Is Free Beer Good For Tips?

Goal

The goal of this project is to test a hypothesis using a variety of techniques: “eyeball” test, t-test, and bootstrapping. Use filename `stats/hyp.py`.

Discussion

Here is a typical statistics question (derived from one by Jeff "The Hammer" Hamrick) that we will solve in multiple ways.

Q. *Psychologists studied the size of the tip in a restaurant when the waitron gave the patron a free beer. Here are tips from 20 patrons, measured in percent of the total bill: 20.8, 18.7, 19.1, 20.6, 21.9, 20.4, 22.8, 21.9, 21.2, 20.3, 21.9, 18.3, 21.0, 20.3, 19.2, 20.2, 21.1, 22.1, 21.0, and 21.7. Does a beer-inspired tip exceed 20 percent or perhaps dip below 20 percent (maybe patrons get drunk and can't do math)? Use a significance level equal to* $\alpha = 0.06$.

Side note: Always pick the significance level before you run your experiment. It is really bad mojo to pick your significance after you know what the p-value is.

Before starting on this, let's interpret that question: It asks whether the mean of the specified sample differs significantly from the usual 20% tip. By “significantly” we refer to the likelihood that the usual population (with mean 20.0) could yield a sample with the observed sample mean. By “usual” we mean our control of approximately: $N(20.0, s^2/n)$ where s is the sample variance of the sample tips and $n = len(tips)$. (We can reasonably assume that tips follow a normal distribution.)

While the population mean is 20.0, the means of any resamples we take will bounce around left and right of 20.0. The question is: does this particular test sample's mean, $m = 20.725$, fall outside of the typical variability of the sample means?

More formally, we would say the following: The **null hypothesis** is that the mean for the specified sample does not differ significantly from $\mu = 20.0$. I think of this as the *control* in my experiment. The **alternate hypothesis** is that the sample mean differs significantly above or below the population mean. Formally,

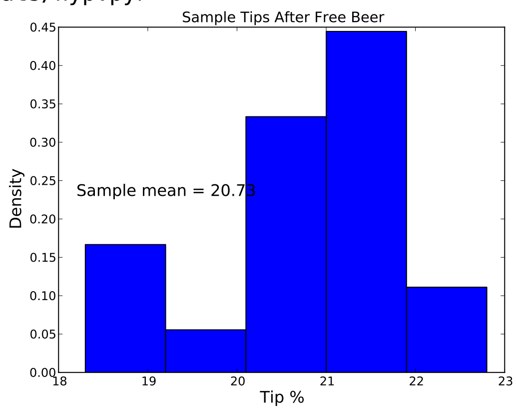
$H_0 : m = 20.0$ (non-free beer situation)
$H_1 : m \neq 20.0$ (free beer situation; two-sided alternative hypothesis)

We could also say that $H_0 : m - \mu = 0$ and $H_1 : |m - \mu| > 0$.

Steps

Eyeballing it

1. First, just draw a histogram of the tips to see what it looks like. For this exercise, create a file called `stats/hyp.py`.

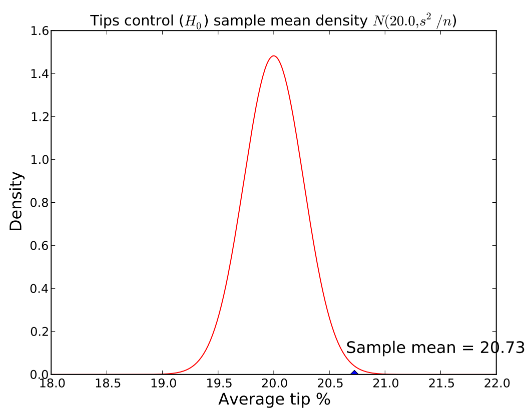


For your convenience, here are the tips in python format:

```
tips = [20.8, 18.7, 19.1, 20.6, 21.9, 20.4, 22.8,
        21.9, 21.2, 20.3, 21.9, 18.3, 21.0, 20.3,
        19.2, 20.2, 21.1, 22.1, 21.0, 21.7]
```

(Use your awesome new skills from previous labs to generate the histogram.) To me, there is a lot of "mass" to the right of the usual 20% tip but my eyeball is not a rigorous significance mechanism.

2. To get a better idea, let's simply plot the distribution of the sample means given our H_0 assumption: $N(20.0, s^2/n)$. We need to use the sample variance s^2 from our test sample because we don't know the variance of the original distribution. It safe to assume that the variance is similar. This is our "control" or the usual tipping distribution: the distribution of the set of average tips per day if H_0, the control, is true.



Looking at that graph, it seems that a sample mean of 20.73 is pretty far in the right tail of a normal curve centered at the control average 20% tip. It looks to be a few standard deviations away from the mean. My gut says that it's pretty likely that giving people a free beer increases tips significantly.

t-test

1. Let's use a *t-test* now to test for significance, just like we would do in statistics class. The t value measures the number of standard deviations a sample mean, m, is away from our presumed population mean μ:

$$t = \frac{m - \mu}{s/\sqrt{N}} \qquad \text{(t-value)}$$

It's just the difference between the means scaled to be in units of standard deviations. Write some code to compute the t-value. When computing s, the sample standard deviation, note that the numpy std() function returns a biased estimate of the standard deviation. Use `np.std(tips, ddof=1)` instead of just `std(tips)`.

2. Print out the value of t. I get $t = 2.69417199392$. That means that m is about 2.7 standard deviations away from μ, which is a very significant departure.

3. To get a p-value, likelihood that we would see such a t value in the nonfree beer situation, look up t in a t-distribution c.d.f. using `1-scipy.stats.t.cdf(t,N-1)`. You should get 0.0071844. Since we need to check both tails, the probability is actually $2x$ that, or, p-value=0.014369 (1.4%). The definition of significance is $\alpha = 0.06$, which means that our sample mean is definitely significant since $1.4\% < 6\%$. There is only a 1.4% chance that the control could generate a value that extreme or beyond.

We must conclude that m differs significantly from $\mu = 20.0$ based upon the significance of $\alpha = 0.06$ and, therefore, we reject H_0 in favor of H_1. Giving out free beers is extremely likely to have increased the average tip in that experiment.

Boostrapping for empirical hypothesis testing

Ok, now, let's use bootstrapping to estimate a *p-value*. A p-value for some point statistic or value is the probability that the control (null hypothesis H_0) could generate that statistic or value. In our case, a p-value can tell us the likelihood that a normal distribution centered around $\mu = 20.0$ with $s^2 = var(tips)$ could generate a sample mean of 20.725. (We approximate the population variance with our sample variance.) *Note and we are sampling from $N(\mu = 20.0, s^2)$ to conjure up samples from the control situation. We are not resampling from the tips list as we are trying to see how the observed sample mean, 20.725, fits within the control distribution not the test distribution. We are also not generating samples from the distribution of a mean random variable, $N(\mu = 20.0, s^2/n)$.*

1. Bootstrap TRIALS=5000 samples of size $n = len(tips)$ from $N(\mu, s^2)$ using your `rnorm()` function created in a previous lab. It's very important that we use the same sample size as $len(tips)$ so we are comparing the same thing. Compute the mean of each sample, X, an add to $\overline{X}$ as you generate samples from the normal distribution.

2. Compute how many means in $\overline{X}$ are greater than or equal to mean(tips):

```
greater = np.sum(X_ >= np.mean(tips))
```

or

```
greater = sum([x>=np.mean(tips) for x in X_]) # the number of true values
```

3. The (one-sided) p-value is just the ratio of values above the observed mean, mean(tips), to the number of trials. Double that because we're doing a two-sided test. With 5000 trials, I see just 13 values greater than $m = 20.725$. That gives us a p-value of $2 * \frac{13}{5000} = 0.0052$ or .52%. That means that, empirically, we find that there is an extremely small probability that the control could generate an extreme value like $m = 20.725$. Certainly the likelihood is less than the required 6% significant value.

Note: we would expect the empirical p-value (.52%) and the p-value derived from the t-test (1.4%) to be very close to each other when the number of trials is large with bootstrapping. Our resident statistician, Jeff Hamrick, explains that the difference is not a problem with our bootstrapping solution and is ok.

"A student t distribution with dof=19, is pretty close to a normal. But the differences are most greatly felt in the tails, and we're in the tails (rejection H_0), thus casting a little bit of sketchiness or your choice to draw the simulated raw data from a normal random variable. If we were performing this exact same operation on a data set with reasonably large size (say, 40 or 50 or 75) the differences would still exist but would be even more minute."

Again, we easily reject the control and conclude that giving out free beers increases tips.

Deliverables. `stats/hyp.py` and a text file call `stats/hyp_results.txt` that gives your t-value and p-value from the t-test. Also give your empirical p-value from bootstrapping with $TRIALS = 5000$. Tag when completed with all stats-related exercises with `stats`.