STAT 113: Lab 5

Randomization Tests In R

Last revised October 27, 2017

Overview

The goal of this lab is to gain additional practice with the logic of hypothesis testing, and also to see how to carry out randomization tests in R. You may be wondering why it is necessary to be able to do tests like this in R, when we already know how to do them in StatKey. One real-world practical reason is that by carrying out the test in R, we have an exact record of what we did (we can create a reproducible report using Markdown, or at least a script file, for example.) A pedagogical reason is that doing the same thing two or three different ways should hopefully highlight the fundamental concepts as abstracted away from a particular procedure.

What to Turn In

The exercises labeled "Homework" should be written up in a Markdown document and turned in by the start of lab on Friday 11/3. Your writeup should adopt best practices for statistical analysis: Visualize the data before doing any inference, document in text what you are doing (make your code chunks small and include descriptive text between them), and always interpret your results in the real context of the problem.

A Note on "I" SLOs: Please note that the I and J SLOs are treated differently than the A-G SLOs. Their objective is to get you to be comfortable taking a dataset and a question and doing all phases of analysis, from importing the data, visualizing it, estimating parameters, carrying out tests, and interpreting results in context. The associated problems are thus more open-ended than most we have had. In addition, since these objectives do not lend themselves to quizzes, the grade for them is based *primarily* on homework.

Before you start, make sure `mosaic` and `Lock5Data` are loaded.

```
library("mosaic")
library("Lock5Data")
```

1 Testing a Single Proportion

Let's walk through how you would do the test for the "Love is Blind" data in R.

Recall the setup: 18 men in relationships with women attempted to guess whether a woman was his partner by touching the back of her hand while blindfolded. In the study, 8 of 18 did so correctly. We defined p to be the "long-run" success rate in the population of "men in relationships with women", and defined the hypotheses:

$$H_0 : p = 1/3$$
$$H_1 : p > 1/3$$

where H_0 represents the hypothesis that the men are no better than chance at this task, and H_1 says that they are better than chance.

There's no dataset to read in here, since we just have a single proportion to represent the entire data set. Let's store that "observed" value in a variable. (We don't have to do this; we could just put in the number directly in subsequent code, but by defining it up front, we can change it in just one place later if our data should change.)

```
obs.prop <- 8/18
```

To simulate random guessing, we can use the `rflip()` function, which simulates flipping a "weighted" coin some number of times, and records the number and proportion of heads and tails.

We'll do one random simulation of random guessing first to see how this works.

```
rflip(n = 18, prob = 1/3)
```

We don't want to do just one simulation of the study; we want to do many, so that we can see how likely we are to get a result of 8 or more correct *assuming random* *guessing*. We can use the `do()` function to do this, just as we did to create sampling distributions and bootstrap distributions.

```
## Remember that in Markdown it can be a good idea to put
## comuptationally intensive commands (like repeating something many times)
## in chunks by themselves and use the cache = TRUE option so that it does
## not redo it every time you Knit the document
Random.props <- do(5000) * rflip(n = 18, prob = 1/3)
```

The `Random.props` object is a data frame. Use the `head()` function to see what variables it contains.

Here's code to produce a dot plot of the results, highlighting the values that lie beyond the observed difference.

```
## We'll highlight the region corresponding to the P-value,
## using the observed difference we calculated above as the
## threshold.
dotPlot(~prop, data = Random.props,
        groups = prop >= obs.prop,
        width = 1/18, cex = 10)
## width= controls the "bin width" and cex= is a size multiplier
## for the dots to make them easier to see
```

Now that we've visualized the area corresponding to the P-value, let's calculate the actual number. Remember that the P-value in this case is a "proportion of proportions"; the proportion of the simulations that yielded a sample proportion of $8/18$ or greater.

We can get proportions of cases (here a case is really one random simulation) that meet some condition with a command of the form

```
### Not a real command; just an illustration of the form
prop(~(CONDITION), data = DataSet)
```

where `CONDITION` is the same sort of expression we use to `filter()` a dataset; but here it calculates the proportion of cases that meet the condition. In this case we want the proportion of "cases" in the simulation for which `prop > obs.prob`. Confusingly, `prop` is both the name of the function that computes proportions and the name of the variable in the `Random.props` data frame.

```
## not strictly needed, but improves readability to add
## parentheses around the condition
P.value <- prop(~(prop >= obs.prop), data = Random.props)
P.value
```

What is the conclusion?

2 Simulating Random Assignment Into Two Groups

Now let's walk through how to do the randomization procedure for the Penguin study in R.

The data is on my website, under `/data/Penguins.csv`.

```
Penguins <- read.file("http://colindawson.net/data/Penguins.csv")
```

The explanatory variable is called `Band` and the response variable is called `Survived` (both binary).

Use the `head()` function to verify that the data was read in correctly and has the expected format.

We can get counts of each of the four combinations (Metal band survived, Metal band didn't survive, Control survived, Control didn't survive) using `tally()`

```
tally(Survived ~ Band, data = Penguins)
```

Much like we get means of a numeric variable by group using the form

```
## Not actual code to run; just an example of a command form
mean(Response ~ Group, data = DataSet)
```

we can get proportions associated with a binary response variable using the form

```
prop((CONDITION) ~ Group, data = DataSet)
```

where `CONDITION` is again same sort of expression we use to `filter()` a dataset; but here it calculates the proportion of cases that meet the condition *within each group* (i.e., for cases with each different value of the variable `Group`). In this case we want the proportion of cases for which `Survived == "Yes"`.

```
prop((Survived == "Yes") ~ Band, data = Penguins)
```

Do the calculation of the proportions on your own based on the output of `tally()` and verify that they match the results produced by `prop()`.

We can also get the difference in these proportions using `diffprop()`.

```
diffprop((Survived == "Yes") ~ Band, data = Penguins)
```

Note that this is the second proportion minus the first, which is the opposite of what we did in class. If you prefer, you can put a minus sign in front to make the difference positive. Just make sure you are doing the subtraction consistently throughout.

This is the "observed difference"; that is, the real difference in sample proportions from the data. Let's save this value as a named variable, since we will need it.

```
## Can call the variable whatever you want
obs.diff <- diffprop((Survived == "Yes") ~ Band, data = Penguins)
```

What we want now is to find a P-value associated with the likelihood of getting a result this large by the process of random assignment to groups alone. We want to simulate "shuffling" the penguins into two random groups, and computing the difference in survival proportions each time.

We can get one random shuffling by applying the `shuffle()` function to the grouping variable. This is like taking the values of the `Survive` variable and putting them in two random piles.

```
prop((Survived == "Yes") ~ shuffle(Band), data = Penguins)
diffprop((Survived == "Yes") ~ shuffle(Band), data = Penguins)
## Note that the difference will not match the proportions, since
## we are doing a new shuffling in the second line.
```

We have now computed the difference in proportions from *one* random reshuffling.

Before you read on, try to anticipate what is going to come next, both at a conceptual level and on a code level.

As before, we can use the `do()` function to repeat this process many times.

```
## cache = TRUE
## The line break is just for readability
Random.diffs <- do(5000) *
    diffprop((Survived == "Yes") ~ shuffle(Band), data = Penguins)
```

Look at the randomization data frame to see what the variable is called.

Here's the code again to produce a dot plot of the results, highlighting the values that lie beyond the observed difference.

```
## I'm highlighting a two-tailed region here by using the abs() function to
## find those simulated differences in proportions that are greater
## *in absolute value* than the observed difference
dotPlot(~diffprop, data = Random.diffs,
        groups = (abs(diffprop) >= abs(obs.diff)),
        width = 1/50, cex = 12)
## This time width = 1/50, since this is the smallest observable
## difference of proportions in any sample where each group has 50 cases.
```

See the teeny tiny pink regions on the extremes? Let's finally compute the *P*-value:

```
## This is a two-tailed P-value due to the absolute value
P.value <- prop(~(abs(diffprop) >= abs(obs.diff)),
                data = Random.diffs)
P.value
```

And we're done.

What do we conclude?

3 Testing a Difference Between Group Means

If we are comparing means instead of proportions, the only things that change conceptually, apart from our hypotheses, are that we compute a difference of means from the real data and from each "scrambled" dataset, rather than a difference in proportions.

In terms of the R code, we replace `prop()` and `diffprop()` with `mean()` and `diffmean()`.

Let's look at the data from a study investigating the effect of either sleep or caffeine consumption on working memory capacity. The data comes from participants who either slept 90 minutes or took a caffeine pill, and then had to recall as many words as they could. We are interested in testing whether mean recall is different depending on which condition participats are in. Our hypotheses are, therefore:

$$H_0 : \mu_{\mathsf{sleep}} - \mu_{\mathsf{caffeine}} = 0$$
$$H_1 : \mu_{\mathsf{sleep}} - \mu_{\mathsf{caffeine}} \neq 0$$

where μ_{sleep} represents the mean number of words that would be recalled from the list if everyone in the population took a 90 minute nap first, etc.

If we assume H_0 is true, then we can assume that each participant would have had the same recall regardless of which condition they are in. Just like the penguins, we can simulate this by randomly and repeatedly re-splitting the cases into two groups. The only difference between this test and that one is that here we are interested in computing the difference between the group means, rather than the difference in proportions. Everything else is the same.

Before doing the formal test, let's plot the data and compute some summary statistics. (It is always a good idea to do this.)

```r
data("SleepCaffeine") # in Lock5Data
## Side-by-side boxplots and paneled dot plots are a useful
## way to visualize the distributions by group
bwplot(Words ~ Group, data = SleepCaffeine)
dotPlot(Words ~ Group, data = SleepCaffeine)
```

Now, some summary statistics.

```r
mean(Words ~ Group, data = SleepCaffeine) #look at the means
obs.diff <- diffmean(~Words | Group, data = SleepCaffeine)
```

We want to know whether this difference in sample means is larger than we would expect to see by random assignment alone, assuming the recall would be the same on average in the population.

We will build up a randomization distribution in the same way as we did for the penguin data.

```r
Random.diffs <- do(5000) * diffmean(Words ~ shuffle(Group), data = SleepCaffeine)
```

Look at the `Random.diffs` data frame to confirm the name of the variable.

Let's plot the randomization distribution and highlight those statistics that are more extreme than the one we observed (this is again a two-tailed test, so we compare absolute values).

```r
dotPlot(~diffmean, data = Random.diffs,
        groups = abs(diffmean) >= abs(obs.diff),
        width = 0.01, cex = 5)
```

```r
P.value <- prop(~(abs(diffmean) >= abs(obs.diff)), data = Random.diffs)
P.value
```

What is our conclusion?

Homework

1. (I1) In a sample of 120 soccer matches played in the Football Association (FA) premier league in Great Britain, the home team won 70 times. How strong is the evidence for a home field advantage in this league? Compute a P-value and also construct a bootstrap 95% confidence interval to estimate the "long run" home winning rate in this league. You can use the `rflip()` function to generate simulated data, based either on the null parameter value or based on a "population" that looks like the sample (note that this is equivalent to sampling with replacement from a dataset with a binary variable).
2. (I2) This problem is adapted from Exercise B.18 in the textbook.

   Studies suggest that when young men interact with a woman who is in the fertile period of her menstrual cycle, they pick up subconsciously on subtle changes in her skin tone, voice, and scent. A new study suggests that they may even change their speech patterns. The experiment included 123 male and 5 female college students, all of them heterosexual. The men were randomly divided into two groups with one group paired with a woman in the fertile phase of her cycle and the other group with a woman in a different stage of her cycle. The women were used equally in the two different stages. For the men paired with a less fertile woman, 38 of the 61 men copied their partner's sentence construction in a task to describe an object. For the men paired with a woman at peak fertility, 30 of the 62 men copied their partner's sentence construction. The experimenters hypothesized that men might be less likely to copy their partner during peak fertility in a (subconscious) attempt to attract more attention to themselves. The data is on my website in the usual place as `Fertility.csv`. Visualize the data using one or more appropriate plots, conduct a hypothesis test to see if the proportion copying sentence structure is different when the woman is at peak fertility (include all details of the test (hypotheses, P-values, conclusion in context, etc.)), and construct a bootstrap confidence interval to estimate the long run/population difference in copying rates.

3. (I4) A social psychologist wants to investigate the effect of social status on the tendency of individuals to suppress their own opinions in favor of another person's opinion. He sets up an experiment where people are asked to make judgments about an ambiguous image. Afterwards, they discuss their responses with a partner, whom they believe to be another participant in the experiment, but who is actually a confederate of the experimenter. After discussion, the participants are asked whether they want to change their judgments. For half of the participants, the partner is a "high status" individual; for the other half, the partner is "low status". Each participant encountered 40 disagreements. Data about their responses is in the dataset `Moore` in the `car` package (load the package first, then the dataset).[1]. The number of times out of 40 that the participant changed their mind is recorded in the `conformity` variable; the status of the partner is recorded in the `partner.status` variable. Visualize the data, set up and test hypotheses to investigate whether the mean rate of conformity (measured by number of times out of 40 that participants will change their mind) differs depending on the status level of the partner, including all details of the test, and compute and interpret a bootstrap confidence interval for the parameter of interest.

[1]The results are published in Moore, J. C., Jr. and Krupat, E. (1971), Relationship between source status, authoritarianism and conformity in a social etting. *Sociometry* **34**, 122-134