Data Manipulation with dplyr

Micro Specialization

Transforming Data with dplyr

- **select() :** Picks variables based on their names.
- **filter() :** Picks cases based on their values.
- **arrange() :** Changes the ordering of the rows.
- **mutate() :** Adds new variables that are functions of existing variables
- **summarise() :** Reduces multiple values down to a single summary.
- **join()** : Join multiple data sets together into a single data frame

Data Set:

- install.packages(“tidyverse”)
- library(tidyverse)

- data("counties")

- view()
- str()
- glimpse()
- summary()
- head()
- tail()

Select()

Select Columns

- Select a part of table

- Create a new table

- Dataset %>% select(coln1,coln2,...)
  - counties %>% select(state,county,population,unemployment)

- counties_selected<- counties %>% select(state,county,population,unemployment)

- counties_selected

- *counties_selected <- counties[, c (“state", “county“, “population”, “unemployment”)]*

Selecting columns

Select the following four columns from the counties variable:

- state
- county
- population
- Poverty

You don't need to save the result to a variable.

```
counties %>%select(state,county,population,poverty)
```

Arrange

Arranging Observations

- Arrange (ascending)

- Arrange (descending)

- counties_selected %>%
  arrange(population)

- counties_selected %>%
  arrange(desc(population))

Filter

Filtering for Conditions

- counties_selected %>% filter(state=='New York')
- *counties_select <- counties_selected[counties_selected$state == 'New York',]*
- counties_selected %>% filter(unemployment < 6)

Filtering and Arranging

- counties_selected %>% arrange(desc(population)) %>% filter(state=='New York')
- counties_selected %>% arrange(desc(population)) %>% filter(unemployment < 6)
- counties_selected %>% arrange(desc(population)) %>% filter(state=='New York‘, unemployment < 6)

Filter

- Find only the counties that have a population above one million (1000000)

- Find only the counties in the state of California that also have a population above one million (1000000)

- counties_population <- counties %>% select(state, county, population)
- counties_population %>%filter(state=='California', population>1000000)

Arranging observations

- Here you see the counties_work dataset with a few interesting variables selected. These variables: private_work, public_work, self_employed.
- Sort these observations by either private_work, public_work, self_employed to find the most interesting cases.
- Sort by multiple variables
- Sort by mixed ascending and descending
- counties_work <- counties %>% select(private_work, public_work, self_employed)
- # Add a verb to sort in descending order of public_work
- counties_work %>% arrange(desc(public_work))

Filtering and arranging

- Filter for counties in the state of Texas that have more than ten thousand people (10000), and sort them in descending order of the percentage of people employed in private work

- counties_popwork <- counties %>% select(state, county, population, private_work, public_work, self_employed)
- counties_popwork %>%filter(state=='Texas', population>10000)%>%arrange(desc(private_work))

Mutate

- Data SubSet

- Unemployment as Nos (not as %age)
  - Population*unemployment/100

- counties_unpop<- counties %>% select(state,county,population,unemployment)

- counties_unpop%>% mutate(unemployed_population=population*unemployment/100)

Calculating the number of people

- Which county has highest number of unemployed people
- Sort in descending order of the public_workers column
- counties_unpop%>% mutate(unemployed_population=population*unemployment/100) %>% arrange(desc(unemployed_population))
- 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 the percentage of women in a county

- The dataset includes columns for the total number (not percentage) of men and women in each county.

- You could use this, along with the population variable, to compute the fraction of men (or women) within each county.

- # Select the columns state, county, population, men, and women
- counties_gender <- counties %>%select(state, county, population, men, women)
- # Calculate proportion_women as the fraction of the population made up of women
- counties_gender %>%select(state, county, population, men, women)%>%mutate(proportion_women=women/population)

Count

- counties%>% count()
- counties%>% count(state)
- counties%>% count(state, sort=TRUE)
- counties%>%count(state, wt=population, sort=TRUE)

Counting by region

- The counties dataset contains columns for region, state, population, and the number of citizens, which we selected and saved as the counties_citizen

- Count by regions and sort

- counties_citizen <- counties %>% select(region, state, population, citizens)
- # Use count to find the number of counties in each region
- counties_citizen %>%count(region, sort=TRUE)

Counting citizens by state

- Count the number of counties in each state, weighted based on the citizens column, and sorted in descending order.

- counties_citizen <- counties %>% select(region, state, population, citizens)
- # Find number of counties per state, weighted by citizens
- counties_citizen %>% count(state, wt=citizens, sort=TRUE)

Summarize

- Summarise Total Population
- counties%>% summarise(total_population = sum(population))

- Summarise Total Population and Average Unemployment

- counties%>% summarise(total_population = sum(population), average_unemployment = mean(unemployment))

- Summary Functions are:
  - sum
  - mean
  - median
  - min
  - max
  - n

Summarise: Aggregate within Group

- Count group by state of population and average unemployment

- Counties %>% **group_by(state)** %>% summarise(total_population = sum(population), average_unemployment = mean(unemployment))

- Count group by state and metro of population and average unemployment

- counties %>% group_by(state, metro) %>% summarise(total_population = sum(population), average_unemployment = mean(unemployment))

Summarise and Arrange

- Count group by state of population and average unemployment, arrange by average unemployment in descending order

- counties %>% group_by(state) %>% summarise(total_population = sum(population), average_unemployment = mean(unemployment)) %>% arrange(desc(average_unemployment))

Summarise

- Summarize the counties dataset to find the following columns:
  - min_population (with the smallest population),
  - max_unemployment (with the maximum unemployment), and
  - average_income (with the mean of the income variable).

- counties_selected <- counties %>% select(county, population, income, unemployment)
- # Summarize to find minimum population, maximum unexployment, and average income
- counties_selected <- counties %>% select(county, population, income, unemployment) %>% summarise(min_population = min(population),max_unemployment = max(unemployment),average_income = mean(income))
- counties_selected

top_n

- Filter n number of rows to return for **top_n**()
- counties_top <- %>% select (state, county, population, unemployment, income)
- counties_top %>% group_by(stata) %>% top_n(1,population)
- Check with (2,population)

?select_helpers

These functions allow you to select variables based on their names.

- **starts_with() :** Starts with a prefix.
- **ends_with()** : Ends with a suffix.
- **contains()** : Contains a literal string.
- **matches()** : Matches a regular expression.
- **num_range()** : Matches a numerical range like x01, x02, x03.
- **one_of()** : Matches variable names in a character vector.
- **everything()** : Matches all variables.
- **last_col()** : Select last variable, possibly with an offset.

Removing a Variable from Table

- counties %>% select(-census_id)

Rename Verb

- counties_rename <- counties %>% select (state, county, population, unemployment)
- counties_rename
- counties_rename %>% rename(unemployment_rate = unemployment)
- counties_rename <- counties %>% select (state, county, population, rename(unemployment_rate = unemployment)

Transmute:Combination of Select & Mutate

- Transmute verb allows you to control which variables you keep, which variables you calculate, and which variables you drop.

- Returns a subset of columns that are transformed and changed at same time

- counties %>% transmute(state, county, fraction_men = men / population)
- counties %>% transmute(state, county, population, unemployment_people = population * unemployment /100)

Thanks