IST 687 Inferential statistics

Corey Jackson

2021-02-02 20:51:29

Today's Agenda

- Announcements
- Review of Week 3 (Async; Chapters 4-6)
- Breakout (Complete Lab 4)
- Homework 4 Tips
- Next week's agenda

Announcements

- Office Hours: 6-7pm EDT and by appointment
  - End of class office hours (Group Project)
- Final project deliverables for project update II
  - 30 mins at the end of Week 5 for team meetings

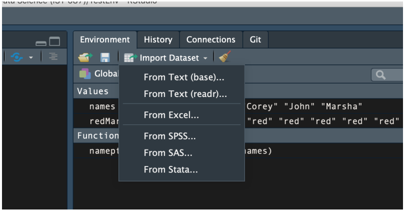
Review of Week 3

Overview of Week 3: (Descriptive Statistics & Functions)

- Importing data
- Data munging
- User-defined Functions
- Exploring distributions

Week 3: Importing data

- Most common formats: txt, csv, Excel, json (Week 5)
- Several packages for importing data from these formats: `read_delim()`, `read_tsv()`, `read_csv()`, `read_csv2()`, `read.xlsx()`.
- These require the `readr` or `xlsx` packages.
- Automatic import using R



Week 3: Cleaning dataframes

- Renaming columns: `names()` or `colnames()`
  - renaming one column: `colnames(dataframe)[1] <- "new_name"`
- Removing columns (scenario: remove columns 1 and 7):
  - `dataframe <- dataframe[,-c(1,7)]` or `dataframe <- dataframe[,c(2:6)]`
- Creating new columns: `dataframe$new_column <- code`
- Coercing datatypes
  - converting columns in data frames: `state$population <- as.numeric(state$population)`

Week 3: Functions

- Basic components of functions: body and arguments

```
function_name <- function(arg)
{
   BODY
}
```

- Functions can have many arguments (seperated by ,)

```
function_name(arg1,arg2,arg3)
```

Week 3: Functions

- A tip for writing functions... start with pseudo-code

```
Distribution <- function(vector,number)
{
  # only keep the elements within the vector that
  # are less than the number, and store the number
  # of eligible elements into the variable "count"

  # calculate the percentage and return the results

}
```

Week 3: Functions

- Stepwise coding with functions

```
vec <- c(8,2,1,4,0)
val <- 2
```

- only keep the elements within the vector that are less than the number, and store the number of eligible elements into the variable “count”

Week 3: Functions

Start simple and add complexity

1. Return elements in vector less than the number

    ```
    vec < val
    ```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE
```

2. Count the number of elements in the vector

    ```
    sum(vec < val)
    ```

```
## [1] 2
```

Week 3: Functions

- Example using length
  vec[vec < val] # returns values

```
## [1] 1 0
```

which(vec < val) # returns index position

```
## [1] 3 5
```

length(vec[vec < val]) OR length(which(vec < val))

Week 3: Functions

```
Distribution <- function(vector,number)
{
  # only keep the elements within the vector that
  # are less than the number, and store the number
  # of eligible elements into the variable "count"

count <- length(vec[vec < val])

  # calculate the percentage and return the results

 perc <- count/length(vec)

}
```

Week 3: Exploring data

- **Descriptive statistics**: (1) central tendency e.g., `mean()` and (2) dispersion gives us the properties of distributions e.g., `sd()`
- **Distibtions**: (1) helps understand your data and (2) helps determine modeling techniques (e.g., non-parametric modeling)
- Simulting distributions in R using e.g., `rnorm()`,`rpareto()`. Simulation helpful when you don't have actual data or limited data.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA): Summarizing data

```
##                    mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0
```

Exploratory Data Analysis (EDA): Summarizing data using `dplyr()`

- The dplyr package is powerful for munging and summarizing data in dataframes.

More on dplyr here: Exploratory Data Analysis with R

Exploratory Data Analysis (EDA): Summarizing data using `dplyr()`

Problem: get the mean hp and mpg by cylinder

```
myCars %>%
  group_by(cyl) %>%
  summarize(
    mean_mpg=mean(mpg),
    mean_hp=mean(hp)
    )
```

```
## # A tibble: 3 x 3
##   cyl   mean_mpg mean_hp
##   <fct>    <dbl>   <dbl>
## 1 4         26.7    82.6
## 2 6         19.7   122.
## 3 8         15.1   209.
```

Exploratory Data Analysis (EDA): Summarizing data using `dplyr()`

Problem: get the mean hp and mpg by each cyl and gear pair

```
myCars %>%
  group_by(cyl,gear) %>%
  summarize(
    mean_mpg=mean(mpg),
    mean_hp=mean(hp)
    )
```

Exploratory Data Analysis (EDA): Summarizing data using `dplyr()`

```
## # A tibble: 8 x 4
## # Groups:   cyl [3]
##   cyl   gear  mean_mpg mean_hp
##   <fct> <fct>    <dbl>   <dbl>
## 1 4     3         21.5     97 
## 2 4     4         26.9     76 
## 3 4     5         28.2    102 
## 4 6     3         19.8    108.
## 5 6     4         19.8    116.
## 6 6     5         19.7    175 
## 7 8     3         15.0    194.
## 8 8     5         15.4    300.
```

Useful resources for EDA

1. R for Data Science (Chapter 7)
2. Exploratory Data Analysis

Useful packages/functions for the future

Here are a few links to site with useful packages/functions for doing data science:

1. Top R libraries for data science
2. Quick list of useful R packages

Week 4 primer

Week 4: Sampling

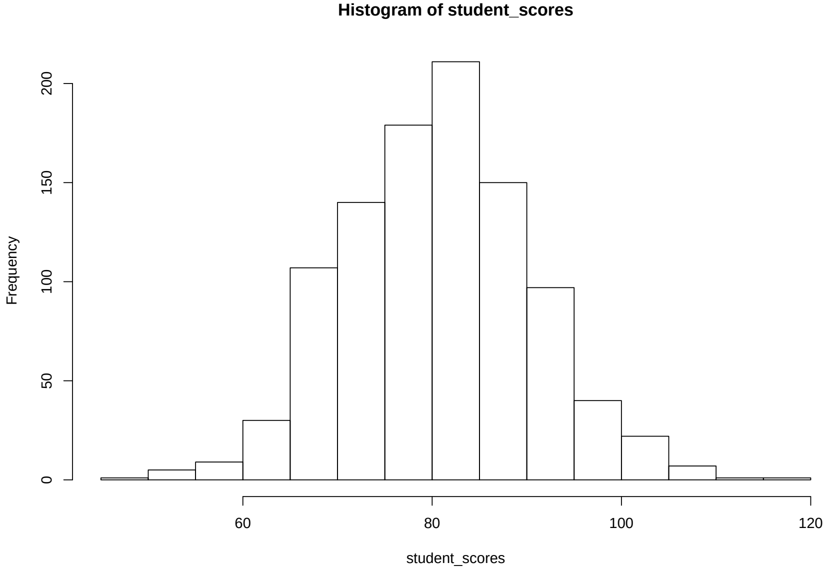
Sampling data

- Helpful when we don't have access to the full population
- Allows us to make assumptions about the underlying truth (i.e., population).
- in R `sample(X =, size =, replace = )`

Week 4: Sampling

A dataset of 1000 student scores from the mid-term

```
student_scores <- rnorm(1000,80,10)
```



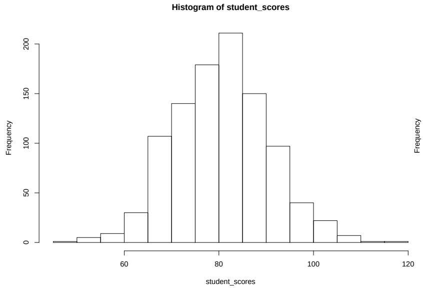
Week 4: Sampling

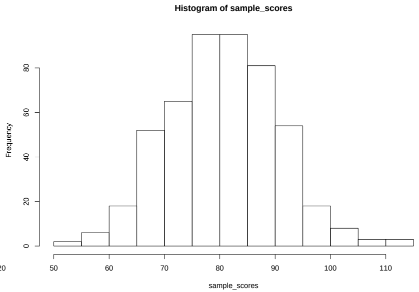
Obtain a sample of size 500 from the student_scores vector with replacement

```
sample(student_scores,500,replace = TRUE)
```

```
## [1]  85.74439  86.20582  81.53518 100.84060  68.21117
```

Week 4: Visualizing sample distributions



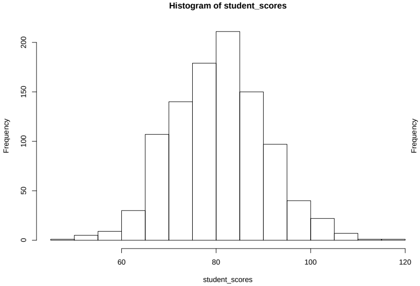


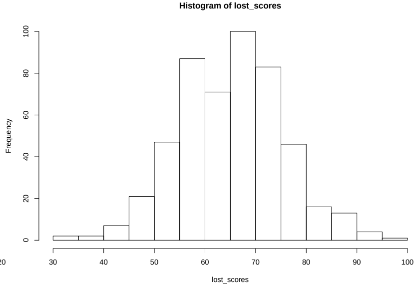
Mean parameters:
- student_scores: 80.54
- sample_scores: 80.51

The distributions and parameters *look* similar.

Week 4: Comparing distributions using sample statistic

What about a new sample of `lost_scores`? Are these scores the same as the `student_scores`?





Mean parameters:
- student_scores: 80.54
- lost_scores: `65.14`

The parameters *do not look* similar.

Week 4: Comparing sample distributions

Comparing two distributions

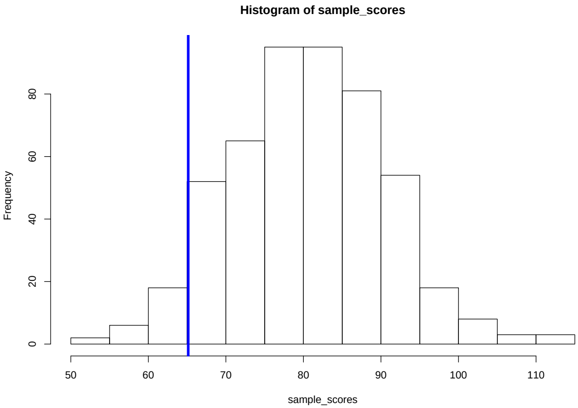
- Helpful for evaluating whether two datasets are the "same" i.e., come from the same population.
- To make this determination we can compare the sample statistic (the mean) from the "unknown" population to the known population

Week 4: Comparing sample distributions

A scenario:

1. You have the parameters of sample_scores and you want to know if lost_scores with a single sample mean of 65.14 is the same data as sample_scores with a mean of 80.51.
2. We can compare the sample mean of lost_scores (65.14) to the distirbution of 'sample_scores' to determine if the lost_scores fits within an "acceptable range" of values.

Week 4: Comparing sample distributions



Is the mean value for unknown_scores within an acceptable range?

Week 4: Comparing sample distributions

We can determine whether the mean for `lost_scores` is within our acceptable range:

- Determine a threshold
- Comparing the mean of `lost_scores` to the distribution of sample_scores.

If the mean falls outside of the threshold we cannot say it is from the same population.

Week 4: Comparing sample distributions

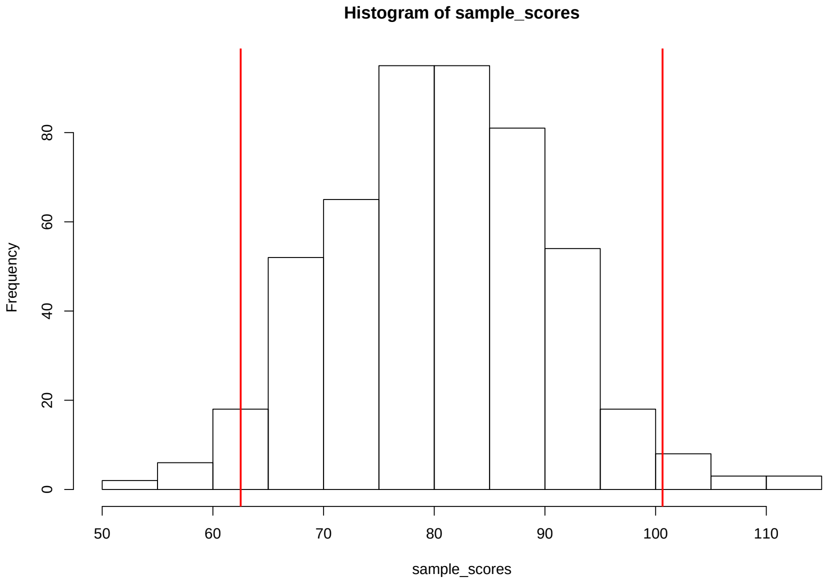
Our acceptable threshold (also called alpha) is .05. The value may differ depending on field.

```
quantile(sample_scores,probs =  c(0.025,0.975))
```

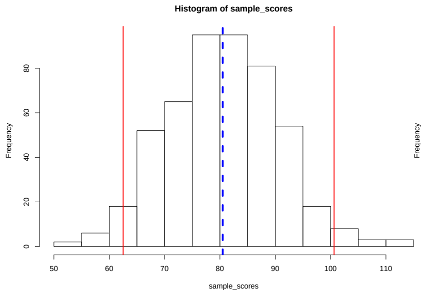
```
##      2.5%     97.5%
##  62.51097 100.62659
```

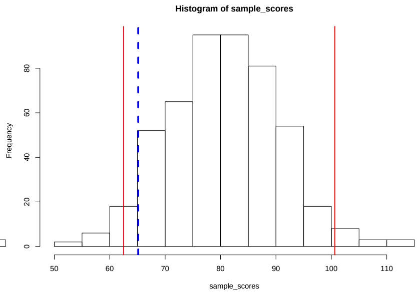
Read as: 2.5% percent of the values in our dataset is smaller than 62.51 and 2.5% percent of the values in our dataset is greater than 100.63.

Week 4: Comparing sample distributions



Week 4: Comparing sample distributions





Week 4: Comparing sample distributions

Our acceptable range of truth lies between these numbers. So, yes, it is likely, `lost_scores` comes from the same population of scores.

Lab 4

Lab 4: Descriptive Stats & Functions

Lab Goals:

- Investigating new functions
- Create samples from a population
- Making inferences about the population based on the sample mean

Groups for Pair Programming

Lab 4: Descriptive Stats & Functions

- New functions for the week (`set.seed()`,`runif()`,`sample()`). Describe what each function does by adding comments in the code:
- A commented line of code uses #

```
# Computes the mean of a vector
mean(vector)
```

Note: Explore the purposes of each using ??help or a search engine

Homework Tips

Homework 4 Tips

- Printing from functions
- Replicating samples
- Working with missing data

Be sure to install and load the `moments()` package.

Homework 4 Tips: Printing in Functions (Step 1)

- `cat()`: take many arguments, but last argument should be a new line "\n"

```
nameptinter <- function(names){
  cat("My name is:",names,"\n")
  cat("There are ",nchar(names),"letters in", names)
}
```

```
nameptinter("Corey")
```

```
## My name is: Corey
## There are 5 letters in Corey
```

Homework 4 Tips: Replicating samples (Step 2)

- Repating a sequence programatically
- Two functions: `replicate(times,process)` or `rep(process,times)`

```
replicate(4,"Corey")
```

```
## [1] "Corey" "Corey" "Corey" "Corey"
```

```
rep(mean(c(10,43,10,46,5)),4)
```

```
## [1] 22.8 22.8 22.8 22.8
```

Homework 4 Tips: "Getting data" (Step 2)

- Counting things in a vector that match some critera. Using `grep()` or `which()`

A vector of names stored in `people`

```
## [1] "Corey"  "Corey"  "Corey"  "Marsha"
```

```
grep("Corey", people)
```

```
## [1] 1 2 3
```

```
which(people=="Corey")
```

```
## [1] 1 2 3
```

Homework 4 Tips: Missing data (Step 3)

- Working with missing values. *A matter of informed choice?*
- Use `summmary()` to investigate missing values
- Choices: ignore, replace, delete
  - `na.omit()` or `complete.cases()` removes observations with NAs in any column

```
##   score1 score2 score3
## 1      9     NA      1
## 2      6      5      3
## 3     NA      2      5
```

`data[complete.cases(data), ]` or `na.omit(data)`

```
##   score1 score2 score3
## 2      6      5      3
```

Homework 4 Tips: Missing data (Step 3)

- Computing on columns with missing values `na.rm = TRUE`

```
mean(data$score1)
```

```
## [1] NA
```

```
mean(data, na.rm=TRUE)
```

```
## [1] 7.5
```

Homework 4 Tips

- Use the results in question 6 as a starting point for step 7. Remember, functions can take other functions as arguments

e.g.,

```
replicate(times,process)
replicate(100, mean(sample(studentPop, size=sampleSize)))
```

Next Week

Week 5 : Connecting with external data sources

- **Asynchronous**
  - Read Chapter 11 in Saltz and Stanton
  - Submit HW 4 and Lab 4 by Monday
  - Continue collaborating on your final project
- **Live Session**
  - Lab 5: Storage Wars
  - Team meetings for final project/update 2