STAT 100 - Important Functions for Data Wrangling

Logical Operators in R

- & - "And"
- | - "Or"
- == - "Equal to"
- != - "Not equal to"
- %in% - "In"

Wrangling Verbs

- %>% - Takes **dataset** to the left and "pipes" it as the first argument in the next line (since the first argument of most wrangling verbs is a dataset)
    - *colleges %>%*
      rename(SAT = sat_avg_2013)
- <- - Defines a new **dataset** (or **model**) to the left using information from the right
    - *colleges_updated <- colleges*
- filter() - Gets specific **rows/observations**
    - *thanksgiving <- flights %>%*
      filter(month == 11)
- select() - Gets specific **variables/columns**
    - *thanksgiving <- flights %>%*
      select(year)
- mutate() - Modifies existing **variables** and/or create new ones
    - **When defining a new variable, for it to stay in the dataset, you must set the dataset equal to itself at the beginning (as shown below)**
        - *hpi <- hpi %>%*
          mutate(LogFootprint = log(Footprint))
- case_when() - Uses logical conditions to mutate() a **variable**, setting it to the value to the RIGHT of the tilda if the statement to the LEFT is true
    - *mutate(seniority_new = case_when(*
      seniority <= 2 ~ "junior",
      seniority == 3 ~ "mid",
      seniority >= 4 ~ "senior"))
    - *hpi <- hpi %>%*
      mutate(Classification = case_when(
      LifeExpectancy >= 72.27 ~ "Above or Equal to Average",
      LifeExpectancy < 72.27 ~ "Below to Average"))
- rename() - Renames **variable**, where new name is the the LEFT of equal sign

    - *rename(INCOME = FINCBTAX)*
- summarize() - Allows for use of summary functions, such as mean(), sd(), cor(), IQR(), and n(), which can be set equal to new **variables**
    - *summarize(mean_INCOME = mean(INCOME),*
        mean_IRAX = mean(IRAX),
        households = n())
- n() - Provides a count, which can be useful after a group_by() with a certain **variable**
    - *hpi %>%*
        group_by(Classification) %>%
        summarize(count = n())
- group_by() - Groups data by **variable(s)**, allowing for **contingency/proportions**
    - *mythbusters %>%*
            group_by(group, yawned) %>%
            summarize(count = n()) %>%
            mutate(prop = count / sum(count))
    - *SaratogaHouses %>%*
        count(waterfront) %>%
        mutate(prop = n/sum(n))
- arrange() - Sorts the data based on values of a certain **variable** (when paired with desc(), arranges the data in descending order of the **variable**)
    - *glassdoor %>%*
        group_by(education, gender) %>%
        summarize(median_pay = median(pay)) %>%
        arrange(desc(median_pay))
- na.omit() - Removes all **rows/observations** with a single missing value for any variable (most aggressive way to deal with missing values)
    - *colleges_aggressive_removal <- colleges %>%*
        na.omit()
- drop_na() - Removes **rows/observations** with missing values for specific variable(s) (moderately aggressive way to deal with missing values)
    - *colleges_moderate_removal <- colleges %>%*
        drop_na(sticker_price_2013)
- na.rm = TRUE - Only temporarily ignores N/A as needed before calculating, without removing any **rows/observations** (least aggressive way to deal with missing values)
    - *colleges_light_removal <- colleges %>%*
        mutate(mean_sticker_price_2013 = mean(sticker_price_2013, na.rm = TRUE))
- factor() - Converts **seemingly-numerical variable** to a **categorical variable**
    - *ggplot(Pollster08, aes(x = Days, y = Margin, color = factor(Charlie)))*

- c() - Concatenates 2 or more values into 1, which is necessary when a function only accepts 1 input
    - *bootstrap_dist <- movies %>%*
      filter(Genre %in% c("Drama", "Action"))
- slice_max() - Filters for n **rows/observations** with the highest values for a certain variable (this may result in more **rows/observations** than specified in the case of ties)
    - *glassdoor %>%*
      drop_na(pay) %>%
      filter(gender == "Female", jobtitle == "Software Engineer") %>%
      slice_max(pay, n = 10) %>%
      select(pay, education)
- fct_relevel() - Manually reorders the factor levels of a **categorical variable**
    - *glassdoor %>%*
      drop_na(pay) %>%
      filter(gender == "Female", jobtitle == "Financial Analyst") %>%
      mutate(pay = pay/1000, education = factor(education)) %>%
      mutate(education = fct_relevel(education, "High School", "College")) %>%

    ggplot(aes(y = pay, x = education)) +
      geom_boxplot()