STAT 100 - Important Code for Week 5 (Simulation-Based Confidence Intervals)

Functions

Set **seed** to save randomly-generated information over time (put this at top of r-chunk)

- set.seed(---)
    - *set.seed(1)*

Compute the **sample statistic**, which will be center of **confidence interval**

- SAMPLE-STATISTIC <- DATASET %>%
  drop_na(RESPONSE-VAR) %>%
  specify(response = RESPONSE-VAR) %>%
  calculate(stat = "TYPE-OF-STAT")
    - *x_bar <- movies %>%*
      drop_na(OpeningWeekend) %>%
      specify(response = OpeningWeekend) %>%
      calculate(stat = "mean")

Generate and graph a **bootstrap distribution** from a **mean**

- BOOTSTRAP-DIST <- DATASET %>%
  drop_na(RESPONSE-VAR) %>%
  specify(response = RESPONSE-VAR) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "TYPE-OF-STAT")
  ggplot(data = BOOTSTRAP-DIST,
  mapping = aes(x = stat)) +
  geom_histogram()
    - *bootstrap_dist <- movies %>%*
      drop_na(OpeningWeekend) %>%
      specify(response = OpeningWeekend) %>%
      generate(reps = 1000, type = "bootstrap") %>%
      calculate(stat = "mean")
      ggplot(data = bootstrap_dist,
      mapping = aes(x = stat)) +
      geom_histogram()

Generate and graph a **bootstrap distribution** from a **difference in means**

- BOOTSTRAP-DIST <- DATASET %>%
  drop_na(RESPONSE-VAR) %>%
  specify(RESPONSE-VAR ~ CAT-EXPLANATORY-VAR) %>%

generate(reps = 1000, type = "bootstrap") %>%
calculate(stat = "diff in means",
order = c("CATEGORY-1", "CATEGORY-2"))
ggplot(data = BOOTSTRAP-DIST,
mapping = aes(x = stat)) +
geom_histogram()

- *bootstrap_dist <- movies %>%*
  drop_na(OpeningWeekend) %>%
  filter(Genre %in% c("Action", "Drama")) %>%
  specify(OpeningWeekend ~ Genre) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "diff in means",
  order = c("Action", "Drama"))
  ggplot(data = bootstrap_dist,
  mapping = aes(x = stat)) +
  geom_histogram()

Compute **confidence interval via SE method**

- CONF-INTERVAL <- BOOTSTRAP-DIST %>%
  get_confidence_interval(type = "se", level = CONF-LEVEL,
  point_estimate = SAMPLE-STATISTIC)
  - *ci <- bootstrap_dist %>%*
    get_confidence_interval(type = "se", level = 0.95,
    point_estimate = x_bar)

Compute **confidence interval via percentile method**

- CONF-INTERVAL <- BOOTSTRAP-DIST %>%
  - get_confidence_interval(type = "percentile", level = CONF-LEVEL)
  - *ci <- bootstrap_dist %>%*
    get_confidence_interval(type = "percentile", level = 0.95)

Quickly graph **bootstrap distribution**

- BOOTSTRAP-DIST %>%
  visualize()
  - *bootstrap_dist %>%*
    visualize()

STAT 100 - Important Code for Week 6 (Simulation-Based Hypothesis Tests)

Functions for Hypothesis Tests (for Sample Proportion)

Set **seed** to save randomly-generated information over time (put this at top of r-chunk)

- set.seed(---)
    - *set.seed(1)*

Construct data frame of sample results for a **sample proportion**

- DATA-FRAME <- data.frame(CAT-RESPONSE-VAR = c(rep("CAT-1", VALUE-1), rep("CAT-2", VALUE-2)))
    - *esp <- data.frame(guess = c(rep("correct", 106), rep("incorrect", 329 - 106)))*

Generate **null distribution** for a **sample proportion**

- NULL-DIST <- DATA-FRAME %>%
  specify(response = RESPONSE-VAR, success = "CAT-OF-INTEREST") %>%
  hypothesize(null = "point", p = NULL-VALUE) %>%
  generate(reps = 1000, type = "draw") %>%
  calculate(stat = "prop")
    - *null_dist <- esp %>%*
      specify(response = guess, success = "correct") %>%
      hypothesize(null = "point", p = 0.25) %>%
      generate(reps = 1000, type = "draw") %>%
      calculate(stat = "prop")

Compute **observed test statistic** for a **sample proportion**

- TEST-STATISTIC <- DATA-FRAME %>%
  specify(response = RESPONSE-VAR, success = "CAT-OF-INTEREST") %>%
  calculate(stat = "prop")
    - *test_stat <- esp %>%*
      specify(response = guess, success = "correct") %>%
      calculate(stat = "prop")

Compute **p-value**

- P-VALUE <- NULL-DIST %>%
  get_p_value(obs_stat = TEST-STAT, direction = "DIRECTION-OF-TEST")
    - *p_value <- null_dist %>%*
      get_p_value(obs_stat = test_stat, direction = "greater")

Functions for Computing Power (for Sample Proportion)

Step #1: Construct a dataframe and generate a **null distribution**

- DATA-FRAME <- data.frame(CAT-RESPONSE-VAR = c(rep("CAT-1", VALUE-1), rep("CAT-2", VALUE-2)))
  - *dat <- data.frame(at_bats = c(rep("hit", 80), rep("miss", 20)))*
- NULL-DIST <- DATA-FRAME %>%
  specify(response = RESPONSE-VAR, success = "CAT-OF-INTEREST") %>%
  hypothesize(null = "point", p = NULL-VALUE) %>%
  generate(reps = 1000, type = "draw") %>%
  calculate(stat = "prop")
  - *null <- dat %>%*
    specify(response = at_bats, success = "hit") %>%
    hypothesize(null = "point", p = 0.25) %>%
    generate(reps = 1000, type = "draw") %>%
    calculate(stat = "prop")

Step #2: Determine the **critical value(s)**

- quantile(NULL-DIST$NULL-VALUE-IN-NULL-DIST, ONE-MINUS-ALPHA)
  - *quantile(null$stat, 0.95)*

Step #3: Generate an **alternative distribution**

- ALT-DIST <- DATA-FRAME %>%
  specify(response = RESPONSE-VAR, success = "CAT-OF-INTEREST") %>%
  hypothesize(null = "point", p = ALT-VALUE) %>%
  generate(reps = 1000, type = "draw") %>%
  calculate(stat = "prop")
  - *alt <- dat %>%*
    specify(response = at_bats, success = "hit") %>%
    hypothesize(null = "point", p = 0.33) %>%
    generate(reps = 1000, type = "draw") %>%
    calculate(stat = "prop")

Step #4: Find probability of **critical value(s)** (or more extreme) under **alternative distribution**

- ALT-DIST %>%
  summarize(power = mean(ALT-VALUE-IN-ALT-DIST > quantile(NULL-DIST$ALT-VALUE-IN-ALT-DIST, 0.95)))
  - *alt %>%*
    summarize(power = mean(stat > quantile(null$stat, 0.95)))