

Stat 579 Statistical Computing II Homework 3

Erik Erhardt

April 13, 2006

All code can be found in the Appendix.

Exercise 1 *Pearson P.*

(a) *Pearson statistic.*

Given $H_0 : \theta_0 = \{.1, .2, .4, .2, .1\}$ with observed counts $x = \{16, 50, 31, 11, 9\}$, we reject H_0 with a Pearson statistic $P = 44.3462$ and associated large sample χ_4^2 distribution p-value = 0.0000000054367.

(b) *Parametric bootstrap evaluation of the multinomial sampling distribution.*

Using 100,000 simulations, no simulated Pearson P s exceeded the observed Pearson P , giving an “exact” p-value = 0. (Running this many times gave 0 each time, so this 0 is a really, really small 0.)

The plot in Figure 1 on page 2 demonstrate that the χ_4^2 distribution approximates well the sampling distribution of the Pearson P statistic based on the Multinomial(θ) distribution with $\theta = \{.1, .2, .4, .2, .1\}$.

Both of these two p-values are nearly 0, so while they agree, this particular vector $x = \{16, 50, 31, 11, 9\}$ does not give us much information about the quality of the approximation.

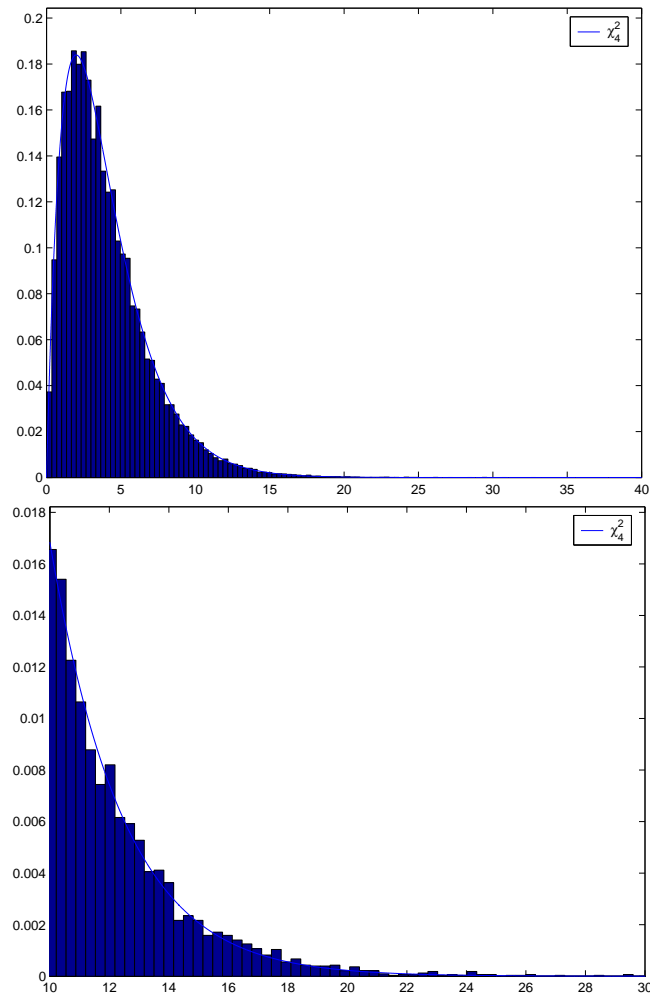


Figure 1: 1b. χ_4^2 distribution approximates Multinomial(.1, .2, .4, .2, .1).

Exercise 2 *Binomial Confidence Intervals.***(a)** *Write functions.*

Functions `pearsonp.m`, `multigen.m`, `pciexact.m`, `pciagresti.m`, `pciscore.m`, and the simulation script in Appendix.

(b) *Study of confidence interval coverage and lengths.*

By the plots in Figure 2 on page 4 and Table 1 on page 3, the score interval is best, followed by Agresti's interval, with the Exact interval being the worst. The score interval not only is the shortest interval, but is most often on target for the correct coverage. The Agresti interval conservatively covers more than the nominal 95%, with the exact over-covering. The Agresti interval is slightly longer than the score, with their length converging as a function of the sample size, but the exact interval is much longer except when θ is near 1 or 0. In these cases, the exact interval is shorter, but grossly overcovering.

n	Method	Coverage			Interval Length		
		Mean	Std	95% CI	Mean	Std	95% CI
5	Exact	0.9905	0.0071	(0.9903, 0.9906)	0.6794	0.0684	(0.6780, 0.6808)
5	Agresti	0.9672	0.0228	(0.9667, 0.9676)	0.6051	0.0267	(0.6046, 0.6056)
5	Score	0.9553	0.0281	(0.9548, 0.9559)	0.5593	0.0536	(0.5582, 0.5604)
15	Exact	0.9796	0.0100	(0.9794, 0.9798)	0.4230	0.0812	(0.4214, 0.4246)
15	Agresti	0.9631	0.0141	(0.9628, 0.9634)	0.3903	0.0474	(0.3893, 0.3912)
15	Score	0.9521	0.0218	(0.9517, 0.9525)	0.3704	0.0658	(0.3691, 0.3717)
25	Exact	0.9756	0.0103	(0.9754, 0.9758)	0.3297	0.0717	(0.3282, 0.3311)
25	Agresti	0.9584	0.0126	(0.9582, 0.9587)	0.3077	0.0490	(0.3068, 0.3087)
25	Score	0.9513	0.0149	(0.9510, 0.9516)	0.2960	0.0613	(0.2948, 0.2972)
50	Exact	0.9685	0.0093	(0.9683, 0.9686)	0.2322	0.0555	(0.2311, 0.2333)
50	Agresti	0.9567	0.0129	(0.9564, 0.9569)	0.2195	0.0443	(0.2186, 0.2204)
50	Score	0.9502	0.0138	(0.9499, 0.9505)	0.2143	0.0504	(0.2133, 0.2154)
100	Exact	0.9649	0.0075	(0.9648, 0.9651)	0.1627	0.0408	(0.1619, 0.1635)
100	Agresti	0.9545	0.0109	(0.9543, 0.9547)	0.1555	0.0360	(0.1548, 0.1562)
100	Score	0.9493	0.0098	(0.9491, 0.9495)	0.1534	0.0386	(0.1526, 0.1542)

Table 1: Average Coverage and Interval length for the three methods.

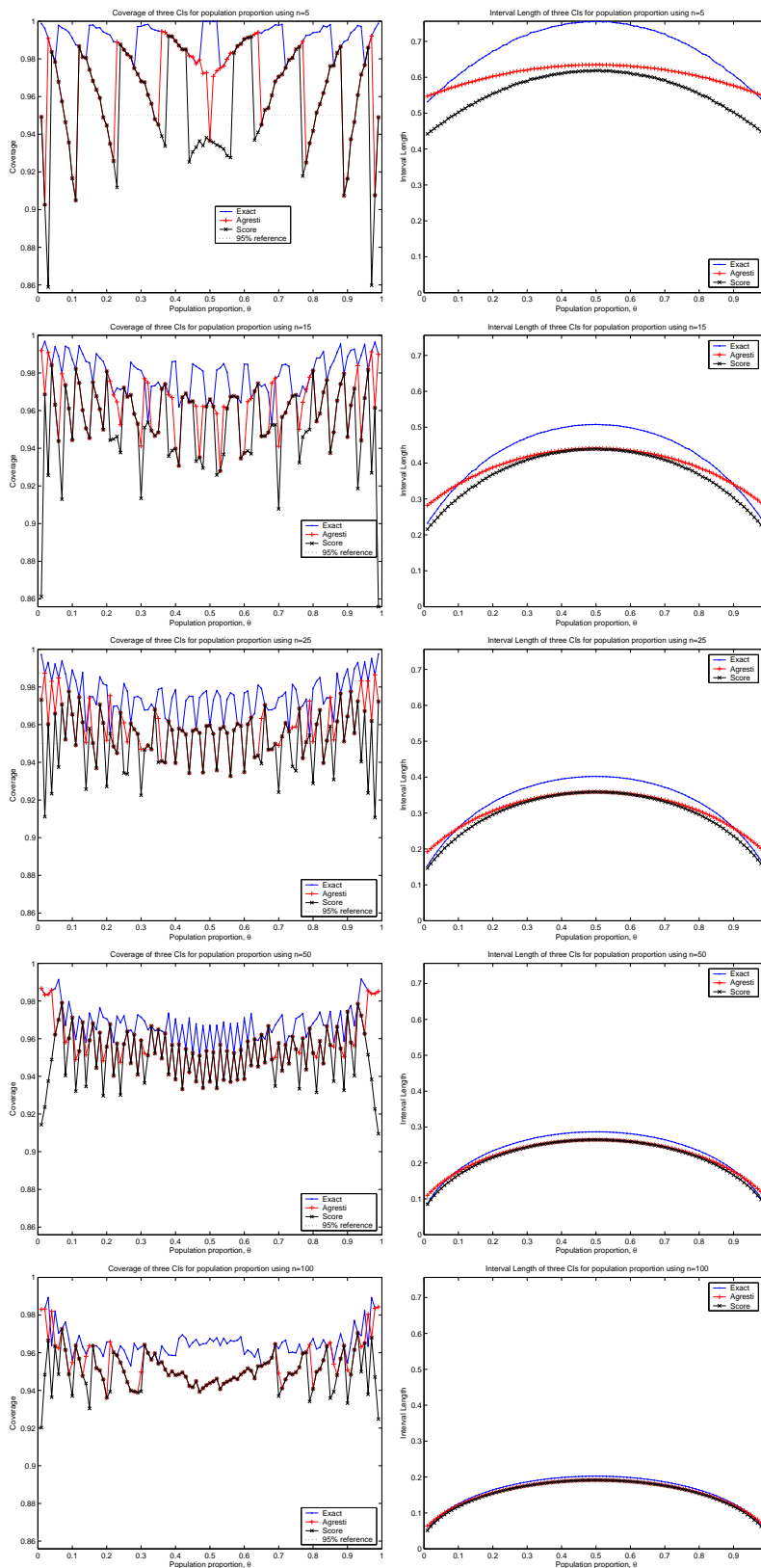


Figure 2: 2b. Binomial Confidence Intervals

Exercise 3 *Winsorized Means.***(a)** *Write function.*Function `winsor.m` to compute Winsorized means in Appendix.**(b)** *Comparison of Winsorized means.*

For a set of symmetric distributions with varying kurtosis, I am interested in the Winsorized (95%) confidence interval coverage and interval length, and the difference between the mean and Winsorized means with different Winsorizing trimming fractions. To study this, I executed a full factorial design with 5 distributions, 10 Winsorizing trimming fractions, 9 sample sizes, and 500 repetitions. The Winsorizing trimming fractions are chosen to cover the entire range possible. The sample sizes are chosen to give both small sample and asymptotic estimates. The design space is provided in the table below.

Distribution	Winsorizing Fraction	Sample size
uniform(-0.5, 0.5)	0.00	10
Normal(0, 1)	0.05	20
t(4)	0.10	30
Cauchy = t(1)	0.15	50
double exponential(1)	0.20	100
	0.25	200
	0.30	500
	0.35	1000
	0.40	10000
	0.45	

Code executing the design in Appendix.

The plots in Figure 3 on page 7 summarize the results of the study. Each row summarizes a distribution. Columns summarize coverage, interval length, or difference between mean and Winsorized mean on the vertical axis. Sample size is summarized with individual lines in each plot. The Winsorized trimming fraction in on the horizontal axis.

The coverage rate of the Winsorized confidence interval remains on target for the Uniform, Normal, and t_4 distributions, is wildly low for small trimming fractions for the Cauchy, and is systematically low for the double exponential.

The confidence interval length tends to be short with Winsorized trimming fractions from 0.1 to 0.35. For all five distributions the length increases dramatically after 0.35, and for the Cauchy, the interval length is also longer without a trimming fraction, which is expected for the Cauchy since extreme outliers are typical.

The difference between the mean and Winsorized mean for the uniform, normal, t_4 , and double exponential distributions remains extremely close to 0, but is more variable with larger trimming fractions. As expected, the Cauchy (t_0) distribution occasionally has extremely different values for the mean and Winsorized mean, since extreme outliers are characteristic for this distribution.

A recommendation for a general trimming fraction is 0.25. This fraction has short intervals that have roughly the correct coverage.

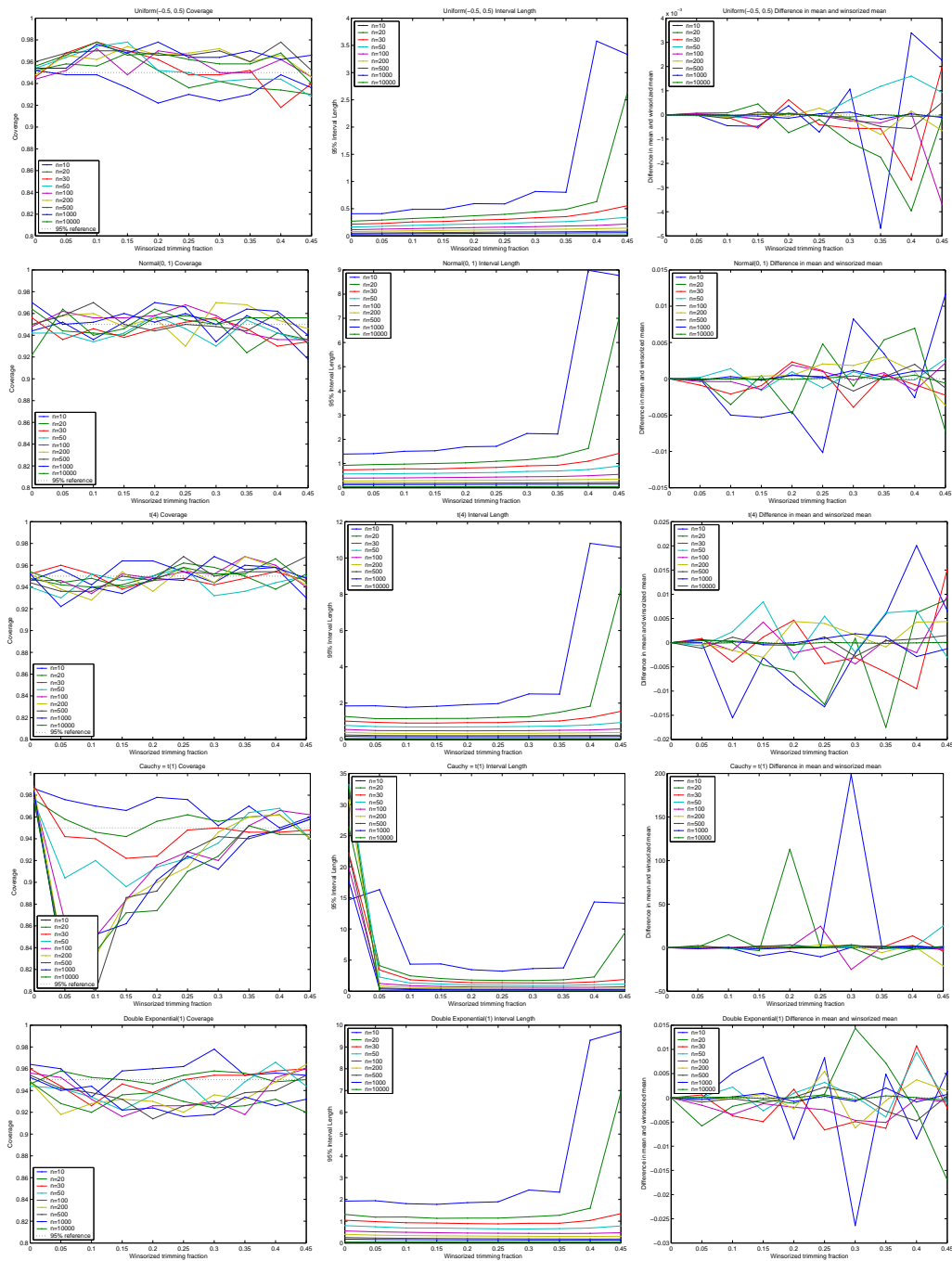


Figure 3: 3b. Winsorized Mean Coverage, Interval Length, and difference between mean and Winsorized mean for five distributions

Appendix

Matlab code used for the analysis above

```

%%% Exercise 1 =====
% 1a =====
function [P,pval]=pearsonp(x,theta0)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pearsonp, Stat 579 Statistical Computing II Homework 3
% Create a M-file to calculate the pearson P statistic,
% and p-value under chi2,
% input (x,theta0) where
% x is a matrix with row vectors of counts (each row an observation), and
% theta0 is a row vector of proportions under H0.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:
% Erik Barry Erhardt, M.S., Ph.D. student Statistics
% Dept of Math & Stats, Univ. of New Mexico, Albuquerque, NM 87131, U.S.A.
% Office: Humanities 328, MSC03 2150
% erike AT stat.unm.edu http://www.stat.unm.edu/~erike/
% (505)277-0757 Fax: (505)277-5505
% American Statistical Association Albuquerque Chapter Representative
%
% Date: 2/18/06
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin~=2;error('PEARSONP requires 2 arguments');return;end;
xs=size(x);ts=size(theta0);
if xs(2)~=ts(2);error('Number columns for x and theta0 must be the same');return;end;
if length(find(x<0)) > 0;error('Counts in x must all be nonnegative');return;end;
if length(find(theta0<=0)) > 0 | length(find(theta0>1)) > 0;error('Proportions in theta0 must between 0 and 1');return
eps=1e-10;
if abs(sum(theta0)-1) > eps;error('Sum of theta0 vector is not 1');return;end;
m=sum(x,2); % sum of counts in x
k=xs(2); % number of categories
P=sum((x-m*theta0.*ones(xs)).^2)./(m*theta0,2); % Pearson P statistic
pval=1-chi2cdf(P,k-1); % p-value based on chi2 with k-1 df
% EOF

theta0=[.1 .2 .4 .2 .1];
x=[16 50 31 11 9];
[P,Ppval] =pearsonp(x,theta0);
% 1b =====
function x=multigen(m,theta0,n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% multigen (efficient), Stat 579 Statistical Computing II Homework 3
% Create a M-file to simulate n samples from a multinomial distribution
% with a total count of m,
% theta0 is a row vector of proportions.
% input (m,theta0) where
% m is a scaler, and
% theta0 is a row vector of proportions under H0.
% returns a matrix with each sample in a row.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:
% Erik Barry Erhardt, M.S., Ph.D. student Statistics
% Dept of Math & Stats, Univ. of New Mexico, Albuquerque, NM 87131, U.S.A.
% Office: Humanities 328, MSC03 2150
% erike AT stat.unm.edu http://www.stat.unm.edu/~erike/
% (505)277-0757 Fax: (505)277-5505
% American Statistical Association Albuquerque Chapter Representative
%
% Date: 2/18/06
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin<2;error('MULTRND requires 2 arguments');return;end;
if nargin==2;n=1;end;
if length(find(theta0<=0)) > 0 | length(find(theta0>1)) > 0;error('Proportions in theta0 must between 0 and 1');return
eps=1e-10;
if abs(sum(theta0)-1) > eps;error('Sum of theta0 vector is not 1');return;end;
t = [0 cumsum(theta0)]; % partitions [0,1] into sections each theta(i) long.
u = unifrnd(0,1,n,m); % generates m unif(0,1) random deviates
x = histc(u',t)'; % counts the number of the m deviates in u which fall into the k bins specified by the partions in t
x = x(:,1:length(theta0)); % remove k+1 entry of x, which is the count of >1, of which there are none.
% EOF

```

```

theta0=[.1 .2 .4 .2 .1];
x=[16 50 31 11 9];
[P0,Ppval0] =pearsonp(x,theta0); % reference value
% simulation for parametric bootstrap evaluation of the p-value
n=1e5;
m=117;
theta0=[.1 .2 .4 .2 .1];
x=multigen(m,theta0,n);
[P,Ppval] =pearsonp(x,theta0); % simulated values
Ppval_bs = sum((P>P0))/n % bootstrap p-value

% plot sampling distribution and overlay chi2(4) distribution
bins=100;
hist(P,bins)
[N,bin_centers]=hist(P,bins);
figure(1);clf;
lower=0;
upper=40;
ind=find((lower<bin_centers) & (bin_centers<upper));
vert=(N/n)*(length(ind)/(upper-lower))*1.2;
bar(bin_centers(ind),vert(ind),1)
hold on;
xx=lower:0.001:upper;yy=chi2pdf(xx,4);
plot(xx,yy);
legend('\chi^2_4',0)
axis([lower upper 0 1.1*max([vert(ind) yy])])
print -depsc2 sc2hw03_1b1.eps

figure(2);clf;
lower=10;
upper=30;
ind=find((lower<bin_centers) & (bin_centers<upper));
vert=(N/n)*(length(ind)/(upper-lower))*1;
bar(bin_centers,vert,1)
hold on;
xx=lower:0.001:upper;yy=chi2pdf(xx,4);
plot(xx,yy);
legend('\chi^2_4',0)
axis([lower upper 0 1.1*max(vert(ind))])
print -depsc2 sc2hw03_1b2.eps

%%% Exercise 2 =====
% 2a =====
function [est,pL,pR]=pciexact(n,y,alpha)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pciexact, Stat 579 Statistical Computing II Homework 3
% Create a M-file to calculate the exact CI for population proportion
% (Leemis & Trivedi, TAS, v.50,1, pp. 63-68, Feb 1996)
% input (n,y,alpha) where
% n is a vector or scalar of sample sizes,
% y is a vector of successes,
% alpha is a vector or scalar of significance levels (0.05 for 95% CI)
% ex. [est,pL,pU]=pciexact(100,20,.05)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:
% Erik Barry Erhardt, M.S., Ph.D. student Statistics
% Dept of Math & Stats, Univ. of New Mexico, Albuquerque, NM 87131, U.S.A.
% Office: Humanities 328, MSC03 2150
% erike AT stat.unm.edu http://www.stat.unm.edu/~erike/
% (505)277-0757 Fax: (505)277-5505
% American Statistical Association Albuquerque Chapter Representative
%
% Date: 3/27/2006
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin<2;error('PCIEXACT requires 2 arguments');return;end;
if nargin==2; alpha = 0.05; warning('Using default: alpha=0.05');end;
if (sum(size(y)<size(n))); error('n vector larger than y vector');return;end;
if (sum(y>n)); error('values in y larger than n');return;end;
est=y./n; % estimate
pL = zeros(size(y));
FcritL = ones(size(y));
il = find(y>0);

```

```

FcritL(il) = finv(alpha./2,2.*y(il),2.*(n-y(il)+1)); % left F critical value
pL(il) = 1./((1+(n-y(il)+1)./(y(il).*FcritL(il)))); % left CI bound
pR = ones(size(y));
FcritR = ones(size(y));
ir = find(y<n);
FcritR(ir) = finv(1-alpha./2,2.*(y(ir)+1),2.*(n-y(ir))); % right F critical value
pR(ir) = 1./((1+(n-y(ir))./(y(ir)+1).*FcritR(ir))); % right CI bound
% EOF

function [est,pL,pR]=pciagresti(n,y,alpha)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pciagresti, Stat 579 Statistical Computing II Homework 3
% Create a M-file to calculate the Agresti add 2 CI for population proportion
% input (n,y,alpha) where
% n is a vector or scalar of sample sizes,
% y is a vector of successes,
% alpha is a vector or scalar of significance levels (0.05 for 95% CI)
% ex. [est,pL,pU]=pciagresti(100,20,.05)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:
% Erik Barry Erhardt, M.S., Ph.D. student Statistics
% Dept of Math & Stats, Univ. of New Mexico, Albuquerque, NM 87131, U.S.A.
% Office: Humanities 328, MSC03 2150
% erike AT stat.unm.edu http://www.stat.unm.edu/~erike/
% (505)277-0757 Fax: (505)277-5505
% American Statistical Association Albuquerque Chapter Representative
%
% Date: 3/27/2006
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin<2;error('PCIAGRESTI requires 2 arguments');return;end;
if nargin==2; alpha = 0.05; warning('Using default: alpha=0.05');end;
if (sum(size(y)<size(n))); error('n vector larger than y vector');return;end;
if (sum(y>n)); error('values in y larger than n');return;end;

add2 = abs(norminv(alpha/2));
est = (y+add2)./(n+2*add2); % estimate
moe = norminv(alpha/2).*sqrt((est.*(1-est))./(n+2.*add2)); % margin of error
pL = est+moe; % left CI bound
pR = est-moe; % right CI bound
% EOF

function [est,pL,pR]=pciscore(n,y,alpha)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pciscore, Stat 579 Statistical Computing II Homework 3
% Create a M-file to calculate the Score CI for population proportion
% Binomial (Score) CI using normal approximation with known std dev.
% Interval endpoints solve a quadratic equation
% input (n,y,alpha) where
% n is a vector or scalar of sample sizes,
% y is a vector of successes,
% alpha is a vector or scalar of significance levels (0.05 for 95% CI)
% ex. [est,pL,pU]=pciscore(100,20,.05)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:
% Erik Barry Erhardt, M.S., Ph.D. student Statistics
% Dept of Math & Stats, Univ. of New Mexico, Albuquerque, NM 87131, U.S.A.
% Office: Humanities 328, MSC03 2150
% erike AT stat.unm.edu http://www.stat.unm.edu/~erike/
% (505)277-0757 Fax: (505)277-5505
% American Statistical Association Albuquerque Chapter Representative
%
% Date: 3/27/2006
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin<2;error('PCISCORE requires 2 arguments');return;end;
if nargin==2; alpha = 0.05; warning('Using default: alpha=0.05');end;
if (sum(size(y)<size(n))); error('n vector larger than y vector');return;end;
if (sum(y>n)); error('values in y larger than n');return;end;

est = y./n; % estimate
z=abs(norminv(alpha/2)); % critical value
temp = z.*sqrt( est.*(1-est)./n + .25.*(z./n).^2 );
pL = ( est + .5.*z.*z./n - temp )./( 1 + z.*z./n ); % left CI bound
pR = ( est + .5.*z.*z./n + temp )./( 1 + z.*z./n ); % right CI bound
% EOF

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation
clear
M=1e4; % number of simulations
theta = 0.01:0.01:0.99; % theta values
n = [5 15 25 50 100]; % sample sizes
methods=1:3; % three methods: exact, agresti, score
results = zeros(length(methods),length(n),length(theta),4); % store results (method, sample size, theta, [cov len len_
cov = zeros(M,length(methods)); % working vector for coverage
len = zeros(M,length(methods)); % working vector for length
alpha = 0.05; % significance level

tic
for i_n = n
    [toc i_n]
    for i_theta = theta
        [toc i_n i_theta]
        % determine indicies
        ind_n=find(n==i_n);
        ind_theta=find(theta==i_theta);
        % random deviates from binomial distribution
        y = binornd(i_n, i_theta, M, 1);
        % calculate estimates and confidence intervals
        [est_e,lci_e,uci_e] = pciexact( i_n, y, alpha);
        [est_a,lci_a,uci_a] = pciagresti(i_n, y, alpha);
        [est_s,lci_s,uci_s] = pciscore( i_n, y, alpha);
        % indicate whether the interval covers the true value of the parameter
        cov(:,1) = ((lci_e<i_theta) & (i_theta<uci_e));
        cov(:,2) = ((lci_a<i_theta) & (i_theta<uci_a));
        cov(:,3) = ((lci_s<i_theta) & (i_theta<uci_s));
        % calculate the length of the intervals
        len(:,1) = uci_e-lci_e;
        len(:,2) = uci_a-lci_a;
        len(:,3) = uci_s-lci_s;
        % store results of coverage, interval length, and CI for interval length
        results(:,ind_n,ind_theta,1)=mean(cov)';
        results(:,ind_n,ind_theta,2)=mean(len)';
        results(:,ind_n,ind_theta,3)=mean(len)-2*std(cov)/sqrt(M);
        results(:,ind_n,ind_theta,4)=mean(len)+2*std(cov)/sqrt(M);
    end
end
toc
save res results
figure(1);clf;
for pn=1:length(n); % loop over plot numbers, pn
    i_n=pn; % index of n
    % Coverage plots
    results_temp = squeeze(results(:,i_n,:,1))'; % condense results into easy matrix to plot (1=coverage)
    plot(theta',results_temp(:,1),'b.-', theta',results_temp(:,2),'r+-', theta',results_temp(:,3),'kx-', theta',ones(1,length(theta)),0);
    legend('Exact','Agresti','Score','95% reference',0); % legend
    xlabel('Population proportion, \theta'); ylabel('Coverage'); % label axes
    tit=strcat('Coverage of three CIs for population proportion using n=',num2str(n(i_n))); title(tit); % title plot
    axis([0 1 min(min(min(results(:, :, 1)))) 1]); % set axes
    plot_name = strcat('sc2hw03b_n',num2str(i_n),'c.eps'); % plot name
    print(gcf, '-depsc2', plot_name); % print plot
    %pause
    % length plots
    results_temp = squeeze(results(:,i_n,:,2))'; % condense results into easy matrix to plot (2=length)
    plot(theta',results_temp(:,1),'b.-', theta',results_temp(:,2),'r+-', theta',results_temp(:,3),'kx-'); % plot length
    legend('Exact','Agresti','Score',0); % legend
    xlabel('Population proportion, \theta'); ylabel('Interval Length'); % label axes
    tit=strcat('Interval Length of three CIs for population proportion using n=',num2str(n(i_n))); title(tit); % title
    axis([0 1 0 max(max(max(results(:, :, 2))))]); % set axes
    plot_name = strcat('sc2hw03b_n',num2str(i_n),'l.eps'); % plot name
    print(gcf, '-depsc2', plot_name); % print plot
    %pause
end;
for pn=1:length(n); % loop over plot numbers, pn
    i_n=pn; % index of n
    results_temp = squeeze(results(:,i_n,:,1))'; % condense results into easy matrix to plot (1=coverage)
    cov_m = mean(results_temp);

```

```

cov_s = std(results_temp);
cov_l = cov_m-2*cov_s/sqrt(M);
cov_u = cov_m+2*cov_s/sqrt(M);
disp(['Coverage for n=',num2str(n(i_n)) ])
[cov_m' cov_s' cov_l' cov_u']

results_temp = squeeze(results(:,i_n,:,2))'; % condense results into easy matrix to plot (2=length)
cov_m = mean(results_temp);
cov_s = std(results_temp);
cov_l = cov_m-2*cov_s/sqrt(M);
cov_u = cov_m+2*cov_s/sqrt(M);
disp(['Length for n=',num2str(n(i_n)) ])
[cov_m' cov_s' cov_l' cov_u']

end;

%%% Exercise 3 =====
% 3a =====

function [winsor_mean, winsor_std, winsor_n_center, winsor_se, winsor_lci, winsor_uci] = winsor(x, winsor_fraction, alpha)
% winsor, Stat 579 Statistical Computing II Homework 2
% Create a M-file to compute the Winsorized mean of a sample
% input (data, winsor_fraction) where
% x is a matrix, each column is a sample of size n.
% winsor_fraction is a number between 0 and 0.5 the fraction to winsorize on both side of the median.
% alpha is the significance level for the confidence interval
% output [winsor_mean, winsor_std, winsor_n_center] where
% winsor_mean is the winsorized mean.
% winsor_std is the winsorized standard deviation.
% winsor_n_center is the number that weren't winsorized in the center of the sample,
% also the degrees of freedom.
% winsor_se is the winsorized standard error.
% winsor_lci is the lower 100(1-alpha)% confidence limit
% winsor_uci is the upper 100(1-alpha)% confidence limit
% Author:
% Erik Barry Erhardt, M.S., Ph.D. student Statistics
% Dept of Math & Stats, Univ. of New Mexico, Albuquerque, NM 87131, U.S.A.
% Office: Humanities 328, MSC03 2150
% erike AT stat.unm.edu http://www.stat.unm.edu/~erike/
% (505)277-0757 Fax: (505)277-5505
% American Statistical Association Albuquerque Chapter Representative
%
% Date: 3/30/2006
%
% WINSOR requires 2 arguments';return;end;
if nargin<2;error('WINSOR requires 2 arguments');return;end;
if nargin==2; alpha = 0.05; warning('Using default: alpha=0.05');end;

[n,n_samples] = size(x); % sample size and number of samples
n_trim_each_side = floor(winsor_fraction*n); % number to winsorize on each side
half = floor(.5*n);

if (n_trim_each_side >= half ) | (n <= 4) % feasibility check
    fprintf(1,'\n trim fraction (%f) too large or n (%d) too small: retry \n', n_trim_each_side, n);
    winsor_mean = 0; winsor_std=0; winsor_n_center=0;
    return
end

if n_trim_each_side == 0; % no trimming/winsorizing to do
    winsor_mean = mean(x); % winsorized mean
    winsor_std = std(x); % winsorized standard deviation
    winsor_n_center = n; % winsorized sample size
    winsor_se = sqrt((n-1)/(n*(winsor_n_center-1)))*winsor_std; % winsorized standard error
    winsor_lci = winsor_mean - tinv(1-alpha/2, winsor_n_center-1)*winsor_se;% lower confidence limit
    winsor_uci = winsor_mean + tinv(1-alpha/2, winsor_n_center-1)*winsor_se;% upper confidence limit
else;
    x_sorted = sort(x); % sorted data
    winsor_n_center = n - 2*n_trim_each_side; % winsorized sample size
    x_center = x_sorted((n_trim_each_side+1):(n-n_trim_each_side),:); % trimmed center of the sample
    x_lower = ones(n_trim_each_side,1)*x_sorted((n_trim_each_side+1),:); % min in trimmed sample
    x_upper = ones(n_trim_each_side,1)*x_sorted((n-n_trim_each_side),:); % max in trimmed sample
    x_winsor = [x_lower;x_center;x_upper]; % construct the winsorized sample
    winsor_mean = mean(x_winsor); % winsorized mean
    winsor_std = std(x_winsor); % winsorized standard deviation
    % winsor_se = sqrt((n-1)/(n*(winsor_n_center-1)))*winsor_std; % winsorized standard error WRONG
    winsor_se = ((n-1)/(winsor_n_center-1))*(1/sqrt(n))*winsor_std; % winsorized standard error
    winsor_lci = winsor_mean - tinv(1-alpha/2, winsor_n_center-1)*winsor_se;% lower confidence limit
    winsor_uci = winsor_mean + tinv(1-alpha/2, winsor_n_center-1)*winsor_se;% upper confidence limit
end

```

```

end
winsor_n_center = winsor_n_center*ones(1,n_samples); % return the winsorized sample size as a vector
% EOF

% 3b
=====
clear
M=5e2; % number of simulations
win_frac = [0.00:0.05:0.45]; % winsorized trimming fractions
n = [10 20 30 50 100 200 500 1000 10000]; % sample sizes
dist=[1:5]; % distributions
results = zeros(length(dist),length(n),length(win_frac),3); % store results (distribution, sample size, win_frac, [cov
cov = zeros(M,1); % working vector for coverage
len = zeros(M,1); % working vector for length
mean_diff = zeros(M,1); % working vector for mean_diff (difference of winsorized a
alpha = 0.05; % significance level

tic
for i_n = n
    for i_win_frac = win_frac
        for i_dist = dist
            [toc i_n i_win_frac i_dist]
            % determine indicies
            ind_n=find(n==i_n);
            ind_win_frac=find(win_frac==i_win_frac);
            ind_dist=find(dist==i_dist);

            % draw deviates from one of the distributions
            if i_dist==1;
                x = unifrnd(-.5,.5,i_n,M); % uniform(-0.5, 0.5)
            elseif i_dist==2;
                x = normrnd(0,1,i_n,M); % Normal(0, 1)
            elseif i_dist==3;
                x = trnd(4,i_n,M); % t(4)
            elseif i_dist==4;
                x = trnd(1,i_n,M); % Cauchy = t(1)
            elseif i_dist==5;
                x = exprnd(1,i_n,M) .* (2.*( rand(i_n,M) < .5 ) - 1); % double exponential(1)
            else
                warning('No Distribution to use.');
```

```

% Coverage plots
figure(1);clf;
results_temp = squeeze(results(i_dist,:,:),1)'; % condense results into easy matrix to plot (1=coverage)
plot(win_frac',results_temp(:,:),'.-', win_frac',ones(length(win_frac),1)*(1-alpha),'k:'); % plot coverages
legend('n=10','n=20','n=30','n=50','n=100','n=200','n=500','n=1000','n=10000','95% reference',3); % legend
xlabel('Winsorized trimming fraction'); ylabel('Coverage'); % label axes
tit=strcat(dist_name,' Coverage'); title(tit); % title plot
axis([0 0.45 min(min(min(results(:,:,:),1))) 1]); % set axes
plot_name = strcat('sc2hw03c_d',num2str(i_dist),'c.eps'); % plot name
print(gcf, '-depsc2', plot_name); % print plot

% pause

% Interval Length plots
figure(1);clf;
results_temp = squeeze(results(i_dist,:,:),2)'; % condense results into easy matrix to plot (2=length)
plot(win_frac',results_temp(:,:),'.-'); % plot lengths
legend('n=10','n=20','n=30','n=50','n=100','n=200','n=500','n=1000','n=10000',2); % legend
xlabel('Winsorized trimming fraction'); ylabel('95% Interval Length'); % label axes
tit=strcat(dist_name,' Interval Length'); title(tit); % title plot
axis([0 0.45 0 max(max(max(results(:,:,:),2)))]); % set axes
plot_name = strcat('sc2hw03c_d',num2str(i_dist),'l.eps'); % plot name
print(gcf, '-depsc2', plot_name); % print plot

% pause

% Difference in mean and winsorized mean plots
figure(1);clf;
results_temp = squeeze(results(i_dist,:,:),3)'; % condense results into easy matrix to plot (2=length)
plot(win_frac',results_temp(:,:),'.-', win_frac',zeros(length(win_frac),1),'k:'); % plot lengths
legend('n=10','n=20','n=30','n=50','n=100','n=200','n=500','n=1000','n=10000',2); % legend
xlabel('Winsorized trimming fraction'); ylabel('Difference in mean and winsorized mean'); % label axes
tit=strcat(dist_name,' Difference in mean and winsorized mean'); title(tit); % title plot
axis([0 0.45 0 max(max(max(results(:,:,:),2)))]); % set axes
plot_name = strcat('sc2hw03c_d',num2str(i_dist),'d.eps'); % plot name
print(gcf, '-depsc2', plot_name); % print plot

% pause
end;
for pn=1:length(dist); % loop over plot numbers, pn
    i_dist=pn; % index of n
    results_temp = squeeze(results(i_dist,:,:),1)'; % condense results into easy matrix to plot (1=coverage)
    cov_m = mean(results_temp);
    cov_s = std(results_temp);
    cov_l = cov_m-2*cov_s/sqrt(M);
    cov_u = cov_m+2*cov_s/sqrt(M);
    disp(['Coverage for dist=',num2str(dist(i_dist)) ])
    [cov_m' cov_s' cov_l' cov_u']
end;
for pn=1:length(dist); % loop over plot numbers, pn
    i_dist=pn; % index of n
    results_temp = squeeze(results(i_dist,:,:),2)'; % condense results into easy matrix to plot (2=length)
    cov_m = mean(results_temp);
    cov_s = std(results_temp);
    cov_l = cov_m-2*cov_s/sqrt(M);
    cov_u = cov_m+2*cov_s/sqrt(M);
    disp(['Length for dist=',num2str(dist(i_dist)) ])
    [cov_m' cov_s' cov_l' cov_u']
end;
for pn=1:length(dist); % loop over plot numbers, pn
    i_dist=pn; % index of n
    results_temp = squeeze(results(i_dist,:,:),3)'; % condense results into easy matrix to plot (2=length)
    cov_m = mean(results_temp);
    cov_s = std(results_temp);
    cov_l = cov_m-2*cov_s/sqrt(M);
    cov_u = cov_m+2*cov_s/sqrt(M);
    disp(['Diff in means dist=',num2str(dist(i_dist)) ])
    [cov_m' cov_s' cov_l' cov_u']
end;

```