Exercise #6: Coin flipping and the bootstrap

Patrick Applegate, patrick.applegate@psu.edu; Klaus Keller, klaus@psu.edu

Learning Goals

After completing this exercise, you should be able to

- perform a simple bootstrap analysis
- use the `sample()` command to both generate series of coin flips and bootstrap replicates from existing data sets

Introduction

In the earth sciences, we often have to make inferences with limited data. This problem can be especially acute in climate science, where observational records of temperature and sea level rise (for example) go back a few hundred years at most. Given these limited data, how can we make inferences about the past and future of the climate system, while accurately representing our uncertainties?

A statistical procedure called the *bootstrap* provides a means of assessing how sure we can be about our answers given available data. The bootstrap involves generating "replicates" of the original data set by sampling, with replacement, from the original data set, and calculating the statistic of interest from each of the replicates. The distribution of this statistic is then a guide to our uncertainty in the statistic as estimated from the original data set.

In this exercise, we'll examine the application of the bootstrap to coin flipping. Coin flips provide a simple, easily-simulated system for thinking about probability; we will generalize the lessons from coin flips to sea level data in the next exercise.

Tutorial

Suppose we have a fair coin, one in which the ratio of heads to the total number of flips $c = 0.5$ over the long term. If we flip this coin many times and calculate our *estimate* of c based on the data we have after each flip, we will get something like Figure 1.

```
# Perform a simulation of successive estimates of c, the ratio of heads to the
# total number of coin flips, for a fair coin.

# Set some values.
true.c <- 0.5          # the true ratio of heads to total flips (0.5 indicates
                       # a fair coin)
n.flips <- 5* 10^ 2    # how many coin flips to perform

# Perform the random sampling and calculate the estimates of c.  1, heads;
# 0, tails.
set.seed(1)
ht <- sample(x = c(0, 1), size = n.flips, replace = TRUE,
             prob = c(true.c, 1- true.c))
flips <- seq(from = 1, to = n.flips, by = 1)
```

```r
c.ests <- cumsum(ht)/ flips

# Plot up the results.
plot(flips, c.ests, type = 'l', col = 'blue', log = 'x',
     xlab = 'Number of flips carried out (log scale)', ylab = 'Estimates of c')
abline(h = true.c, lty = 2)
```

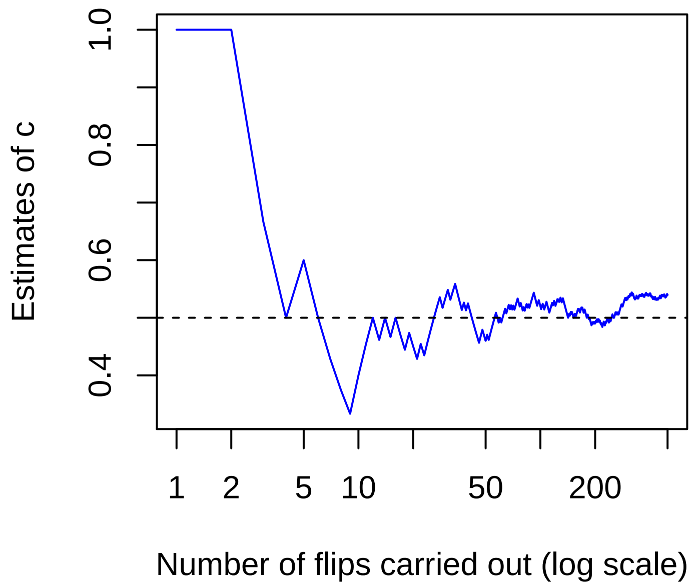

Figure 1: Successive estimates of c (blue curve) after varying numbers of flips with a fair coin that has a true value of $c = 0.5$ (black, dashed line).

Looking at Figure 1, we can see that our c estimates only settle down close to the true value after a few tens of flips. Estimates of c based on smaller numbers of coin flips can be quite inaccurate.

Generating bootstrap replicates from a vector of coin flips

Now assume that we have just ten coin flips. If this data set consists of five heads and five tails, our best estimate of c is 0.5:

$c_{est} = n_{heads}/n_{flips} = 5/10 = 0.5$

The following code block stores the data in `ht.data`, generates just one bootstrap replicate and stores it in `ht.boot`, and calculates the c values associated with the data and the bootstrap replicate. It then prints the bootstrap replicate and its associated c value to the screen.

```r
# Set up a vector of coin flips that represents the data.  1, heads; 0, tails.
ht.data <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1)

# Calculate the value of c based on ht.data.
c.data <- sum(ht.data)/ length(ht.data)

# Generate one bootstrap replicate based on ht.data and calculate the
# associated estimate of c.
set.seed(1)
ht.boot <- sample(x = ht.data, size = length(ht.data), replace = TRUE,
                  prob = NULL)
c.boot <- sum(ht.boot)/ length(ht.boot)

# Print the bootstrap replicate and the c estimate to the screen.
paste('ht.boot =', paste(ht.boot, collapse = ' '))
paste('c.boot =', c.boot)
```

```
## [1] "ht.boot = 0 0 1 1 0 1 1 1 1 0"
## [1] "c.boot = 0.6"
```

In this case, we get a bootstrap replicate in `ht.boot` that contains six heads (`1`) and four tails (`0`), and therefore has a `c.boot` of 0.6.

Examine the `help()` file for `sample()`; can you see why the arguments `x`, `size` and `prob` differ between the two code blocks above? Note that we used `sample()` to accomplish two different tasks:

- In the first code block, we used `sample()` to generate a sequence of coin flips from a coin with a known c.
- In the second code block, we used the same function to generate a single bootstrap realization by random sampling, with replacement, from a data vector.

Exercise

Part 1. Using the second code block above and what you learned about `for` loops in Exercise #2, write an R script that

- generates `n.boot <- 10^3` bootstrap replicates of the ten flips stored in `ht.data`, above
- calculates a c value for each bootstrap replicate, and stores these c values in a vector `c.vals` (you should define this vector *before* the `for` loop and use the loop's counter variable to save each new value of c in a different element of `c.vals`)

```
c.vals <- rep(NA, n.boot)
for (i in 1: n.boot) {
  # one or more commands for generating the bootstrap replicates go here
  c.vals[i] <- # calculate your c values on the right-hand side here
}
```

- makes a histogram of these c values and plots `c.data` on top of the histogram as a vertical, red line (this line should coincide with the peak of the histogram)
- calculates the 95% range of `c.vals` using `quantile(c.vals, probs = c(0.025, 0.975))`.

Part 2. Save a copy of your script from Part 1 using a different file name. Incorporate code from the first code block above into this new script so that it generates a synthetic data set of a given length and `true.c`, and then performs the bootstrap analysis from Part 1.

Questions

1. What is the 95% range of `c.vals` in Part 1?

2. `source()` your code from Part 2 repeatedly using a true c value of 0.5 and a length for the original data set of 10 flips. How does the histogram change depending on the number of heads in the original data set?
3. Again using your code from Part 2, generate synthetic data sets with lengths of 100 and 1000 coin flips and repeat the bootstrap analysis each time. Use a true c value of 0.5. How do the histograms and 95% ranges in these experiments compare to that from Question 1?
4. Set your code from Part 2 to use 1000 coin flips and change the true c value to 0.3 and 0.8. Compare the histograms and 95% ranges in these experiments to those from Question 3.