Social Sciences Intro to Statistics

Week 3.1 Descriptive Statistics

Week 3: Learning goal - Identify descriptive statistics in order to describe individual variables.

Introduction

Load packages:

```
library(tidyverse)
library(ggplot2)
best_netflix <- read_csv("https://raw.githubusercontent.com/bcl96/Social-Sciences-Stats/main/
```

How do we understand data?

Before we can interpret and run statistical tests, we need to first understand our data. One way to understand our data is using descriptive statistics to describe data. We can use summary functions to find out about the mean, median, percentiles, and standard deviation of our data.

Mean, Median, and Percentiles

Mean is the average of the dataset, the sum of all the values divided by the number of values. While the mean represents the typical value of a dataset, it is also sensitive to outliers. Outliers can drastically change the value of our mean.

```
mean(best_netflix$SCORE)
#> [1] 8.093496
```

Median on the other hand, is less sensitive to outliers. The median is the value where 50% of the dataset would be greater and 50% of the dataset would be lesser than it. To find the median, you arrange the dataset in ascending order (smallest to largest) and then find the value that is in the middle. If the dataset has an odd number of values, our median is the middle value. If the dataset has an even number of values, our median is the average of the two median values.

```
median(best_netflix$SCORE)
#> [1] 8
```

Percentiles help us visualize our dataset as 100 equal parts. The 25th percentile, also known as a the first quartile, is the value where 25% of the data would fall below. The 75th percentile, or third quartile, is the value where 75% of the data would fall below The median is the 50th percentile, it equally divides our dataset and 50% of the data would fall below. Similar to the median, because percentiles provide information on the spread of our dataset, it is less affected by outliers than compared to the mean.

```
# Find the 25th percentile (first quartile)
q1 <- quantile(best_netflix$SCORE, probs = 0.25)

# Find the median (50th percentile)
median <- quantile(best_netflix$SCORE, probs = 0.5)

# Find the 75th percentile (third quartile)
q3 <- quantile(best_netflix$SCORE, probs = 0.75)

# Print the results
print(q1)
#> 25%
#> 7.7
print(median)
#> 50%
#>   8
print(q3)
#> 75%
#> 8.4
```

Introduce IPEDS data

What is IPEDS?

- IPEDS is the Integrated Postsecondary Education Data System. It is a system of interrelated surveys conducted annually by the U.S. Department of Education's National Center for Education Statistics (NCES). IPEDS gathers information from every college, university, and technical and vocational institution that participates in the federal student financial aid programs. The Higher Education Act of 1965, as amended, requires that institutions that participate in federal student aid programs report data on enrollments, program completions, graduation rates, faculty and staff, finances, institutional prices, and student financial aid. These data are made available to students and parents through the College Navigator college search Web site and to researchers and others through the IPEDS Data Center.

Which IPEDS variables will we be using to teach statistical inference

- Variables about annual cost of attendance for full-time graduate programs
  - average tuition (in-state vs. out-of-state)
  - required fees (in-state vs. out-of-state)
  - books and supplies; off-campus room and board; off-campus "other expenses"
    - these are calculated for undergraduate students, but should be highly correlated with graduate students
  - **Note**: none of these variables include the cost of healthcare
    - Therefore, assuming you must pay for the cost of healthcare, the measures of cost of attendance (COA) we create will understate the total cost of graduate school + life

Why use IPEDS data rather than *College Scorecard* data? and why these variables?

- *College Scorecard* brings together variables from many data sources, including IPEDS
  - The most interesting variables in *College Scorecard* are debt and earnings from degree programs, which come from Office of Federal Student Aid and IRS
- These university-level debt and earnings variables are measured as the mean (or median) of all students who graduated from the university
  - So the underlying data (that we don't see) are student-level; these student-level data are aggregated to the university-level to create measures mean (or median) debt/earnings at that university
  - Using these measures would make teaching statistical inference more confusing (e.g., trying to test hypotheses about the population mean of mean earnings)
- Why teach inferential statistics using IPEDS measures about graduate tuition (and cost of attendance)?
  - These measures are truly measured at the university-level

  - Tuition and cost-of-attendance are the big drivers of student debt, so learning more about how tuition/cost-of-attendance varies across universities will help you make more informed decisions about pursuing graduate education in the future

Some definitions

Some definitions related to tuition, fees, expenses, etc; from the IPEDS "Student Charges for Full Academic Year" 2022-2023 academic year data dictionary [LINK]:

- "Full-time student" (graduate)
  - Graduate - A student enrolled for 9 or more semester credits, or 9 or more quarter credits, or students involved in thesis or dissertation preparation that are considered full time by the institution
- "Academic year"
  - The period of time generally extending from September to June; usually equated to 2 semesters or trimesters, 3 quarters, or the period covered by a 4-1-4 plan
- "Tuition"
  - Amount of money charged to students for instructional services. Tuition may be charged per term, per course, or per credit
- "In-state tuition"
  - The tuition charged by institutions to those students who meet the state's or institution's residency requirements
- "Out-of-state tuition"
  - The tuition charged by institutions to those students who do not meet the state's or institution's residency requirements
- "Required fees"
  - Fixed sum charged to students for items not covered by tuition and required of such a large proportion of all students that the student who does NOT pay the charge is an exception
- "In-state fees"
  - The fees charged by institutions to those students who meet the state's or institution's residency requirements.
- "Out-of-state fees"
  - The fees charged by institutions to those students who do not meet the state's or institution's residency requirements

- "Books and supplies" (undergraduate)
  - Do not include unusual costs for special groups of students (e.g., engineering or art majors), unless they constitute the majority of students at your institution
- "Room charges"
  - The charges for an academic year for rooming accommodations for a typical student sharing a room with one other student.
- "Board charges"
  - The charge for an academic year for meals, for a specified number of meals per week.
- "Other expenses"
  - The amount of money (estimated by the financial aid office) needed by a student to cover expenses such as laundry, transportation, entertainment, and furnishings. (For the purpose of this survey room and board and tuition and fees are not included.)

Note: the IPEDS measures of full-time graduate tuition (both in-state and out-of-state) are "average" tuition price across different graduate degree programs (excluding "first-professional" degree programs like law and medicine)

- e.g., full-time annual tuition price for an MBA program is likely higher than that of an MA in education
- But in the context of using these measures to teach statistical inference, pretend these prices are not "averages" but rather the official tuition price
- make the same assumption for variables about fees, books/supplies, living expenses, etc.

Note: "required fees" do not include the cost of healthcare (I think)

Now let's load the IPEDS data and create some data frames from it.

```
# Libraries
  #install.packages('tidyverse') # if you haven't installed already
  #install.packages('labelled') # if you haven't installed already
  #install.packages('patchwork') # if you haven't installed already

library(tidyverse) # load tidyverse package
library(labelled) # load labelled package package
library(patchwork)


##########
```

```r
########## IPEDS
##########

# Load ipeds dataset from course website
load(url('https://raw.githubusercontent.com/bcl96/Social-Sciences-Stats/main/data/ipeds/outpu

# Create ipeds data frame with fewer variables/observations
df_ipeds_pop <- panel_data %>%
  # keep data from fall 2022
  filter(year == 2022) %>%
  # which universities to keep:
    # 2015 carnegie classification: keep research universities (15,16,17) and master's univer
  filter(c15basic %in% c(15,16,17,18,19,20)) %>%
  # which variables to keep
  select(instnm,unitid,opeid6,opeid,control,c15basic,stabbr,city,zip,locale,obereg, # basic i
         tuition6,fee6,tuition7,fee7, # avg tuition and fees for full-time grad, in-state and
         isprof3,ispfee3,osprof3,ospfee3, # avg tuition and fees for MD, in-state and out-of-
         isprof9,ispfee9,osprof9,ospfee9, # avg tuition and fees for Law, in-state and out-of
         chg4ay3,chg7ay3,chg8ay3) %>% # [undergraduate] books+supplies; off-campus (not with
  # rename variables; syntax <new_name> = <old_name>
  rename(region = obereg, # revion
         tuit_grad_res = tuition6, fee_grad_res = fee6, tuit_grad_nres = tuition7, fee_grad_n
         tuit_md_res = isprof3, fee_md_res = ispfee3, tuit_md_nres = osprof3, fee_md_nres = o
         tuit_law_res = isprof9, fee_law_res = ispfee9, tuit_law_nres = osprof9, fee_law_nres
         books_supplies = chg4ay3, roomboard_off = chg7ay3, oth_expense_off = chg8ay3) %>% #
  # create measures of tuition+fees
  mutate(
    tuitfee_grad_res = tuit_grad_res + fee_grad_res, # graduate, state resident
    tuitfee_grad_nres = tuit_grad_nres + fee_grad_nres, # graduate, non-resident
    tuitfee_md_res = tuit_md_res + fee_md_res, # MD, state resident
    tuitfee_md_nres = tuit_md_nres + fee_md_nres, # MD, non-resident
    tuitfee_law_res = tuit_law_res + fee_law_res, # Law, state resident
    tuitfee_law_nres = tuit_law_nres + fee_law_nres) %>% # Law, non-resident
  # create measures of cost-of-attendance (COA) as the sum of tuition, fees, book, living exp
  mutate(
    coa_grad_res = tuit_grad_res + fee_grad_res + books_supplies + roomboard_off + oth_expens
    coa_grad_nres = tuit_grad_nres + fee_grad_nres + books_supplies + roomboard_off + oth_exp
    coa_md_res = tuit_md_res + fee_md_res + books_supplies + roomboard_off + oth_expense_off
    coa_md_nres = tuit_md_nres + fee_md_nres + books_supplies + roomboard_off + oth_expense_o
    coa_law_res = tuit_law_res + fee_law_res + books_supplies + roomboard_off + oth_expense_o
    coa_law_nres = tuit_law_nres + fee_law_nres + books_supplies + roomboard_off + oth_expens
  # keep only observations that have non-missing values for the variable coa_grad_res
```

```
    # this does cause us to lose some interesting universities, but doing this will eliminate
  filter(!is.na(coa_grad_res))

# Add variable labels to the tuit+fees variables and coa variables
  # tuition + fees variables
    var_label(df_ipeds_pop[['tuitfee_grad_res']]) <- 'graduate, full-time, resident; avg tuit
    var_label(df_ipeds_pop[['tuitfee_grad_nres']]) <- 'graduate, full-time, non-resident; avg
    var_label(df_ipeds_pop[['tuitfee_md_res']]) <- 'MD, full-time, state resident; avg tuitio
    var_label(df_ipeds_pop[['tuitfee_md_nres']]) <- 'MD, full-time, non-resident; avg tuition
    var_label(df_ipeds_pop[['tuitfee_law_res']]) <- 'Law, full-time, state resident; avg tuit
    var_label(df_ipeds_pop[['tuitfee_law_nres']]) <- 'Law, full-time, non-resident; avg tuiti

  # COA variables
    var_label(df_ipeds_pop[['coa_grad_res']]) <- 'graduate, full-time, state resident COA; ==
    var_label(df_ipeds_pop[['coa_grad_nres']]) <- 'graduate, full-time, non-resident COA; ==
    var_label(df_ipeds_pop[['coa_md_res']]) <- 'MD, full-time, state resident COA; == tuition
    var_label(df_ipeds_pop[['coa_md_nres']]) <- 'MD, full-time, non-resident COA; == tuition
    var_label(df_ipeds_pop[['coa_law_res']]) <- 'Law, full-time, state resident COA; == tuiti
    var_label(df_ipeds_pop[['coa_law_nres']]) <- 'Law, full-time, non-resident COA; == tuitio

df_ipeds_pop %>% glimpse()
#> Rows: 965
#> Columns: 38
#> $ instnm           <chr> "Alabama A & M University", "University of Alabama a~
#> $ unitid           <dbl> 100654, 100663, 100706, 100724, 100751, 100830, 1008~
#> $ opeid6           <chr> "001002", "001052", "001055", "001005", "001051", "0~
#> $ opeid            <chr> "00100200", "00105200", "00105500", "00100500", "001~
#> $ control          <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, ~
#> $ c15basic         <dbl+lbl> 18, 15, 16, 19, 16, 18, 16, 20, 18, 18, 19, 18, ~
#> $ stabbr           <chr+lbl> "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL", ~
#> $ city             <chr> "Normal", "Birmingham", "Huntsville", "Montgomery", ~
#> $ zip              <chr> "35762", "35294-0110", "35899", "36104-0271", "35487~
#> $ locale           <dbl+lbl> 12, 12, 12, 12, 13, 12, 13, 12, 23, 43, 21, 13, ~
#> $ region           <dbl+lbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
#> $ tuit_grad_res    <dbl> 10128, 8424, 10632, 7416, 11100, 7812, 10386, 15325,~
#> $ fee_grad_res     <dbl> 1414, 0, 1054, 2740, 690, 766, 1784, 900, 1000, 190,~
#> $ tuit_grad_nres   <dbl> 20160, 19962, 24430, 14832, 31460, 17550, 31158, 153~
#> $ fee_grad_nres    <dbl> 1414, 0, 1054, 2740, 690, 766, 1784, 900, 1000, 190,~
#> $ tuit_md_res      <dbl> NA, 31198, NA, NA, 31198, NA, NA, NA, NA, NA, NA, NA~
#> $ fee_md_res       <dbl> NA, 3464, NA, NA, 0, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ tuit_md_nres     <dbl> NA, 62714, NA, NA, 62714, NA, NA, NA, NA, NA, NA, NA~
#> $ fee_md_nres      <dbl> NA, 3464, NA, NA, 0, NA, NA, NA, NA, NA, NA, NA, NA,~
```

```
#> $ tuit_law_res      <dbl> NA, NA, NA, NA, 24080, NA, NA, 39000, NA, NA, NA, NA~
#> $ fee_law_res       <dbl> NA, NA, NA, NA, 300, NA, NA, 325, NA, NA, NA, NA, 65~
#> $ tuit_law_nres     <dbl> NA, NA, NA, NA, 44470, NA, NA, 39000, NA, NA, NA, NA~
#> $ fee_law_nres      <dbl> NA, NA, NA, NA, 300, NA, NA, 325, NA, NA, NA, NA, 65~
#> $ books_supplies    <dbl> 1600, 1200, 2416, 1600, 800, 1200, 1200, 1800, 998, ~
#> $ roomboard_off     <dbl> 9520, 14330, 11122, 7320, 14426, 10485, 14998, 8020,~
#> $ oth_expense_off   <dbl> 3090, 6007, 4462, 5130, 4858, 4030, 6028, 4600, 3318~
#> $ tuitfee_grad_res  <dbl> 11542, 8424, 11686, 10156, 11790, 8578, 12170, 16225~
#> $ tuitfee_grad_nres <dbl> 21574, 19962, 25484, 17572, 32150, 18316, 32942, 162~
#> $ tuitfee_md_res    <dbl> NA, 34662, NA, NA, 31198, NA, NA, NA, NA, NA, NA, NA~
#> $ tuitfee_md_nres   <dbl> NA, 66178, NA, NA, 62714, NA, NA, NA, NA, NA, NA, NA~
#> $ tuitfee_law_res   <dbl> NA, NA, NA, NA, 24380, NA, NA, 39325, NA, NA, NA, NA~
#> $ tuitfee_law_nres  <dbl> NA, NA, NA, NA, 44770, NA, NA, 39325, NA, NA, NA, NA~
#> $ coa_grad_res      <dbl> 25752, 29961, 29686, 24206, 31874, 24293, 34396, 306~
#> $ coa_grad_nres     <dbl> 35784, 41499, 43484, 31622, 52234, 34031, 55168, 306~
#> $ coa_md_res        <dbl> NA, 56199, NA, NA, 51282, NA, NA, NA, NA, NA, NA, NA~
#> $ coa_md_nres       <dbl> NA, 87715, NA, NA, 82798, NA, NA, NA, NA, NA, NA, NA~
#> $ coa_law_res       <dbl> NA, NA, NA, NA, 44464, NA, NA, 53745, NA, NA, NA, NA~
#> $ coa_law_nres      <dbl> NA, NA, NA, NA, 64854, NA, NA, 53745, NA, NA, NA, NA~
```

```
##########
########## Create data frame of generated variables, with each variable meant to represent t
##########


num_obs <- 10000

# Generate normal distribution w/ custom mean and sd
set.seed(124)
norm_dist <- rnorm(n = num_obs, mean = 50, sd = 5)

# Generate right-skewed distribution
set.seed(124)
rskew_dist <- rbeta(n = num_obs, shape1 = 2, shape2 = 5)

# Generate left-skewed distribution
set.seed(124)
lskew_dist <- rbeta(n = num_obs, shape1 = 5, shape2 = 2)

# Generate standard normal distribution (default is mean = 0 and sd = 1)
set.seed(124)
```

```r
stdnorm_dist <- rnorm(n = num_obs, mean = 0, sd = 1)  # equivalent to rnorm(10)

# Create dataframe
df_generated_pop <- data.frame(norm_dist, rskew_dist, lskew_dist, stdnorm_dist)

# drop individual objects associated with each variable
rm(norm_dist,rskew_dist,lskew_dist,stdnorm_dist)
rm(num_obs)


##########
########## Create sample versions of generated population data frame and IPEDS population da
##########

# create sample version of our generated data
  set.seed(124) # set seed so that everyone ends up with the same random sample

  df_generated_sample <- df_generated_pop %>% sample_n(size = 200)
  df_generated_sample %>% glimpse()
#> Rows: 200
#> Columns: 4
#> $ norm_dist    <dbl> 42.70513, 50.24400, 61.29008, 45.47494, 44.74406, 47.9912~
#> $ rskew_dist   <dbl> 0.34451771, 0.31359906, 0.09375337, 0.05581678, 0.0744584~
#> $ lskew_dist   <dbl> 0.6554823, 0.6864009, 0.9062466, 0.9441832, 0.9255415, 0.~
#> $ stdnorm_dist <dbl> -1.45897348, 0.04880097, 2.25801577, -0.90501164, -1.0511~


# create sample version of our ipeds data

  set.seed(124) # set seed so that everyone ends up with the same random sample

  df_ipeds_sample <- df_ipeds_pop %>% sample_n(size = 200)

  # compare mean of coa_grad_res between population and sample
  mean(df_ipeds_pop$coa_grad_res, na.rm = TRUE)
#> [1] 32528.35
  mean(df_ipeds_sample$coa_grad_res, na.rm = TRUE)
#> [1] 31620.8


##########
# Create function to generate plots of variable distributions
```

```
##########

plot_distribution <- function(data_vec, plot_title = '') {
  p <- ggplot(as.data.frame(data_vec), aes(x = data_vec)) +
    ggtitle(plot_title) + xlab('') + ylab('') +
    geom_histogram(aes(y = ..density..), alpha = 0.4, position = 'identity') +
    geom_density() +
    geom_vline(aes(xintercept = mean(data_vec, na.rm = T), color = 'mean'),
               linetype = 'dotted', size = 0.8, alpha = 0.8) +
    geom_vline(aes(xintercept = median(data_vec, na.rm = T), color = 'median'),
               linetype = 'dotted', size = 0.8, alpha = 0.8) +
    scale_color_manual(name = 'Statistics',
                       labels = c(paste('Mean:', round(mean(data_vec, na.rm = T), 2),
                                        '\nStd Dev:', round(sd(data_vec, na.rm = T), 2)),
                                  paste('Median:', round(median(data_vec, na.rm = T), 2))),
                       values = c(mean = 'blue', median = 'red')) +
    theme(plot.title = element_text(size = 10, face = 'bold', hjust = 0.5),
          legend.title = element_text(size = 9, face = 'bold'),
          legend.text = element_text(size = 8))

  p
}


##########
# Write function to get the sampling distribution from a variable (defaults equal 500 samples
##########

get_sampling_distribution <- function(data_vec, num_samples = 1000, sample_size = 200) {
  sample_means <- vector(mode = 'numeric', num_samples)

  for (i in 1:length(sample_means)) {
    samp <- sample(data_vec, sample_size)
    sample_means[[i]] <- mean(samp, na.rm = T)
  }

  sample_means
}

##########
# Write Function to generate sampling distribution (with t-test value) assuming null hypothes
##########
```

```r
# Function to generate t-distribution plot
plot_t_distribution <- function(data_vec, mu, alpha = 0.05, alternative = 'two.sided', plot_t

  data_vec <- na.omit(data_vec)

  # Calculate t-statistics
  sample_size <- length(data_vec)
  deg_freedom <- sample_size - 1
  xbar <- mean(data_vec)
  s <- sd(data_vec)

  std_err <- s / sqrt(sample_size)
  t <- (xbar - mu) / std_err

  # Calculate critical value and p-value
  if (alternative == 'less') {  # left-tailed
    cv_lower <- qt(p = alpha, df = deg_freedom, lower.tail = T)
    cv_legend <- round(cv_lower, 2)
    cv_legend2 <- round(cv_lower * std_err + mu, 2)
    pval <- round(pt(q = t, df = deg_freedom, lower.tail = T), 4)
  } else if (alternative == 'greater') {  # right-tailed
    cv_upper <- qt(p = alpha, df = deg_freedom, lower.tail = F)
    cv_legend <- round(cv_upper, 2)
    cv_legend2 <- round(cv_upper * std_err + mu, 2)
    pval <- round(pt(q = t, df = deg_freedom, lower.tail = F), 4)
  } else {  # two-tailed
    cv_lower <- qt(p = alpha / 2, df = deg_freedom, lower.tail = T)
    cv_upper <- qt(p = alpha / 2, df = deg_freedom, lower.tail = F)
    cv_legend <- str_c('\u00B1', round(cv_upper, 2))
    cv_legend2 <- str_c(round(cv_lower * std_err + mu, 2), ' & ', round(cv_upper * std_err +
    pval_half <- round(pt(q = t, df = deg_freedom, lower.tail = t < 0), 4)
    pval <- str_c(pval_half, ' + ', pval_half, ' = ', 2 * pval_half)
  }

  # Plot t-distribution
  p <- ggplot(data.frame(x = -c(-4, 4)), aes(x)) +
    ggtitle(plot_title) + xlab('') + ylab('') +
    stat_function(fun = dt, args = list(df = deg_freedom), xlim = c(-4, 4))

  # Shade rejection region using critical value
  if (alternative != 'greater') {
```

```
  p <- p + geom_vline(aes(xintercept = cv_lower, color = 'cval'),
                      linetype = 'dotted', size = 0.8, alpha = 0.8)

  if (shade_rejection) {
    p <- p + stat_function(fun = dt, args = list(df = deg_freedom),
                           xlim = c(-4, cv_lower),
                           geom = 'area', alpha = 0.3, fill = 'red')
  }

  if (shade_pval) {
    p <- p + stat_function(fun = dt, args = list(df = deg_freedom),
                           xlim = c(-4, if_else(alternative == 'two.sided', -abs(t), t)),
                           geom = 'area', alpha = 0.3, fill = 'blue')
  }
}
if (alternative != 'less') {
  p <- p + geom_vline(aes(xintercept = cv_upper, color = 'cval'),
                      linetype = 'dotted', size = 0.8, alpha = 0.8)

  if (shade_rejection) {
    p <- p + stat_function(fun = dt, args = list(df = deg_freedom),
                           xlim = c(cv_upper, 4),
                           geom = 'area', alpha = 0.3, fill = 'red')
  }

  if (shade_pval) {
    p <- p + stat_function(fun = dt, args = list(df = deg_freedom),
                           xlim = c(if_else(alternative == 'two.sided', abs(t), t), 4),
                           geom = 'area', alpha = 0.3, fill = 'blue')
  }
}

# Legend text
legend_text <- c('t-statistics / p-value', 'critical value / alpha')

if (stacked) {
  legend_text <- c(str_c('t-statistics: ', round(t, 2),
                   '\n(p-value: ', str_extract(pval, '[\\d.-]+$'), ')'),
                   str_c('Critical value: ', cv_legend,
                   '\n(alpha: ', round(alpha, 2), ')'))
}
```

```
  stats_text <- c(str_c('t-statistics: ', round(t, 2)),
                  str_c('SE: ', round(std_err, 2)),
                  str_c('p-value: ', pval),
                  str_c('Critical value: ', cv_legend),
                  str_c('alpha: ', round(alpha, 2)))

  if (!stacked) {
    p <- p +
      annotate('text', size = 9*5/14, x = 4.84, y = 0.14, hjust = 0,
               label = 'bold(Statistics)', parse = T) +
      annotate('text', size = 8*5/14, x = 4.89, y = 0:4 * -0.015 + 0.12, hjust = 0,
               label = stats_text)
  }

  # Label plot
  p <- p +
    geom_vline(aes(xintercept = t, color = 'tstat'),
               linetype = 'dotted', size = 0.8, alpha = 0.8) +
    scale_x_continuous(sec.axis = sec_axis(trans = ~ . * std_err + mu)) +
    scale_color_manual(name = if_else(stacked, 'Statistics', 'Legend'),
                       breaks = c('tstat', 'cval'),
                       labels = legend_text,
                       values = c(tstat = 'blue', cval = 'red')) +
    theme(plot.title = element_text(size = 10, face = 'bold', hjust = 0.5),
          plot.margin = unit(c(5.5, if_else(stacked, 5.5, 30), 5.5, 5.5), 'pt'),
          legend.title = element_text(size = 9, face = 'bold'),
          legend.text = element_text(size = 8)) +
    coord_cartesian(xlim = c(-4, 4),
                    clip = 'off')

  p
}
```

Investigate `df_ipeds_pop`

Show variable labels

```
df_ipeds_pop %>% var_label()
#> $instnm
#> [1] "Institution (entity) name"
#>
```

```
#> $unitid
#> [1] "Unique identification number of the institution"
#>
#> $opeid6
#> [1] "First 6 digits of OPEID"
#>
#> $opeid
#> [1] "Office of Postsecondary Education (OPE) ID Number"
#>
#> $control
#> [1] "Control of institution"
#>
#> $c15basic
#> [1] "Carnegie Classification 2015: Basic"
#>
#> $stabbr
#> [1] "State abbreviation"
#>
#> $city
#> [1] "City location of institution"
#>
#> $zip
#> [1] "ZIP code"
#>
#> $locale
#> [1] "Degree of urbanization (Urban-centric locale)"
#>
#> $region
#> [1] "Bureau of Economic Analysis (BEA) regions"
#>
#> $tuit_grad_res
#> [1] "In-state average tuition full-time graduates"
#>
#> $fee_grad_res
#> [1] "In-state required fees for full-time graduates"
#>
#> $tuit_grad_nres
#> [1] "Out-of-state average tuition full-time graduates"
#>
#> $fee_grad_nres
#> [1] "Out-of-state required fees for full-time graduates"
#>
```

```
#> $tuit_md_res
#> [1] "Medicine: In-state tuition"
#>
#> $fee_md_res
#> [1] "Medicine: In-state required fees"
#>
#> $tuit_md_nres
#> [1] "Medicine: Out-of-state tuition"
#>
#> $fee_md_nres
#> [1] "Medicine: Out-of-state required fees"
#>
#> $tuit_law_res
#> [1] "Law: In-state tuition"
#>
#> $fee_law_res
#> [1] "Law: In-state required fees"
#>
#> $tuit_law_nres
#> [1] "Law: Out-of-state tuition"
#>
#> $fee_law_nres
#> [1] "Law: Out-of-state required fees"
#>
#> $books_supplies
#> [1] "Books and supplies 2022-23"
#>
#> $roomboard_off
#> [1] "Off campus (not with family), room and board 2022-23"
#>
#> $oth_expense_off
#> [1] "Off campus (not with family), other expenses 2022-23"
#>
#> $tuitfee_grad_res
#> [1] "graduate, full-time, resident; avg tuition + required fees"
#>
#> $tuitfee_grad_nres
#> [1] "graduate, full-time, non-resident; avg tuition + required fees"
#>
#> $tuitfee_md_res
#> [1] "MD, full-time, state resident; avg tuition + required fees"
#>
```

```
#> $tuitfee_md_nres
#> [1] "MD, full-time, non-resident; avg tuition + required fees"
#>
#> $tuitfee_law_res
#> [1] "Law, full-time, state resident; avg tuition + required fees"
#>
#> $tuitfee_law_nres
#> [1] "Law, full-time, non-resident; avg tuition + required fees"
#>
#> $coa_grad_res
#> [1] "graduate, full-time, state resident COA; == tuition + fees + (ug) books/supplies + (
#>
#> $coa_grad_nres
#> [1] "graduate, full-time, non-resident COA; == tuition + fees + (ug) books/supplies + (ug
#>
#> $coa_md_res
#> [1] "MD, full-time, state resident COA; == tuition + fees + (ug) books/supplies + (ug) of
#>
#> $coa_md_nres
#> [1] "MD, full-time, non-resident COA; == tuition + fees + (ug) books/supplies + (ug) off-
#>
#> $coa_law_res
#> [1] "Law, full-time, state resident COA; == tuition + fees + (ug) books/supplies + (ug) o
#>
#> $coa_law_nres
#> [1] "Law, full-time, non-resident COA; == tuition + fees + (ug) books/supplies + (ug) off
```

Show value labels for variables that are `labelled` class (code note run)

```
df_ipeds_pop %>% select(control,locale,region,c15basic) %>% val_labels()
```

Investigate data structure

- confirm one observation per unitid

```
df_ipeds_pop %>% group_by(unitid) %>% summarise(n_per_key=n()) %>% ungroup() %>% count(n_per
#> # A tibble: 1 x 2
#>   n_per_key     n
#>       <int> <int>
#> 1         1   965
```

Standard Deviation

Standard deviation is a measure of variability in the values of a dataset. Standard deviation measures how much the values deviate away from the mean.

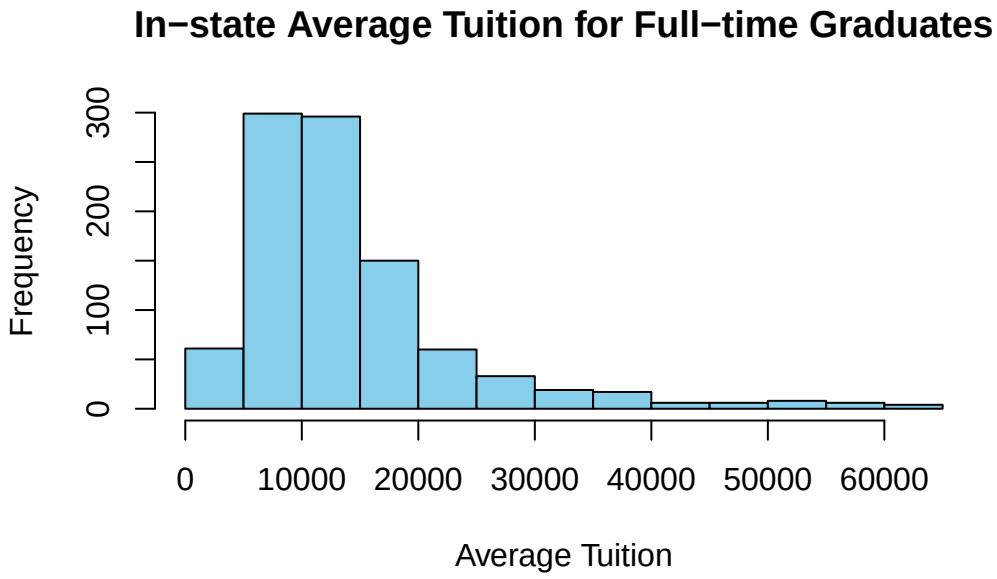
- If all of the values in our dataset are the same, then the mean would be the same value and our standard deviation would be zero.
- A high standard deviation indicates that our values are spread out and there's more variability in our dataset.
- A low standard deviation indicates that our values are close to the mean.

```
# Here is how we would find the standard deviation for in-state average tuition for full-time

sd(df_ipeds_pop$tuit_grad_res)
#> [1] 9807.926

# Since our standard deviation is high, this implies our values are spread out. Let's look at

hist(df_ipeds_pop$tuit_grad_res, breaks = 20, col = "skyblue", main = "In-state Average Tuiti
```



Describing individual variables using R

Using R, we can also use summary() to reduces multiple values down to a single summary output. This would give us the summary statistics such as mean, median, minimum, maximum, and 1st and 3rd quartiles for specific variables in the data frame. Another similar function, but has the ability to collapse a data frame into a single row, summarise() can be used to create new variables that we build from existing variables.

```
#summary(x)
summary(df_ipeds_pop$tuit_grad_res)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1209    8108   11700   14312   17072   63468

#syntax of 'summarise()'
#summarize(data, new_variable = function(existing_variable))
```

Measures of central tendency (Mean, median, mode)

Mean, median, and mode are the measures of central tendency. They are quite important as they are usually the numbers reported out. These usually represent the center of many data points, which means they provide a good summary of what we can expect from the group or dataset as a whole.

- Mean: The "average" or the "expectation" of the dataset. The mean takes the sum of all the values in a dataset and divides by the number of data points. The mean is useful at giving us insight on data that is normally distributed (we will cover types of distributions later in the week).
- Median: The value you find in the middle when you arrange the dataset in ascending order (smallest to largest). When there is no exact middle value, you take the mean of the two values closest to the middle. The median is less influenced by outliers.
- Mode: The value that appears the most in the dataset. Unlike the mean or the median, the mode is always a number that actually occurs in our dataset. Mode is most useful when there is a relatively large sample, so you're likely to have a large number of popular values. Mode is also helpful when the values are not numeric.

```
#Let's double check if we will get the same values as we did above with the summary() functi
mean(df_ipeds_pop$tuit_grad_res)
#> [1] 14311.72
median(df_ipeds_pop$tuit_grad_res)
#> [1] 11700
```

```r
# R does not have a function for mode. Well, there is mode(), but that tells us the storage m

# define data_mode() function
data_mode = function() {

  # calculate mode of in-state average tuition for full-time graduates
  return(names(sort(-table(df_ipeds_pop$tuit_grad_res)))[1])
}

# call data_mode() function
data_mode()
#> [1] "7176"

# We can try this on something that isn't numeric. Let's take a look at the most common city

# define data_mode2() function
data_mode2 = function() {

  # calculate mode of the city that most institutions in our dataset are from
  return(names(sort(-table(df_ipeds_pop$city)))[1])
}

# call data_mode() function
data_mode2()
#> [1] "New York"
```

Measures of dispersion (Standard deviation, variance, range, interquartile range)

Range, interquartile range, variance, and standard deviation are the measures of dispersion or spread. They tell us how well the data is spread around the middle of the dataset, which can tell us how well the mean or median represents the data.

- Range: The distance we find when we take the largest value in the dataset and subtract the smallest value. The larger the range or distance between our most extreme points, the more spread out our dataset is.
- Interquartile range (IQR): The distance between the 75th percentile (Q3) and the 25th percentile (Q1). The IQR doesn't account for the extreme values or outliers, it focuses on the middle 50% of our dataset. Two different datasets could have the same range, but the IQR gives information about their core data points.

- Variance: The measure of how far the values in a dataset are spread out from the mean. To find the variance, first find the mean of the dataset. Then find the difference between each data point and the mean value. Square each of those values, add up all the squared values, then divide up the sum of the square values by number of data points or values minus 1. Sample variance will always be slightly smaller than the population variance. To counter this bias, we usually divide the number of samples minus 1 (n-1), to getter a better guess for the population variance. Variance tells us how much variability is in our data.

  $Variance = \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2$

- Standard deviation: The measure of how dispersed the data point is to the mean of the dataset. To find the standard deviation, first find the mean of the dataset. Then subtract the mean from each data point. Square each of those differences (which will make all the distances positive). Add up the squared differences together, then divide the sum by the number of data points minus 1. Finally, take the square root of the result.

  $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2}$

```
# Let's find the range of in-state average tuition for full-time graduates
max(df_ipeds_pop$tuit_grad_res, na.rm = TRUE) - min(df_ipeds_pop$tuit_grad_res, na.rm = TRUE)
#> [1] 62259

# The function range() in R doesn't give you the range automatically, it gives you the smalle
range(df_ipeds_pop$tuit_grad_res)
#> [1]  1209 63468
63468 - 1209
#> [1] 62259

# Calculate interquartile range (IQR) of in-state average tuition for full-time graduates
IQR(df_ipeds_pop$tuit_grad_res)
#> [1] 8964

# To double check the function, we can use the code we learned earlier about percentiles
quantile(df_ipeds_pop$tuit_grad_res, probs = 0.75) - quantile(df_ipeds_pop$tuit_grad_res, pro
#>  75%
#> 8964

# Calculate variance of in-state average tuition for full-time graduates
var(df_ipeds_pop$tuit_grad_res)
#> [1] 96195409

# Calculate standard deviation of in-state average tuition for full-time graduates
```

```r
sd(df_ipeds_pop$tuit_grad_res)
#> [1] 9807.926
```