

Lab 7: A little precision

Due 3/3, 3 Questions

With this lab we consider the ways in which we can quantify the variability in an estimate. We have seen two basic approaches in lecture: One computational and the other analytical. We will make explicit links between these two and in the process firm up our language and our methods a bit (“precision” in two senses of the word).

We will again appeal to simulation for this little study. We will simulate the act of conducting a survey. The pieces we need to assemble to do this include: a) a population of interest, b) a numerical value we can assign to all the items/subjects in the population and a summary of these values (the mean, median, proportion) that we will estimate c) a method for drawing simple random samples from the population.

The (custom) R function `survey` will do this for us. We will take as our population the 20,718 residents of Millbrae, CA (a city in the San Francisco Bay Area). We would like to learn something about this group of people by taking small surveys. This lab will consider how precise our estimates are, and how this precision varies with sample size.

1. Getting started

We will first load up data that we’ll need for this lab.

```
source("http://www.stat.ucla.edu/~cocteau/stat13/data/precise.R")  
ls()
```

You should see a function called `survey` and a couple of other data sets that constitute our populations (`people` and `households`). The values in `people` refer to the ages of every resident of Millbrae; and those in `households` refer to household incomes in 2007 (in thousands of dollars). We will begin by taking a simple survey of the households Millbrae.

```
x = survey(100,households)
```

```
mean(x)
```

At a conceptual level, you can imagine selecting 100 households at random (say, from the phonebook), interviewing the head of the household and finding out how much money they made last year. Admittedly this setup is a little strained because household incomes in Millbrae are published by the U.S. Census Bureau (which is ultimately the source of our data). For the purposes of this lab, however, we will let household income represent some aspect of the community that is not known. Think instead about how you might ask about more interesting questions like Internet usage – “How many hours a day are you on the web?” – or perhaps the effects of the current recession – “Have you put off a big purchase in the last six months?”.

```
y = survey(200,people)
mean(y)
```

With these commands, we imagine creating a simple random sample of 200 residents in Millbrae, contacting them and recording their ages. As with household income, these data are carefully collected by the Census Bureau and so the example is a little artificial from a practical standpoint. You can instead imagine needing information about residents that is not quite so readily available. Think about a health-related question like “Have you eaten peanut butter in the last week?”, a question the CDC and FDA was asking a number of people last month as they tried to uncover the source of the salmonella outbreak.

In each case we constructed a simple random sample of the population, collected some numerical measure for each member of the sample, and then computed the sample mean. As we saw in lecture, this statistic is an estimate of the population mean (the average age of residents in Millbrae or the average income of households in the city). Let’s repeat the last two lines 10 times or so.

```
y = survey(200,people)
mean(y)
```

You should notice that the mean of your sample varies each time you conduct a survey. If this were a real survey, the effort in contacting 200 people could be considerable. So each time you hit ENTER, you should think about the effort it would take to perform this survey and appreciate the luxury of this magical simulation world where we have the population so readily at our disposal.

2. Confidence intervals

In the last lab, we made use of the bootstrap to form a 95% confidence interval. Let's take a survey and come up with a confidence interval for the ages of residents in Millbrae. We will use the built-in packages for bootstrapping now:

```
library(bootstrap)

y = survey(200,people)
boots = bootstrap(y,5000,mean)
```

For those of you working on your own computers, you will need to install another package. This time it is called `bootstrap`. You will again issue the command `install.packages("bootstrap")` and follow the instructions, selecting the location of a computer near you (a host for the code you will be downloading).

The command above takes our survey results `y` and applies the estimate `mean`. It then forms 5000 bootstrap replicates of the mean, and we have stored them in `boots`. Below we have a look at the bootstrap samples (they are stored in a variable called `thetastar` in `boots`) and then form the percentile confidence interval.

```
hist(boots$thetastar)

lo = quantile(boots$thetastar,0.025)
hi = quantile(boots$thetastar,0.975)
```

Have a look at the values of `lo` and `hi`. What does the interval they represent `[lo,hi]` tell you about the plausible range for the average age of all the residents in Millbrae?

We can now perform a little piece of magic that is never really possible when you are conducting a survey. We can see if the true population parameter is contained in our confidence interval. Again, this is **ONLY POSSIBLE BECAUSE WE ARE SIMULATING FROM A KNOWN POPULATION**.

```
contains(lo,hi,mean(people))
```

The last command asks whether or not the mean of the population (in this case the mean age of people living in Millbrae). The function returns **TRUE** if the population parameter is contained in the interval and **FALSE** otherwise. We can now give a precise meaning to a confidence interval. Let us repeat the process above 100 times, each time taking a different survey, forming a confidence interval and seeing how many of the intervals contain the truth.

```
results = rep(0,100)

for(i in 1:100)
{
  y = survey(200,people)
  boots = bootstrap(y,5000,mean)

  lo = quantile(boots$thetastar,0.025)
  hi = quantile(boots$thetastar,0.975)

  results[i] = contains(lo,hi,mean(people))
  print(i)
}
```

This looks bad, I know. But notice that all the commands are things you've seen before. You have a loop of 100 iterations and for each one we are conducting a survey, computing a confidence interval from the survey results (the ages of 200 randomly selected people in the city), and then peaking to see if the confidence intervals contained the truth or not. Oh, and we've added a call to the command **print** so you can see R working. If you think about it, we're doing quite a lot and the whole process will take a little time. By printing a count you know how far R has gotten.

Ultimately, **results** contains a 1 if the confidence interval contained the population parameter and 0 otherwise. Let's have a look.

```
results
sum(results)
```

Question 1. (a) How many of your 100 surveys produced confidence intervals that contained the population parameter? Is this what you expected? Why? (b) Define what we mean by a confidence interval; feel free to use this experiment as a reference.

3. A comparison

In the last lecture we discussed an analytical approach to computing confidence intervals. We began with the fact that for the mean, we have a formula for its standard error. Recall that the standard error of an estimate (like the sample mean), is defined to be the standard deviation of its sampling distribution. The narrower the sampling distribution, the more precise our estimate. If our sampling distribution is roughly normal looking, then we can form confidence intervals by adding and subtracting two standard errors from the sample mean. (To be precise, as that is the theme of this lab, we should add and subtract 1.96 times the standard error; technically `pnorm(-1.96)` is 0.025 and `1-pnorm(1.96)` is 0.025. The values of `pnorm(-2)` and `1-pnorm(2)` are 0.0228.)

In a previous lab, we estimated the standard error of the sample mean using our bootstrap replicates. Specifically, we computed something like the following.

```
y = survey(200,people)
boots = bootstrap(y,5000,mean)

seboot = sd(boots$thetastar)
```

Again, our 5000 bootstrap replicates represent an estimate of the sampling distribution and so we can estimate the standard error with their standard deviation.

In class we found that the sample mean, as an estimate, was fairly special in that we have a formula that describes exactly its standard error. With a little math, we can show that the standard error of the sample mean is σ/\sqrt{n} where σ is the standard deviation of the population.

This suggests that another route to estimating the standard error would be to estimate σ/\sqrt{n} . In code this might look like the following.

```
seformula = sd(y)/sqrt(200)
```

Now, have a look at the two.

```
seboot  
seformula
```

The two should be very very close. This is by design. When a classical formula exists, the bootstrap tends to match it.

Finally, we will use this standard error to form a 95% confidence interval “by the book”, meaning with a standard formula. Recall from class that the appropriate multiplier for a 95% confidence interval is 1.96. We will use this in the simulation below. (If we appeal to Student’s t distribution, with a sample of size 200, the multiplier is 1.97 – close enough for the purpose of this lab. We’re not quite up to speed on how to use Student’s results yet, so we are assuming our sample size is big enough.)

Here is the code to now form and examine the Central Limit Theorem formula for a 95% confidence interval. Give it a go!

```
results = rep(0,100)  
  
for(i in 1:100)  
{  
  
  y = survey(200,people)  
  
  lo = mean(y) - 1.96*sd(y)/sqrt(200)  
  hi = mean(y) + 1.96*sd(y)/sqrt(200)  
  
  results[i] = contains(lo,hi,mean(people))  
  
  print(i)  
}
```

Question 2. (a) How many of your 100 surveys produced confidence intervals that contained the population parameter? Is this what you expected? Why? (b) What is the first thing that struck you about running this code as compared to the code involving the bootstrap? What explains the difference?

4. The median

Finally, we will consider a statistic not covered in our book. Given the skew in the data, it is often common for people to report median household incomes rather than means. In this last part of the lab, we will consider the performance of the bootstrap confidence intervals for the population median. We will conduct surveys of size 100 this time, and our subjects will be drawn from the `households` data set.

Below we present the code. It is virtually the same as the code for the means, with a couple slight alterations to reflect the fact that we are working with the income data now and our surveys only involve 100 households.

```
results = rep(0,100)

for(i in 1:100)
{

  y = survey(100,households)
  boots = bootstrap(y,5000,median)

  lo = quantile(boots$thetastar,0.025)
  hi = quantile(boots$thetastar,0.975)

  results[i] = contains(lo,hi,median(households))

  print(i)
}
```

Question 3. (a) How many of your 100 surveys produced confidence intervals that contained the population parameter? Is this what you expected? Why? (b) Look at `lo` and `hi` (these are the values from the last simulation); interpret what the interval `[lo,hi]` means.