Hypothesis Testing

Answering Questions about Data

Assignment Contents

Overview

EXPECTED TIME 2 HRS

This assignment will reveiw the calculation of confidence intervals and p-values in Python. The examples from lecture will be reproduced using Python functions and packages. Eventually you will be asked to calculate confidence intervals and p-values on your own.

Activities in this Assignment

- Calculate confidence intervals
  - For means
  - For differences of means
  - Using t- and z- distributions
- Calculate p-values

NOTE: The z-multipliers MUST be calculated with the stats package as demonstrated below. e.g. for a 95% confidence interval, 1.959963... must be used rather than just 1.96

A quick reminder about a couple of useful "`numpy`" functions.

```
import numpy as np

# Calculations of:
print("Taking the mean:" ,
      np.mean([1,2,3,4,5,6,7]))
print("Finding Standard Deviation:",
      np.std([1,2,3,4,5,6,7]))
print("Finding Square-root:",
      np.sqrt(1738))
```

Calculate CI Using Python

Lets go through the example given in Lecture 7-1 around the 8 minute mark:

Observations: $n = 56$
Sample Standard Deviation: $s = 25$
Population Standard Deviation: UNKNOWN

Likelihood that $\bar{X}$ (the sample mean) is within ± 5 of μ?

$P(-5\le\bar{X}-\mu\le5) = P(-5\le Error\le5)$
$P(\frac{-5}{\sigma/\sqrt{n}}\le Z \le \frac{5}{\sigma/\sqrt{n}})$

Using sample standard deviation as estimate of σ:

$P(\frac{-5}{25/\sqrt{56}}\le Z \le \frac{5}{25/\sqrt{56}})$
$P(\frac{-5}{3.34…}\le Z \le \frac{5}{3.34…})$
$P(-1.49666\le Z \le 1.49666)$

Because of the symmetry of the normal distribution, this is equal to:

$=1 - 2 * P(Z \ge 1.49)$
$ = 1- 2 * (1-P(Z\le 1.49))$
$= 1-2*P(Z\le-1.49))$
(Rounding used to match lecture)

Note above how $P(Z \ge 1.49)$ is changed to $P(Z\le -1.49)$. The reason for this is $P(Z\le y)$ maybe easily calculated for any value of y with the "`.cdf()`" function in the stats package.

Shown below:

```
import scipy.stats as stats
1 - 2 * stats.norm.cdf(-1.49)
```

Continuing the example,creating the confidence interval.

The interval is:
$$\Big(\bar{X} - Z_{\alpha/2}\cdot\frac{\sigma}{\sqrt{n}}\ ,\ \bar{X} + Z_{\alpha/2}\cdot\frac{\sigma}{\sqrt{n}} \Big)$$

To calculate $Z_{\alpha/2}$ for a 95% confidence interval we can use the "`.interval()`" function or the "`.ppf()`" function. Both of which were covered more extensively in earlier assignments.

```
print(stats.norm.interval(.95))
```

```
print(stats.norm.ppf( 1- ((1 - .95)/2)))
print(stats.norm.ppf( (1 - .95)/2))
```

Calculating the confidence interval:

```
alpha = .95
interval_end = 1-((1-alpha)/2)
print(interval_end)
z_mult = stats.norm.ppf(interval_end)
sd = 25
x_bar = 135
n = 56

# Using our calculated values from above and python to create CI
print("\nConventional Calulation:\n",
      (x_bar - z_mult*(sd/np.sqrt(n)),x_bar + z_mult*(sd/np.sqrt(n)) ))

# Calculating CI using the .interval function
# (FOR THOSE INTERESTED IN SEEING MORE OF THE STATS PACKAGE)
print("\nCalculation with .interval():\n",  # "loc" is the mean of the distribution, "scale" is the sd
      stats.norm.interval(alpha = .95, loc = x_bar, scale= sd/ np.sqrt(n)))
```

Question 1

```
### GRADED

### Given point estimate (x_bar), confidence level(Z[alpha/2]), n, and sample standard deviation,
### calculate a confidence interval.

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### e.g. In the above example, the correct answer would be:

# lower = 128.45222
# upper = 141.54778

### Answers will be tested to three decimal places

### Calculate a 95% confidence interval where the sample mean of 92 observations was 130 with a
### sample standard deviation of 12

### YOUR ANSWER BELOW

x_bar = 130
n = 92
alpha = .95
sd = 12
interval_end = 1-((1-alpha)/2)


z_mult = stats.norm.ppf(interval_end)


lower = x_bar - z_mult*(sd/np.sqrt(n))

upper = x_bar + z_mult*(sd/np.sqrt(n))
```

Question 2

```
### GRADED

### Given point estimate (x_bar), confidence level(Z[alpha/2]), n, and sample standard deviation,
### calculate a confidence interval.

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### Answers will be tested to three decimal places

### Calculate a 90% confidence interval where the sample mean of 22 observations was 150 with a
### sample standard deviation of 40

### YOUR ANSWER BELOW

x_bar = 150
n = 22
alpha = .90
sd = 40

interval_end = 1-((1-alpha)/2)


z_mult = stats.norm.ppf(interval_end)


lower = x_bar - z_mult*(sd/np.sqrt(n))

upper = x_bar + z_mult*(sd/np.sqrt(n))
```

Question 3

```
### GRADED

### Given point estimate (x_bar), confidence level(Z[alpha/2]), n, and sample standard deviation,
### calculate a confidence interval.

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### Answers will be tested to three decimal places
```

```
### Calculate a 99% confidence interval where the sample mean of 2000 observations was 140 with a
### sample standard deviation of 40

### YOUR ANSWER BELOW

x_bar = 140
n = 2000
alpha = .99
sd = 40

interval_end = 1-((1-alpha)/2)


z_mult = stats.norm.ppf(interval_end)


lower = x_bar - z_mult*(sd/np.sqrt(n))

upper = x_bar + z_mult*(sd/np.sqrt(n))
```

Question 4

```
### GRADED

### Given point estimate (x_bar), confidence level(Z[alpha/2]), n, and sample standard deviation,
### calculate a confidence interval.

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### Answers will be tested to three decimal places

### Calculate a 95% confidence interval where the sample mean of 40 observations was 120 with a
### sample standard deviation of 9

### YOUR ANSWER BELOW

x_bar = 120
n = 40
alpha = .95
sd = 9

interval_end = 1-((1-alpha)/2)


z_mult = stats.norm.ppf(interval_end)


lower = x_bar - z_mult*(sd/np.sqrt(n))

upper = x_bar + z_mult*(sd/np.sqrt(n))
```

The T-distribution

When σ is unknown, and "n" is small: Lecture 7-3

The " `stats` " package has a library for the t distribution.

The " `t` " library functions similarly to the " `norm` " library, except that degrees of freedom must be specified. Remember, degrees of freedom (df) in these cases is $n - 1$. Thus 21 observations would yeild $df = 20$

The below shows how to calculate the values from the t-distribution discussed and looked up in the t-table in lecture 7-3.

```
print("normal distributions; 2.5% in tails; interval",
      stats.norm.interval(.95))

print("\nt-distribution; 2.5% in tails; df = 20; interval",
      stats.t.interval(.95, df = 20))

print("t-distribution; 2.5% in tails; df = 5; interval",
      stats.t.interval(.95, df = 5))

print("t-distribution; 2.5% in tails; df = 5; ppf",
      stats.t.ppf(.975, df = 5))
```

Example:

Given a set of observations, and a confidence level of 95%, calculate the confidence interval with a t-distribution:

```
observations = [121, 110, 126, 112, 129, 118, 126, 127, 126, 111, 127, 113, 126, 115, 114, 116]

n = len(observations) # find "n" -- the number of observations
x_bar = np.mean(observations) # find "x_bar"-- the sample mean
sd = np.std(observations) # find the sample standard deviation
alpha = .95
t_mult = stats.t.interval (alpha, df = n-1)[1]

print("Sample Mean:  ", x_bar)
print("Observations (n):  ", n)
print("Sample sd:  ", sd)
print("t-multiplier:  ", t_mult)

print("\nConfidence Interval:  ", x_bar, "+/-", round(t_mult * (sd / np.sqrt(n)),4))
```

Thus, given the above 16 observations, we can caluclate the 95% confidence interval from a t-distribution; spanning from ~116.3 to ~123.3

Question 5

```
### GRADED

### Calculate the 95% confidence interval (with a t-distribution)
### of the data stored in the"observations" variable below

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### Answers will be tested to three decimal places

### YOUR ANSWER BELOW

observations = [104, 148, 109, 104, 108, 120, 134, 129, 140, 128, 142, 113, 125, 111, 132, 133, 109, 107]
alpha = .95

n = len(observations) # find "n"
x_bar = np.mean(observations) # find "x_bar"- the sample mean
sd = np.std(observations) # find the sample standard deviation

t_mult = stats.t.interval (alpha, df = n-1)[1] # Find multiplier

lower = x_bar - t_mult * (sd / np.sqrt(n))
upper = x_bar + t_mult * (sd / np.sqrt(n))
```

```
### BEGIN HIDDEN TESTS
observations = [104, 148, 109, 104, 108, 120, 134, 129, 140, 128, 142, 113, 125, 111, 132, 133, 109, 107]

n = len(observations) # find "n"
x_bar = np.mean(observations) # find "x_bar"- the sample mean
sd = np.std(observations) # find the sample standard deviation
alpha = .95

t_mult = stats.t.interval (alpha, df = n-1)[1]

lower_T = x_bar - t_mult * (sd / np.sqrt(n))
upper_T = x_bar + t_mult * (sd / np.sqrt(n))

assert round(lower_T,3) == round(lower,3)
assert round(upper_T,3) == round(upper,3)

### END HIDDEN TESTS
```

Question 6

```
### GRADED

### Calculate the 95% confidence interval (with a t-distribution)
### of the data stored in the "observations" variable below

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### Answers will be tested to three decimal places

### YOUR ANSWER BELOW

observations = [124, 147, 136, 136, 100, 133, 137, 117, 121, 127, 130, 132, 143,
       146, 130, 149, 119, 146, 107, 148, 125, 105, 116, 130, 117, 117,
       108, 105, 139, 130]

alpha = .95

n = len(observations) # find "n"
x_bar = np.mean(observations) # find "x_bar"- the sample mean
sd = np.std(observations) # find the sample standard deviation

t_mult = stats.t.interval (alpha, df = n-1)[1] # Find multiplier

lower = x_bar - t_mult * (sd / np.sqrt(n))
upper = x_bar + t_mult * (sd / np.sqrt(n))
```

Question 7

NOTE: The below asks for you to calculate the confidence interval using the *z (normal) distribution* instead of the t-distribution

```
### GRADED

### Calculate the 95% confidence interval **(with a Z- (NORMAL) DISTRIBUTION)**
### of the data stored in the "observations" variable below

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### Answers will be tested to three decimal places

### YOUR ANSWER BELOW

observations = [124, 147, 136, 136, 100, 133, 137, 117, 121, 127, 130, 132, 143,
       146, 130, 149, 119, 146, 107, 148, 125, 105, 116, 130, 117, 117,
       108, 105, 139, 130]
alpha = .95

n = len(observations) # find "n"
x_bar = np.mean(observations) # find "x_bar"- the sample mean
sd = np.std(observations) # find the sample standard deviation

z_mult = stats.norm.interval (alpha)[1] # Find multiplier

lower = x_bar - z_mult * (sd / np.sqrt(n))
upper = x_bar + z_mult * (sd / np.sqrt(n))
```

Confidence Intervals: Difference of Means

Let's review the Central Park calculations from lectures 7-5 and 7-6:

Mean temperature between 1869 and 1968 ($\bar{Y}$) is 35.0 with a standard deviation (s_y) of 3.8. 100 observations (n_y)

Mean temperature between 1969 and 2015 ($\bar{X}$) is 38.1 with a standard deviation (s_x) of 4.4. 47 observations (n_x)

Difference of means: $\bar{X} - \bar{Y} = 38.1 - 35.0 = 3.1$
Standard Error: $s.e. = \sqrt{\frac{s^2_x}{n_x}+ \frac{s^2_y}{n_y}}$

```
se = np.sqrt(((4.4**2)/47)+((3.8**2)/100))
se
```

To create a confidence interval:

$\bar{X} - \bar{Y} \pm Z_{\alpha/2} \cdot s.e$

Calculate $Z_{\alpha/2}$ for the 95% confidence interval:

```
stats.norm.interval(.95)
```

Of note, when the observations are $n\ge30$, the normal distribution may be used for calculating confidence intervals instead of the t-distribution.

```
# Calculate CI
(3.1 + stats.norm.interval(.95)[0]*se, 3.1 + stats.norm.interval(.95)[1]*se)
```

To Calculate the p value:

$1-P\Big(-\frac{\bar{X} - \bar{Y}}{s.e}<Z<\frac{\bar{X} - \bar{Y}}{s.e}\Big)$
$= 2\cdot P \Big(Z<-\frac{3.1}{.745…}\Big)$
$\approx 2\cdot 1-P\Big(Z<-4.156\Big)$
$\approx .0000324$

Calculations below:

```
# Find test statistic using standard error calculated above
test_stat = (38.1-35.0)/ se
print(test_stat)

# Find p-value
print("p-value: ", round((2*(stats.norm.cdf(-test_stat))),7))
```

Final Example:

One additional example of finding the confidence interval of the difference of two means, and finding the p-value:

```
obs1 = [32.42, 34.61, 35.09, 35.67, 32.04, 34.31, 33.03, 35.55, 34.7 ,
       34.91, 36.02, 32.68, 35.65, 34.14, 32.65, 34.55, 32.78, 37.7 ,
       33.91, 33.53, 31.32, 33.25, 35.07, 36.66, 36.55, 33.52, 33.32,
       32.55, 33.69, 36.05, 30.66, 35.02, 34.05, 34.67, 37.61, 33.71,
       35.72, 34.54, 35.05, 33.69, 30.33, 32.01, 33.16, 36.3 , 32.66,
       31.73, 33.35, 33.16, 33.76, 33.92, 32.05, 34.18, 34.45, 31.49,
       31.9 , 34.33, 33.2 , 31.37, 34.56, 32.61]

obs2 = [36.2 , 41.98, 38.58, 33.59, 36.55, 33.5 , 30.78, 40.87, 42.25,
       39.08, 28.09, 36.74, 44.41, 29.22, 38.55, 24.41, 28.93, 31.97,
       36.6 , 36.  , 37.96, 33.92, 43.8 , 36.96, 41.44, 40.54, 35.88,
       30.82, 38.7 , 29.1 ]

# Calculate sample means
mean_x = np.mean(obs1)
mean_y = np.mean(obs2)

# Calculate sample standard deviations
sd_x = np.std(obs1)
sd_y = np.std(obs2)

# Count number of observations in each sample
n_x = len(obs1)
n_y = len(obs2)

# Set alpha
alpha = .95


print("Means: ", mean_x, ",", mean_y)
print("Standard Deviations: ", sd_x, ",", sd_y)
print("Number of Observations: ", n_x, ",", n_y)


# Calculate Observed Difference of means
diff = mean_x - mean_y

# Calculate Standard Error
se = np.sqrt( (sd_x**2/n_x) + (sd_y**2/n_y))

# Find z-multiplier
z_mult = stats.norm.interval(alpha)[1]
```

```
print("\n\nZ-Multiplier: ", z_mult)
print("Difference of Means: ", diff)
print("Standard Error: ", se)

# Calculate confidence interval
lower = diff - z_mult * se
upper = diff + z_mult * se

print("Confidence Interval: ", lower, " , ", upper)
```

With our 95% confidence interval spanning from -3.86 to -0.2 w e can say w ith 95% confidence that the difference in the means of the tw o sets of observations does not include 0.

Thus w e know that the p-value w ill be less than .05.

Below the exact p-value is calculated

```
# Calculate Test Statistic
test_stat = diff/ se
print(test_stat)

# Find p-value
print("p-value: ", round((2*(stats.norm.cdf(test_stat))),4))
```

Notice, becuase the difference w as negative, the test statistic w as also negative, and thus did not need to be made negative w hen passed to " `stats.norm.cdf()` "

See below w hat w ould happen if it w ere made negative -- a nonsensical p-value

```
print("WRONG p-value: ", round((2*(stats.norm.cdf(-test_stat))),4))
```

Question 8

```
### GRADED
### Calculate the 95% confidence interval (with a Z- (NORMAL) DISTRIBUTION)
### of the difference of the means of the collections stored in obs1 and obs2

### NOTE: Specifically find the CI for the mean of obs1 - mean of obs2

### Assign the lower bound as a number to "lower" and the upper bound as a number to "upper"

### Answers will be tested to three decimal places
### YOUR ANSWER BELOW

obs1 = [22.9 , 26.08, 25.04, 22.09, 24.28, 31.3 , 25.47, 24.17, 23.42,
       25.64, 23.96, 23.94, 25.35, 20.92, 27.74, 25.93, 26.9 , 27.87,
       22.43, 23.73, 29.25, 25.66, 23.6 , 26.77, 17.38, 26.26, 17.67,
       24.04, 19.42, 27.41, 30.02, 25.22, 26.47, 24.47, 22.85, 20.07,
       29.46, 23.61, 26.54, 25.37]

obs2 = [26.37, 32.62, 22.13, 22.64, 32.33, 25.62, 18.69, 26.86, 17.87,
       18.16, 26.37, 25.77, 22.57, 27.41, 17.2 , 22.61, 26.97, 28.78,
       24.02, 25.41, 27.88, 28.99, 30.06, 30.23, 24.19, 17.06, 24.38,
       24.13, 25.87, 31.58, 21.19, 32.07, 30.07, 24.23, 27.37]

alpha = .95

# Calculate sample means
mean_x = np.mean(obs1)
mean_y = np.mean(obs2)

# Calculate sample standard deviations
sd_x = np.std(obs1)
sd_y = np.std(obs2)

# Count number of observations in each sample
n_x = len(obs1)
n_y = len(obs2)

# Calculate Observed Difference of means
diff = mean_x - mean_y

# Calculate Standard Error
se = np.sqrt( (sd_x**2/n_x) + (sd_y**2/n_y))

# Find z-multiplier
z_mult = stats.norm.interval(alpha)[1]

# Calculate confidence interval
lower = diff - z_mult * se
upper = diff + z_mult * se
```

Question 9

```
### GRADED

### Calculate the p-value for the difference of the means of the two samples.

### Answers will be tested to 3 decimal place

### Assign numeric answer to "p_val"
### YOUR ANSWER BELOW

obs1 = [22.9 , 26.08, 25.04, 22.09, 24.28, 31.3 , 25.47, 24.17, 23.42,
       25.64, 23.96, 23.94, 25.35, 20.92, 27.74, 25.93, 26.9 , 27.87,
       22.43, 23.73, 29.25, 25.66, 23.6 , 26.77, 17.38, 26.26, 17.67,
       24.04, 19.42, 27.41, 30.02, 25.22, 26.47, 24.47, 22.85, 20.07,
       29.46, 23.61, 26.54, 25.37]
```

```
obs2 = [26.37, 32.62, 22.13, 22.64, 32.33, 25.62, 18.69, 26.86, 17.87,
       18.16, 26.37, 25.77, 22.57, 27.41, 17.2 , 22.61, 26.97, 28.78,
       24.02, 25.41, 27.88, 28.99, 30.06, 30.23, 24.19, 17.06, 24.38,
       24.13, 25.87, 31.58, 21.19, 32.07, 30.07, 24.23, 27.37]

# Calculate sample means
mean_x = np.mean(obs1)
mean_y = np.mean(obs2)

# Calculate sample standard deviations
sd_x = np.std(obs1)
sd_y = np.std(obs2)

# Count number of observations in each sample
n_x = len(obs1)
n_y = len(obs2)

# Calculate Observed Difference of means
diff = mean_x - mean_y

# Calculate Standard Error
se = np.sqrt( (sd_x**2/n_x) + (sd_y**2/n_y))

test_stat = diff/ se

# Find p-value
p_val = 2*(stats.norm.cdf(test_stat))
```