Project1

Alexis Kolecki, Jessie Heise

2025-06-17

Data Processing

Step 1

```
# Load libraries
library(dplyr)
library(tidyr)

# Read in EDU01a.csv data
first_data <- read.csv("EDU01a.csv")

# Select only the following columns:
# Area_name (rename as area_name),
# STCOU, any column that ends in 'D'
selected_columns <- first_data |>
    select(area_name = Area_name, STCOU,
        ends_with("D"))

# Display the first 5 rows of the new
# data set
head(selected_columns, n = 5)
```

```
      area_name STCOU EDU010187D EDU010188D EDU010189D EDU010190D EDU010191D
1 UNITED STATES     0   40024299   39967624   40317775   40737600   41385442
2       ALABAMA  1000     733735     728234     730048     728252     725541
3   Autauga, AL  1001       6829       6900       6920       6847       7008
4   Baldwin, AL  1003      16417      16465      16799      17054      17479
5   Barbour, AL  1005       5071       5098       5068       5156       5173
  EDU010192D EDU010193D EDU010194D EDU010195D EDU010196D
```

```
1   42088151   42724710   43369917   43993459   44715737
2     726150     728014     730509     727989     736825
3       7137       7152       7381       7568       7834
4      17983      18735      19384      19961      20699
5       5252       5135       5111       5017       5053
```

Step 2

```
# Convert data into long format Each
# row only has one enrollment value for
# that Area_name
pivoted_data <- selected_columns |>
    pivot_longer(cols = 3:12, names_to = "Enrollment",
        values_to = "Enrollment_Value") |>
    select(-STCOU)  #drop the STCOU column since we aren't using it

# Display first 5 rows of the new data
# set
head(pivoted_data, n = 5)
```

```
# A tibble: 5 x 3
  area_name     Enrollment Enrollment_Value
  <chr>         <chr>                 <int>
1 UNITED STATES EDU010187D         40024299
2 UNITED STATES EDU010188D         39967624
3 UNITED STATES EDU010189D         40317775
4 UNITED STATES EDU010190D         40737600
5 UNITED STATES EDU010191D         41385442
```

Step 3

```
# Separate enrollment variable
long_updated <- pivoted_data |>
   mutate(Survey = substr(Enrollment, 1,7), #pull measurement info
          Year = as.numeric(substr(Enrollment, 8,9))) |> #pull yr/make numeric
  mutate(Year=ifelse(Year>80, 1900+Year,2000+Year)) |> #format year to four digits
  select(area_name, Survey, Year, Enrollment_Value) #Select variables of interest
```

```r
# Display first 5 rows of new data set
head(long_updated, n = 5)
```

```
# A tibble: 5 x 4
  area_name     Survey   Year Enrollment_Value
  <chr>         <chr>   <dbl>            <int>
1 UNITED STATES EDU0101  1987         40024299
2 UNITED STATES EDU0101  1988         39967624
3 UNITED STATES EDU0101  1989         40317775
4 UNITED STATES EDU0101  1990         40737600
5 UNITED STATES EDU0101  1991         41385442
```

Step 4

```r
# Create two new data sets: one with only county data and one with only non-county data
county_tibble <- long_updated|>
  filter(grepl(",",area_name)) |> #filter to those with an area_Name
  mutate(county=grep(pattern =", \\w\\w", area_name))

class(county_tibble) <- c("county", class(county_tibble)) #Update class to county


state_tibble <- long_updated|>
  filter(!grepl(",",area_name)) #filter to those without an area_name

class(state_tibble) <- c("state", class(state_tibble)) #updates class to state

# Print the first 10 rows of the county tibble
head(county_tibble, n = 10)
```

```
# A tibble: 10 x 5
   area_name   Survey   Year Enrollment_Value county
   <chr>       <chr>   <dbl>            <int>  <int>
 1 Autauga, AL EDU0101  1987             6829      1
 2 Autauga, AL EDU0101  1988             6900      2
 3 Autauga, AL EDU0101  1989             6920      3
 4 Autauga, AL EDU0101  1990             6847      4
 5 Autauga, AL EDU0101  1991             7008      5
 6 Autauga, AL EDU0101  1992             7137      6
```

```
 7 Autauga, AL EDU0101  1993             7152      7
 8 Autauga, AL EDU0101  1994             7381      8
 9 Autauga, AL EDU0101  1995             7568      9
10 Autauga, AL EDU0101  1996             7834     10
```

```
# Print the first 10 rows of the state tibble
head(state_tibble, n = 10)
```

```
# A tibble: 10 x 4
   area_name     Survey   Year Enrollment_Value
   <chr>         <chr>   <dbl>            <int>
 1 UNITED STATES EDU0101  1987         40024299
 2 UNITED STATES EDU0101  1988         39967624
 3 UNITED STATES EDU0101  1989         40317775
 4 UNITED STATES EDU0101  1990         40737600
 5 UNITED STATES EDU0101  1991         41385442
 6 UNITED STATES EDU0101  1992         42088151
 7 UNITED STATES EDU0101  1993         42724710
 8 UNITED STATES EDU0101  1994         43369917
 9 UNITED STATES EDU0101  1995         43993459
10 UNITED STATES EDU0101  1996         44715737
```

Step 5

```
# Creating a new variable that
# describes which state one of these
# county measurements corresponds to
county_tibble <- county_tibble |>
    mutate(state = substr(area_name, nchar(area_name) -
        2, nchar(area_name)))
```

Step 6

```
# Creating a variable called division
# for non-county level tibble to denote
# the state
state_tibble <- state_tibble |>
    mutate(division = case_when(area_name %in%
```

```
        c("CONNECTICUT", "MAINE", "MASSACHUSETTS",
            "NEW HAMPSHIRE", "RHODE ISLAND",
            "VERMONT") ~ "New England", area_name %in%
        c("NEW JERSEY", "NEW YORK", "PENNSYLVANIA") ~
        "Mid-Atlantic", area_name %in% c("ILLINOIS",
        "INDIANA", "MICHIGAN", "OHIO", "WISCONSIN") ~
        "East North Central", area_name %in%
        c("IOWA", "KANSAS", "MINNESOTA",
            "MISSOURI", "NEBRASKA", "NORTH DAKOTA",
            "SOUTH DAKOTA") ~ "West North Central",
        area_name %in% c("DELAWARE", "DISTRICT OF COLUMBIA",
            "FLORIDA", "GEORGIA", "MARYLAND",
            "NORTH CAROLINA", "SOUTH CAROLINA",
            "VIRGINIA", "WEST VIRGINIA") ~
            "South Atlantic", area_name %in%
            c("ALABAMA", "KENTUCKY", "MISSISSIPPI",
                "TENNESSEE") ~ "East South Central",
        area_name %in% c("ARKANSAS", "LOUISIANA",
            "OKLAHOMA", "TEXAS") ~ "West South Central",
        area_name %in% c("ARIZONA", "COLORADO",
            "IDAHO", "MONTANA", "NEVADA",
            "NEW MEXICO", "UTAH", "WYOMING") ~
            "Mountain", area_name %in% c("ALASKA",
            "CALIFORNIA", "HAWAII", "OREGON",
            "WASHINGTON") ~ "Pacific", TRUE ~
            "ERROR"))
```

Data Processing Function Writing

Write one function that does steps 1 & 2 above

```
# Function for reading in file from a
# url
read_csv_code <- function(filename, column_name) {
    library(dplyr)
    library(tidyr)
    first_data <- read.csv(filename)
    return(first_data)
}
```

```
# Takes our code from steps 1 and 2
# above and turns them into a function
# this function will require a data
# file and a column_name
function_for_steps_1_2 <- function(first_data,
    column_name) {
    selected_columns <- first_data |>
        select(area_name = Area_name, STCOU,
            ends_with("D"))
    pivoted_data <- selected_columns |>
        pivot_longer(cols = 3:12, names_to = column_name,
            values_to = "Enrollment_Value") |>
        select(-STCOU)
    print(pivoted_data)
    return(pivoted_data)
}
```

Write a function that takes in the output of step 2 and does step 3 above

```
# Function taking in our code from step
# 2 and executing step 3 This function
# takes the file from function step 1
# and the column_name
function_for_step_3 <- function(pivoted_data,
    column_name) {
    long_updated <- pivoted_data |>
        mutate(Survey = substr(pivoted_data[[column_name]],
            1, 7), Year = as.numeric(substr(pivoted_data[[column_name]],
            8, 9))) |>
        mutate(Year = ifelse(Year > 50, 1900 +
            Year, 2000 + Year)) |>
        select(area_name, Survey, Year, Enrollment_Value)
    return(long_updated)
}
```

Write a function to do step 5

```
# Function using code from step 4 (need
# before step 5)
```

```
function_for_step_4 <- function(long_updated) {
    county_tibble <- long_updated |>
        filter(grepl(",", area_name)) |>
        mutate(county = grep(pattern = ", \\w\\w",
            area_name))
    class(county_tibble) <- c("county", class(county_tibble))
    state_tibble <- long_updated |>
        filter(!grepl(",", area_name))
    class(state_tibble) <- c("state", class(state_tibble))
    return(list = c(county_tibble, state_tibble))
}

# Function for our code in step 5
function_for_step_5 <- function(county_tibble) {
    county_tibble <- county_tibble |>
        mutate(state = substr(area_name,
            nchar(area_name) - 2, nchar(area_name)))
    return(county_tibble)
}
```

Write a function to do step 6

```
# Function for step 6
function_for_step_6 <- function(state_tibble) {
    state_tibble <- state_tibble |>
        mutate(division = case_when(area_name %in%
            c("CONNECTICUT", "MAINE", "MASSACHUSETTS",
                "NEW HAMPSHIRE", "RHODE ISLAND",
                "VERMONT") ~ "New England",
            area_name %in% c("NEW JERSEY",
                "NEW YORK", "PENNSYLVANIA") ~
                "Mid-Atlantic", area_name %in%
                c("ILLINOIS", "INDIANA",
                  "MICHIGAN", "OHIO", "WISCONSIN") ~
                "East North Central", area_name %in%
                c("IOWA", "KANSAS", "MINNESOTA",
                  "MISSOURI", "NEBRASKA",
                  "NORTH DAKOTA", "SOUTH DAKOTA") ~
                "West North Central", area_name %in%
                c("DELAWARE", "DISTRICT OF COLUMBIA",
```

```
                  "FLORIDA", "GEORGIA", "MARYLAND",
                  "NORTH CAROLINA", "SOUTH CAROLINA",
                  "VIRGINIA", "WEST VIRGINIA") ~
                "South Atlantic", area_name %in%
                c("ALABAMA", "KENTUCKY",
                  "MISSISSIPPI", "TENNESSEE") ~
                "East South Central", area_name %in%
                c("ARKANSAS", "LOUISIANA",
                  "OKLAHOMA", "TEXAS") ~
                "West South Central", area_name %in%
                c("ARIZONA", "COLORADO",
                  "IDAHO", "MONTANA", "NEVADA",
                  "NEW MEXICO", "UTAH", "WYOMING") ~
                "Mountain", area_name %in%
                c("ALASKA", "CALIFORNIA",
                  "HAWAII", "OREGON", "WASHINGTON") ~
                "Pacific", TRUE ~ "ERROR"))
    return(state_tibble)
}
```

Write a function that takes the output from step 3 and creates the two tibbles in step 4, calls the above two functions to perform steps 5 and 6, and returns two final tibbles

```
# Function for steps 4, 5, 6
function_for_steps_4_5_6 <- function(long_updated) {
    county_tibble <- long_updated |>
        filter(grepl(",", area_name)) |>
        mutate(county = grep(pattern = ", \\w\\w",
            area_name))
    class(county_tibble) <- c("county", class(county_tibble))
    state_tibble <- long_updated |>
        filter(!grepl(",", area_name))
    class(state_tibble) <- c("state", class(state_tibble))
    results5 <- function_for_step_5(county_tibble)
    results6 <- function_for_step_6(state_tibble)
    return(list = c(county_data = results5,
        state_data = results6))
}
```

Put the above functions into a wrapper function

```
# Wrapper function that takes in the
# URL of a .csv file and the optional
# argument for the variable name, calls
# the functions written above, and
# returns two tibbles
processing_wrapper <- function(url, column_name) {
    result <- read_csv_code(url) |>
        (\(data) function_for_steps_1_2(data,
            column_name))() |>
        (\(data) function_for_step_3(data,
            column_name))() |>
        function_for_steps_4_5_6()
    return(result)
}
```

Write Combining Function

```
# Combining Function with minimum 2 and
# maximum 4 input datasets required for
# the pulling it all together stage
CombiningFunction <- function(data1, data2,
    data3 = NULL, data4 = NULL) {
    input <- list(data1, data2, data3, data4)
    county_data <- bind_rows(input[[1]][1:6],
        input[[2]][1:6], input[[3]][1:6],
        input[[4]][1:6])
    noncounty_data <- bind_rows(input[[1]][7:11],
        input[[2]][7:11], input[[3]][7:11],
        input[[4]][7:11])
    combined <- list(county_data = tibble(county_data),
        noncounty_data = tibble(noncounty_data))
    return(combined)
}
```

Writing a Generic Function for Summarizing

Plot Function for State Data

```
#Plot function for state data
library(ggplot2)
plot.state <- function(df,var_name="state_data.Enrollment_Value"){
  plot_data <- df|> group_by(state_data.division, state_data.Year)|>
    #groups our data by division and year
    summarize(y_axis=mean(get(var_name))) |> #gets our mean of enrollment by year and divisi
    filter(state_data.division != 'ERROR') #filter out the ERROR

  print(plot_data)
  ggplot(data=plot_data,
         aes(x=state_data.Year,y= y_axis,color=state_data.division)) +
    geom_line() +
    labs(y="Average Enrollment", x="Year", color="Division")
  #Set up a line plot of year and mean enrollment by division
}
```

Plot Function for County Data

```
#Plot function for county data
plot.county <- function(df,var_name="county_data.Enrollment_Value",
                        state = ' AL' , order='Top', n=5){
  plot_data <- df |>
    filter(county_data.state== state) |> #filter by state
    group_by(county_data.area_name) |> #group by area name
    summarize(y_axis=mean(get(var_name)))  #obtain mean of enrollment for each area name

#if else for the top or bottom parameter
  if(order=='Top'){
    plot_data2 <- plot_data |>
      arrange(desc(y_axis)) |>
      slice_head(n= n)
  }
  else {
    plot_data2 <- plot_data |>
      arrange(y_axis)|>
      slice_head(n= n)
```

```
  }

  print(plot_data2)
  #Create plot of the mean enrollment for each area_name in the state
  #plot specifically the top or bottom n of mean enrollment in that area
  ggplot(data=plot_data2, aes(x=county_data.area_name,y= y_axis/1000, group = 1)) +
    #divide y-axis values by 1000 so that the y-axis labels are more readable
    geom_point() + #create scatterplot
    labs(y="Average Enrollment (in thousands)", x="County") +
    scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
    theme(axis.text.x =element_text(size=5))
  #stagger x-axis labels so no overlap
}
```

Putting it Together

Running Original Enrollment Data

```
EDU01a <- processing_wrapper("https://www4.stat.ncsu.edu/~online/datasets/EDU01a.csv",
    "Enrollment")
```

```
# A tibble: 31,980 x 3
   area_name     Enrollment Enrollment_Value
   <chr>         <chr>                 <int>
 1 UNITED STATES EDU010187D         40024299
 2 UNITED STATES EDU010188D         39967624
 3 UNITED STATES EDU010189D         40317775
 4 UNITED STATES EDU010190D         40737600
 5 UNITED STATES EDU010191D         41385442
 6 UNITED STATES EDU010192D         42088151
 7 UNITED STATES EDU010193D         42724710
 8 UNITED STATES EDU010194D         43369917
 9 UNITED STATES EDU010195D         43993459
10 UNITED STATES EDU010196D         44715737
# i 31,970 more rows
```

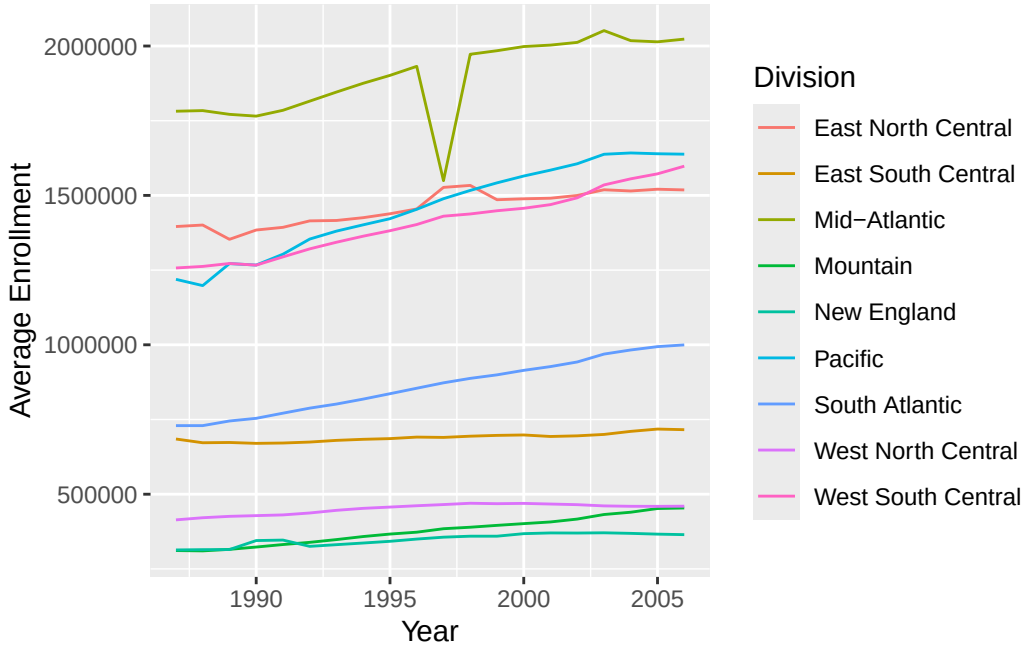
```
EDU01b <- processing_wrapper("https://www4.stat.ncsu.edu/~online/datasets/EDU01b.csv",
    "Enrollment")
```

```
# A tibble: 31,980 x 3
   area_name     Enrollment Enrollment_Value
   <chr>         <chr>                 <int>
 1 UNITED STATES EDU010197D         44534459
 2 UNITED STATES EDU010198D         46245814
 3 UNITED STATES EDU010199D         46368903
 4 UNITED STATES EDU010200D         46818690
 5 UNITED STATES EDU010201D         47127066
 6 UNITED STATES EDU010202D         47606570
 7 UNITED STATES EDU015203D         48506317
 8 UNITED STATES EDU015204D         48693287
 9 UNITED STATES EDU015205D         48978555
10 UNITED STATES EDU015206D         49140702
# i 31,970 more rows
```

```
EDU01ab <- CombiningFunction(EDU01a, EDU01b)

plot.state(EDU01ab[[2]])
```

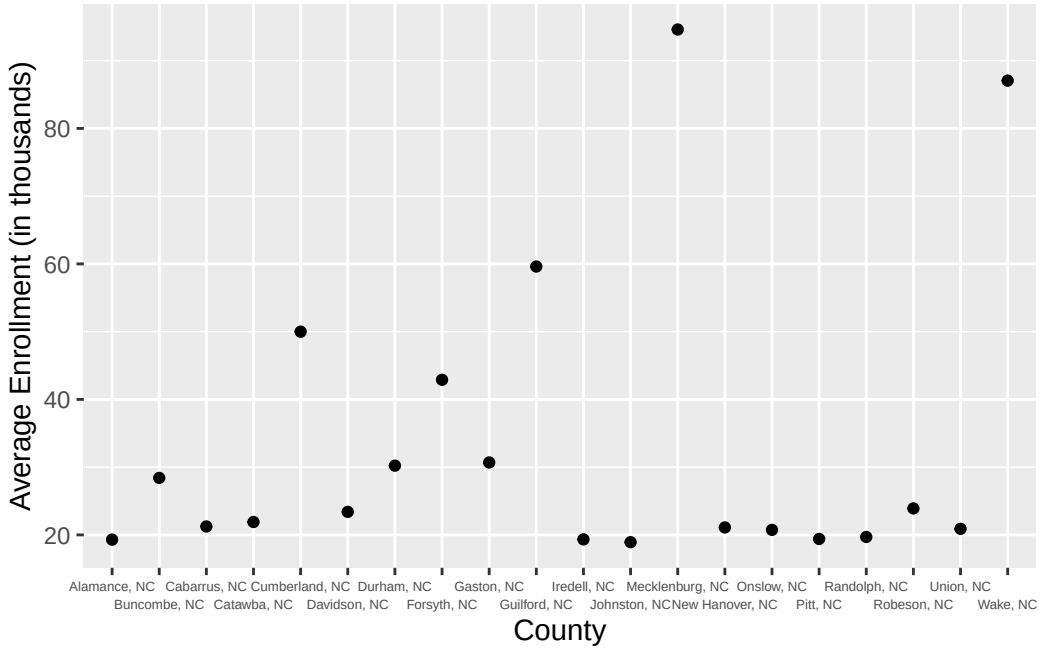
```
# A tibble: 180 x 3
# Groups:   state_data.division [9]
   state_data.division state_data.Year   y_axis
   <chr>                         <dbl>    <dbl>
 1 East North Central             1987 1395868.
 2 East North Central             1988 1400830.
 3 East North Central             1989 1352998.
 4 East North Central             1990 1384188.
 5 East North Central             1991 1393162.
 6 East North Central             1992 1414759.
 7 East North Central             1993 1416112.
 8 East North Central             1994 1425489
 9 East North Central             1995 1438632
10 East North Central             1996 1454695.
# i 170 more rows
```




```
plot.county(EDU01ab[[1]], state = " NC",
    order = "Top", n = 20)
```

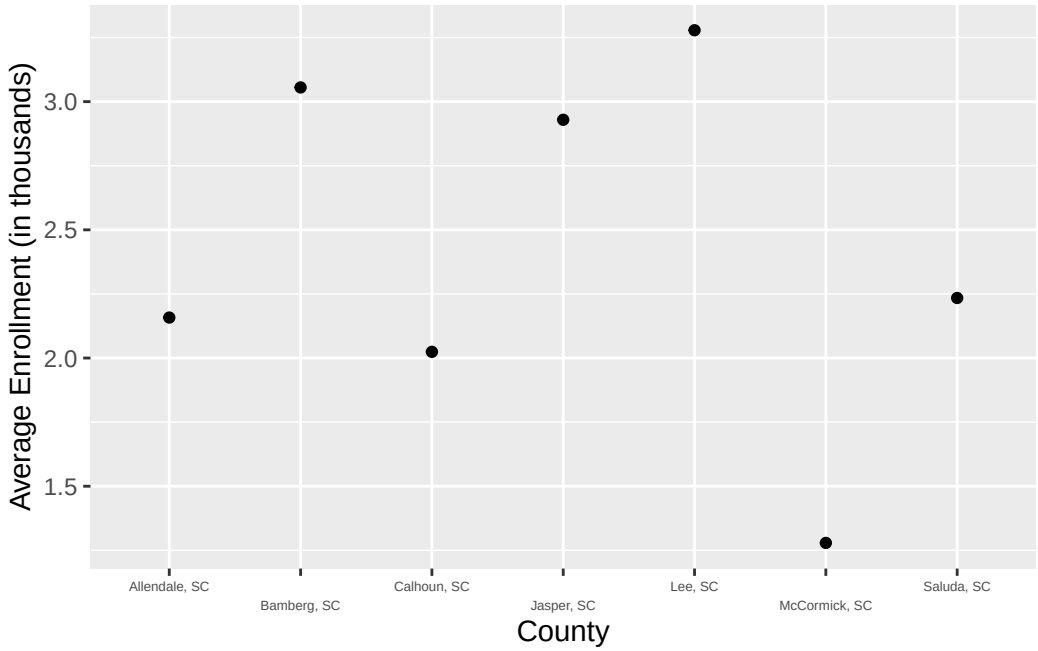
```
# A tibble: 20 x 2
   county_data.area_name y_axis
   <chr>                  <dbl>
 1 Mecklenburg, NC       94598.
 2 Wake, NC              87053.
 3 Guilford, NC          59615.
 4 Cumberland, NC        49999.
 5 Forsyth, NC           42897.
 6 Gaston, NC            30694.
 7 Durham, NC            30223.
 8 Buncombe, NC          28411.
 9 Robeson, NC           23909.
10 Davidson, NC          23405.
11 Catawba, NC           21907.
12 Cabarrus, NC          21244.
13 New Hanover, NC       21103.
14 Union, NC             20900.
15 Onslow, NC            20744.
16 Randolph, NC          19697.
```

```
17 Pitt, NC               19407.
18 Iredell, NC            19341.
19 Alamance, NC           19303.
20 Johnston, NC           18926.
```



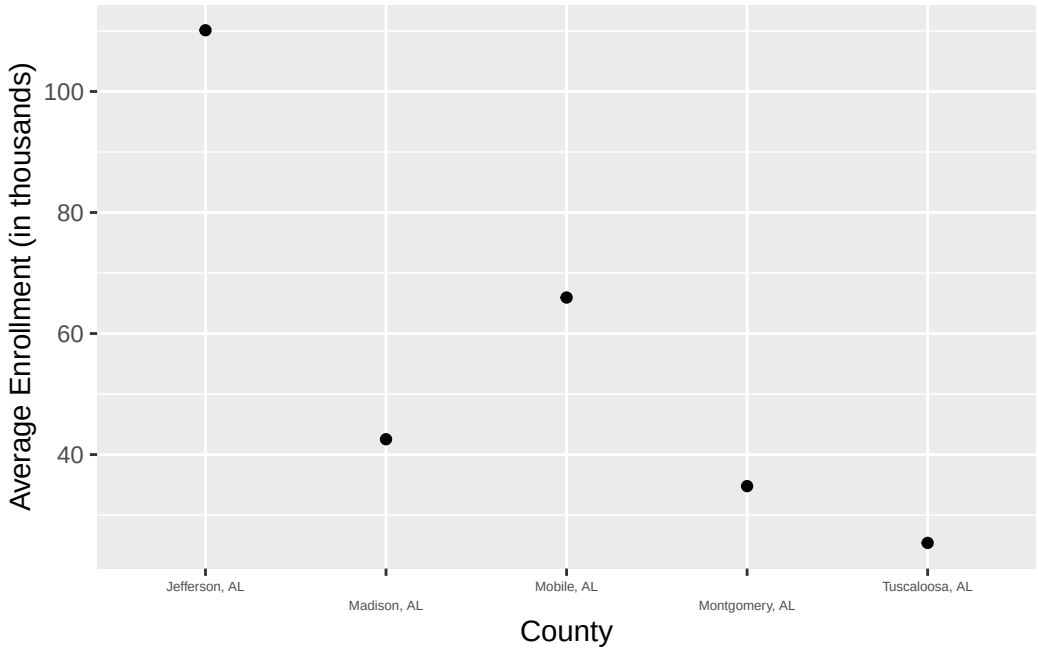

```
plot.county(EDU01ab[[1]], state = " SC",
    order = "Bottom", n = 7)
```

```
# A tibble: 7 x 2
  county_data.area_name y_axis
  <chr>                  <dbl>
1 McCormick, SC          1279.
2 Calhoun, SC            2024.
3 Allendale, SC          2158.
4 Saluda, SC             2234.
5 Jasper, SC             2929.
6 Bamberg, SC            3055.
7 Lee, SC                3278.
```



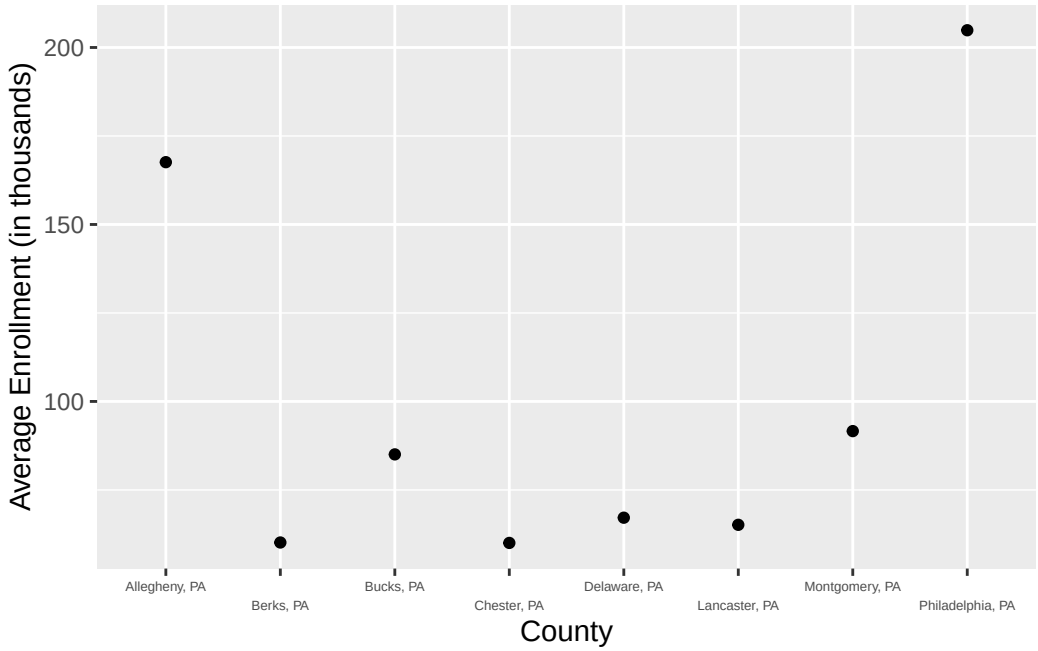

```
plot.county(EDU01ab[[1]])
```

```
# A tibble: 5 x 2
  county_data.area_name  y_axis
  <chr>                   <dbl>
1 Jefferson, AL         110119.
2 Mobile, AL             65952.
3 Madison, AL            42537.
4 Montgomery, AL         34789.
5 Tuscaloosa, AL         25402
```




```
plot.county(EDU01ab[[1]], state = " PA",
    order = "Top", n = 8)
```

```
# A tibble: 8 x 2
  county_data.area_name  y_axis
  <chr>                   <dbl>
1 Philadelphia, PA      204874.
2 Allegheny, PA         167602.
3 Montgomery, PA         91621.
4 Bucks, PA              85044.
5 Delaware, PA           67154.
6 Lancaster, PA          65147.
7 Berks, PA              60150.
8 Chester, PA            60029
```



Run Data Processing Function on Four Other Data Sets

```
PST01a <- processing_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01a.csv",
    "Enrollment")
```

```
# A tibble: 31,980 x 3
   area_name     Enrollment Enrollment_Value
   <chr>         <chr>                 <int>
 1 UNITED STATES PST015171D        206827028
 2 UNITED STATES PST015172D        209283904
 3 UNITED STATES PST015173D        211357490
 4 UNITED STATES PST015174D        213341552
 5 UNITED STATES PST015175D        215465246
 6 UNITED STATES PST015176D        217562728
 7 UNITED STATES PST015177D        219759860
 8 UNITED STATES PST015178D        222095080
 9 UNITED STATES PST015179D        224567234
10 UNITED STATES PST025181D        229466391
# i 31,970 more rows
```

```
PST01b <- processing_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01b.csv",
    "Enrollment")
```

```
# A tibble: 31,980 x 3
   area_name     Enrollment Enrollment_Value
   <chr>         <chr>                 <int>
 1 UNITED STATES PST025182D        231665106
 2 UNITED STATES PST025183D        233792697
 3 UNITED STATES PST025184D        235825544
 4 UNITED STATES PST025185D        237924311
 5 UNITED STATES PST025186D        240133472
 6 UNITED STATES PST025187D        242289738
 7 UNITED STATES PST025188D        244499776
 8 UNITED STATES PST025189D        246819839
 9 UNITED STATES PST030190D        248790925
10 UNITED STATES PST035190D        249622814
# i 31,970 more rows
```

```
PST01c <- processing_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01c.csv",
    "Enrollment")
```

```
# A tibble: 31,980 x 3
   area_name     Enrollment Enrollment_Value
   <chr>         <chr>                 <int>
 1 UNITED STATES PST035191D        252980941
 2 UNITED STATES PST035192D        256514224
 3 UNITED STATES PST035193D        259918588
 4 UNITED STATES PST035194D        263125821
 5 UNITED STATES PST035195D        266278393
 6 UNITED STATES PST035196D        269394284
 7 UNITED STATES PST035197D        272646925
 8 UNITED STATES PST035198D        275854104
 9 UNITED STATES PST035199D        279040168
10 UNITED STATES PST040200D        281424602
# i 31,970 more rows
```

```
PST01d <- processing_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01d.csv",
    "Enrollment")
```

```
# A tibble: 31,980 x 3
```

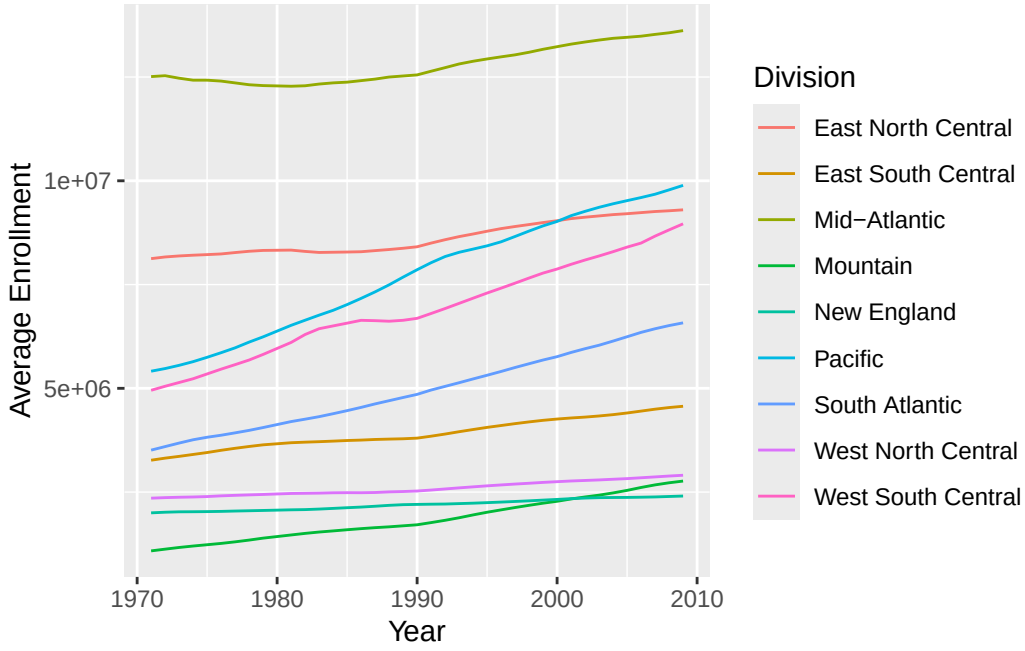
```
   area_name     Enrollment Enrollment_Value
   <chr>         <chr>                 <int>
 1 UNITED STATES PST045200D        282171957
 2 UNITED STATES PST045201D        285081556
 3 UNITED STATES PST045202D        287803914
 4 UNITED STATES PST045203D        290326418
 5 UNITED STATES PST045204D        293045739
 6 UNITED STATES PST045205D        295753151
 7 UNITED STATES PST045206D        298593212
 8 UNITED STATES PST045207D        301579895
 9 UNITED STATES PST045208D        304374846
10 UNITED STATES PST045209D        307006550
# i 31,970 more rows
```

```
PST01abcd <- CombiningFunction(PST01a, PST01b,
    PST01c, PST01d)
```

Plot Other Data Sets with Plot Function

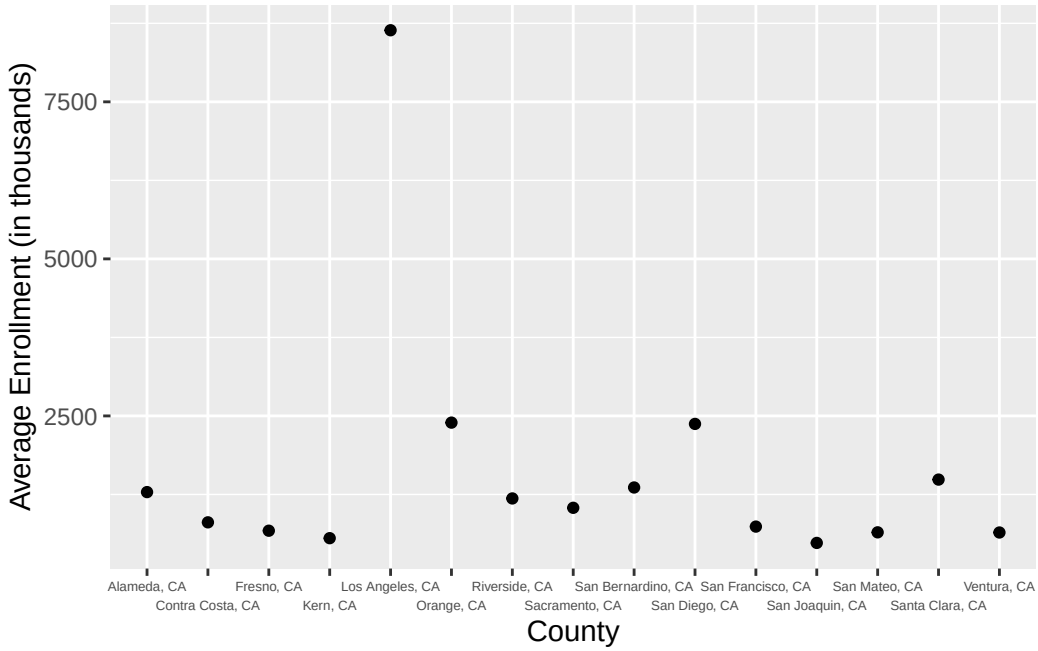
```
plot.state(PST01abcd[[2]])
```

```
# A tibble: 342 x 3
# Groups:   state_data.division [9]
   state_data.division state_data.Year   y_axis
   <chr>                         <dbl>    <dbl>
 1 East North Central             1971 8124464.
 2 East North Central             1972 8164855.
 3 East North Central             1973 8189360
 4 East North Central             1974 8207358.
 5 East North Central             1975 8221077
 6 East North Central             1976 8237345.
 7 East North Central             1977 8270633
 8 East North Central             1978 8301830.
 9 East North Central             1979 8322196.
10 East North Central             1981 8329600.
# i 332 more rows
```



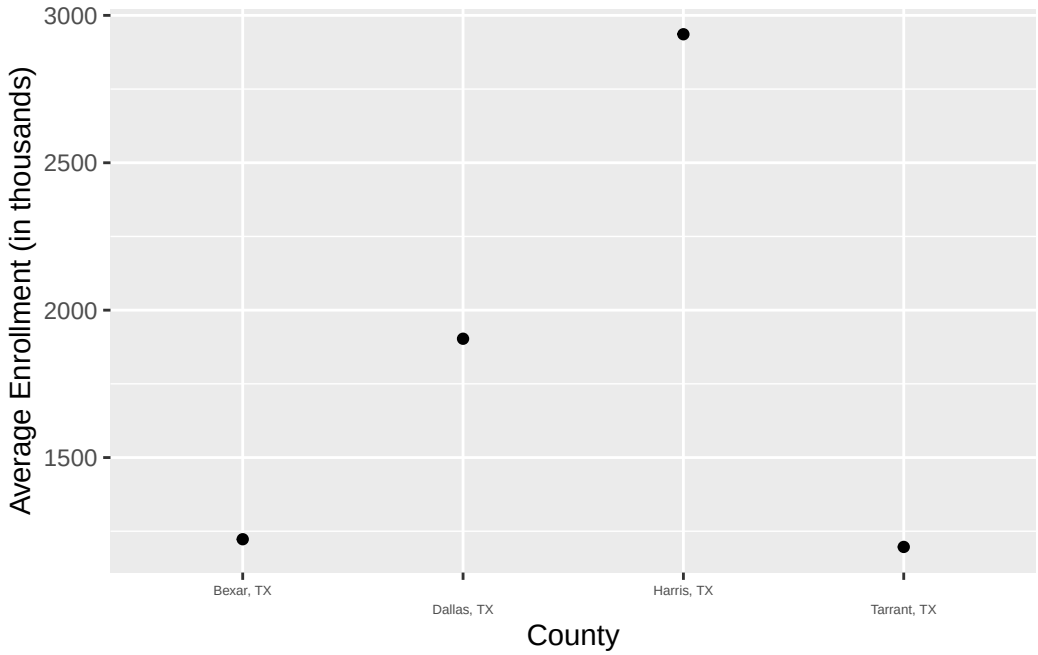

```
plot.county(PST01abcd[[1]], state = " CA",
    order = "Top", n = 15)
```

```
# A tibble: 15 x 2
   county_data.area_name   y_axis
   <chr>                    <dbl>
 1 Los Angeles, CA       8639795.
 2 Orange, CA            2393272.
 3 San Diego, CA         2372674.
 4 Santa Clara, CA       1486544.
 5 San Bernardino, CA    1360795.
 6 Alameda, CA           1287280.
 7 Riverside, CA         1186328.
 8 Sacramento, CA        1037222.
 9 Contra Costa, CA       805928.
10 San Francisco, CA      738149.
11 Fresno, CA             672464.
12 San Mateo, CA          646729.
13 Ventura, CA            644290.
14 Kern, CA               553970.
15 San Joaquin, CA        478151.
```



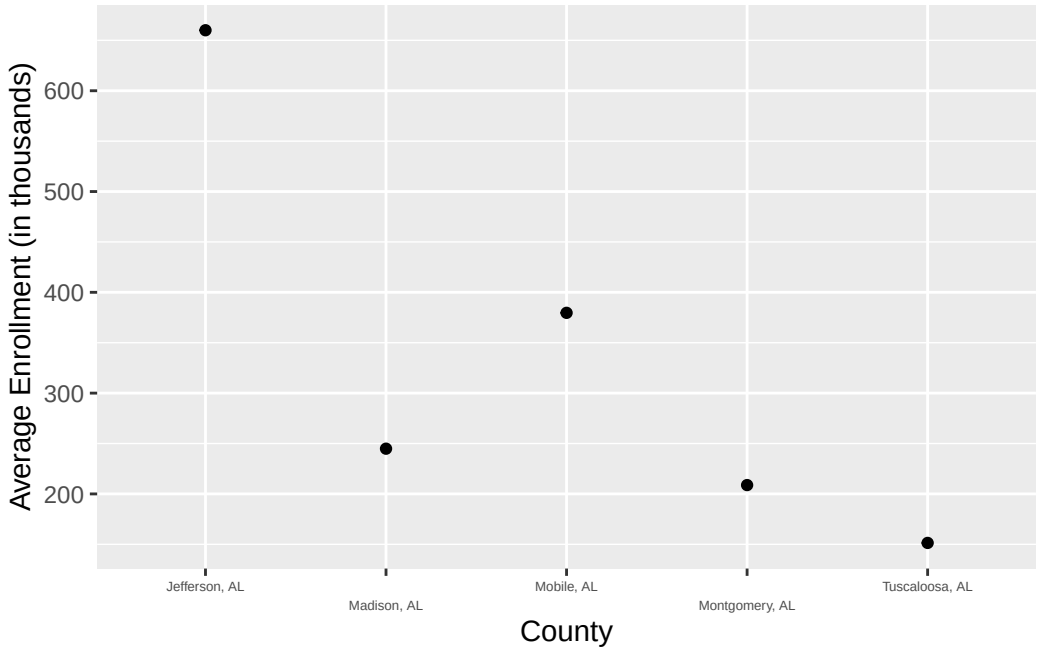

```
plot.county(PST01abcd[[1]], state = " TX",
    order = "Top", n = 4)
```

```
# A tibble: 4 x 2
  county_data.area_name   y_axis
  <chr>                    <dbl>
1 Harris, TX            2935994.
2 Dallas, TX            1903013.
3 Bexar, TX             1222903.
4 Tarrant, TX           1196789.
```



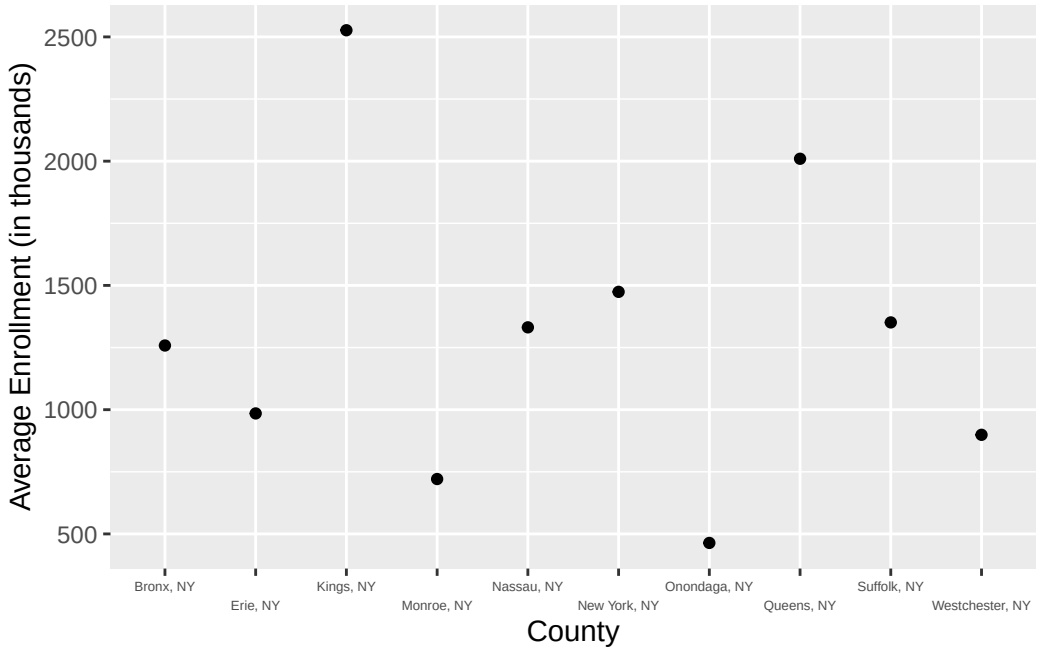

```
plot.county(PST01abcd[[1]])
```

```
# A tibble: 5 x 2
  county_data.area_name  y_axis
  <chr>                   <dbl>
1 Jefferson, AL         660014.
2 Mobile, AL            379642
3 Madison, AL           244899.
4 Montgomery, AL        208781.
5 Tuscaloosa, AL        151387.
```




```
plot.county(PST01abcd[[1]], state = " NY",
    order = "Top", n = 10)
```

```
# A tibble: 10 x 2
   county_data.area_name   y_axis
   <chr>                    <dbl>
 1 Kings, NY             2526834.
 2 Queens, NY            2009665.
 3 New York, NY          1474154.
 4 Suffolk, NY           1351148.
 5 Nassau, NY            1331286.
 6 Bronx, NY             1258213.
 7 Erie, NY               984911.
 8 Westchester, NY        898835.
 9 Monroe, NY             720902.
10 Onondaga, NY           463793.
```


Average Enrollment (in thousands)
2500
2000
1500
1000
500
Bronx, NY
Erie, NY
Kings, NY
Monroe, NY
Nassau, NY
New York, NY
Onondaga, NY
Queens, NY
Suffolk, NY
Westchester, NY
County