Statistics 572
Homework 4

Alejandro Robles

October 11th, 2018

1 Exercise 7.2

Using the information in example 7.3, plot the probability of Type II error as a function of μ . How does this compare with figure 7.2?

MATLAB Code:

```
mualt = 40:60;
cv = 1.645;
sig = 1.5;
ct = cv*1.5 + 45;
ctv = ct*ones(size(mualt));
beta = normcdf(ctv,mualt,sig);
pow = 1 - beta;
plot(mualt,pow,'*-',mualt,beta,'ro-')
xlabel('True Mean \mu')
legend('Power','TypeII error')
```

Result:

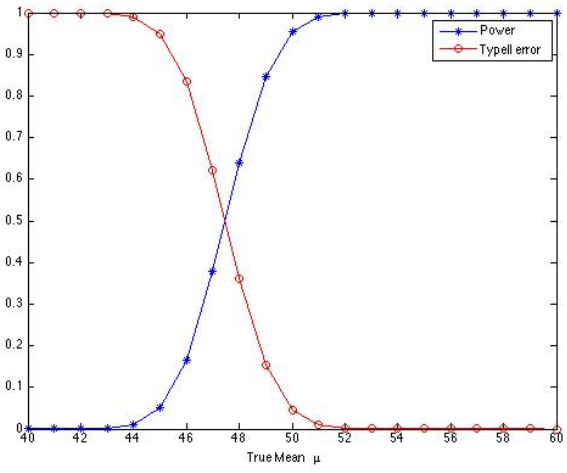


Figure 1: Histogram

Discussion:

As we can see, they are a mirror image of each other. This is because Power = 1 - β.

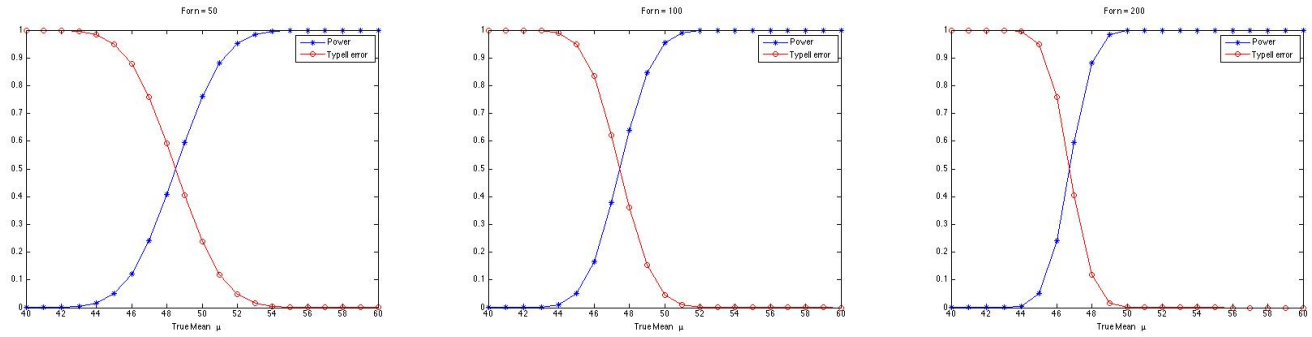
2 Exercise 7.4

Using the same value for the sample mean, repeat example 7.3 for different sample sizes of n = 50, 100, 200 . What happens to the curve showing the power as a function of the true mean as the sample size changes?

MATLAB Code:

```
for n=[50, 100, 200];
    mualt = 40:60;
    cv = 1.645;
    sig = 15/sqrt(n); % Only thing that changes with sample size
    ct = cv*sig + 45;
    ctv = ct*ones(size(mualt));
    beta = normcdf(ctv,mualt,sig);
    pow = 1 - beta;
    figure;
    plot(mualt,pow,'*-',mualt,beta,'ro-')
    xlabel('True Mean \mu')
    legend('Power','TypeII error')
    title(sprintf('For n = %d\n',n));
end
```

Result:



(a) N = 50 (b) N = 100 (c) N = 200

Figure 2: Power Versus Type II Error

Discussion: Power is the probability of rejecting H_0 when it is really false. As we can see, as the sample size increases the curve becomes more steeper and less flat. Because the critical region is in the upper tail of the distribution of the test statistic, then as μ increases the power quickly goes up as the sample size increases.

3 Exercise 7.5

Repeat example 7.6 using a two-tail test. In other words, test for the alternative hypothesis that the mean is not equal to 454.

MATLAB Code:

```
load mcdata
n = length(mcdata);
% Population sigma is known.
sigma = 7.8;
sigxbar = sigma/sqrt(n);
Tobs = (mean(mcdata)-454)/sigxbar;
M = 1000; % Number of Monte Carlo trials
Tm = zeros(1,M); % Storage for test statistics from the MC trials.
for i = 1:M % Start the simulation.
xs = sigma*randn(1,n) + 454;
Tm(i) = (mean(xs) - 454)/sigxbar;
end
% This is a two-tail test, so it is the
alpha = 0.05;
cv = [csquantiles(Tm,alpha/2) csquantiles(Tm,1-alpha/2)];
% Check to see if we reject null hypothesis
if Tobs < cv(1)
    sprintf('Because %g < %g, then we reject the Null Hypothesis ',Tobs, cv(1))
elseif Tobs > cv(2)
    sprintf('Because %g > %g, then we reject the Null Hypothesis ',Tobs, cv(2))
else
    disp('We do not Null Hypothesis')
end
```

Result:

```
>> exercise_7_5
Because -2.5641 < -1.85231, then we reject the Null Hypothesis.
```

Discussion: As we can see, because t_0 falls in the critical region, then we reject the null hypothesis.

4 Exercise 7.6

Repeat example 7.8 for larger M. Does the estimated Type I error get closer to the true value?

MATLAB Code:

```
largerM = [1000 1500 2000 2500 3000];
alpha = 0.05; sigma = 7.8;
cv = norminv(alpha,0,1);
Im = 0; index = 1;
alphas = zeros(1,length(largerM));
load mcdata
n = length(mcdata);
sigxbar = sigma/sqrt(n);
for M = largerM
    for i = 1:M
            % Generate a random sample under H_0.
            xs = sigma*randn(1,n) + 454;
            Tm = (mean(xs)-454)/sigxbar;
            if Tm <= cv   % then reject H_0
                Im = Im +1;
            end
    end
    alphas(index) = Im/M;
    index = index + 1;
    Im = 0;
end
figure(1);
plot(largerM, alphas, '-o');
xlabel('M Trials'); ylabel('\alpha');
```

Result:

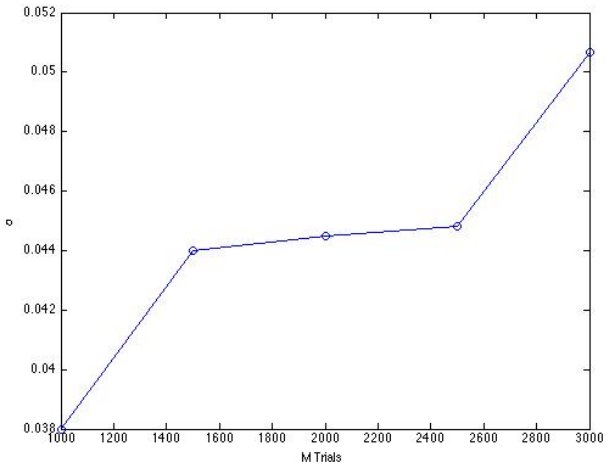


Figure 3: Type I Error

Discussion: As we can see, as the number of M trials increases the closer it reaches the true value of 0.05.

5 Exercise 7.7

Write MATLAB code that implements the parametric bootstrap. Test it using the forearm data. Assume that the normal distribution is a reasonable model for the data. Use your code to get a bootstrap estimate of the standard error and the bias of the coefficient of skewness and the coefficient of kurtosis. Get a bootstrap percentile interval for the sample central second moment using your parametric bootstrap approach.

MATLAB Code:

```
% Parameters
load forearm
n = length(forearm);
sample_mu = mean(forearm);
sample_sigma = std(forearm);
B = 100;
alpha = 0.05;

% Get the value of the statistic of interest with original data.
theta1 = skewness(forearm);
theta2 = kurtosis(forearm);

% Sample from Gaussian B times with sample size of n
sampled_bs = normrnd(sample_mu,sample_sigma,[n,B]);

% Calculate statistics from sampled data
theta1_bs = skewness(sampled_bs);
theta2_bs = kurtosis(sampled_bs);

% Get Standard Error from both estimates
se_theta1 = std(theta1_bs);
se_theta2 = std(theta2_bs);

% Get Bias from both estimates
bias_theta1 = mean(theta1_bs) - theta1;
bias_theta2 = mean(theta2_bs) - theta1;

% Get bootstrap percentile interval
bvals = zeros(1,B);
for i = 1:B
   bvals(i) = mom(sampled_bs(:,i)); % function found in toolbox
end
k = round(B*alpha/2);
sbval = sort(bvals);
blo = sbval(k);
bhi = sbval(B-k);

sprintf('Bootstrap Estimate of the Standard Error (Skewness): %g', se_theta1)
sprintf('Bootstrap Estimate of the Standard Error (Kurtosis): %g', se_theta2)
sprintf('Bootstrap Estimate of the Bias (Skewness): %g', bias_theta1)
sprintf('Bootstrap Estimate of the Bias (Kurtosis): %g', bias_theta2)
sprintf('Bootstrap Percentile Interval for the sample central second moment: (%g, %g)', blo,bhi )
```

Result:

```
>> exercise_7_7
Bootstrap Estimate of the Standard Error (Skewness): 0.180723
Bootstrap Estimate of the Standard Error (Kurtosis): 0.363058
Bootstrap Estimate of the Bias (Skewness): 0.0993879
Bootstrap Estimate of the Bias (Kurtosis): 3.09387
Bootstrap Percentile Interval for the sample central second moment: (0.977131, 1.51873)
>>
```

Discussion:

If the estimated bias is small relative to the estimated error then it's recommended to not correct for the bias. This is the case for skewness but not for kurtosis.

6 Exercise 7.8

Write MATLAB code that will get the bootstrap standard confidence interval. Use it with the forearm data to get a confidence interval for the sample central second moment. Compare this interval with the ones obtained in the examples and in the previous problem.

MATLAB Code:

```
% Parameters
load forearm
n = length(forearm);
sample_mu = mean(forearm);
sample_sigma = std(forearm);
B = 100;
alpha = 0.05;

% Get the value of the statistic of interest with original data.
theta = mom(forearm);

% Sample from Gaussian B times with sample size of n
sampled_bs = normrnd(sample_mu,sample_sigma,[n,B]);

% Get bootstrap percentile interval
bvals = zeros(1,B);
for i = 1:B
   bvals(i) = mom(sampled_bs(:,i)); % function found in toolbox
end
k = round(B*alpha/2);
sbval = sort(bvals);
blo = sbval(k);
bhi = sbval(B-k);

% Get Standard Error from both estimates
se_theta = std(bvals);

% Get Bootstrap Standard Confidence Interval
ci_low = theta - blo*se_theta;
ci_high = theta + blo*se_theta;

sprintf('Bootstrap Standard Confidence Interval for the sample central second moment: (%g, %g)', ci_
```

Result:

```
>> exercise_7_8
Bootstrap Standard Confidence Interval for the sample central second moment: (1.10903, 1.38396).
```

Discussion: We have a 95% Standard Confidence Interval for the sample central second moment to be between (1.10903, 1.38396). Compared to previous confidence interval, this one is more narrower.

7 Exercise 7.9

Use your program from problem 7.8 and the forearm data to get a bootstrap confidence interval for the mean. Compare this to the theoretical one.

MATLAB Code:

```
% Parameters
load forearm
n = length(forearm);
sample_mu = mean(forearm);
sample_sigma = std(forearm);
B = 100;
alpha = 0.05;

% Get the value of the statistic of interest with original data.
theta = mean(forearm);

% Sample from Gaussian B times with sample size of n
sampled_bs = normrnd(sample_mu,sample_sigma,[n,B]);


% Get bootstrap percentile interval
bvals = zeros(1,B);
for i = 1:B
   bvals(i) = mean(sampled_bs(:,i)); % function found in toolbox
end
k = round(B*alpha/2);
sbval = sort(bvals);
blo = sbval(k);
bhi = sbval(B-k);

% Get Standard Error from both estimates
se_theta = std(bvals);

% Get Bootstrap Standard Confidence Interval
ci_low = theta - blo*se_theta;
ci_high = theta + blo*se_theta;

sprintf('Bootstrap Standard Confidence Interval for the mean: (%g, %g)', ci_low,ci_high )
```

Result:

```
>> exercise_7_9
Bootstrap Standard Confidence Interval for the mean: (17.0769, 20.5273)
```

Discussion:

We use the assumption that this follows a N(18.8021, 1.1205) so therefore this process did capture correct mean for this instance.

8 In-Class:

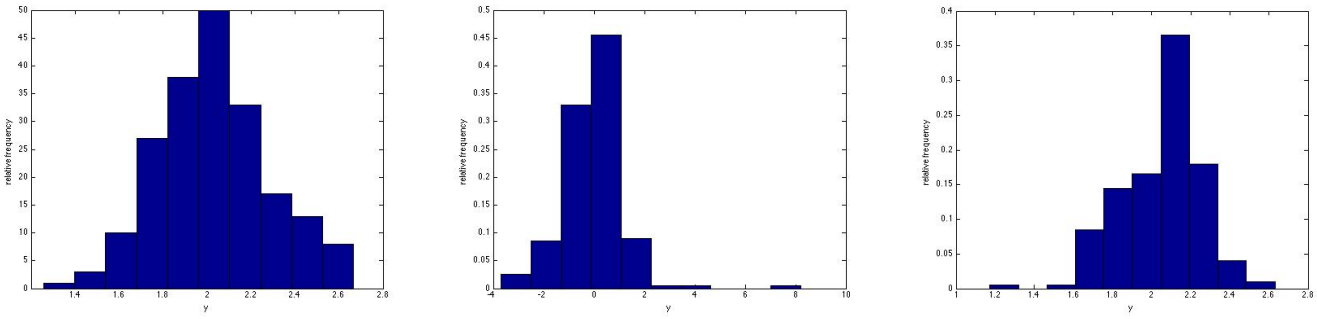
Part A: Take random sample of size 20 from N(2, 1) to represent as your raw data. Calculate 95% Confidence Interval of median using the three methods below. Plot the relative frequency histogram of the bootstrap estimates.

1. Standard Method
2. Boot-T
3. Boot Percentile Interval

MATLAB Code:

```
% Parameters
n = 20; mu = 2; sigma = 1;
B = 200; alpha = 0.05;

% Sample from Gaussian with sample size of n
data = normrnd(mu,sigma,[n,1]);

% Bootstrap-T UDF
[ci1, thetas] = bootstrap_t(data, @median, 'median', alpha,B);
figure(1);
rhist(thetas); %UDF

% Bootstrap-Standard UDF
figure(2);
[ci2, thetas] = bootstrap_standard(data, @median, 'median', alpha,B, [2,1]);
rhist(thetas);
% Bootstrap-Percentile UDF

figure(3);
[ci3, thetas] = bootstrap_percentile(data, 'median', alpha, B);
rhist(thetas);
```

Result:



(a) Standard Method (b) Boot T (c) Percentile Interval

Figure 4: BootStrap Confidence Intervals

Discussion: As we can see, they all contain the true median of 2 but the Standard Method looks the best, which makes sense because are assuming the right distribution.

Part B: Repeat Part A 100 times and calculate coverage of each method where

$$Coverage = \frac{Number\ of\ CI's\ Including\ True\ Median}{100} * 100$$

MATLAB Code:

```
% Parameters
n = 20; mu = 2; sigma = 1;
B = 100; alpha = 0.05; trials = 100;

% Sample from Gaussian with sample size of n
data = normrnd(mu,sigma,[n,1]);

correct1 = 0;
correct2 = 0;
correct3 = 0;

for i = 1:trials
    % Bootstrap-T UDF
    ci1 = bootstrap_t(data, @median, 'median', alpha,B);
    ci2 = bootstrap_standard(data, @median, 'median', alpha,B, [2,1]);
    ci3 = bootstrap_percentile(data, 'median', alpha, B);

    % Check how many were correct
    correct1 = correct1 + (mu > ci1(1) || mu < ci1(2));
    correct2 = correct2 + (mu > ci2(1) || mu < ci2(2));
    correct3 = correct3 + (mu > ci3(1) || mu < ci3(2));
end

sprintf('Coverage for Boot-T = %g', correct1/100*100 )
sprintf('Coverage for Standard = %g', correct2/100*100 )
sprintf('Coverage for Percentile = %g', correct3/100*100 )
```

Result:

```
Coverage for Boot-T = 100%
Coverage for Standard = 100%
Coverage for Percentile = 100%
```

Discussion: So even though we produced 95% Confidence Intervals, then theoretically we should only see coverage of around 95. I imagine if we were to work with real life data then maybe there will be more errors in the process that may converge the coverage to 95.

User Defined Functions:

Standard Method:

```
function [ci, bthetas] = bootstrap_standard(data, statistic , stat_name, alpha, B, params)
    % Parameters
    n = length(data);

    if isempty(params)
        sample_mu = mean(data);
        sample_sigma = std(data);
    else
        sample_mu = params(1);
        sample_sigma = params(2);
    end

    % Get the value of the statistic of interest with original data.
    theta = statistic(data);

    % Sample from Gaussian B times with sample size of n
    sampled_bs = normrnd(sample_mu,sample_sigma,[n,B]);


    % Get bootstrap percentile interval
    bvals = zeros(1,B);
    for i = 1:B
       bvals(i) = statistic(sampled_bs(:,i)); % function found in toolbox
    end
    k = round(B*alpha/2);
    sbval = sort(bvals);
    blo = sbval(k);
    bhi = sbval(B-k);

    % Get Standard Error from both estimates
    se_theta = std(bvals);

    % Get Bootstrap Standard Confidence Interval
    ci_low = theta - blo*se_theta;
    ci_high = theta + blo*se_theta;
    ci = [ci_low, ci_high];
    bthetas = sbval;
    sprintf('Bootstrap Standard Confidence Interval for the %s: (%g, %g)', stat_name, ci_low, ci_hig
```

Boot T:

```
function [ci, bthetas] = bootstrap_t(data, statistic , stat_name, alpha, B)
    thetahat = statistic(data);

    % Get the bootstrap replicates and samples.
    [bootreps, bootsam] = bootstrp(B,stat_name,data);
    % Set up some storage space for the SEs.
    sehats = zeros(size(bootreps));

    bvals = zeros(1,B);
    % Each column of bootsam contains indices
    % to a bootstrap sample.
    for i = 1:B
                % extract the sample from the data
```

```
        xstar = data(bootsam(:,i));
        bvals(i) = statistic(xstar);
        % Do bootstrap using that sample to estimate SE.
        sehats(i) = std(bootstrp(25,stat_name,xstar));
    end
    zvals = (bootreps - thetahat)./sehats;

    % Estimate the SE using the bootstrap.
    SE = std(bootreps);

    % Get the quantiles.
    k = B*alpha/2;
    szval = sort(zvals);
    tlo = szval(k);
    thi = szval(B-k);
    % Get the endpoints of the interval.
    blo = thetahat-thi*SE;
    bhi = thetahat-tlo*SE;

    ci = [blo bhi];
    bthetas = szval;
    sprintf('Bootstrap Standard Confidence Interval for the %s: (%g, %g)', stat_name, blo,bhi )
```

Boot Percentile Interval:

```
function [ci, bthetas] = bootstrap_percentile(data, stat_name, alpha, B)
    bvals = bootstrp(B,stat_name,data);
    % Find the upper and lower endpoints
    k = round(B*alpha/2);
    sbval = sort(bvals);
    blo = sbval(k);
    bhi = sbval(B-k);
    %sprintf('Bootstrap Percentile Interval for the %s: (%g, %g)',stat_name, blo,bhi )
    ci = [blo, bhi];
    bthetas = sbval;
```

Relative Frequency:

```
function [no,xo] = rhist(varargin)
error(nargchk(1,inf,nargin));
[cax,args,nargs] = axescheck(varargin{:});
y = args{1};
if nargs == 1
    x = 10;
elseif nargs == 2
    x = args{2};
else
    if isempty(args{2})
        x = 10;
    else
        x = args{2};
    end
end
[m,n] = size(y);
[nn,x]=hist(y,x); % frequency
nn = nn./m;     % relative frequency
%  relative frequency density
if nargs == 3
```

```
    binwidth = x(2)-x(1);
    nn = nn./binwidth;
end
if nargout == 0
  if ~isempty(cax)
    bar(cax,x,nn,[min(y(:)) max(y(:))],'hist');
  else
    bar(x,nn,[min(y(:)) max(y(:))],'hist');
  end
  xlabel('y')
  if nargs == 3
    ylabel('relative frequency density')
  else
     ylabel('relative frequency')
  end
else
  no = nn;
  xo = x;
end
```