Project 1

Trever Yoder and Koji Takagi

Load Packages and Functions

In this section, we load all necessary libraries.

```r
library(tidyverse)
library(readr)
library(ggplot2)
```

Task 1: Data Processing

Question 1: Read in the dataset

We want to read in some of this Census data set, but not all of it. Here we specify which columns we want to read in and we named this data set: `df_selected`. We then `slice` the first 5 lines to display them to confirm we read the data in correctly.

```r
#Read in the data while selecting specific columns
df_selected <- read_csv(
  "https://www4.stat.ncsu.edu/~online/datasets/EDU01a.csv",
  show_col_types = FALSE) %>%
  select(Area_name, STCOU, ends_with("D")) %>% #select specified columns
  rename(area_name = Area_name) #rename "Area_name" as directed

#Display the first 5 lines
df_selected %>%
slice(1:5)
```

```
# A tibble: 5 x 12
  area_name     STCOU EDU010187D EDU010188D EDU010189D EDU010190D EDU010191D
  <chr>         <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
```

```
1 UNITED STATES 00000   40024299   39967624   40317775   40737600   41385442
2 ALABAMA       01000     733735     728234     730048     728252     725541
3 Autauga, AL   01001       6829       6900       6920       6847       7008
4 Baldwin, AL   01003      16417      16465      16799      17054      17479
5 Barbour, AL   01005       5071       5098       5068       5156       5173
# i 5 more variables: EDU010192D <dbl>, EDU010193D <dbl>, EDU010194D <dbl>,
#   EDU010195D <dbl>, EDU010196D <dbl>
```

Question 2: Convert to long format

Now we want to convert the data into long format where each row has only one enrollment value for `area_name`. This converted data will be called `df_long`. We then display the first 5 rows to make sure everything looks as expected.

```r
df_long <- pivot_longer(
  df_selected,
  cols = ends_with("D"),
  names_to = "Survey",
  values_to = "Enrollment Value"
)

#Display the first 5 lines
df_long %>%
  slice(1:5)
```

```
# A tibble: 5 x 4
  area_name     STCOU Survey     `Enrollment Value`
  <chr>         <chr> <chr>                   <dbl>
1 UNITED STATES 00000 EDU010187D           40024299
2 UNITED STATES 00000 EDU010188D           39967624
3 UNITED STATES 00000 EDU010189D           40317775
4 UNITED STATES 00000 EDU010190D           40737600
5 UNITED STATES 00000 EDU010191D           41385442
```

Question 3: Split a variable into 2 variables

Now we need to separate some values that are currently combined in `Survey`. The first 7 digits of `Survey` are currently a `Item_ID` (public school enrollment) and the last 2 digits followed by D are the school year. We want to separate these values to create 2 corresponding variables and turn the year into a 4 digit format. Since we will not be working with any data that was before the year 1925 or after the year 2025, we can do some simple math. The `Year` 1987 will be referring to the Fall 1986-1987 school year.

```
#Separate and create variables from Survey
long_updated <- df_long %>%
  mutate(
    Year = as.numeric(substr(Survey, 8, 9)),
    Year = ifelse(Year > 25, Year + 1900, Year + 2000),
    Item_ID = substr(Survey, 1, 7)
  )

#Display the first 5 lines
long_updated %>%
  slice(1:5)
```

```
# A tibble: 5 x 6
  area_name     STCOU Survey     `Enrollment Value`  Year Item_ID
  <chr>         <chr> <chr>                   <dbl> <dbl> <chr>
1 UNITED STATES 00000 EDU010187D           40024299  1987 EDU0101
2 UNITED STATES 00000 EDU010188D           39967624  1988 EDU0101
3 UNITED STATES 00000 EDU010189D           40317775  1989 EDU0101
4 UNITED STATES 00000 EDU010190D           40737600  1990 EDU0101
5 UNITED STATES 00000 EDU010191D           41385442  1991 EDU0101
```

Question 4: Create county and state data sets

Now we want to create a data set for non-county data and a data set for only county level data. As directed, we will add a class to the county level data tibble that's called `county` and we will create a class for the non-county data called `state`. Then we will print the first 10 rows of each tibble to make sure they look correct.

```
#Create the county and state data sets
county_idx <- grep(", \\w\\w", long_updated$area_name)
county_tibble <- long_updated[county_idx, ]
state_tibble <- long_updated[-county_idx, ]

#add class accordingly
class(county_tibble) <- c("county", class(county_tibble ))
class(state_tibble) <- c("state", class(state_tibble))

#display first 10 lines of county data
```

```
county_tibble  %>%
slice(1:10)
```

```
# A tibble: 10 x 6
   area_name   STCOU Survey     `Enrollment Value`  Year Item_ID
   <chr>       <chr> <chr>                   <dbl> <dbl> <chr>  
 1 Autauga, AL 01001 EDU010187D               6829  1987 EDU0101
 2 Autauga, AL 01001 EDU010188D               6900  1988 EDU0101
 3 Autauga, AL 01001 EDU010189D               6920  1989 EDU0101
 4 Autauga, AL 01001 EDU010190D               6847  1990 EDU0101
 5 Autauga, AL 01001 EDU010191D               7008  1991 EDU0101
 6 Autauga, AL 01001 EDU010192D               7137  1992 EDU0101
 7 Autauga, AL 01001 EDU010193D               7152  1993 EDU0101
 8 Autauga, AL 01001 EDU010194D               7381  1994 EDU0101
 9 Autauga, AL 01001 EDU010195D               7568  1995 EDU0101
10 Autauga, AL 01001 EDU010196D               7834  1996 EDU0101
```

```
#display first 10 lines of state data
state_tibble %>%
slice(1:10)
```

```
# A tibble: 10 x 6
   area_name     STCOU Survey     `Enrollment Value`  Year Item_ID
   <chr>         <chr> <chr>                   <dbl> <dbl> <chr>  
 1 UNITED STATES 00000 EDU010187D           40024299  1987 EDU0101
 2 UNITED STATES 00000 EDU010188D           39967624  1988 EDU0101
 3 UNITED STATES 00000 EDU010189D           40317775  1989 EDU0101
 4 UNITED STATES 00000 EDU010190D           40737600  1990 EDU0101
 5 UNITED STATES 00000 EDU010191D           41385442  1991 EDU0101
 6 UNITED STATES 00000 EDU010192D           42088151  1992 EDU0101
 7 UNITED STATES 00000 EDU010193D           42724710  1993 EDU0101
 8 UNITED STATES 00000 EDU010194D           43369917  1994 EDU0101
 9 UNITED STATES 00000 EDU010195D           43993459  1995 EDU0101
10 UNITED STATES 00000 EDU010196D           44715737  1996 EDU0101
```

Question 5: Add the state to each county

Now we want to add the state that each `county` corresponds to as a new variable in our tibble. This new variable will be called `State`.

```
# Add State to county
county_tibble <- county_tibble %>%
  mutate(State = substr(area_name, nchar(area_name) - 1, nchar(area_name)))

#display first 10 lines of county data
county_tibble  %>%
slice(1:10)
```

```
# A tibble: 10 x 7
   area_name   STCOU Survey     `Enrollment Value`  Year Item_ID State
   <chr>       <chr> <chr>                   <dbl> <dbl> <chr>   <chr>
 1 Autauga, AL 01001 EDU010187D               6829  1987 EDU0101 AL
 2 Autauga, AL 01001 EDU010188D               6900  1988 EDU0101 AL
 3 Autauga, AL 01001 EDU010189D               6920  1989 EDU0101 AL
 4 Autauga, AL 01001 EDU010190D               6847  1990 EDU0101 AL
 5 Autauga, AL 01001 EDU010191D               7008  1991 EDU0101 AL
 6 Autauga, AL 01001 EDU010192D               7137  1992 EDU0101 AL
 7 Autauga, AL 01001 EDU010193D               7152  1993 EDU0101 AL
 8 Autauga, AL 01001 EDU010194D               7381  1994 EDU0101 AL
 9 Autauga, AL 01001 EDU010195D               7568  1995 EDU0101 AL
10 Autauga, AL 01001 EDU010196D               7834  1996 EDU0101 AL
```

Question 6: Add Division to the state tibble

Now for our non-county level tibble, we want to create a new variable called `Division` that corresponds to the state's classification of division. Since we will have many conditions, we will use `case_when`. For rows that correspond to a non-state (such as UNITED STATES) we will return `ERROR`.

```
# Add Division to non-county tibble
state_tibble <- state_tibble %>%
  mutate(area_name = toupper(area_name)) %>%
  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"
        ) ~ "Middle 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", "MARYLAND", "DISTRICT OF COLUMBIA",
        "VIRGINIA", "WEST VIRGINIA", "NORTH CAROLINA",
        "SOUTH CAROLINA", "GEORGIA", "FLORIDA"
        ) ~ "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"
    )
  )

# Check Division table (no zeros!)
state_tibble %>% count(Division, area_name) %>% print(n = 52)
```

```
# A tibble: 52 x 3
   Division           area_name                n
   <chr>              <chr>                <int>
 1 ERROR              UNITED STATES           10
 2 East North Central ILLINOIS                10
 3 East North Central INDIANA                 10
 4 East North Central MICHIGAN                10
 5 East North Central OHIO                    10
 6 East North Central WISCONSIN               10
 7 East South Central ALABAMA                 10
 8 East South Central KENTUCKY                10
 9 East South Central MISSISSIPPI             10
10 East South Central TENNESSEE               10
11 Middle Atlantic    NEW JERSEY              10
12 Middle Atlantic    NEW YORK                10
13 Middle Atlantic    PENNSYLVANIA            10
14 Mountain           ARIZONA                 10
15 Mountain           COLORADO                10
16 Mountain           IDAHO                   10
17 Mountain           MONTANA                 10
18 Mountain           NEVADA                  10
19 Mountain           NEW MEXICO              10
20 Mountain           UTAH                    10
21 Mountain           WYOMING                 10
22 New England        CONNECTICUT             10
23 New England        MAINE                   10
24 New England        MASSACHUSETTS           10
25 New England        NEW HAMPSHIRE           10
26 New England        RHODE ISLAND            10
27 New England        VERMONT                 10
28 Pacific            ALASKA                  10
29 Pacific            CALIFORNIA              10
30 Pacific            HAWAII                  10
31 Pacific            OREGON                  10
32 Pacific            WASHINGTON              10
33 South Atlantic     DELAWARE                10
34 South Atlantic     DISTRICT OF COLUMBIA    20
35 South Atlantic     FLORIDA                 10
36 South Atlantic     GEORGIA                 10
37 South Atlantic     MARYLAND                10
38 South Atlantic     NORTH CAROLINA          10
39 South Atlantic     SOUTH CAROLINA          10
40 South Atlantic     VIRGINIA                10
41 South Atlantic     WEST VIRGINIA           10
42 West North Central IOWA                    10
43 West North Central KANSAS                  10
44 West North Central MINNESOTA               10
45 West North Central MISSOURI                10
46 West North Central NEBRASKA                10
47 West North Central NORTH DAKOTA            10
48 West North Central SOUTH DAKOTA            10
49 West South Central ARKANSAS                10
50 West South Central LOUISIANA               10
```

```
51 West South Central OKLAHOMA                10
52 West South Central TEXAS                   10
```

Task 2: Simplify Task 1 through 6 with functions

Questions 1-2 with a function

We created a function that has an optional argument to allow the user to specify the name of the column representing the value (which is `Enrollment Value` in this case)

```
#Create the function
long_format_conversion <- function(df, value = "Enrollment Value") {
  selected_data <- df %>%
    # Select required columns
    select(Area_name, STCOU, ends_with("D")) %>%
    # Rename Area_name to area_name
    rename(area_name = Area_name)

  # Convert to long format
  long_updated <- pivot_longer(
    selected_data,
    cols = ends_with("D"),
    names_to = "Survey",
    values_to = value
  )
  return(long_updated)
}
```

Question 3 with a function

Now we want to break `survey` into 2 variables just like we did in question 3, however, we are going to make a function this time.

```
survey_function <- function(long_updated) {
  long_data_updated <- long_updated %>%
    # Extract the last two digits as Year and convert to numeric
    mutate(Year = as.numeric(substr(Survey, start = 8, stop = 9))) %>%
    # Convert two-digit Year to four-digit Year
    mutate(Year = ifelse(Year > 25, Year + 1900, Year + 2000)) %>%
    # Extract the first 7 characters as Item_ID
    mutate(Item_ID = substr(Survey, start = 1, stop = 7))
```

```r
  return(long_data_updated)
}
```

Question 5 with a function

Now we are creating a function to do step 5.

```r
state_function <- function(county_tibble) {
  new_county_tibble <- county_tibble %>%
    mutate(State = substr(area_name, start = nchar(area_name) - 1,
                          stop = nchar(area_name)))
  return(new_county_tibble)
}
```

Question 6 with a function

Now we are completing step 6, but with a function

```r
division_function <- function(state_tibble) {
  state_tibble_updated <- state_tibble %>%
    mutate(area_name = toupper(area_name)) %>%  # Ensure uppercase for matching
    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"
        ) ~ "Middle 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", "MARYLAND", "DISTRICT OF COLUMBIA",
          "VIRGINIA", "WEST VIRGINIA", "NORTH CAROLINA",
          "SOUTH CAROLINA", "GEORGIA", "FLORIDA"
        ) ~ "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_updated)
}
```

Questions 4-6 with 1 function

In order to create two new tibbles from the output from step 3 and apply steps 5-6, we need to create a new function to perform the following: take in the output from step 3, Creates 2 tibbles like in step 4, then calls the functions from steps 5 and 6 to return two final tibbles.

```
create_datasets <- function(long_updated) {
  # Split by presence of ", XX" at end of area_name
  county_indices <- grep(pattern = ", \\w\\w", long_updated$area_name)
  state_tibble <- long_updated[-county_indices, ]
  class(state_tibble) <- c("state", class(state_tibble))

  county_tibble <- long_updated[county_indices, ]
  class(county_tibble) <- c("county", class(county_tibble))

  final_county_tibble <- state_function(county_tibble)
  final_state_tibble <- division_function(state_tibble)

  return(list(county = final_county_tibble, state = final_state_tibble))
}
```

Create a wrapper function

We can combine all of these functions into one function call. We want this function to take in a URL of a `.csv` file and apply all of the functions listed above. We still want to be able to easily change `Enrollment Value` for when we use other census data sets.

```r
my_wrapper <- function(url, value = "Enrollment Value") {
  result <- read_csv(url, show_col_types = FALSE) %>%
    long_format_conversion(value = value) %>%
    survey_function() %>%
    create_datasets()
  return(result)
}
```

Call and combine Our data

We want to combine our data from two different URLs. We will create a single short function that takes in the results of the two calls in our wrapper function. At the end of this report, we will apply our wrapper functions to a few different data sets.

```r
combine_wrapper_results <- function(wrapper1, wrapper2) {
  combined_county <- bind_rows(wrapper1$county, wrapper2$county)
  combined_state <- bind_rows(wrapper1$state, wrapper2$state)
  list(county = combined_county, state = combined_state)
}
```

Task 3: Writing a Generic Function for Summarizing

Plot function for state

Now we want to summarize some of our data. We want to do this efficiently, so we will create some functions. Here, we do not want to include any of the `Divisions` that are listed as `ERROR` in our data sets. This function will be used to create our state plots.

```r
plot.state <- function(df, var_name = "Enrollment Value") {
  df %>%
    filter(Division != "ERROR") %>% #exclude error divisions
    group_by(Division, Year) %>%
    summarize(mean_value = mean(get(var_name), na.rm = TRUE), .groups = "drop") %>%
    ggplot(aes(x = Year, y = mean_value, color = Division)) +
    geom_line() +
    labs(
      title = "Mean Value per Division over Years",
      y = paste("Mean of", var_name),
      x = "Year"
    )+
    scale_y_continuous(labels = scales::comma) +  # normal number format
    guides(color = guide_legend(ncol = 2))        # multi-column legend
}
```

Plot function for county

We will now create a similar function for our county data set, however we will add some calculations to this function. We want to be able to sort by the top or bottom mean enrollment values with our function, so we have to do some sorting. It's important to note that we are sorting based on mean values, but we are plotting the actual statistic values (`Enrollmen Value` and later we will display `Resident Population`).

```
plot.county <- function(df, var_name = "Enrollment Value",
                        state = "NC", top_or_bottom = "top", n = 5) {

  # Filter data for selected state
  df_state <- df %>% filter(State == state)

  # Calculate mean for each county
  mean_df <- df_state %>%
    group_by(area_name) %>%
    summarize(mean_value = mean(get(var_name), na.rm = TRUE), .groups = "drop")

  # Sort by mean and pick top or bottom n
  if (top_or_bottom == "top") {
    mean_df <- mean_df %>% arrange(desc(mean_value))
  } else {
    mean_df <- mean_df %>% arrange(mean_value)
  }

  selected_areas <- mean_df %>% slice_head(n = n) %>% pull(area_name)

  # Filter original data for selected counties
  df_selected <- df_state %>% filter(area_name %in% selected_areas)

  # Plot actual values (not means)
  ggplot(df_selected, aes(x = Year, y = get(var_name), color = area_name)) +
    geom_line() +
    labs(
      title = paste(top_or_bottom, n, "counties in", state),
      y = var_name,
      x = "Year"
    )+
    scale_y_continuous(labels = scales::comma) +  # normal number format
    guides(color = guide_legend(ncol = 2))        # multi-column legend
}
```

Task 4: Putting it Together

Process the EDU Data Sets

We run our wrapper function on the two EDU data sets and inspect the results. Since we are looking at enrollment values, all of our plots and code use `value = Enrollment value`. This is referring to the census observed enrollment counts at given locations and given years. Again, With our current data sets, we are looking at the Public school enrollment for school years.

```
edu1 <- my_wrapper(
  "https://www4.stat.ncsu.edu/~online/datasets/EDU01a.csv",
  value = "Enrollment Value"
  )
edu2 <- my_wrapper(
  "https://www4.stat.ncsu.edu/~online/datasets/EDU01b.csv",
  value = "Enrollment Value"
  )

# Inspect to ensure correctness
head(edu1$county)
```

```
# A tibble: 6 x 7
  area_name   STCOU Survey     `Enrollment Value`  Year Item_ID State
  <chr>       <chr> <chr>                   <dbl> <dbl> <chr>   <chr>
1 Autauga, AL 01001 EDU010187D               6829  1987 EDU0101 AL
2 Autauga, AL 01001 EDU010188D               6900  1988 EDU0101 AL
3 Autauga, AL 01001 EDU010189D               6920  1989 EDU0101 AL
4 Autauga, AL 01001 EDU010190D               6847  1990 EDU0101 AL
5 Autauga, AL 01001 EDU010191D               7008  1991 EDU0101 AL
6 Autauga, AL 01001 EDU010192D               7137  1992 EDU0101 AL
```

```r
head(edu1$state)
```

```
# A tibble: 6 x 7
  area_name     STCOU Survey     `Enrollment Value`  Year Item_ID Division
  <chr>         <chr> <chr>                   <dbl> <dbl> <chr>   <chr>
1 UNITED STATES 00000 EDU010187D           40024299  1987 EDU0101 ERROR
2 UNITED STATES 00000 EDU010188D           39967624  1988 EDU0101 ERROR
3 UNITED STATES 00000 EDU010189D           40317775  1989 EDU0101 ERROR
4 UNITED STATES 00000 EDU010190D           40737600  1990 EDU0101 ERROR
5 UNITED STATES 00000 EDU010191D           41385442  1991 EDU0101 ERROR
6 UNITED STATES 00000 EDU010192D           42088151  1992 EDU0101 ERROR
```

Combine EDU Data Sets

Here we use our combining function to merge the two processed data sets. We then display the first few rows to make sure this worked as intended.

```r
edu_combined <- combine_wrapper_results(edu1, edu2)
head(edu_combined$county)
```

```
# A tibble: 6 x 7
  area_name   STCOU Survey     `Enrollment Value`  Year Item_ID State
  <chr>       <chr> <chr>                   <dbl> <dbl> <chr>   <chr>
1 Autauga, AL 01001 EDU010187D               6829  1987 EDU0101 AL
2 Autauga, AL 01001 EDU010188D               6900  1988 EDU0101 AL
3 Autauga, AL 01001 EDU010189D               6920  1989 EDU0101 AL
4 Autauga, AL 01001 EDU010190D               6847  1990 EDU0101 AL
5 Autauga, AL 01001 EDU010191D               7008  1991 EDU0101 AL
6 Autauga, AL 01001 EDU010192D               7137  1992 EDU0101 AL
```

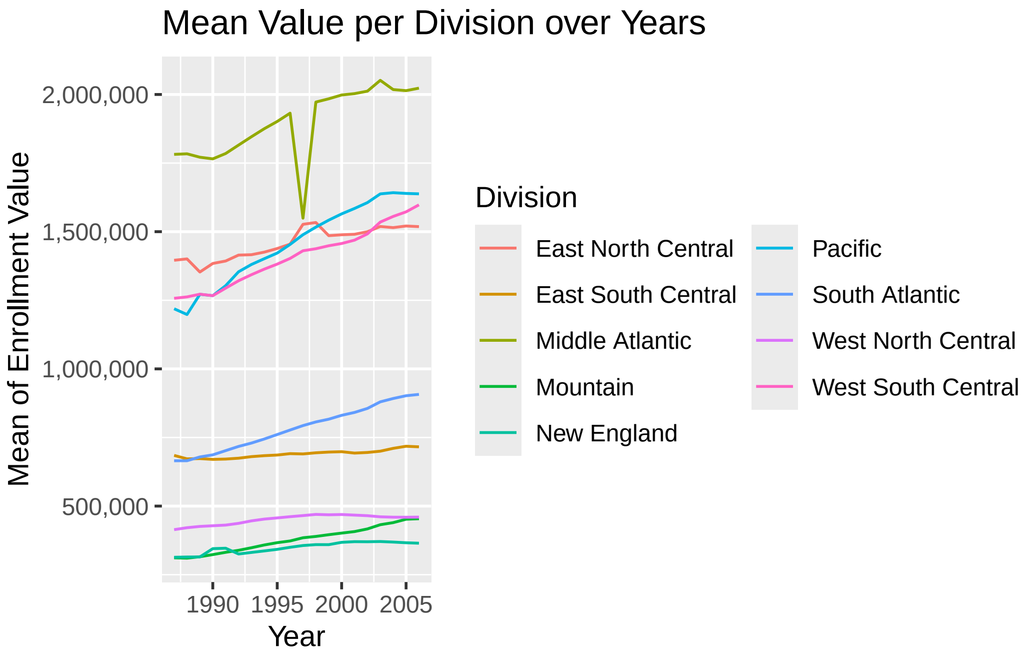
```r
head(edu_combined$state)
```

```
# A tibble: 6 x 7
  area_name     STCOU Survey     `Enrollment Value`  Year Item_ID Division
  <chr>         <chr> <chr>                   <dbl> <dbl> <chr>   <chr>
1 UNITED STATES 00000 EDU010187D           40024299  1987 EDU0101 ERROR
2 UNITED STATES 00000 EDU010188D           39967624  1988 EDU0101 ERROR
3 UNITED STATES 00000 EDU010189D           40317775  1989 EDU0101 ERROR
4 UNITED STATES 00000 EDU010190D           40737600  1990 EDU0101 ERROR
5 UNITED STATES 00000 EDU010191D           41385442  1991 EDU0101 ERROR
6 UNITED STATES 00000 EDU010192D           42088151  1992 EDU0101 ERROR
```

State Plot for EDU Data

This plot shows the mean enrollment by `Division` across years. We are excluding `Error` divisions.

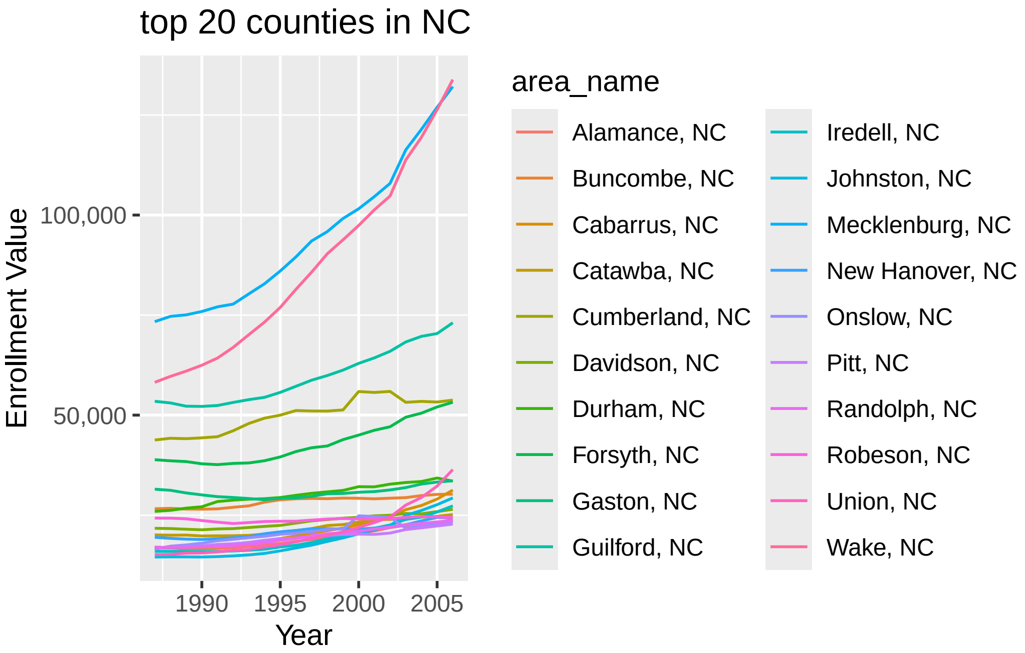
```
plot(edu_combined$state, var_name = "Enrollment Value")
```



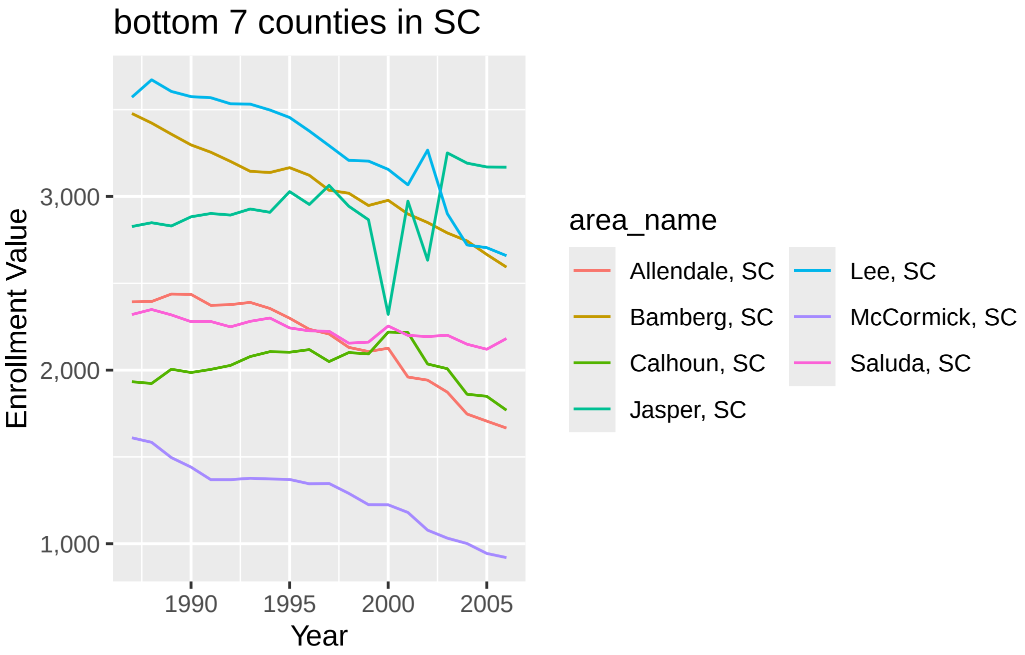
County Plots for EDU Data

Below are various plots for county data, demonstrating flexibility in selecting state, top/bottom, and count.

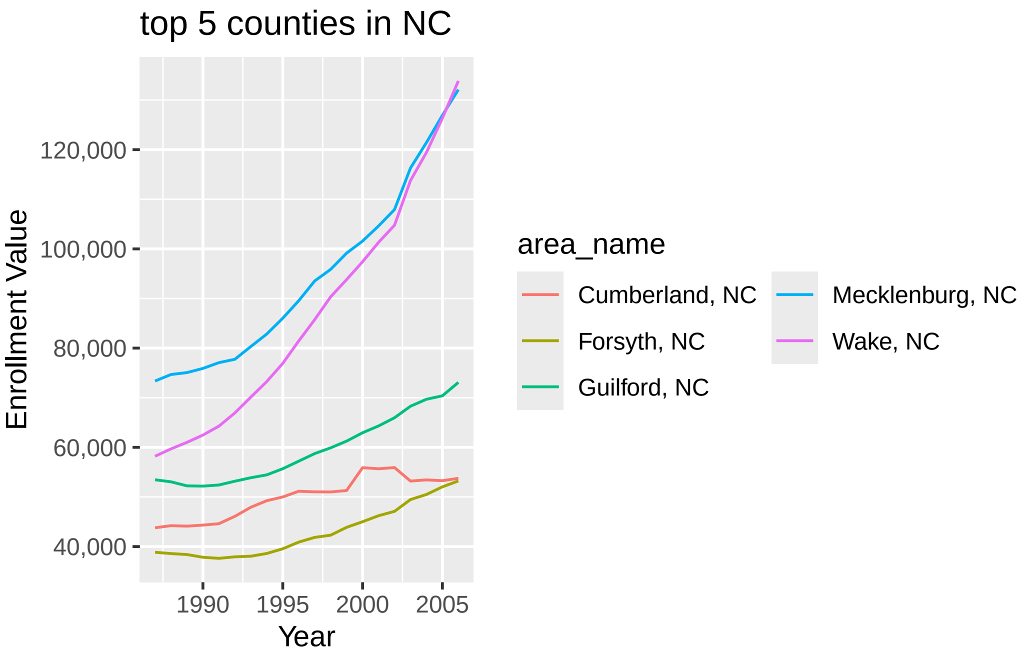
```
# NC, top 20
plot(edu_combined$county, var_name = "Enrollment Value",
     state = "NC", top_or_bottom = "top", n = 20)
```



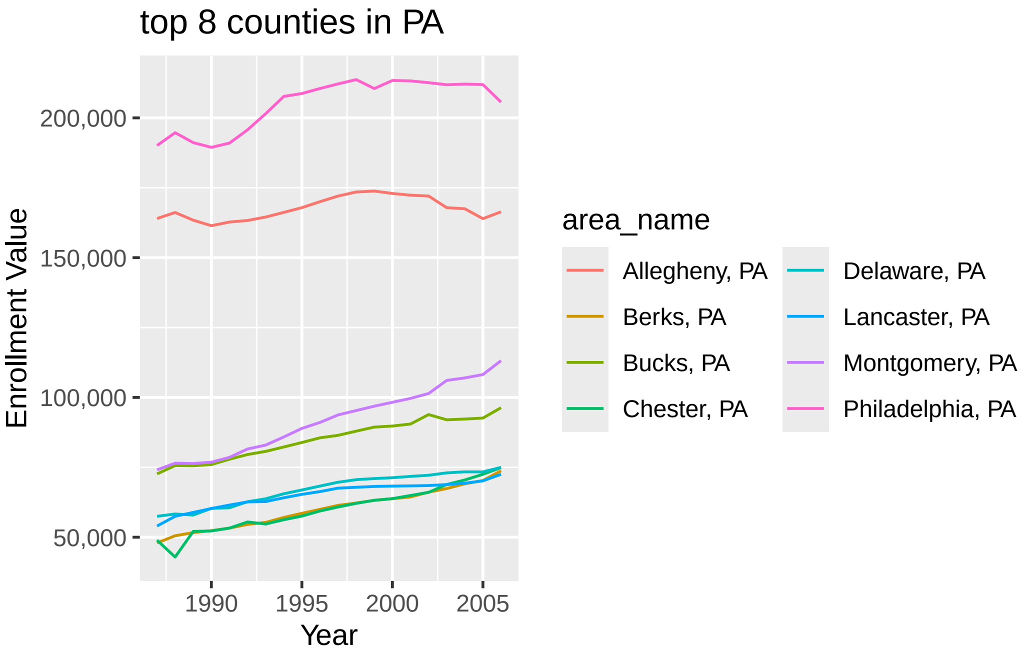
```
# SC, bottom 7
plot(edu_combined$county, var_name = "Enrollment Value",
     state = "SC", top_or_bottom = "bottom", n = 7)
```



```r
# Default (uses NC top 5)
plot(edu_combined$county, var_name = "Enrollment Value")
```



```r
# PA, top 8
plot(edu_combined$county, var_name = "Enrollment Value",
     state = "PA", top_or_bottom = "top", n = 8)
```



##Task 4.5: Apply workflow to PST data sets

Process PST Data Sets

We repeat the same workflow for the four PST data sets. We start by processing that data then combining the data sets. For the PST data, we are looking at Resident total population estimates at given locations on given years. `Year` 1971 is referencing July 1st 1971. We will be using `Resident Population` as our statistic value.

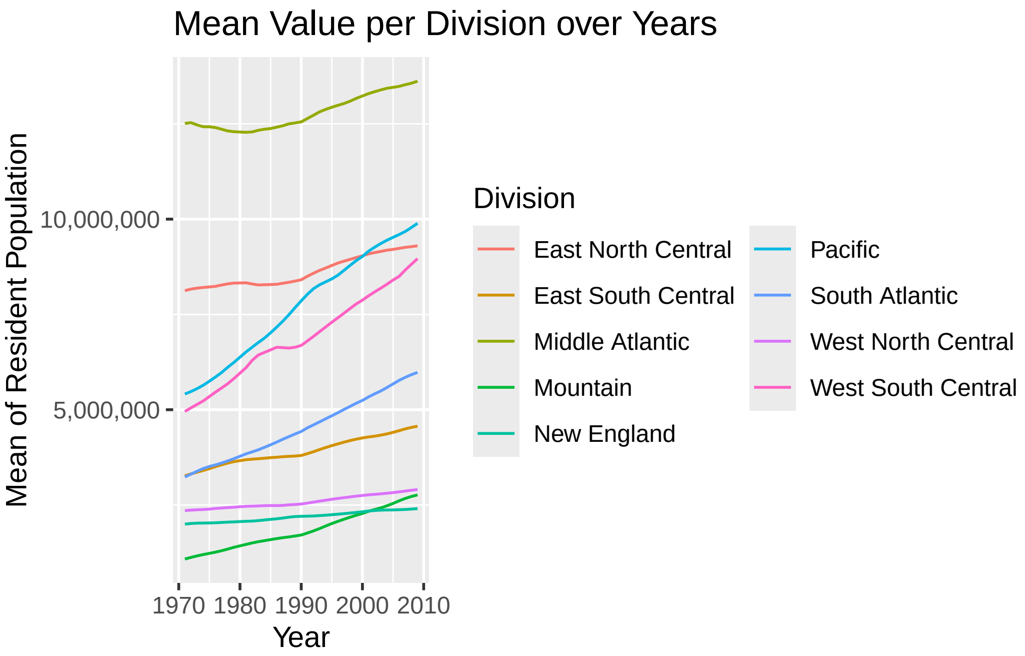
```
pst1 <- my_wrapper(
  "https://www4.stat.ncsu.edu/~online/datasets/PST01a.csv",
  value = "Resident Population"
  )
pst2 <- my_wrapper(
  "https://www4.stat.ncsu.edu/~online/datasets/PST01b.csv",
  value = "Resident Population"
  )
pst3 <- my_wrapper(
  "https://www4.stat.ncsu.edu/~online/datasets/PST01c.csv",
  value = "Resident Population"
  )
pst4 <- my_wrapper(
  "https://www4.stat.ncsu.edu/~online/datasets/PST01d.csv",
  value = "Resident Population"
  )

# Combine step by step
pst12 <- combine_wrapper_results(pst1, pst2)
pst34 <- combine_wrapper_results(pst3, pst4)
pst_combined <- combine_wrapper_results(pst12, pst34)
```

State Plot for PST Data

Now that the PST data is processed, we will make our state plot.

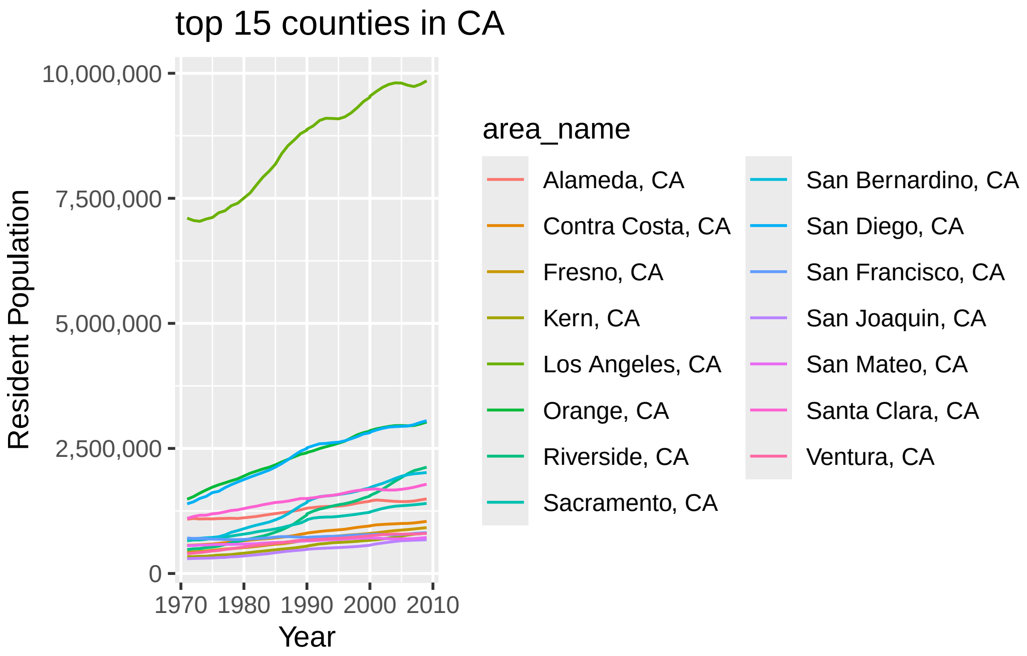
```
plot(pst_combined$state, var_name = "Resident Population")
```



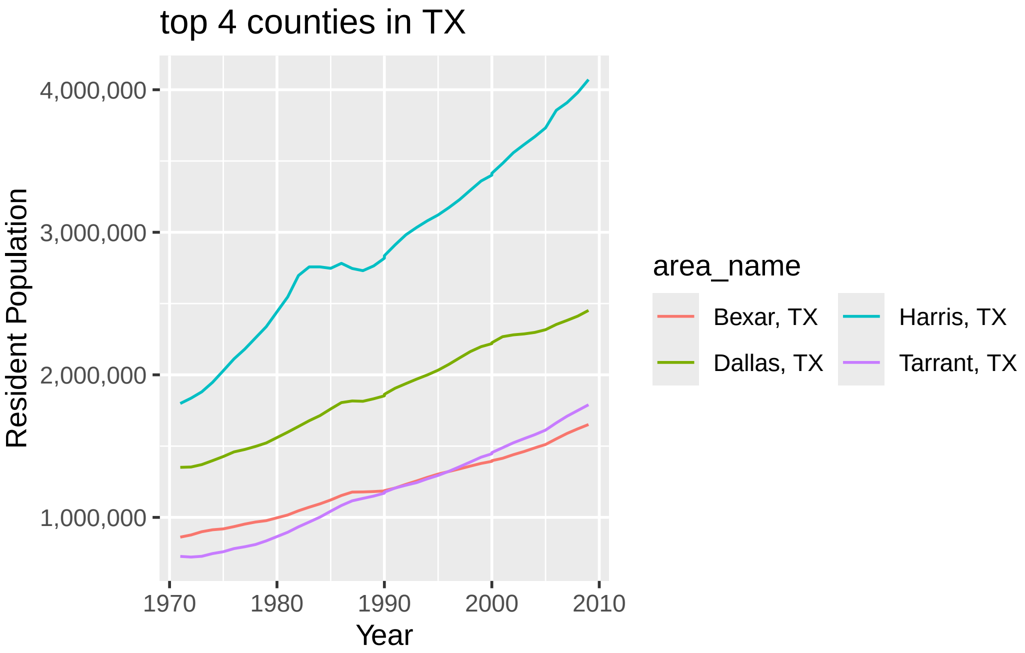
County Plots for PST Data

Finally, we want to demonstrate the flexibility of our summary functions with the PST Data. As noted in the code chunk below, we created plots that show: CA top 15, TX top 4, the default plot function, and NY top 10.

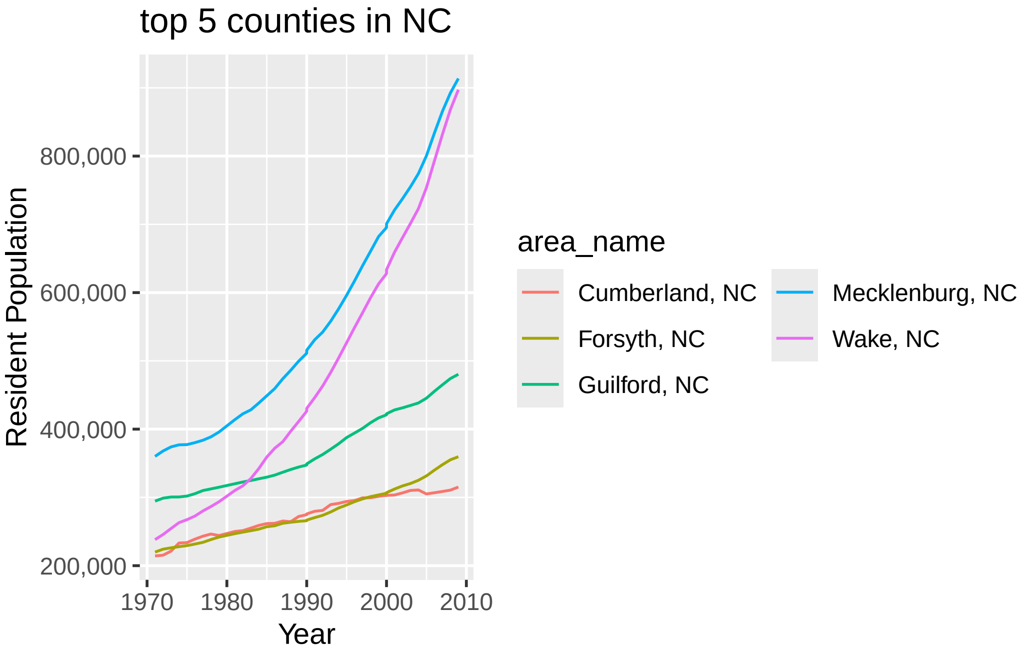
```
# CA, top 15
plot(pst_combined$county, var_name = "Resident Population",
     state = "CA", top_or_bottom = "top", n = 15)
```



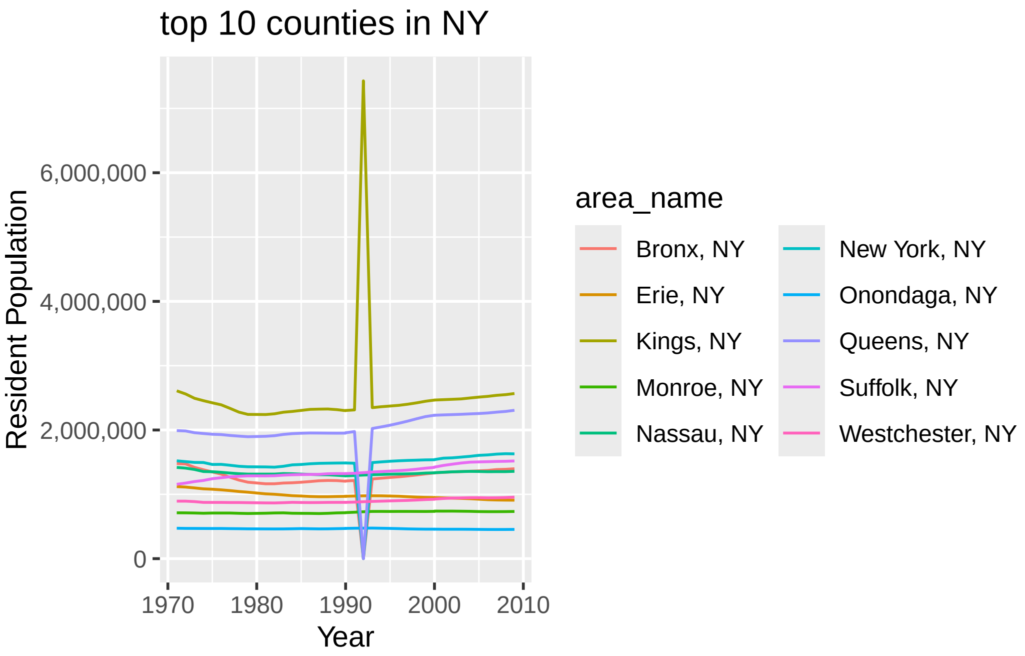
```
# TX, top 4
plot(pst_combined$county, var_name = "Resident Population",
     state = "TX", top_or_bottom = "top", n = 4)
```



```
# Default
plot(pst_combined$county, var_name = "Resident Population")
```



```
# NY, top 10
plot(pst_combined$county, var_name = "Resident Population",
     state = "NY", top_or_bottom = "top", n = 10)
```


top 10 counties in NY
Resident Population
Year
area_name
Bronx, NY
Erie, NY
Kings, NY
Monroe, NY
Nassau, NY
New York, NY
Onondaga, NY
Queens, NY
Suffolk, NY
Westchester, NY