

Stat 579 Statistical Computing II Homework 1

Erik Erhardt

February 5, 2006

All code can be found in the Appendix.

Exercise 1 `cubicfn`

Function is provided in appendix.

Exercise 2 *Cubic function examples.*

(a) $y_1 = 0 + x + x^2 + 0x^3$

(b) $y_2 = 1 - 3x - x^2 + 3x^3$

(c) *both*

The plots in Figure 1 on page 2 illustrate the use of `cubicfn`.

Exercise 3 `polyfn`

Function is provided in appendix.

The plots in Figure 2 on page 2 illustrate the use of `polyfn`.

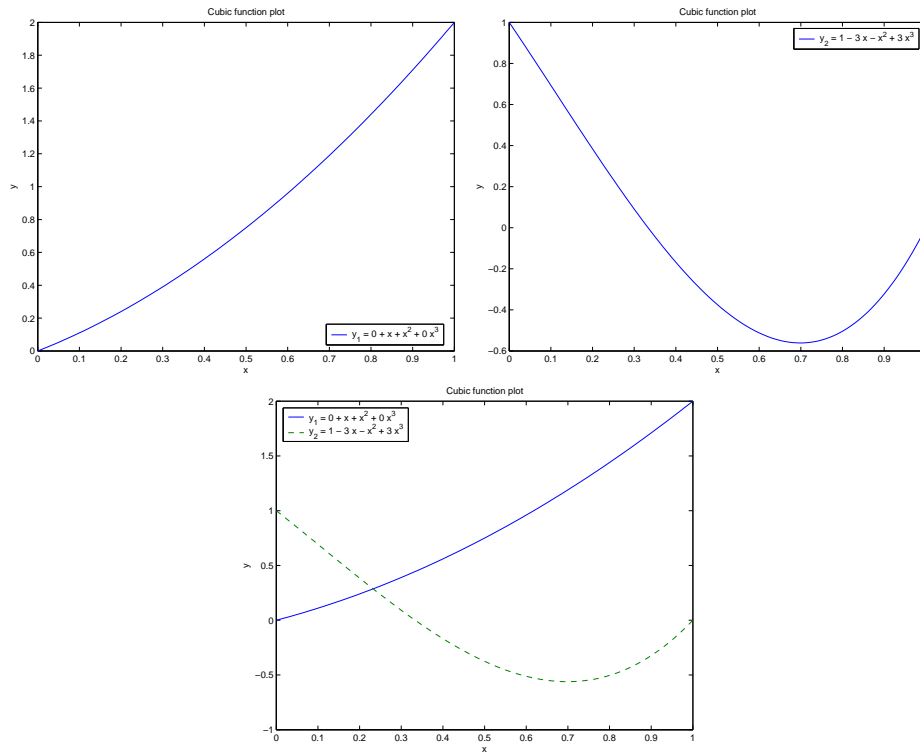


Figure 1: Exercise 2a, b, c, Cubic function.

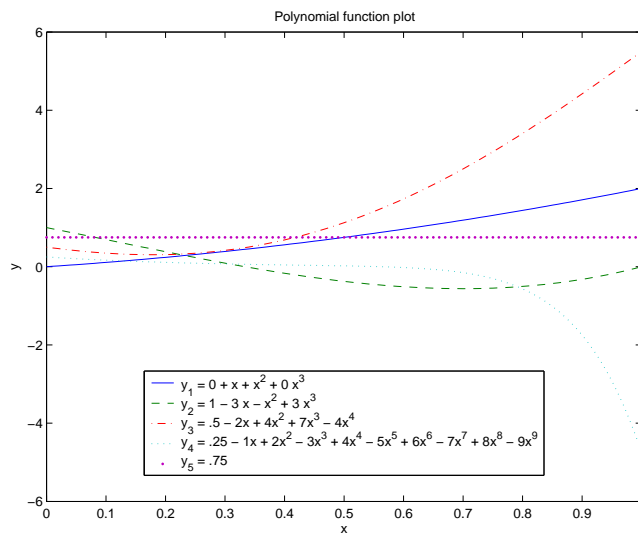


Figure 2: Exercise 3, Poly function.

Exercise 4 help stats

The `*fit` functions give MLEs for the parameters of distributions

```
>> [phat,pci]=betafit(betarnd(12, 3, 1, 500), .05)
phat =
    12.6423    3.1124
pci =
    11.1834    2.7902
    14.1011    3.4346
```

The `*pdf` and `*cdf` functions give the density function and distribution function for the named distribution.

The `*inv` functions give the critical values for a given left-tail probability.

```
>> crit=betainv(.5,12,3)
crit =
    0.8135
```

The `*rnd` functions generate random numbers for the named distribution.

The `*stat` functions give the first two moments of the named distribution.

```
>> [m,v]=betastat(12,3)
m =
    0.8000
v =
    0.0100
```

The `*like` functions give the negative log-likelihood function for the given parameters for a set of data for the named distribution. It can also return the observed Fisher's information.

```
>> [logl, avar]=betalike([12,3],betarnd(12,3,1,500))
logl =
 -458.8368
avar =
    0.5054    0.1055
    0.1055    0.0276
```

The `mean`, `var`, `skewness`, and `kurtosis` functions give the first four moments of observed data.

```
>> m=mean(betarnd(12,3,1,500))
m =
    0.8080
>> v=var(betarnd(12,3,1,500))
v =
    0.0107
>> s=skewness(betarnd(12,3,1,500))
s =
 -0.5562
>> k=kurtosis(betarnd(12,3,1,500))
k =
    2.7468
```

The `kruskalwallis` function gives the nonparametric one-way ANOVA, plotting boxplots of the samples.

```
>> [p,anovatab,stats]=kruskalwallis([betarnd(12,3,1,10)',betarnd(6,6,1,10)',betarnd(3,12,1,10)']);title('Kruskal-Wallis')
p =
    9.4422e-006
anovatab =
    'Source'      'SS'      'df'      'MS'      'Chi-sq'      'Prob>Chi-sq'
    'Columns'    [1.7934e+003] [ 2]    [896.7000]    [23.1406]    [9.4422e-006]
    'Error'      [ 454.1000] [27]    [16.8185]     []          []
    'Total'      [2.2475e+003] [29]     []          []          []
stats =
    gnames: [3x1 char]
    n: [10 10 10]
    source: 'kruskalwallis'
    meanranks: [25.3000 14.8000 6.4000]
    sumt: 0
```

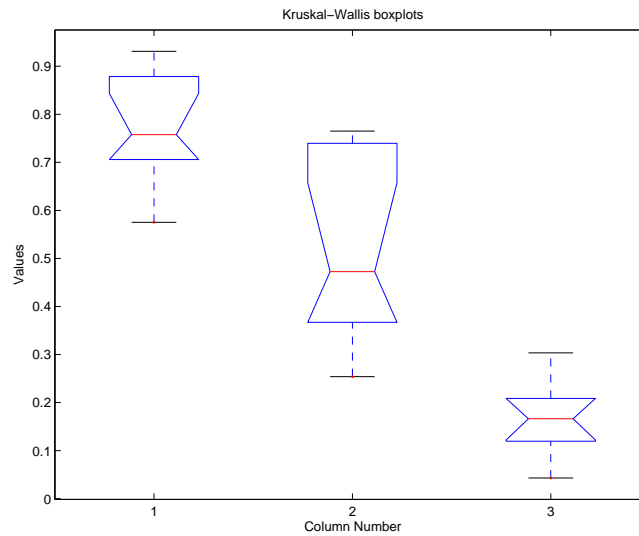


Figure 3: Kruskal-Wallis boxplots.

The plots in Figure 3 on page 4 illustrate the use of `kruskalwallis`. The `ttest` function computes a one-sample t -test.

```
>> [h,p,ci,stats]=ttest(betarnd(12,3,1,10),.7)
h =
    1
p =
    8.5997e-004
ci =
    0.7620    0.8688
stats =
    tstat: 4.8895
         df: 9
         sd: 0.0746
```

The `qqplot` function makes an empirical QQ-plot of the quantiles of a data set versus the quantiles of a standard Normal distribution.

```
>> qqplot(betarnd(12,3,1,100))
```

The plots in Figure 4 on page 5 illustrate the use of `qqplot`.

The `combnk` and `nchoosek` functions are the same, displaying all realizations of n objects in a vector taken k at a time.

```
>> combnk([1 2 3 4 5],3)
>> nchoosek([1 2 3 4 5],3)
ans =
     3     4     5
     2     4     5
     2     3     5
     2     3     4
     1     4     5
     1     3     5
     1     3     4
     1     2     5
     1     4     4
     1     2     3
```

The `perms` function is similar, but gives all permutations of a vector. This is useful when writing your own nonparametric tests.

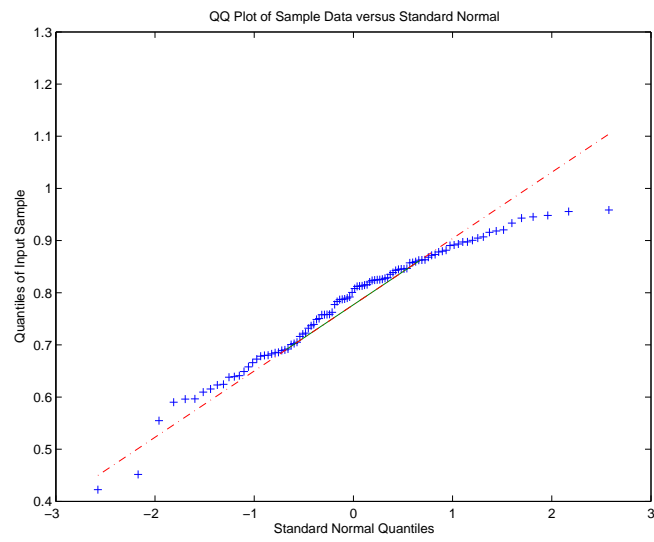


Figure 4: QQ-plot.

```
>> perms(1:3)
ans =
     3     2     1
     3     1     2
     2     3     1
     2     1     3
     1     2     2
     1     3     2
```

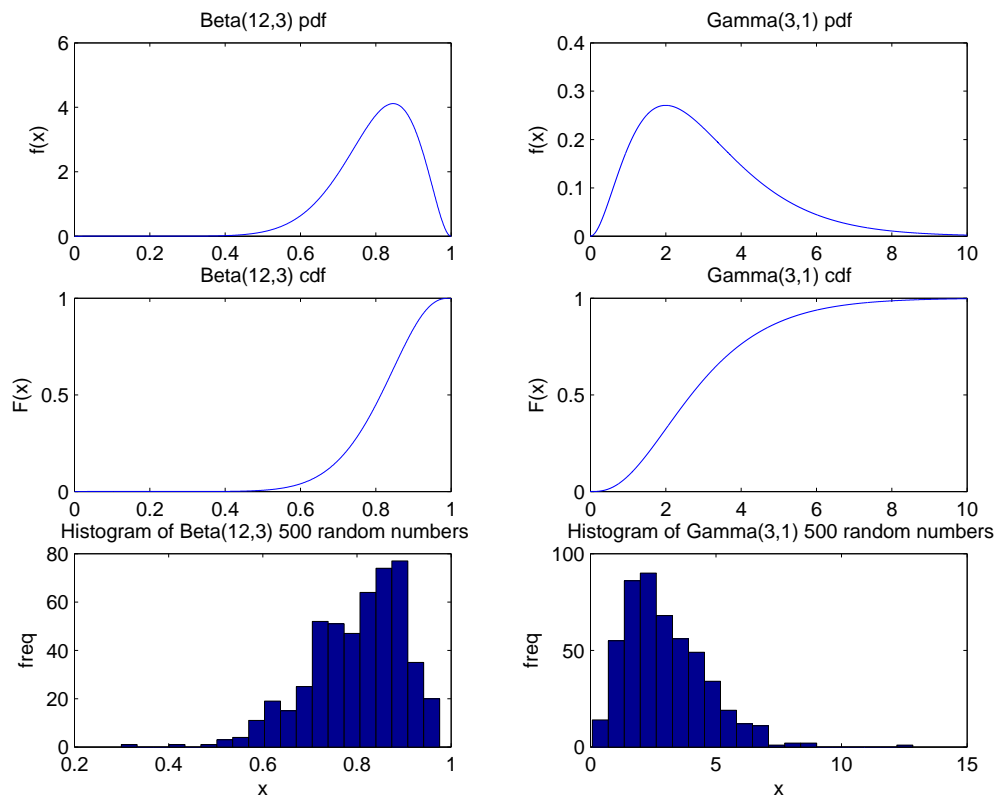


Figure 5: Exercise 5 Beta and Gamma functions.

Exercise 5 *Statistical Functions — beta and gamma distributions*

The appendix shows functions for the beta and gamma distributions: `betapdf`, `betacdf` and `betarnd`, and `gampdf`, `gamcdf`, and `gamrnd`. The plots in Figure 5 on page 6 illustrate the use of those functions.

Exercise 6

(a) The `disttool` demo allows quick distribution viewing duplicating all of the `*pdf`, `*cdf`, and `*inv` functions. The plot in Figure 6 on page 7 illustrate the use of this demo.

(b) The `randtool` demo allows quick random variable realization and viewing duplicating the `*rnd` and `hist` functions. The plot in Figure 7 on page 7 illustrate the use of this demo.

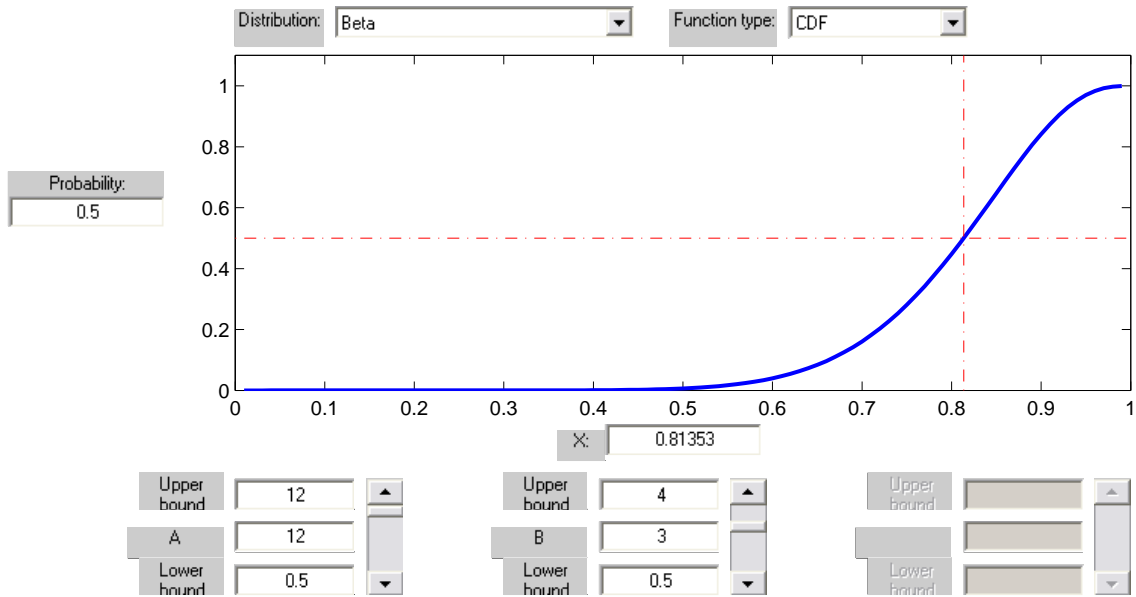


Figure 6: Exercise 6 Disttool.

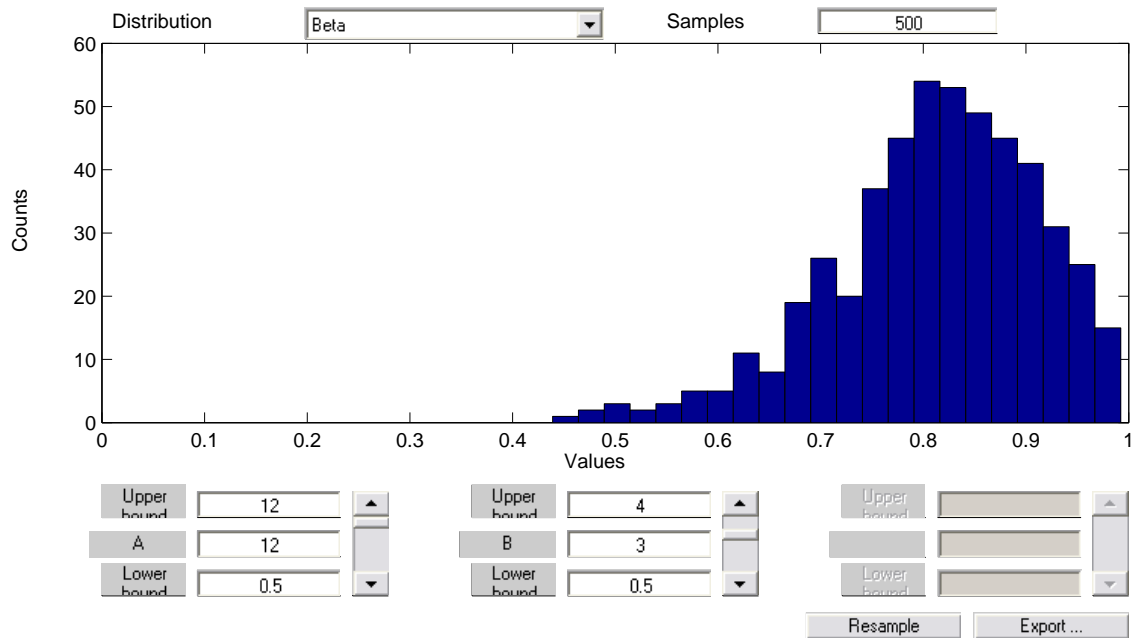


Figure 7: Exercise 6 Randtool.

Appendix

Matlab code used for the analysis above

```

%%% Exercise 1 =====
function [y]=cubicfn(a,b,c,d,x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% cubicfn, Stat 579 Statistical Computing II Homework 1#1
% Create a M-file to define a function, call it cubicfn, that takes
% input (a,b,c,d,x) where a,b,c,d are SCALARS (i.e. constants), and
% x is a MATRIX, and returns the cubic polynomial
%
%           y =  a + b*x + c*x^2 + d*x^3
%
% where the polynomial is evaluated for each element in x.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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: 1/29/06
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin ~= 5
    error('CUBICFN requires 5 arguments');
    return;
end
y =  a + b*x + c*x.^2 + d*x.^3;
%EOF

%%% Exercise 2 =====
x=0:.01:1;
y1 = cubicfn(0,1,1,0,x);
y2 = cubicfn(1,-3,-1,3,x);

%%% Exercise 2a =====
fignum=figure;clf;
plot(x,y1,'-');
title('Cubic function plot')
xlabel('x');
ylabel('y');
legend('y_1 = 0 + x + x^2 + 0 x^3',0);
fn='sc2hw1_2a';
print(fignum,'-depsc2',fn);

%%% Exercise 2b =====
fignum=figure;clf;
plot(x,y2,'-');
title('Cubic function plot')
xlabel('x');
ylabel('y');
legend('y_2 = 1 - 3 x - x^2 + 3 x^3',0);
fn='sc2hw1_2b';
print(fignum,'-depsc2',fn);

%%% Exercise 2c =====
fignum=figure;clf;
plot(x,y1,'-', x,y2,'--');
title('Cubic function plot')
xlabel('x');
ylabel('y');
legend('y_1 = 0 + x + x^2 + 0 x^3','y_2 = 1 - 3 x - x^2 + 3 x^3',0);
fn='sc2hw1_2c';
print(fignum,'-depsc2',fn);

%%% Exercise 3 =====
function [y]=polyfn(a,x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% polyfn, Stat 579 Statistical Computing II Homework 1#3
% Create a Matlab function file to define a function, call it polyfn,
% that takes input (a,x) where a is a VECTOR of ARBITRARY LENGTH and x
% is a MATRIX, and returns the polynomial
%

```

```

%           y = a(1) + a(2)*x + a(3)*x^2 + .... + a(n)*x^(n-1)
%
% where the polynomial is evaluated for each element in x. Here n is the
% the number of elements in a, or the length of the input vector a.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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: 1/29/06
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin ~= 2
    error('CUBICFN requires 2 arguments');
    return;
end
len_a=length(a);
y=zeros(1,length(x));
for i=1:length(a);
    y = y + a(i)*x.^(i-1);
end
%EOF

%%% Exercise 3a =====
x=0:.01:1;
y1 = polyfn([0,1,1,0],x);
y2 = polyfn([1,-3,-1,3],x);
y3 = polyfn([.5,-2,4,7,-4],x);
y4 = polyfn([.25,-1,2,-3,4,-5,6,-7,8,-9],x);
y5 = polyfn([.75],x);
figure(gcf);
plot(x,y1,'-', x,y2,'--', x,y3,'-.', x,y4,':', x,y5,'.');
title('Polynomial function plot')
xlabel('x');
ylabel('y');
legend('y_1 = 0 + x + x^2 + 0 x^3', 'y_2 = 1 - 3 x - x^2 + 3 x^3', 'y_3 = .5 - 2x + 4x^2 + 7x^3 - 4x^4', 'y_4 = .25 - 1x - 1x^2 + 1x^3 - 1x^4', 'y_5 = .75');
print(figure,'-depsc2',fn);

%%% Exercise 4 =====
in text

%%% Exercise 5 =====
% Beta and Gamma pdf, cdf, and rnd
x=0:.001:1;
subplot(3,2,1); plot(x,betapdf(x,12,3),'b-'); title('Beta(12,3) pdf'); xlabel(''); ylabel('f(x)');
subplot(3,2,3); plot(x,betacdf(x,12,3),'b-'); title('Beta(12,3) cdf'); xlabel(''); ylabel('F(x)');
subplot(3,2,5); n=500; hist(betarnd(12,3, 1, n), 20); title('Histogram of Beta(12,3) 500 random numbers'); xlabel('x');
x=0:.001:10;
subplot(3,2,2); plot(x,gampdf(x,3,1),'b-'); title('Gamma(3,1) pdf'); xlabel(''); ylabel('f(x)');
subplot(3,2,4); plot(x,gamcdf(x,3,1),'b-'); title('Gamma(3,1) cdf'); xlabel(''); ylabel('F(x)');
subplot(3,2,6); n=500; hist(gamrnd(3,1, 1, n), 20); title('Histogram of Gamma(3,1) 500 random numbers'); xlabel('x');

%%% Exercise 6 =====
disttool
randtool

```