Summary Statistics

Max Turgeon

DATA 2010–Tools and Techniques in Data Science

Lecture Objectives

- Compute summary statistics in R
- Use the Central Limit Theorem to construct confidence intervals for means and proportions
- Import data into R

Motivation

- It’s one of the great paradoxes of statistics:
    - To better understand data, summarise it.
- The mean/average occupies a special place, because of its nice properties.
- But looking at multiple summaries gives us a fuller picture.

Mean

- Recall the definition: if we have n observations $X_1, \dots, X_n$, their **mean** (or average) is their sum divided by n:

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

- The mean is a measure of *central tendency*, i.e. where the bulk of the observations tend to fall.
- In R, we can compute the mean using the function `mean`.

Examples i

- We'll use the dataset `mtcars`, which comes with `R` by default.
- Datasets are usually stored in `data.frame`s.
  - We can inspect `data.frame`s using `str` or `head`/`tail`.

```
str(mtcars)
```

```
## 'data.frame': 32 obs. of 11 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3
24.4 22.8 19.2 ...
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
## $ disp: num 160 160 108 258 360 ...
```

Examples ii

```
## $ hp : num 110 110 93 110 175 105 245 62 95
123 ...
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21
3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
## $ qsec: num 16.5 17 18.6 19.4 17 ...
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

Examples iii

- There are many ways to compute the average miles per gallon (mpg) for this dataset. I will demonstrate two ways.
  - Extract column
  - Use **tidyverse** (next week)

```r
# Extract column with $ and use mean function
mpg_vec <- mtcars$mpg
mean(mpg_vec)
```

```
## [1] 20.09062
```

Examples iv

```r
# Or in one line of code
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

- We can also compute the variance, standard deviation, IQR, etc.

```r
mpg_vec <- mtcars$mpg
c(var(mpg_vec), sd(mpg_vec))
```

Examples v

```
## [1] 36.324103  6.026948
```

```
IQR(mpg_vec)
```

```
## [1] 7.375
```

Exercise

Use the `median` function to compute the median `mpg`.

Solution

```r
# Extract column first
mpg_vec <- mtcars$mpg
median(mpg_vec)
```

```
## [1] 19.2
```

```r
# Or in one line of code
median(mtcars$mpg)
```

```
## [1] 19.2
```

Missing data i

- In R, missing data will usually be represented by the value `NA`.
    - Unless the data collection used a different value...
- If you try to summarize a vector with missing value, you will get `NA`.
- Therefore, missing values need to be removed first using `na.rm = TRUE`.

```r
# Create a vector with NA
vect <- c(1, 4, NA, 5.5)
```

Missing data ii

```r
mean(vect)
```

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

```r
mean(vect, na.rm = TRUE)
```

```
## [1] 3.5
```

Summaries by group i

- We may want to summarise by groups, i.e. for each subgroup we want the sample mean.
  - We usually want to compare them!

```r
# Let's look at cyl
table(mtcars$cyl)
```

```
## 
##  4  6  8 
## 11  7 14 
```

Summaries by group ii

```r
mpg_4c <- mtcars$mpg[mtcars$cyl == 4]
mpg_6c <- mtcars$mpg[mtcars$cyl == 6]
mpg_8c <- mtcars$mpg[mtcars$cyl == 8]
```

```r
c(mean(mpg_4c), mean(mpg_6c), mean(mpg_8c))
```

```
## [1] 26.66364 19.74286 15.10000
```

Exercise

Compute the median `mpg` for each value of `cyl`.

Solution

```r
c(median(mpg_4c), median(mpg_6c),
  median(mpg_8c))
```

```
## [1] 26.0 19.7 15.2
```

Central Limit Theorem i

- The sample mean gives us some idea about the *population* mean.
    - “What if we could measure the height of all Canadians, instead of a sample?”
- But how certain can we be that the mean of our data is close to the true population mean?
- The Central Limit Theorem tells us that, whatever the distribution of the data, its *sample mean* behaves like a normal random variable.

Central Limit Theorem ii

- More precisely, if we have n independent observations that come from a distribution with mean μ and variance σ^2, then the sample mean is approximately normal:

$$\bar{X} \approx N(\mu, \sigma^2/n).$$

- The most important consequence: we can construct confidence intervals for the *population* mean.
    - For 95% CI: $\bar{X} \pm 1.96\hat{\sigma}/\sqrt{n}$, where $\hat{\sigma}$ is the *standard deviation* of the data (use the function `sd`).

Central Limit Theorem iii

- Important observations:
    - As we collect more data, the standard deviation of the data can go up or down. On the other hand, the confidence interval will become narrower and narrower.
    - In practice, we don't really know if our data is independent, or if it all comes from the same distribution. This uncertainty has to be reflected in the strength of our conclusions about the data.
- **Vocabulary**: $\hat{\sigma}$ is the standard *deviation*, and $\hat{\sigma}/\sqrt{n}$ is the standard *error*.

Examples i

```r
# Recall----
c(mean(mtcars$mpg), sd(mtcars$mpg), nrow(mtcars))
```

```
## [1] 20.090625  6.026948 32.000000
```

```r
# 95% Confidence interval
c(20.09062 - 1.96*6.026948/sqrt(32),
  20.09062 + 1.96*6.026948/sqrt(32))
```

```
## [1] 18.00239 22.17885
```

Examples ii

```r
# Alternative: save values in variables
# and use variables
mean_mpg <- mean(mtcars$mpg)
sd_mpg <- sd(mtcars$mpg)
n <- nrow(mtcars)

c(mean_mpg - 1.96*sd_mpg/sqrt(n),
  mean_mpg + 1.96*sd_mpg/sqrt(n))
```

```
## [1] 18.00239 22.17886
```

Exercise

Compute the *average and standard deviation* for `qsec`, which is the quarter-mile time (i.e. the time it takes the car to travel a quarter mile starting from a standstill).

Compute a 95% confidence interval for the average quarter-mile time.

Solution

```
mean_qsec <- mean(mtcars$qsec)
sd_qsec <- sd(mtcars$qsec)
n <- nrow(mtcars)
mean_qsec
```

```
## [1] 17.84875
```

```
sd_qsec
```

```
## [1] 1.786943
```

```
c(mean_qsec - 1.96*sd_qsec/sqrt(n),
  mean_qsec + 1.96*sd_qsec/sqrt(n))
```

```
## [1] 17.22961 18.46789
```

Proportions are means too!

- If I want the proportion of apples among fruits, I can take the mean of binary observations:
  - $X_i = 1$ if the i-th fruit is an apple.
  - $X_i = 0$ otherwise.
- This means we can use the CLT for proportions too.
  - **Note**: It doesn't work well when the proportion $\hat{p}$ and the number of observations n are small.

Examples i

```r
cyl6 <- as.numeric(mtcars$cyl == 6)
table(cyl6)
```

```
## cyl6
##  0  1
## 25  7
```

```r
mean(cyl6)
```

```
## [1] 0.21875
```

Examples ii

- **Note**: If $\hat{p}$ is the proportion, then $\hat{\sigma} = \sqrt{\hat{p}(1 - \hat{p})}$.

```r
n <- nrow(mtcars)
phat <- mean(cyl6)
sigma <- sqrt(phat*(1 - phat))

c(phat - 1.96*sigma/sqrt(n),
  phat + 1.96*sigma/sqrt(n))
```

```
## [1] 0.07551468 0.36198532
```

Importing CSV files i

- As we've seen, R comes with built-in datasets.
- To analyze your *own* data, you would need to import it.
- Two most common file formats:
    - Comma-Separated Value (CSV)
    - Excel file.
- We will use different functions for each file format.

```
# For CSV files----
library(readr)

dataset <- read_csv("path/to/file.csv")
```

Importing CSV files ii

```r
# For Excel files----
library(readxl)

dataset <- read_excel(”path/to/file.xlsx”)
# read_excel can also read older xls files
```

Example i

- The csv file can be on your local computer or even online.

```
library(readr)

url <- paste0("https://biostat.app.vumc.org/",
              "wiki/pub/Main/DataSets/",
              "diabetes.csv")
```

```
dataset <- read_csv(url)
```

Example ii

```
##
## -- Column specification ------------------------
## cols(
##   id = col_double(),
##   chol = col_double(),
##   stab.glu = col_double(),
##   hdl = col_double(),
##   ratio = col_double(),
##   glyhb = col_double(),
##   location = col_character(),
##   age = col_double(),
##   gender = col_character(),
```

Example iii

```
##   height = col_double(),
##   weight = col_double(),
##   frame = col_character(),
##   bp.1s = col_double(),
##   bp.1d = col_double(),
##   bp.2s = col_double(),
##   bp.2d = col_double(),
##   waist = col_double(),
##   hip = col_double(),
##   time.ppn = col_double()
## )
```

Example iv

```r
# How many rows and columns?
dim(dataset)
```

```
## [1] 403  19
```

```r
# Average height and weight
c(mean(dataset$height), mean(dataset$weight))
```

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

Example v

```r
# Missing values!
c(mean(dataset$height, na.rm = TRUE),
  mean(dataset$weight, na.rm = TRUE))
```

```
## [1]  66.0201 177.5920
```