Devin Trejo

**ECE 3522: Stochastic Processes in Signals and Systems**

**Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19122**

# Problem Statement

Today we will use MatLab to perform hypothesis testing. From Oliver-Ibe Fundamental of Applied Probability (2nd Edition) chapter 9 we base our analysis.

**“Example 9.9:** The mean lifetime E[X] of the light bulbs produced by Lighting Systems Corporation is 1570 hours with a standard deviation of 120 hours. The president of the company claims that a new production process has led to an increase in the mean lifetimes of the light bulbs. If Joe tested 100 light bulbs made from the new production process and found that their mean lifetime is 1600 hours, test the hypothesis that E[X] is not equal to 1570 hours.”

To test the problem we first must generate an array of Gaussian distributed random variables. From the samples we generate we will compute an actual mean for our data (given our expected mean is 1600). Then we will check to see if the mean is statistically significant by applying a two tailed test. If the test proves not to be significant we will determine the minimum value of N samples which would make our test significant. We will test two different scenarios, one where our generate variables have a standard deviation of 120 and another where our standard deviation is 240.

To test significance we first must compute the z-score for our actual mean (seen below). Depending on the level of significance we want to test (significance = 1-cofidence), we test to see if the z-score is within the accepted range. If it falls outside the range we say the actual mean is statistically significant.

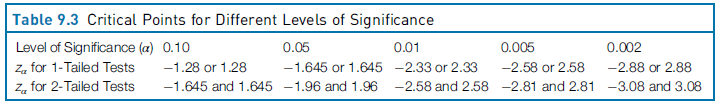


Figure 1: Level of significance and correspond max z-score for both 1 tailed and 2 tailed tests

# Approach and Results

When running the code we encounter that multiple times our randomly generated variables produce significant deviation from the expected mean. For sake of showcasing the true results we obtained we have provided multiple cases for each standard deviation (120, 240) below. For each we have printed to the console the standard deviation we are testing, the actual mean of our generated variables, and a matrix that tells us whether our test was significant different or not. If the test was not significant we test to see what is the minimum value of N that would produce significant difference. In the second matrix printed to the console we see the rows correspond to the different significance levels, (0.2 0.1 0.05 0.01 .005 0.002) and the row correspond to the number of samples we generated (10, 100, 1000).

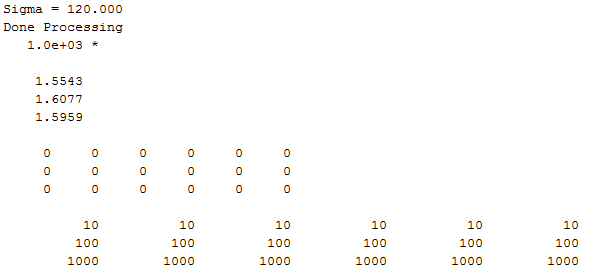


Figure 2: Trial 1 for

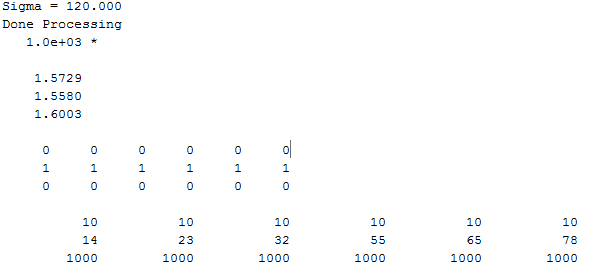


Figure : Trial 2 for

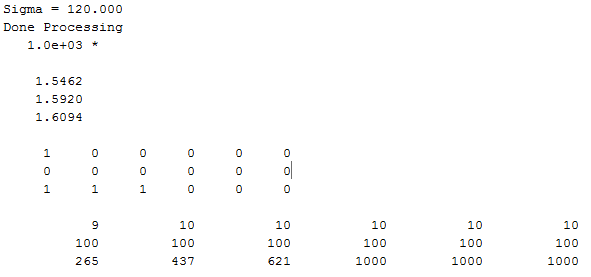


Figure 4: Trial 3 for

For the above trials the standard deviation was 120. Since the standard deviation is relatively low we do not expect there to many times where our true mean is different from the expected mean. We can see from our three trials that the first time we do not obtain any mean that was too different from the expected. In our second trial our mean for when N=100 we saw an actual mean of 1558.0 which is statistically different for all confidence levels we tested. Our final test showcases an instance where N=1000 and our data was statically different. We observe that at 80% confidence, the minimum value N that we could ensure would fall within the actual mean is 265. At 264 we would say the means are statistically significant.

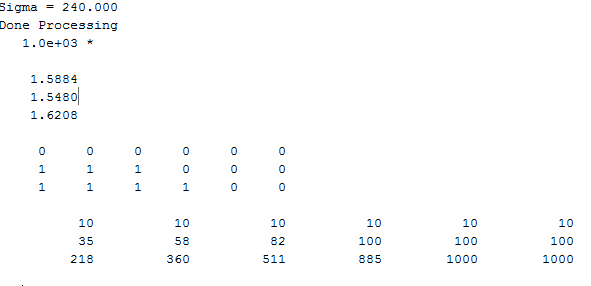


Figure 5: Trial 1 for

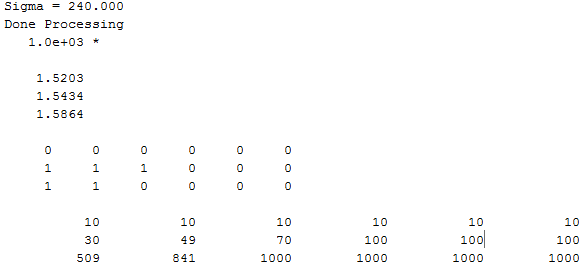


Figure 6: Trial 2 for

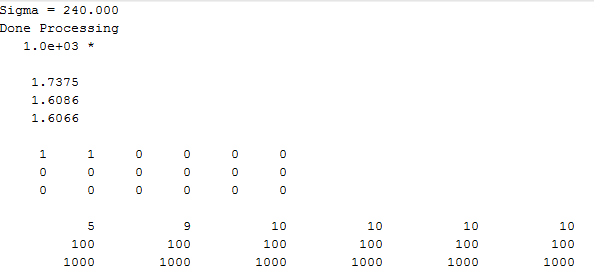


Figure 7: Trial 2 for

When we increased our standard deviation for our random number generator we saw more cases where our data was statistically significant. Our actual means have a lot more variation in values than we saw previously.

# MATLAB Code

‘devNormDist’ explained in CA 8.

%% Main Script

clear; clc; close all

nRange = [10, 100, 1E3];

stdDev2 = [120, 240];

meanX = 1600;

for sigma = stdDev2

fprintf(sprintf('Sigma = %0.3f\n',sigma));

stoCa10(sigma, meanX, nRange)

end

1 Our main script that runs the function. We loop through two different standard deviations to test whether our data is different.

function stoCa10(stdDev2, meanX, nRange)

signfRange = [0.2 0.1 0.05 0.01 .005 0.002];

i = 1;

for nSam = nRange

j = 1;

% Generate Array

X = devNormDist(meanX, stdDev2, nSam);

% Find the Estimate of Mean

muX = sum(X)/nSam;

stdDev = sqrt(var(X));

for significance = signfRange

% Determine if Significant

isSignf(i,j) = chk\_signf(meanX, muX, stdDev2, nSam, significance);

if isSignf(i, j) == 1

signfN(i,j) = det\_signf(muX, meanX, stdDev2, significance);

else

signfN(i,j) = nSam;

end

j = j + 1;

end

meanActual(i) = muX;

i = i + 1;

end

% Show Results

disp('Done Processing');

disp(meanActual')

disp(isSignf)

disp(signfN)

end

2 The function allows us to test over our range of sample sizes whether our means are different. We use the devNormDist generate our array of Gaussian distrusted data. Then we find the actual mean of the data that the computer generated. We then test to see if the true mean is different from the expected mean at various significance levels. If the test proved to be significant we test to see what minimum value N would produce not significant means.

function status = chk\_signf(mean1, mean2, stdev, nSam, significance)

% Find z Score

z = abs((mean1-mean2)/(stdev/sqrt(nSam)));

% Lookup Table for Range

switch significance

case 0.0020

acceptanceRMax = 3.0800;

case 0.0050

acceptanceRMax = 2.8100;

case 0.0100

acceptanceRMax = 2.5800;

case 0.0500

acceptanceRMax = 1.9600;

case 0.1000

acceptanceRMax = 1.6450;

case 0.2000

acceptanceRMax = 1.28;

otherwise

error(sprintf('Confidence level (%0.3f) is not defined in look-up table', 1-significance))

end

% Check if z-score is within the accepted range

% 1 if significant

% 0 if not significant

if (z <= acceptanceRMax)

status = 0;

else

status = 1;

end

end

3 ‘checksignf’ runs to see if two means are different at agiven significance when provided a standard devivation, N samples. The function is simply a switch statement that first computes the z-score and tests to see if it falls inside the accepeted range put forth by Figure 1.

function N = det\_signf(mean1, mean2, stdev, significance)

space = linspace(1, 1E5, 1E5);

% Loop through all possible ns

for n = space

% Use our function we developed before to check if significant

status = chk\_signf(mean1, mean2, stdev ,n, significance);

% If the value n is significant we save the value and return n

if status == 1

N = n;

return

end

end

N = inf;

end

4 ‘det\_signf’ is a function that loops through various n values to test what minimum value n produces significance. Using the ‘checksignf’ function we can easily test the condition. Once we found a value n that produces significance we return the value to the user.

# Conclusions

We conclude by showing how we can test use hypothesis testing to ensure our batch of data falls within a certain range. We can apply a level of confidence that can ensure our customers that what we produce is what we expect to produce. There are always variations in manufacturing that will cause your assembly line to produce different qualities of the same product. The more confident you are that your product falls within a certain range the less samples you take. If you have less samples you can ensure your consumer that most of the batch you are selling falls within an expected range.