 11
##    State       `2018` twentysvtnt twentysvtn `2016` `2015` `2014` `2013 (39)`
##    <chr>        <dbl>       <dbl>      <dbl>  <dbl>  <dbl>  <dbl>       <dbl>
##  1 Alabama      49936       50865      51113  47221  44509  42278       47320
##  2 Alaska       68734       77987      72231  75723  75112  67629       72472
##  3 Arizona      62283       59700      61125  57100  52248  49254       52611
##  4 Arkansas     49781       49751      48829  45907  42798  44922       39376
##  5 California   70489       70038      69759  66637  63636  60487       60794
##  6 Colorado     73034       74984      74172  70566  66596  60940       67912
##  7 Connecticut  72812       74304      72780  75923  72889  70161       69291
##  8 Delaware     65012       64961      62318  58046  57756  57522       54091
##  9 D.C.         85750       81282      83382  70982  70071  68277       60057
## 10 Florida      54644       53086      53681  51176  48825  46140       48532
## # ... with 41 more rows, and 3 more variables: 2013 (38) <dbl>, 2012 <dbl>,
## #   2011 <dbl>
```

v.) Used mutate to create new columns for years 2013 and 2017 to replace the other two ambiguous columns, that existed for each of them, by taking their averages, and removed the ambiguous ones

```
smi <- states_medianincome_2011to2018 %>%
    mutate("2013" = (`2013 (38)` + `2013 (39)`)/2) %>%
    mutate("2017" = (twentysvtn + twentysvtnt)/2) %>%
     select(-c(twentysvtnt,twentysvtn,`2013 (38)`,`2013 (39)`))
smi
```

```
## # A tibble: 51 x 9
##    State       `2018` `2016` `2015` `2014` `2012` `2011` `2013` `2017`
##    <chr>        <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 Alabama      49936  47221  44509  42278  43464  42590 44350. 50989
##  2 Alaska       68734  75723  75112  67629  63648  57431 66804. 75109
##  3 Arizona      62283  57100  52248  49254  47044  48621 51606. 60412.
##  4 Arkansas     49781  45907  42798  44922  39018  41302 39648. 49290
##  5 California   70489  66637  63636  60487  57020  53367 59161  69898.
##  6 Colorado     73034  70566  66596  60940  57255  58629 65642. 74578
##  7 Connecticut  72812  75923  72889  70161  64247  65415 68536  73542
##  8 Delaware     65012  58046  57756  57522  48972  54660 53155  63640.
##  9 D.C.         85750  70982  70071  68277  65246  55251 60366  82332
## 10 Florida      54644  51176  48825  46140  46071  45105 48209  53384.
## # ... with 41 more rows
```

v.) 2019 and 2020 median income data were gathered separately, so I loaded them separately as well, and cleaned it in an almost similar fashion as shown.

```
statesmedianincome_2019 <- read_csv(here::here("Statesmedianincome", "medianstateincome2019.csv"))
```

```
## Warning: Missing column names filled in: 'X2' [2]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   `Title: Median Annual Household Income | KFF` = col_character(),
##   X2 = col_character()
## )
```

```
statesmedianincome_2020 <- read_csv(here::here("Statesmedianincome", "medianstateincome_2020.csv"))
```

```
## Warning: Missing column names filled in: 'X2' [2]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   `Median household income in the United States by state 2020` = col_character(),
##   X2 = col_double()
## )
```

```
smi_2019<- statesmedianincome_2019 %>%
  rename(State = `Title: Median Annual Household Income | KFF`,"2019"=X2)
smi_2020 <- statesmedianincome_2020 %>%
  rename(State = `Median household income in the United States by state 2020`,"2020" = X2)

smi2019_2020 <- smi_2019 %>%  #joining 2019 and 2020 data
    inner_join(smi_2020, by = "State")

smi2019_2020$`2019` = as.numeric(gsub("[\\$,]", "", smi2019_2020$`2019`))
```

vi.) Lastly, used inner join to merge the 2011-2018 data with 2019-2020 data to finally get tidy 2011 to 2020 states median income in one place.

```
smi2019_2020[10,1] <- "D.C." #Renaming state to match name in other table before inner join

statesmedianincome_2011to2020 <- smi %>%
      inner_join(smi2019_2020, by = "State")
statesmedianincome_2011to2020
```

```
## # A tibble: 51 x 11
##    State   `2018` `2016` `2015` `2014` `2012` `2011` `2013` `2017` `2019` `2020`
##    <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 Alabama  49936  47221  44509  42278  43464  42590 44350. 50989   51734  54393
##  2 Alaska   68734  75723  75112  67629  63648  57431 66804. 75109   75463  74476
##  3 Arizona  62283  57100  52248  49254  47044  48621 51606. 60412.  62055  66628
##  4 Arkans~  49781  45907  42798  44922  39018  41302 39648. 49290   48952  50540
##  5 Califo~  70489  66637  63636  60487  57020  53367 59161  69898.  80440  77358
##  6 Colora~  73034  70566  66596  60940  57255  58629 65642. 74578   77127  82611
##  7 Connec~  72812  75923  72889  70161  64247  65415 68536  73542   78833  79043
##  8 Delawa~  65012  58046  57756  57522  48972  54660 53155  63640.  70176  69132
##  9 D.C.     85750  70982  70071  68277  65246  55251 60366  82332   92266  88311
```

```
## 10 Florida  54644  51176  48825  46140  46071  45105 48209  53384.  59227  57435
## # ... with 41 more rows
```

Objective 2 : Create graphical displays and numerical summaries of data for exploratory analysis and presentations.

For this, I will use college_recent_grads data from activity 5 and a section of my project.

```
college_recent_grads <- read_csv("recent-grads.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   major = col_character(),
##   major_category = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
college_recent_grads
```

```
## # A tibble: 173 x 21
##     rank major_code major           major_category total sample_size   men women
##    <dbl>      <dbl> <chr>           <chr>          <dbl>       <dbl> <dbl> <dbl>
##  1     1       2419 Petroleum Engi~ Engineering     2339          36  2057   282
##  2     2       2416 Mining And Min~ Engineering      756           7   679    77
##  3     3       2415 Metallurgical ~ Engineering      856           3   725   131
##  4     4       2417 Naval Architec~ Engineering     1258          16  1123   135
##  5     5       2405 Chemical Engin~ Engineering    32260         289 21239 11021
##  6     6       2418 Nuclear Engine~ Engineering     2573          17  2200   373
##  7     7       6202 Actuarial Scie~ Business        3777          51  2110  1667
##  8     8       5001 Astronomy And ~ Physical Scie~  1792          10   832   960
##  9     9       2414 Mechanical Eng~ Engineering    91227        1029 80320 10907
## 10    10       2408 Electrical Eng~ Engineering    81527         631 65511 16016
## # ... with 163 more rows, and 13 more variables: sharewomen <dbl>,
## #   employed <dbl>, employed_fulltime <dbl>, employed_parttime <dbl>,
## #   employed_fulltime_yearround <dbl>, unemployed <dbl>,
## #   unemployment_rate <dbl>, p25th <dbl>, median <dbl>, p75th <dbl>,
## #   college_jobs <dbl>, non_college_jobs <dbl>, low_wage_jobs <dbl>
```

Learnt many powerful functions for numerical summaries across rows and across columns as well as group wise. Some are demonstrated in the code chunk below.

group_by and summarise functions, together with 'sum', groups our data and returns their group sums.

median belongs to the min,mean,median, and max numerical summaries.

while min_rank belongs to the group of numerical summaries that together with mutate return new columns of summaries across rows. In this case, min-rank is creating a new column that display rankings.

```
category_rankings= college_recent_grads %>%
  group_by(major_category) %>%
```

```
    summarise(total_mctgry=sum(total,na.rm = TRUE),sharewomen_median=median(sharewomen), unemploymentra
  mutate(tota_mctgry_rank=min_rank(total_mctgry),sharewomen_rank=min_rank(sharewomen_median), unemploym
  arrange(major_category)

category_rankings
```

```
## # A tibble: 16 x 8
##    major_category    total_mctgry sharewomen_medi~ unemploymentrate median_earns
##    <chr>                    <dbl>            <dbl>            <dbl>        <dbl>
##  1 Agriculture & Na~        75620           NA               0.0553        35000
##  2 Arts                    357130            0.667           0.0895        30750
##  3 Biology & Life S~       453862            0.583           0.0680        36300
##  4 Business               1302376            0.441           0.0697        40000
##  5 Communications &~       392601            0.672           0.0722        35000
##  6 Computers & Math~       299008            0.269           0.0908        45000
##  7 Education               559129            0.769           0.0488        32750
##  8 Engineering             537583            0.227           0.0598        57000
##  9 Health                  463230            0.783           0.0643        35000
## 10 Humanities & Lib~       713468            0.690           0.0817        32000
## 11 Industrial Arts ~       229792            0.232           0.0557        35000
## 12 Interdisciplinary        12296            0.771           0.0709        35000
## 13 Law & Public Pol~       179107            0.476           0.0825        36000
## 14 Physical Sciences       185479            0.520           0.0511        39500
## 15 Psychology & Soc~       481007            0.799           0.0651        30000
## 16 Social Science          529966            0.543           0.0972        38000
## # ... with 3 more variables: tota_mctgry_rank <int>, sharewomen_rank <int>,
## #   unemploymentrate_rank <int>
```

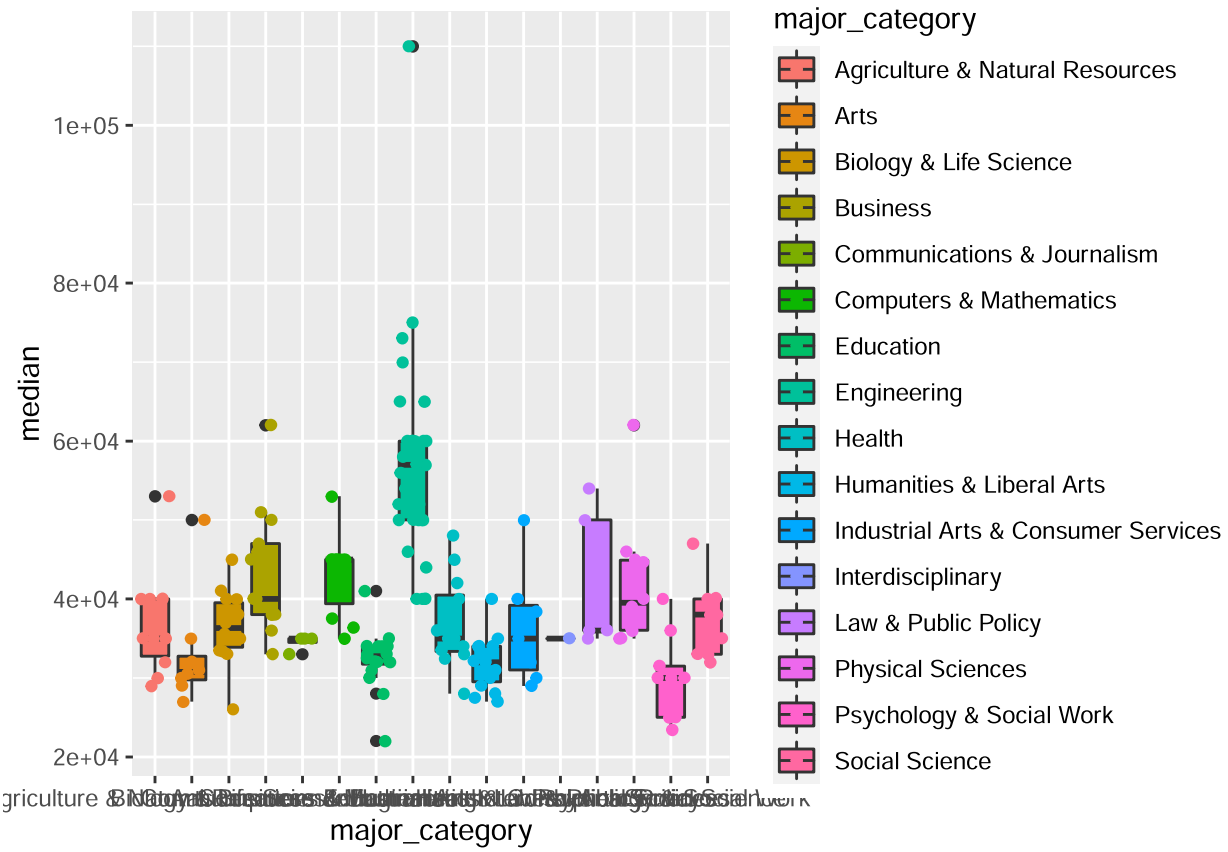
As for visualizations, ggplot2 is really big. It's powerful as it lets one start from creating a basic visualization object, and then keep adding layers of other aspects of the object one at a time to eventually achieve a visually striking and greatly informative visualization.

Below is a box plot of median income for each major category plotted with ggplot2.

First, I added the geom layer that specifies the type of geometry I want for my graph, in this case geom_boxplot.

Then I added another geom, geom_jitters, to get jitter plots on top of the box plot. However, I could get it under the box plot by starting with jitters geom first.

```
major_boxplot=college_recent_grads %>%
  select(employed_fulltime_yearround,median,major_category) %>%
  ggplot()+
    geom_boxplot(aes(x=major_category,y=median,fill=major_category))+
   geom_jitter(aes(x=major_category,y=median,color=major_category))

major_boxplot
```



For our Ranking USA states project, we plotted a line graph for the most improved 5 states that, at first, showed the rankings on the y_axis starting from 1 to 51 as follows:

-First had to use write.csv(most_improved_states,"mostimprovedstates.csv") to get that portion of the data and then load it from our project as it wouldn't knit without it

```
most_improved_states <- read_csv("mostimprovedstates.csv")
```

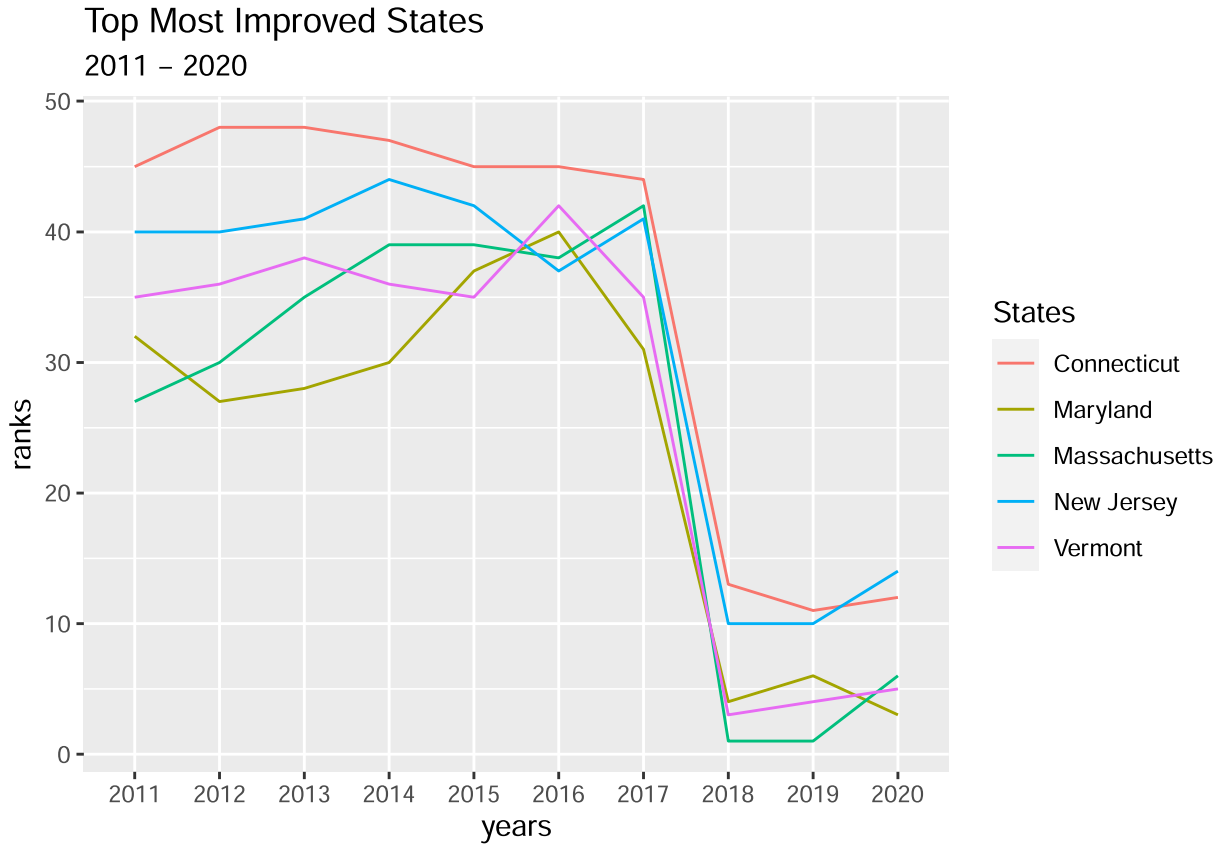
```
## Warning: Missing column names filled in: 'X1' [1]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   States = col_character(),
##   change = col_double(),
##   `2011` = col_double(),
##   `2012` = col_double(),
##   `2013` = col_double(),
##   `2014` = col_double(),
##   `2015` = col_double(),
##   `2016` = col_double(),
##   `2017` = col_double(),
##   `2018` = col_double(),
```

```
##   `2019` = col_double(),
##   `2020` = col_double()
## )
```

plot 1:

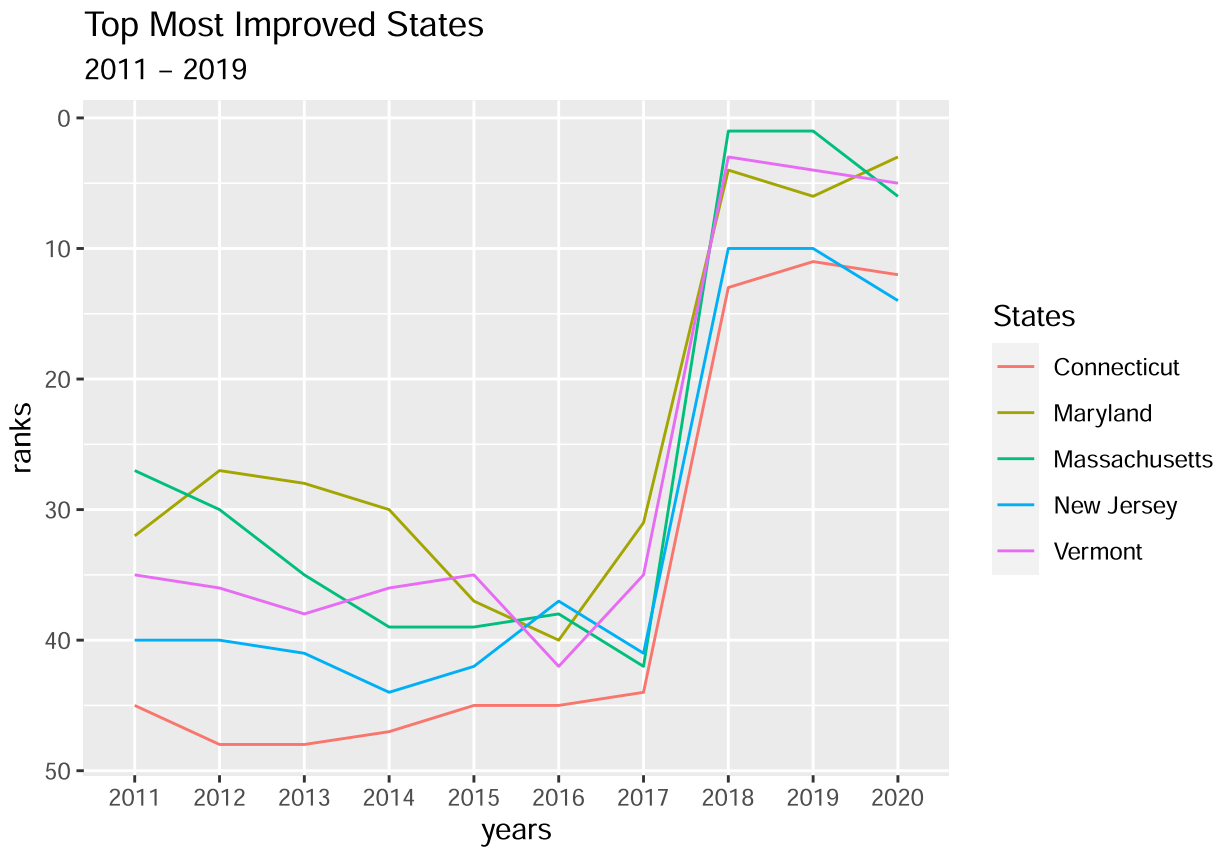
```
top5_mi_states <- most_improved_states[1:5,]

top5_mistates <- top5_mi_states%>%
 select(States, c(`2011`:`2020`))

top5_improved_states <- top5_mistates%>%
  pivot_longer(!States, names_to = "years", values_to = "ranks") %>%
  ggplot()+
  geom_line(aes(x = years , y = ranks , colour = States, group = States)) +
    labs(title = "Top Most Improved States",
       subtitle = "2011 - 2020")


top5_improved_states
```



Later on, after presenting our project, we were advised to flip the y-axis scales as it makes more sense to most people that way. ggplot came in handy, as all we had to do was add a layer that flips the y scale axis and here we have it!

plot 2

```
top5_mi_states <- most_improved_states[1:5,]

top5_mistates <- top5_mi_states%>%
 select(States, c(`2011`:`2020`))

top5_improved_states <- top5_mistates%>%
  pivot_longer(!States, names_to = "years", values_to = "ranks") %>%
  ggplot()+
  geom_line(aes(x = years , y = ranks , colour = States, group = States)) +
    labs(title = "Top Most Improved States",
       subtitle = "2011 - 2019")


top5_improved_states +scale_y_reverse()
```



Objective 3: Write R programs for simulations from probability models and randomization-based experiments.

We used multiple regression to model the factors in our project that we used to get US states rankings against ranks obtained from US NEWS 2021 rankings as we were curious to see which of our factors had more influence as per their rankings.

-Like before had to first write out the data and import to enable eventual knitting

```
regression_rank_data <- read_csv("regressionrankdata.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   `10yr_overall_erank` = col_double(),
##   `10yr_overall_hrank` = col_double(),
##   `10yr_overall_irank` = col_double(),
##   `10year_edurank` = col_double(),
##   `10yr_corrank` = col_double(),
##   Ranks = col_double()
## )
```

Below is the code for the first multiple regression model and it's summary

Model 1

```
multiple_regression <- lm(Ranks ~. , data = regression_rank_data)
summary(multiple_regression)
```

```
##
## Call:
## lm(formula = Ranks ~ ., data = regression_rank_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.360  -3.658   1.240   3.734  21.289
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.538471  15.289837  -0.035   0.9721
## X1                    0.213081   0.987595   0.216   0.8302
## `10yr_overall_erank`  0.466627   0.377369   1.237   0.2228
## `10yr_overall_hrank` -0.093563   0.167260  -0.559   0.5787
## `10yr_overall_irank`  0.062662   0.252964   0.248   0.8055
## `10year_edurank`      0.007679   0.376510   0.020   0.9838
## `10yr_corrank`        0.363965   0.159403   2.283   0.0273 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.399 on 44 degrees of freedom
## Multiple R-squared:  0.6483, Adjusted R-squared:  0.6003
```

```
## F-statistic: 13.52 on 6 and 44 DF,  p-value: 1.264e-08
```

Setting an alpha level of 0.1 for the p_values, we determined that the most influential of our factors were economy, corrections, and education.

So created another multiple regression model for just those three as shown.

Model 2

```
regression_rank_data2 <- regression_rank_data %>%
  select(Ranks,`10yr_overall_erank`,`10yr_corrank`,`10year_edurank`)

multiple_regression2 <- lm(Ranks ~. , data = regression_rank_data2)
summary(multiple_regression2)
```

```
##
## Call:
## lm(formula = Ranks ~ ., data = regression_rank_data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.6316  -2.6826   0.1326   3.8408  18.8274
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.27002    3.53694  -0.359  0.72115
## `10yr_overall_erank`  0.51831    0.10187   5.088 6.24e-06 ***
## `10yr_corrank`        0.32619    0.10011   3.258  0.00209 **
## `10year_edurank`      0.10805    0.04532   2.384  0.02121 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.313 on 47 degrees of freedom
## Multiple R-squared:  0.6311, Adjusted R-squared:  0.6076
## F-statistic:  26.8 on 3 and 47 DF,  p-value: 2.967e-10
```

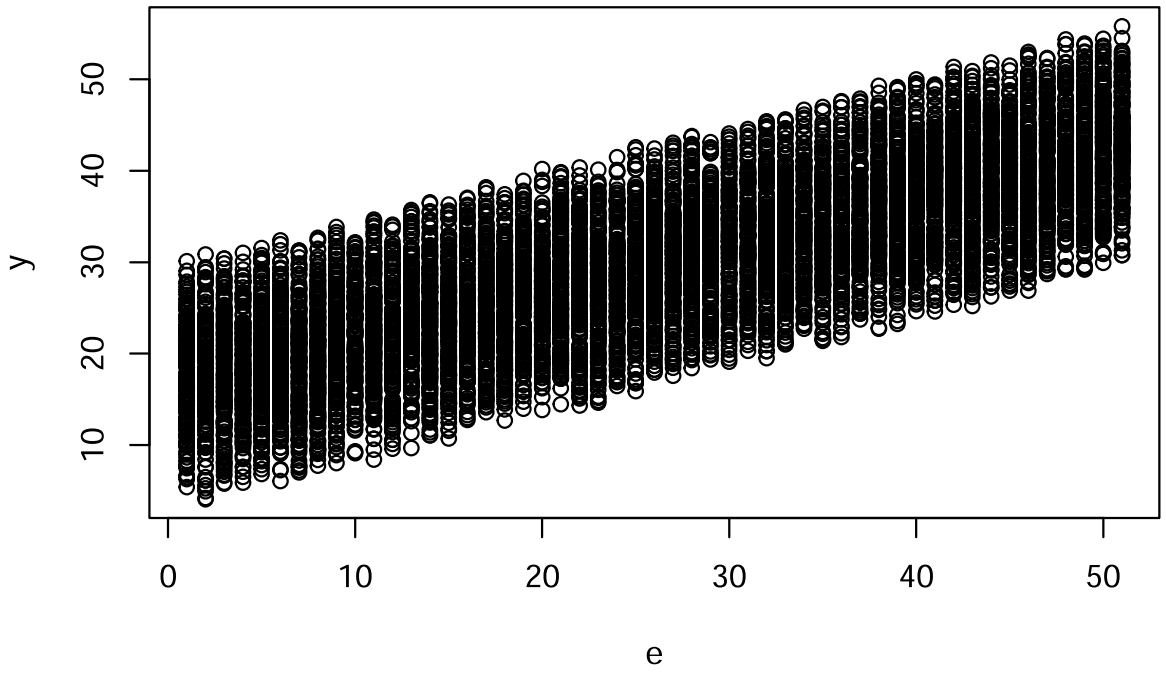
From the second model, we obtained a multiple linear regression model that then enabled us to simulate it with many random rankings and plotted a graph of simulated rankings for each of the factors as follows:

```
n <- c(1:51)
e <- sample(n,10000, replace = TRUE)  #Based on the 51 rankings for each state
cor <-  sample(n,10000, replace = TRUE)

edu <- sample(n,10000, replace = TRUE)



y <- -1.378 + 0.518*e + 0.326*cor+ 0.216*edu +3.56170
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.088  22.838  29.828  29.862  36.876  55.812
```

Economics simulation graph

```
plot(e, y)
```



Plot above shows that there's a strong positive correlation between our economics factor and US News 2021 Rankings.

Objective 4: Use source documentation and other resources to troubleshoot and extend R programs

Learnt how to use the '?' together with a name of some R function at the console in order to get thorough source documentation on it

Also, learnt the various packages needed for to handle different variables in R in order to extend it's programs such as the 'lubridate' for dates and times, and "purr" for maps .

In addition, got exposed to various google sites that helps with learning R such as stackoverflow among others

```
library(lubridate) #for dates and times
```

```
##
## Attaching package: 'lubridate'
```

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

```
library(purrr)  #for maps and other vector  functions
library(stringr)  #for string variables
library(forcats)   #for factor variables
```

Objective 5 : Write clear, efficient, and well-documented R programs

I believe every objective covered thus far, together with the provided examples show my ability to write clear, efficient, and well-documented R programs.