Learn

Part 0: Introduction to Power

Suppose you are interested in testing if on average a bottle of coke weighs 20.4 ounces. You have collected simple random samples of 130 bottles of coke and weighed them.

1. Load the data in with `np.loadtxt('data/coke_weights.txt')`.

```
import numpy as np

coke_weights =
np.loadtxt('data/coke_weights.txt')
```

2. State your null and alternative hypothesis.

```
H0: mu = 20.4
H1: mu !=
20.4
```

3. Compute the mean and standard error of the sample. State why you are able to apply the Central Limit Theorem here to approximate the sampling distribution to be normal.

```
def sample_sd(arr):
  return np.sqrt(np.sum((arr - np.mean(arr)) ** 2) / (len(arr) -
1))

def standard_error(arr):
  return sample_sd(arr) / np.sqrt(len(arr))

m = coke_weights.mean()
se = standard_error(coke_weights)
print 'The sample mean is', m
print 'The standard error is', se
```

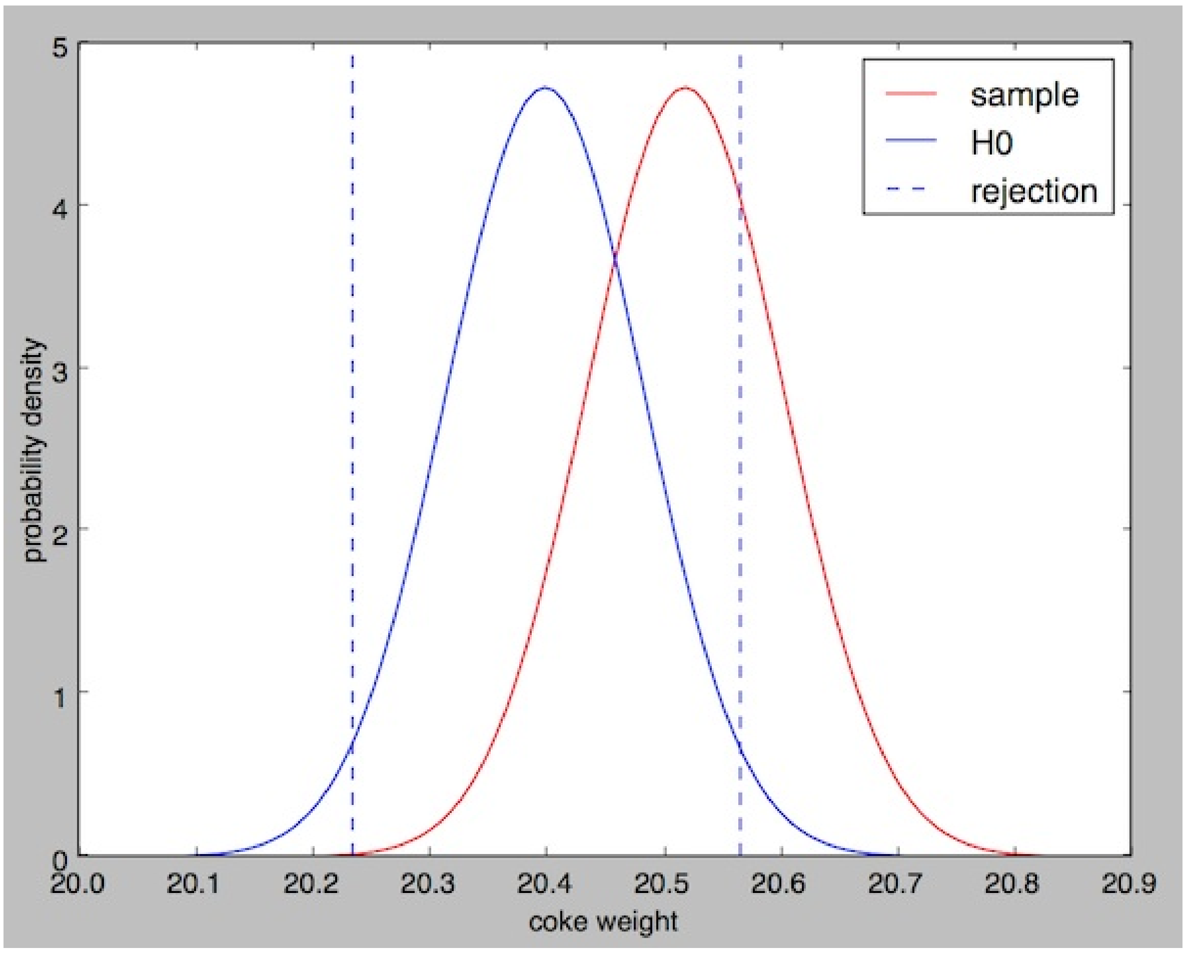
The CLT applies here because the sample size is large enough, where **the mean** of a large enough sample (> 30) would approximate to a normal distribution.

4. Use `scipy` to make a random variable of the sampling distribution. Plot the PDF over `+/- 4 standard error`. Make another random variable of the sampling distribution formulated by the null hypothesis (null distribution). Use the standard error from the sample as an estimator for the standard error of the null distribution. Plot the PDF of the second random variable on the same plot.

```
import scipy.stats as scs

def plot_dist(m ,se, c, label, plot=True):
  rv = scs.norm(m, se)
  x_range = np.linspace( m - (4 * se), m + (4 * se),
1000)
  y = rv.pdf(x_range)
  if plot:
     plt.plot(x_range, y, c=c, label=label)

     plt.axvline(x=m, c=c, linestyle='--')
  return rv
```



5. Plot vertical lines to indicate the bounds for rejecting the null hypothesis assuming a significance level of 0.05. Based on the bounds for rejecting the null, what conclusion can we draw based on the sample of 130 bottles of coke?

```
See vertical lines on image above. Code:

We would fail to reject the null hypothesis in this case, as the mean of
our
sampling distribution falls within these bounds.
```

```
mean = coke_weights.mean()
sem = scs.sem(coke_weights)

bounds = np.linspace(20.4- 4*sem, mean+4*sem,100)

reject_low = 20.4 - 1.96*sem
reject_high = 20.4 + 1.96*sem

CI_low = mean - 1.96*sem
CI_high = mean + 1.96*sem

def plot_part0(bounds, mean, sem, reject_low, reject_high, CI_low, CI_high):
  plt.plot(bounds,scs.norm.pdf(bounds,loc=mean,scale=sem), label='sample',
color='r')
  plt.plot(bounds,scs.norm.pdf(bounds,loc=20.4,scale=sem), label='H0')
  plt.axvline(reject_low, color='b', linestyle='--', label='rejection')
  plt.axvline(reject_high, color='b', linestyle='--')


  plt.ylabel('probability density')
  plt.xlabel('coke weight')
  plt.legend()
  plt.show()
```

6. Build a 95% confidence interval based on the sample data. Does your interval suggest that the weight of a bottle of coke is different than 20.4 ounces? Explain what a false negative is in the context of this problem.

```
def plot_ci(m, se, c, plot=True, ci=0.95):
  z = scs.norm.ppf(ci + (1 - ci) / 2)
  lower_ci = m - z * se
  upper_ci = m + z * se
  rv = scs.norm(m, se)
  x_range = np.linspace( m - (4 * se), m + (4 * se),
1000)
  y = rv.pdf(x_range)
  if plot:
     plt.plot(x_range, y, c=c)
     plt.axvline(x=lower_ci, c=c, alpha=.3,
linestyle=':')
     plt.axvline(x=upper_ci, c=c, alpha=.3,
linestyle=':')
     plt.show()
  return lower_ci, upper_ci


plot_ci(np.mean(coke_weights),
np.std(coke_weights)/np.sqrt(len(coke_weights)), 'blue')


- Based on the 95% CI we conclude that the weight of a bottle of coke on
average is not different
 from 20.4, i.e. we cannot reject the null that the weight is 20.4

A false negative in this test is when the mean weight of the coke bottles
is actually different from 20.4, but we fail to reject the null hypothesis.
```

7. Under the null specified in part 2, using a 5% type I error rate, and considering the true mean being equal

to the one found in our sample; compute the power of the test. Explain what power means in the context of this problem.

```
def calc_power(data, null_mean, ci=0.95):
  m = data.mean()
  se = standard_error(data)
  z1 = scs.norm(null_mean, se).ppf(ci + (1 - ci) / 2)
  z2 = scs.norm(null_mean, se).ppf((1 - ci) / 2)
  return 1 - scs.norm(data.mean(), se).cdf(z1) + scs.norm(data.mean(),
se).cdf(z2)


calc_power(coke_weights, 20.4)
# 0.29549570806327596
```

```
Power, in this context, is the probability of detecting the mean weight of a
bottle of coke is different from 20.4 given the that the weight of a bottle
of
coke is indeed different from 20.4.  In this case, we have a 29.5%
probability
of choosing the alternative correctly if the true mean value is larger by
0.12 ounces.
```

Part 1: Factors that Influence Power of a Test

1. Write a function `explore_power` that includes all the steps in `Part 0`. The input will be the mean value under the null hypothesis (i.e. `20.4` ounce as we have specified above) and the output is the power.

```
def explore_power(data, null_mean, ci=0.95):


  data_mean = np.mean(data)
  data_se = np.std(data, ddof=1) /
np.sqrt(len(data))


  null_norm = scs.norm(null_mean, data_se)
  data_norm = scs.norm(data_mean, data_se)


  reject_low = null_norm.ppf((1 - ci) / 2)
  reject_high = null_norm.ppf(ci + (1 - ci) / 2)


  power_lower = data_norm.cdf(reject_low)
  power_higher = 1 - data_norm.cdf(reject_high)
  power = (power_lower + power_higher) * 100
  return power
```

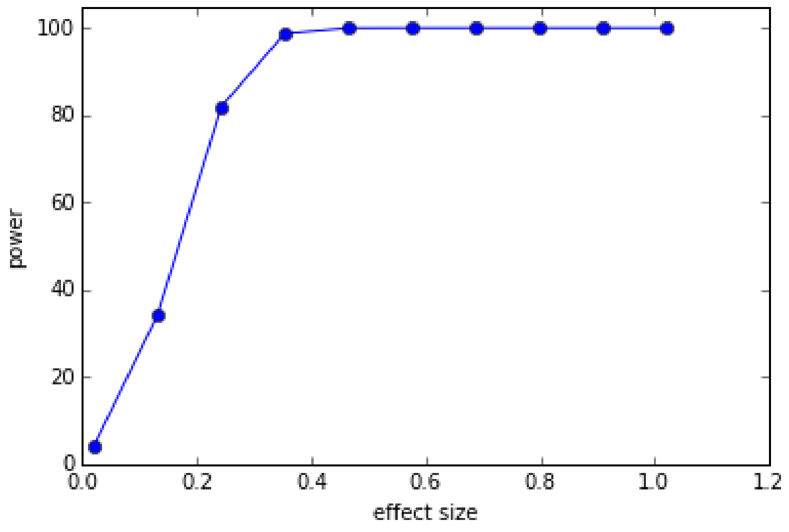
Assume now the null hypothesis is that a bottle of coke weights `20.2` ounces. Run `explore_power()` with the new null hypothesis. Did the power increase or decrease? Explain your observation.

```
Power increased. Changing the null hypothesis here to 20.2 ounces increases
the
effect size, which increases our ability to detect a shift and thus our
power.
Specifically, we now have a power of 96.6%.
```

2. Make a plot of **effect size (x)** against **power (y)**. The effect size is the absolute difference between the value under the null hypothesis and the true value of the mean. Consider the true value of the mean is what we find in our sample (i.e. `20.4` `effect size = 20.519 -` if the null is `20.4`).

```
def plot_es_v_power(data, null_mean):
  effect_sizes = np.linspace(0, 1, 10)
  power_size = [explore_power(data, data.mean() + effect_size) for
effect_size in effect_sizes]
  plt.plot(effect_sizes, power_size, marker='o')
  plt.ylim(0, 105)
  plt.xlim(0, 1.2)
  plt.xlabel('effect size')
  plt.ylabel('power')
  plt.show()

plot_es_v_power(coke_weights, 20.4)
```



3. Without writing any code, explain why the standard error decreases as the sample size increases. Furthermore, extrapolate and explain the relationship between **sample size** and **power**. Verify your result by computing power on a larger dataset with 1000 data points (`numpy.loadtxt('data/coke_weights_1000.txt')`). Is the power higher or lower with a larger sample size given the effect size and significance level held constant?

```
Standard error decreases as the sample size increases because with a larger
and
larger sample, we're better able to judge where the true population mean is
'hiding'.
From a mathematical perspective, sem = std/sqrt(n) and as n increases,
the denominator increases.

This means that the rejection interval (and confidence interval) gets
smaller.
This increases the power of the test, because with the better resolution of
the
intervals comes the ability to detect smaller shifts in the factor of
interest.
```

```
coke_weights_2 = np.loadtxt('./data/coke_weights_1000.txt')

print 'power of smaller sample: ',explore_power(coke_weights,20.4)
print 'power of larger sample: ',explore_power(coke_weights_2,20.4)
```

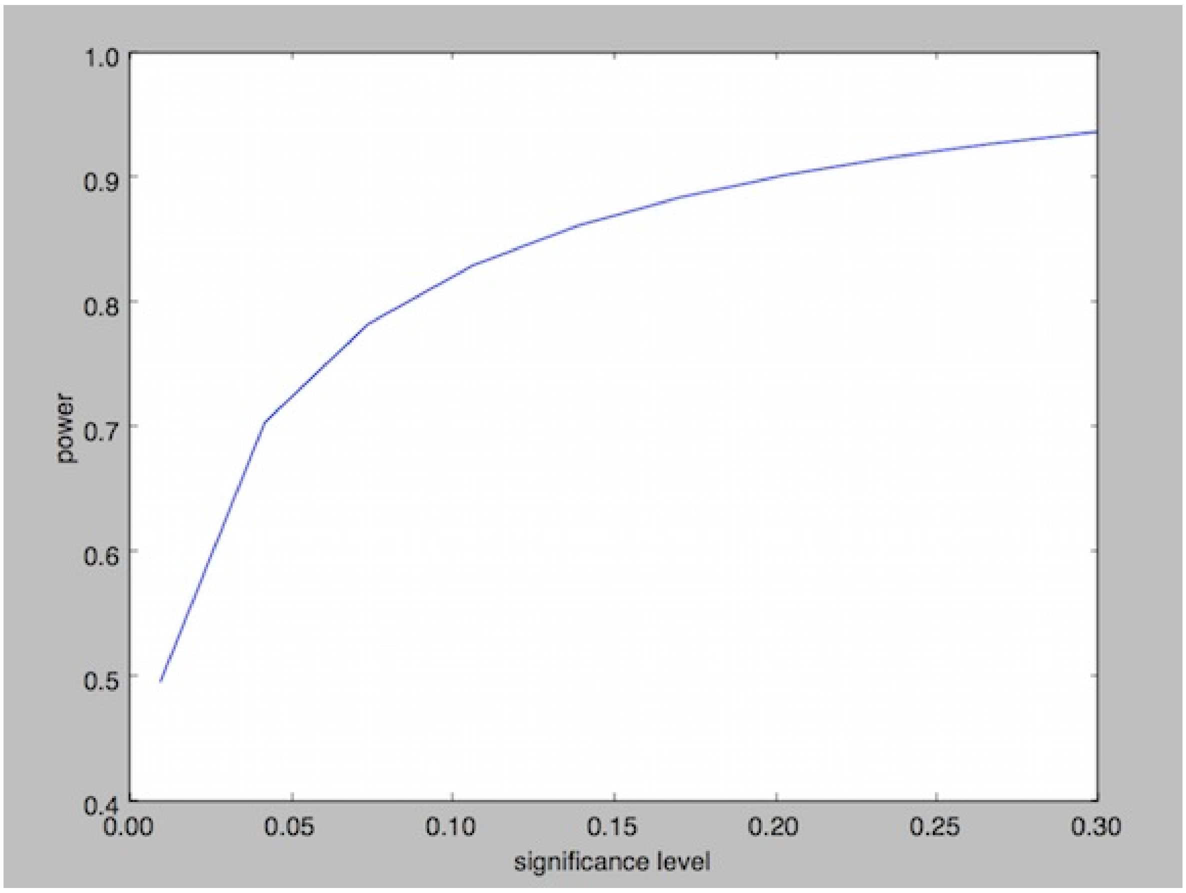
```
As expected, the power level increases with a larger size but with a fixed
alpha
level and effect size.  We are able to better detect differences with smaller
effect sizes
because of the larger sample size.
```

4. How does the power change if the significance level is increased from `0.05` to `0.1`. Explain your observation in terms of the increase/decrease probability of false positive/false negative. Plot **significance level (x)** (over a range of `0.01 - 0.3`) against **power (y)**.

```
print 'power with alpha of 0.1:
',explore_power(coke_weights,20.4,.95)
print 'power with alpha of 0.1:
',explore_power(coke_weights,20.4,.99)

alphavals = np.linspace(0.01,0.3,10)
power_vals = []

def plot_alpha_v_power(data, null_mean, alphavals):
  for val in alphavals:
     newval = explore_power(data, null_mean, 1 - val)
     power_vals.append(newval)
  plt.plot(alphavals,power_vals,'-o')
  plt.ylabel('power')
  plt.xlabel('significance level')
  plt.show()
```



```
As we increase our significance level, the power also increases. If
we
want to reduce the Type I error rate (false positives), we will see
an
increase in the Type II error rate.

In this case, we are allowing for more Type I errors.  We are
providing a decision boundary with greater area in the alternative
space, which in turn provides the pros of higher power along with
the potential cons of more Type I errors.
```

Part 2: Power Calculations for A/B testing

One common problem in A/B testing is to decide when to stop the experiment. Power calculations are very useful in determining the required minimum sample size necessary to reach a certain power (usually 80%), given an effect size and a significance level.

A powerful test would ensure we are able to detect differences in conversion the majority of the time given the difference in fact exists. To gain insights about the effect size, a small-scale pilot experiment is usually launched. The minimum sample size is computed. Subsequently, a full-scale experiment is run until the minimum sample size is reached.

Continuing from yesterday's Etsy case study, get the conversion data for the new and old landing pages with the code below.

```
data = pd.read_csv('data/experiment.csv')
old_data = data[data['landing_page'] == 'old_page']
['converted']
new_data = data[data['landing_page'] == 'new_page']
['converted']
```

Historically, the old page has a conversion of 10% and we wish to test if the new page provides a 0.1% increase (1% lift) in conversion. Recall the null and alternative hypotheses below:

```
# Set X as a random variable which is the (new conversion - old
conversion)
X ~ p_new - p_old

H0: pi_new - pi_old = 0.001
H1: pi_new - pi_old > 0.001
```

Part 2.1: Computing Power for Pilot Sample

In this part, we are going to compute statistical power for the pilot experiment given the null hypothesis.

1. By the CLT, we can approximate the sampling distribution of proportions (`p_new, p_old`) to be normal (since proportions are effectively a mean of 0's and 1's). We can further assume the sampling distribution of `p_new - p_old` to be normal.

   Compute `p_new - p_old` and the standard error from the sample and define a normal distribution random variable. Plot the PDF of the random variable as you have done previously.

   Hint: Standard Error for difference of proportions

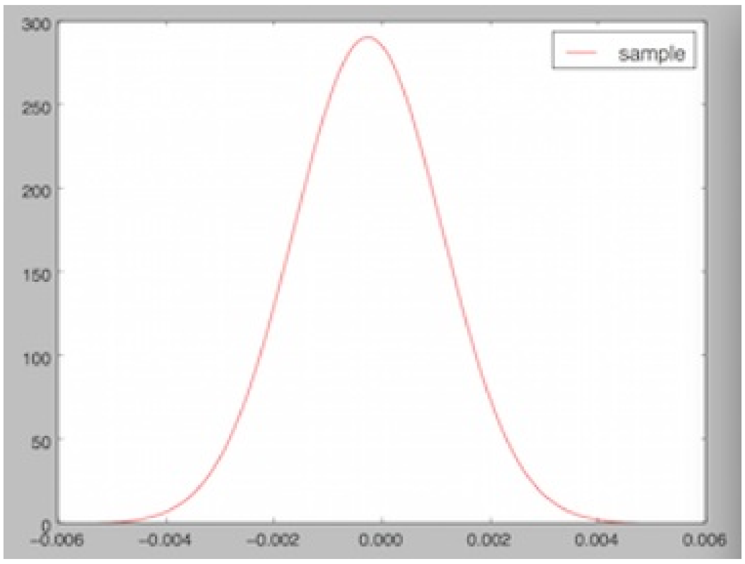
   - `p` is a weighted average of the `p1` and `p2`
   - `n1` is the number of subjects sampled from the first population
   - `n2` is the number of subjects sampled from the second population

```
data = pd.read_csv('data/experiment.csv')
old_data = data[data['landing_page'] == 'old_page']
['converted']
new_data = data[data['landing_page'] == 'new_page']
['converted']


n_old = old_data.count()
n_new = new_data.count()
p_old = old_data.sum()/float(n_old)
p_new = new_data.sum()/float(n_new)
p = (n_old*p_old + p_new*n_new)/(n_old+n_new)
se = np.sqrt(p*(1-p)/n_old + p*(1-p)/n_new)
center = p_new-p_old
sample = scs.norm(loc=center,scale=se)


fig, ax = plt.subplots(1,1)
xvals2 = np.linspace(center-4*se,center+4*se,200)
ax.plot(xvals2,sample.pdf(xvals2),color='r', label='sample')
plt.legend()
plt.show()
```

2. Define another random variable for the null distribution and plot the PDF of the random variable. Add a vertical line on the plot to indicate the bound for rejecting the null hypothesis given a significance level of 5% (not shown in plot below).

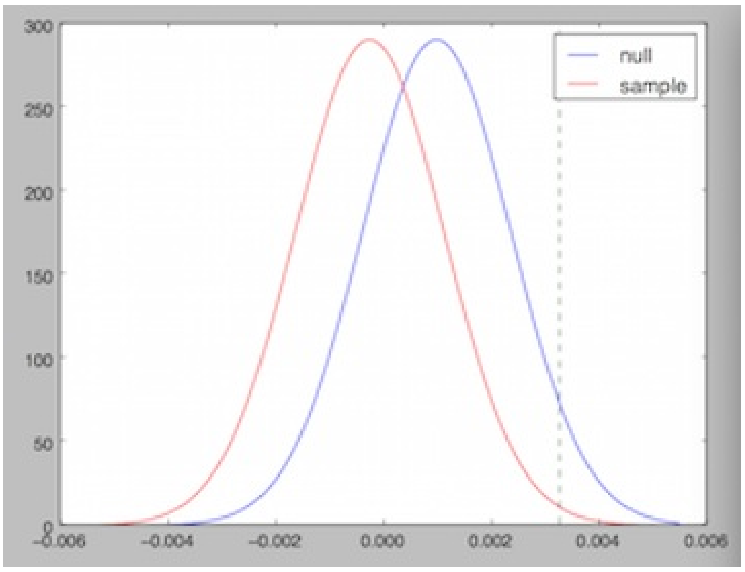


```
null = scs.norm(loc=0.001,scale=se)


xvals = np.linspace(.001-4*se,.001+4*se,200)
xvals2 = np.linspace(center-4*se,center+4*se,200)


fig, ax = plt.subplots(1,1)
ax.plot(xvals,null.pdf(xvals),color='b', label='null')
ax.plot(xvals2,sample.pdf(xvals2),color='r',
label='sample')
alphahighval = null.ppf(0.975)
ax.vlines(alphahighval,0,300,linestyle='--',color='g')
plt.legend()
plt.show()
```

3. Compute the power of the test given the null hypothesis if the true difference is the difference found in the samples taken. If the result seems strange to you, move onto `5.`.



```
The probability of correctly identifying the alternative is
true
if the true difference is as shown by the sample is 0.002.
This should seem strange because there is not a difference that
suggests that we have an increase by the new.
```

```
beta =
sample.cdf(alphahighval)
print "Power is: ",1-beta
```

4. What problem do you spot here with the plot from `2.`? Is increasing sample size going to increase power? If the effect size in the pilot is indeed representative of the ground truth, will the test ever be statisitcally significant? Explain your answer and suggest what the next steps should be.

```
The probability of correctly identifying the alternative is true
if the true difference is as shown by the sample is 0.002.
This should seem strange because there is not a difference
that suggests that we have an increase by the new. Obtaining more
data will not help us better the power, because in this case,
we have a true difference that is not in favor of our
alternative.
```

5. Assume after reviewing the data, Etsy decided the pilot is a plausible enough representation of the company's daily traffic. As a result, Esty decided on a two-tailed test instead, which is as follows:

```
X ~ p_new - p_old

H0: \pi_new - \pi_old = 0.001
H1: \pi_new - \pi_old !=
0.001
```

Recompute the power of the test under the same assumptions as in the previous part

```
Again, this is 1 minus the probability that we commit a type 2 error.
Where a type 2 error is choosing the null when the alternative is actually
true.


newbeta = sample.cdf(null.ppf(0.975)) -
sample.cdf(null.ppf(0.025))
print "two tailed power is: ",1-newbeta
```

Part 2.2: Computing Minimum Sample Size

Assume Etsy is staying with the two-tailed test described in `Part 2.1: 5`. A decision then would have to be made about how long the test is running.

The minimum sample size is calculated by following the exact same process with calculating power, except power is a given (80%) and sample size is omitted

1. Write a function `calc_min_sample_size` that would take

   - 2 lists/arrays of data (i.e. new page converts and old page converts)
   - Significance Level (Default: 0.05)
   - One-tailed to two-tailed test
   - Effect Size
   - Power (Default: 0.8)

   And return the minimum sample size required (rounded up to the nearest whole number).

```
import math

def calc_min_sample_size(a1, a2, eff_size, alpha=0.05, two_tail=True,
power=0.8):
  av1 = np.mean(a1)
  av2 = np.mean(a2)
  n1 = len(a1)
  n2 = len(a2)
  p = (av1 * n1 + av2 * n2)/(n1 + n2)
  sem = np.sqrt(p * (1-p) / n1 + p * (1-p)/ n2)
  pdiff = abs(av2 - av1)

  beta = 1-power
  if two_tail:
     alpha = alpha/2
  if pdiff >= eff_size:
     b = scs.norm(pdiff, sem).ppf(beta) + pdiff
     z_sem = b - eff_size
     z = scs.norm().ppf(1-alpha)
  else:
     b = scs.norm(pdiff, sem).ppf(1-beta) + pdiff
     z_sem = eff_size - b
     z = scs.norm().ppf(alpha)
  sem_des = z_sem / z
  n_des = (p * (1-p) / sem_des) ** 2
  return int(math.ceil(n_des))

print calc_min_sample_size(old_data, new_data, 0.001)
```