**#THE COUNTIES DATASET**

-Select only few columns

**\*counties %>%**

**\*select(state, counties, population, unemployement)**

-Selecting Columns

# Select the columns

counties %>%

  select(state, county, population, poverty)

-Arranging Observation

counties\_selected <- counties %>%

  select(state, county, population, private\_work, public\_work, self\_employed)

# Add a verb to sort in descending order of public\_work

counties\_selected %>%

  arrange(desc(public\_work))

-Filtering for condition

counties\_selected <- counties %>%

  select(state, county, population)

# Filter for counties in the state of California that have a population above 1000000

counties\_selected %>%

  filter(state=="California",

  population>1000000)

-Filtering and Arrange

counties\_selected <- counties %>%

  select(state, county, population, private\_work, public\_work, self\_employed)

# Filter for Texas and more than 10000 people; sort in descending order of private\_work

counties\_selected %>%

  filter(state=="Texas", population>10000)%>%

  arrange(desc(private\_work))

-Calculating the number of employee

counties\_selected <- counties %>%

  select(state, county, population, public\_work)

# Sort in descending order of the public\_workers column

counties\_selected %>%

  mutate(public\_workers = public\_work \* population / 100) %>%

arrange(desc(public\_workers))

-Calculating percentage of women in county

# Select the columns state, county, population, men, and women

counties\_selected <- counties %>%

  select(state, county, population, men, women)

# Calculate proportion\_women as the fraction of the population made up of women

counties\_selected %>%

  mutate(proportion\_women = women/population)

-Put them all

counties %>%

  # Select the five columns

  select(state, county, population, men, women) %>%

  # Add the proportion\_men variable

  mutate(proportion\_men = men/population) %>%

  # Filter for population of at least 10,000

  filter(population>= 10000)%>%

  # Arrange proportion of men in descending order

  arrange(desc(proportion\_men))

-The count verb

-sort in count :

**\*counties %>%**

**\*count(state, sort = TRUE)**

-Add Weight

**\*counties %>%**

**\*count(state, wt = population, sort = TRUE) -> instead of calculating number of counties in every state, this formula calculate total population in each state.**

-Counting by region

# Use count to find the number of counties in each region

counties\_selected %>%

  count(region, sort=TRUE)

-Counting citizens by state

# Find number of counties per state, weighted by citizens

counties\_selected %>%

 count(state, wt=citizens, sort=TRUE)

-Mutating and Counting

counties\_selected %>%

  # Add population\_walk containing the total number of people who walk to work

  mutate(population\_walk = population \* walk / 100) %>%

  # Count weighted by the new column

  count(state, wt = population\_walk, sort = TRUE)

-Summarizing

# Summarize to find minimum population, maximum unemployment, and average income

counties\_selected %>%

  summarize(min\_population=min(population),

  max\_unemployment=max(unemployment),

  average\_income=mean(income))

-Summarizing by state

# Add a density column, then sort in descending order

counties\_selected %>%

  group\_by(state) %>%

  summarize(total\_area = sum(land\_area),

            total\_population = sum(population)) %>%

  mutate(density = total\_population / total\_area) %>%

  arrange(desc(density))

-Summarizing by state and region

# Add a density column, then sort in descending order

counties\_selected %>%

  group\_by(state) %>%

  summarize(total\_area = sum(land\_area),

            total\_population = sum(population)) %>%

  mutate(density = total\_population / total\_area) %>%

  arrange(desc(density))

-top\_n : mengambil nilai tertinggi, tergantung dr jumlah yg kita inginkan

**\*counties\_selected %>%**

**\*group\_by(state) %>%**

**\*top\_n(3, unemployement)** -> mengambil 3 unemployment tertinggi di dalam state

-top\_walk in every region

# Group by region and find the greatest number of citizens who walk to work

counties\_selected %>%

  group\_by(region)%>%

  top\_n(1, walk)

\*highest income in every region and state

counties\_selected %>%

  group\_by(region, state) %>%

  # Calculate average income

  summarize(average\_income=mean(income))%>%

  # Find the highest income state in each region

  top\_n(1, average\_income)

-put theme all together : How many metro and non metro in each state

# Count the states with more people in Metro or Nonmetro areas

counties\_selected %>%

  group\_by(state, metro) %>%

  summarize(total\_pop = sum(population)) %>%

  top\_n(1, total\_pop) %>%

  ungroup() %>%

  count(metro)

-Selecting and Transforming Data

-used contains :

**\*counties %>%**

**\*select (state, county, contains(“work”)) -> Select all words with work**

-start\_with :

**\*counties %>%**

**\*select (state, county, starts\_with(“income”))-> select column with word start with income**

**-other helpers beside contains and start\_with : ends\_with(), last\_col()**

-Removing Variable :

**\*counties %>%**

**\*select (-census\_id)**

-Selecting Columns :

# Glimpse the counties table

glimpse(counties)

counties %>%

  # Select state, county, population, and industry-related columns

  select(state, county, population, professional:production)%>%

  # Arrange service in descending order

  arrange(desc(service))

-select helpers :

counties %>%

  # Select the state, county, population, and those ending with "work"

  select(state, county, population, ends\_with("work")) %>%

  # Filter for counties that have at least 50% of people engaged in public work

  filter(public\_work >= 50)

-Rename a Column :

**\*counties\_selected %>%**

**\*rename(new\_column = old\_column)**

-combine select and rename :

**\*counties\_selected %>%**

**\*select (state, county, population, new\_name = old\_name)**

-Renaming a column after count :

# Rename the n column to num\_counties

counties %>%

  count(state)%>%

  rename(num\_counties=n)

-Renaming a column as part of select :

# Select state, county, and poverty as poverty\_rate

counties %>%

  select(state, county,poverty\_rate=poverty)

-Transmute : combination of select and mutate

**\*counties %>%**

**\*transmute(state, county, fraction\_men = men/population)**

-Using transmute :

counties %>%

  # Keep the state, county, and populations columns, and add a density column

  transmute(state, county, population, density=population/land\_area)%>%

  # Filter for counties with a population greater than one million

  filter(population>1000000)%>%

  # Sort density in ascending order

  arrange(density)

-Choosing among the four verbs :

# Change the name of the unemployment column

counties %>%

  rename(unemployment\_rate = unemployment)

# Keep the state and county columns, and the columns containing poverty

counties %>%

select(state, county, contains("poverty"))

# Calculate the fraction\_women column without dropping the other columns

counties %>%

mutate(fraction\_women = women / population)

# Keep only the state, county, and employment\_rate columns

counties %>%

transmute(state, county, employment\_rate = employed / population)

-Case Babynames Dataset

-Filter for multiple name :

**\*babynames\_multiple <- babynames %>%**

**\*filter(name %in% c(“Amy”, “Christopher”)**

-Filtering and Arranging for one year :

babynames %>%

  # Filter for the year 1990

  filter(year==1990)%>%

  # Sort the number column in descending order

  arrange(desc(number))

-Top names in each year :

# Find the most common name in each year

babynames %>%

  group\_by(year)%>%

  top\_n(1,number)

-Visualizing names with ggplot2 :

  # Filter for the names Steven, Thomas, and Matthew

  selected\_names <- babynames %>%

    filter(name %in% c("Steven", "Thomas", "Matthew"))

  # Plot the names using a different color for each name

  ggplot(selected\_names, aes(x = year, y = number, color = name)) +

    geom\_line()

-Group Mutate and add the fraction column :

**\*babynames %>%**

**\*group\_by(year) %>%**

**\*mutate(year\_total = sum(number)) >%>**

**\*ungroup() %>%**

**\*mutate(fraction = number/year\_total)**

-Finding the year each name that most common

# Calculate the fraction of people born each year with the same name

babynames %>%

  group\_by(year) %>%

  mutate(year\_total = sum(number)) %>%

  ungroup() %>%

  mutate(fraction = number / year\_total) %>%

# Find the year each name is most common

  group\_by(name) %>%

  top\_n(1, fraction)

-Adding total and maximum to each name

babynames %>%

  group\_by(name) %>%

  mutate(name\_total = sum(number),

         name\_max = max(number)) %>%

  # Ungroup the table

  ungroup() %>%

  # Add the fraction\_max column containing the number by the name maximum

  mutate(fraction\_max = number / name\_max)

-Visualizing above example

# Filter for the names Steven, Thomas, and Matthew

names\_filtered <- names\_normalized %>%

  filter(name %in% c("Steven", "Thomas", "Matthew"))

# Visualize these names over time

ggplot(names\_filtered, aes(x=year, y=fraction\_max, color=name))+geom\_line()

-Window Function : Menggeser vector ke sebelah kanan. It useful if we want to compare with previous vector

Contoh : v(1,3,6,14)

**\*lag(v) : NA,1,3,6**

-To see percentage of growth :

**\*babynames\_fraction %>%**

**\*filter(name==”Mathew”) %>%**

**\*arrange(year)%>%**

**\*mutate(difference = fraction – lag(fraction))**

**\*arrange(desc(difference)) -> biggest jump**

-Big changes in name and year :

**\*babynames\_fraction %>%**

**\*arrange(name, year)%>%**

**\*mutate(difference = fraction – lag(fraction))%>%**

**\*group\_by(name) %>%**

**\*arrange(desc(difference))**

-Using ratio to describe the frequency of a name

babynames\_fraction %>%

  # Arrange the data in order of name, then year

  arrange(name, year) %>%

  # Group the data by name

  group\_by(name) %>%

  # Add a ratio column that contains the ratio between each year

  mutate(ratio = fraction / lag(fraction))

-Biggest jump in a name(in ratio) :

babynames\_ratios\_filtered %>%

  # Extract the largest ratio from each name

  top\_n(1,ratio)%>%

  # Sort the ratio column in descending order

  arrange(desc(ratio))%>%

  # Filter for fractions greater than or equal to 0.001

  filter(fraction>0.001)