Studio 9 Simulating confidence intervals
18.05, Spring 2022

Overview of the studio

This studio explores confidence intervals using simulated data

R introduced in this studio

There is no new R introduced in this studio. It makes use of familiar functions like `rnorm`, `qnorm`, `qt`, `rbinom`

Download the zip file

- You should have downloaded the file `mit18_05_s22_studio9.zip` from our MITx site.
- Unzip it in your 18.05 studio folder.
- You should see the following R files
  `mit18_05_s22_studio9.r`
  `mit18_05_s22_studio9-samplecode.r`
  `mit18_05_s22_studio9-test.r`

  and the following other files

  `mit18_05_s22_studio9-test-answers.html`

Prepping R Studio

- In R studio, open `mit18_05_s22_studio9-samplecode.r` and `mit18_05_s22_studio9.r`
- Using the Session menu, set the working directory to source file location. (This is a good habit to develop!)
- Answer the questions in the detailed instructions just below. Your answers should be put in `mit18_05_s22_studio9.r`
- Solution code will be posted tomorrow at 10 pm

Detailed instructions for the studio

• Go through **mit18_05_s22_studio9-samplecode.r** as a tutorial.

Summary of questions
1a. Compute the simulated type 1 CI error rate for z-confidence intervals
1b. Same as part a, except use t-confidence intervals
1c. Based on a prior, find the prior and posterior probability that theta is in a given confidence interval.

2. (OPTIONAL) Simulate a poll and give the rule-or-thumb 95% confidence interval.

Problem 1

Problem 1. This problem will explore the meaning of c in a c-confidence interval for the mean. We will track the simulated type 1 confidence interval error rate. In part c, we will look at the Bayesian posterior probability the parameter of interest is in a given confidence interval.

In order to count results we will be omniscient and always know the true value of the mean and its prior probability.

Recall that the **type 1 confidence interval error rate** is the fraction of trials where the true mean is not in the confidence interval.

Problem 1a. Here you will finish the code for the function
`studio9_problem_1a(theta_vals, theta_prior, sigma, n_data, confidence, n_trials)`

The arguments to this function are:
`theta_vals` = possible values for the mean of the normal distribution
`theta_prior` = probabilities for choosing a θ from `theta_vals`
`sigma` = standard deviation of the normal distribution
`n_data` = the number of data values in each trial
`confidence` = the confidence level, e.g. 0.95, 0.9 etc
`n_trials` = number of trials in the simulation

Our data will be drawn from a normal distribution $N(\theta, \sigma^2)$, where the value of θ is unknown and the value of σ is known. The possible values and prior probabilities of θ are given in the arguments `theta_vals, theta_prior`.

For problem 1(a), we will run an experiment `n_trials` times and keep track of the type 1 CI-error rate. The experiment will consist of the following steps

Step 1. Choose a random value of `theta` using `theta_vals` and `theta_prior`.

Step 2. draw `n_data` data points from a $N(\theta, \sigma^2)$ distribution.

Step 3. Create a z-confidence interval with the confidence given in the argument `confidence`. Here you will use the known value of σ.

Step 4. Check if the true value of θ is in the interval. If it isn't we call it a type 1 CI-error. (We can only do this because we are omniscient and know the true value of theta.)

Run the experiment `n_trials` times. Print out the last confidence interval and the fraction of type 1 CI-errors.

Problem 1b. Here you will finish the code for the function
`studio9_problem_1b(theta_vals, theta_prior, sigma, n_data, confidence, n_trials)`

The arguments to this function are:
`theta_vals` = possible values for the mean of the normal distribution
`theta_prior` = probabilities for choosing a θ from `theta_vals`
`sigma` = standard deviation of the normal distribution

`n_data` = the number of data values in each trial
`confidence` = the confidence level, e.g. 0.95, 0.9 etc
`n_trials` = number of trials in the simulation

This problem is almost identical to 1(a). The only difference is that you will compute t-confidence intervals. So, you will use the given value of σ to generate the data, but you won't use σ when computing the t-confidence interval.

Problem 1c. Here you will finish the code for the function
`studio9_problem_1c(theta_vals, theta_prior, sigma, n_data, confidence, xbar)`

The arguments to this function are:
`theta_vals` = possible values for the mean of the normal distribution
`theta_prior` = probabilities for choosing a θ from `theta_vals`
`sigma` = standard deviation of the normal distribution
`n_data` = the number of data values in each trial
`confidence` = the confidence level, e.g. 0.95, 0.9 etc
`xbar` = the mean of the data

Here we will put on our Bayesian hats and find the probability the true value of θ is in our confidence interval.

We assume our data is sampled from a $\mathrm{N}(\theta, \sigma^2)$ distribution, where θ is unknown and σ is known. We give you the data mean in the argument `xbar`. Using this do the following:

(i) Use the data to update the given prior to a posterior distribution on the possible values of θ.

(ii) Find the z-confidence interval with the given `confidence`.

(iii) Compute both the prior and posterior probabilities that θ is in the confidence interval computed in (ii).

Print out, the prior and posterior distributions, the confidence interval and the prior and posterior probabilites found in (iii).

Problem 2 (OPTIONAL)

Here you will finish the code for the function
`studio9_problem_2(true_theta, n)`

The arguments to this function are:
`true_theta` = the true proportion of the population who prefer Lincoln.
`n` = the number of people polled.

Here we imagine taking a poll in 1860 to find out the the fraction of Massachusetts residents who support Lincoln. The argument `true_theta` is the true proportion supporting Lincoln.

The function should simulate (using `true_theta`) polling `n` people. It should then compute and print out the rule-of-thumb 95% confidence interval. Print this out as an estimated proportion plus or minus a margin of error.

Testing your code

For each problem, we ran the problem function with certain parameters. You can see the function call and the output in `mit18_05_s22_studio9-test-answers.html`. If you call the same function with the same parameters, you should get the same results as in `mit18_05_s22_studio9-test-answers.html` – if there is randomness involved the answers should be close but not identical.

For your convenience, the file `mit18_05_s22_studio9-test.r` contains all the function calls used to make `mit18_05_s22_studio9-test-answers.html`.

Before uploading your code

1. Make sure all your code is in `mit18_05_s22_studio9.r`. Also make sure it is all inside the functions for the problems.
2. Clean the environment and plots window.
3. Source the file.
4. Call each of the problem functions with the same parameters as the test file `mit18_05_s22_studio9-test-answers.html`.
5. Make sure it runs without error and outputs just the answers asked for in the questions.
6. Compare the output to the answers given in `mit18_05_s22_studio9-test-answers.html`.

Upload your code

Use the upload link on our MITx site to upload your code for grading.

Leave the file name as `mit18_05_s22_studio9.r`. (The upload script will automatically add your name and a timestamp to the file.)

You can upload more than once. We will grade the last file you upload.

Due date

Due date: The goal is to upload your work by the end of class.

If you need extra time, you can upload your work any time before 10 PM ET the day after the studio.

Solutions uploaded: Solution code will be posted on MITx at 10 PM the day after the studio.