Summary Sheet

AUTHOR
Julia Gallucci

Class 3

Data manipulation:

- `glimpse()` to view your data as columns down the page, rows across; easier to view entire columns in data set

- `filter()` to subset data set and retain only those columns that meet a certain condition

  - i.e., `filter(data_set, certain_column < certain value)`

- `arrange()` to order rows in a data set by a specific column, default as ascending order, use `-` for descending

```
-   i.e., `arrange(data_set, -certain_column)` to order descending
-   i.e., `arrange(data_set, certain_column)` to order ascending
```

- `select()` to extract only a specified column from data set, can use `-` ro remove a certain column from data set

  - i.e., `select(data_set, certain_column)` to pick only that column
  - i.e., `arrange(data_set, -certain_column)` to pick all columns BUT that one

- `mutate()` to create a new column from existing columns or modify an existing column

  - i.e., `mutate(data_set,new_column_name = existing_column_name + some value)`

Pipe operator

`%>%` is used to combine multiple operations at once

i.e., `data_set %>%`

`filter(``certain_column < certain value``) %>%`

`arrange(certain_column) %>%`

`select(certain_column)`

- filter the data set to only view rows with a specific column under a specific value

- arrange the data set to view rows of a certain column in ascending

- select only specified columns to extract from data set

Data summary:

- `summary()` provides an overview of data; i.e., central tendencies, dispersion and distribution
- `pull()` to extract a specific column from a data frame as a numeric vector
  - can combine `pull` with a mathematical operation

i.e., `data_set %>%`

`pull(certain_column) %>%`

`mean() #can also be things like median, variance, sd etc.`

Note: use na.rm = TRUE when column observations contain NA

- `summarise()` to create a new data table summarizing observations
  - i.e., `data_set %>%`

    `summarise(name_column = mean(certain_column),`

    `name_column2 = sd(certain_column))`
- can also `group_by()` prior to summarizing to get a summary table based on categories
  - i.e., `data_set %>%`

    `group_by(certain_categorical_column) %>% summarise(n_column = n(),`

    `name_column = mean(certain_column),`

    `name_column = sd(certain_column))`

Factors and grouping:

We can mutate categorical columns to factors before grouping our data, to ensure that the levels of the factors are in the desired order. This allows for more intuitive and meaningful presentation of our data during summarization.

```r
library(tidyverse)
CES_data %>%
  filter(cps19_province == "Ontario") %>%
  select(cps19_prov_gov_sat,
         cps19_prov_id,
         cps19_income_number) %>%
  mutate(cps19_prov_id = factor(cps19_prov_id,
         levels = c("Liberal", "Progressive Conservative","NDP","Green","Another party","
  group_by(cps19_prov_id) %>%
  summarise(median_income = median(cps19_income_number, na.rm = TRUE),
            count = n())
```

Here, we are mutating our provincial identification category from the CES data into a factor and providing specified levels that make logical sense.

Note: you can group data by more than one category (in the in-class example, we grouped by both provincial identification and provincial government satisfaction). Use `spread()` or `pivot_wider()` for improved readibility of summary table (see below)

```r
CES_data %>%
  filter(cps19_province == "Ontario") %>%
  select(cps19_prov_gov_sat,
         cps19_prov_id,
         cps19_income_number) %>%
  mutate(cps19_prov_id = factor(cps19_prov_id, levels = c(
    "Liberal",
    "Progressive Conservative",
    "NDP",
    "Green",
    "Another party",
    "None",
    "Don't know/prefer not to answer"
  ))) %>%
  mutate(cps19_prov_gov_sat = factor(cps19_prov_gov_sat, levels = c(
    "Not at all satisfies",
    "Not very satisfied",
    "Fairly satisfied",
    "Very satisfied",
    "Don't know/prefer not to answer"
  ))) %>%
  group_by(cps19_prov_gov_sat, cps19_prov_id) %>%
  summarise(median_income = median(cps19_income_number, na.rm = TRUE)) %>%
  spread(key = cps19_prov_gov_sat,
         value = median_income)
```