## Contents



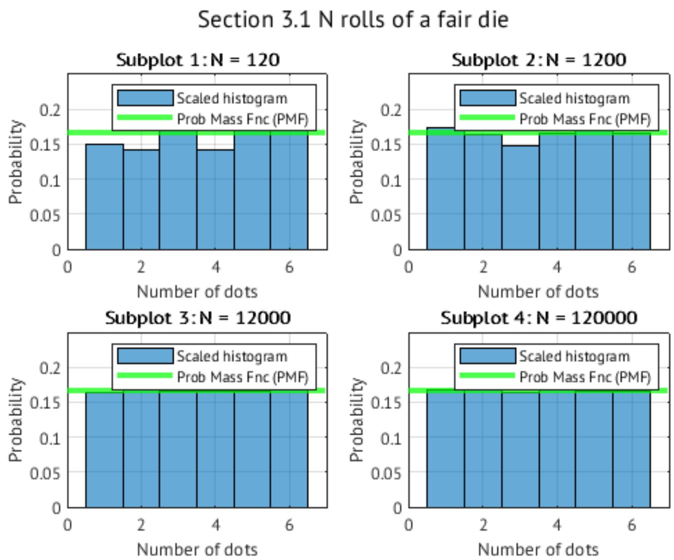
```
% S21 CMPE320 Project 1 Skeleton
%    Histograms, PDFs and PMFs
%    EFCL   1/14/2022
%
close all  % remove all existing figures (very useful to avoid confusion)
clear      % remove all existing variables, but not existing break points in scripts or functions
```

### 3.1 PMF for a single die

```
%bin_edges contain the upper and lower edges for the histogram that we will
%create. Therefore, it always has one more column than the number of bins,
%with the upper edge of one bin corresponding to the lower edge of the
%next.

bin_edges = [0.5:6.5]; % there are only 6 sides, start 0.5 less than 1 and finish 0.5 more than 6
Ntrials = [120, 1200 12000 120000]; % per the assignment
disp(' ');
disp('Section 3.1 PMF of a single fair die');
figure();

for ktrials = 1:length(Ntrials);  %loop on the number of trials

    %  set a new figure (or figures) for this number of trials
    subplot(2,2,ktrials);

    rolls = randi(6,1, Ntrials(ktrials)); % do the random trials

    sample_mean = mean(rolls); % compute the sample mean
    sample_var = var(rolls); % compute the sample variance

    %Each call to histogram will produce a plot.  You might consider
    %putting these in separate figures or subplots

    % These calls are correct, but I won't do it again.  You're CMPEs; you
    % can figure it out from here

    %Raw histogram
    hist_raw=histogram(rolls,'BinEdges',bin_edges); %histogram returns a structure, check it out!

    %Normalized histogram, note calling parameters
    hist_norm = histogram(rolls,'BinEdges',bin_edges,'Normalization','probability');


    hold on;
    % the theoretical pmf is pk = 1/6, k=1,2,...,6 because the die is fair
    %plot(1/6); % plot the theoretical ! This function didn't do what it
    %was supposed to
    yline(1/6, 'color', 'g', 'LineWidth', 3);
    hold off;

    % Professional quality plots always have axes labels
    xlabel('Number of dots');
    ylabel('Probability');
    axis([0 7 0 0.25]);  %...and always have an appropriate scale

    % ...and always have a title
    title(['Subplot ', int2str(ktrials), ': N = ', int2str(Ntrials(ktrials))]);

    %...and almost always have a grid
    grid on

    %...and a legend.
    legend('Scaled histogram','Prob Mass Fnc (PMF)');
    disp(['For ',int2str(Ntrials(ktrials)),' sample mean: ',num2str(sample_mean),' sample variance: ',num2str(sample_var)]);
end; % loop on the trials

sgtitle('Section 3.1 N rolls of a fair die');

mean_th = 3.5; % compute the theoretical mean or average
var_th = 2.9167; % compute the theoretical variance
```

```
disp(['Theoretical mean = ',num2str(mean_th),' theoretical variance: ',num2str(var_th)]);
disp('-----------');
disp(' ');

% account for the fact that you want separate plots for each section
```

```
Section 3.1 PMF of a single fair die
For 120 sample mean: 3.6417 sample variance: 2.9546
For 1200 sample mean: 3.5158 sample variance: 2.9806
For 12000 sample mean: 3.5202 sample variance: 2.926
For 120000 sample mean: 3.4952 sample variance: 2.9237
Theoretical mean = 3.5 theoretical variance: 2.9167
-----------
```

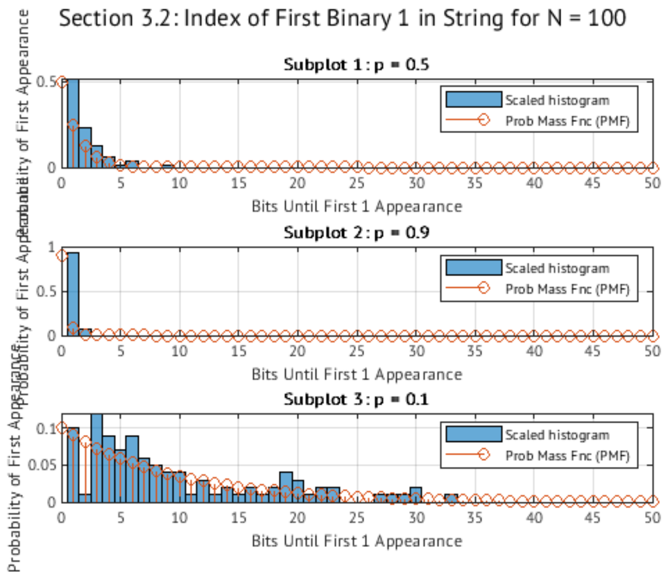


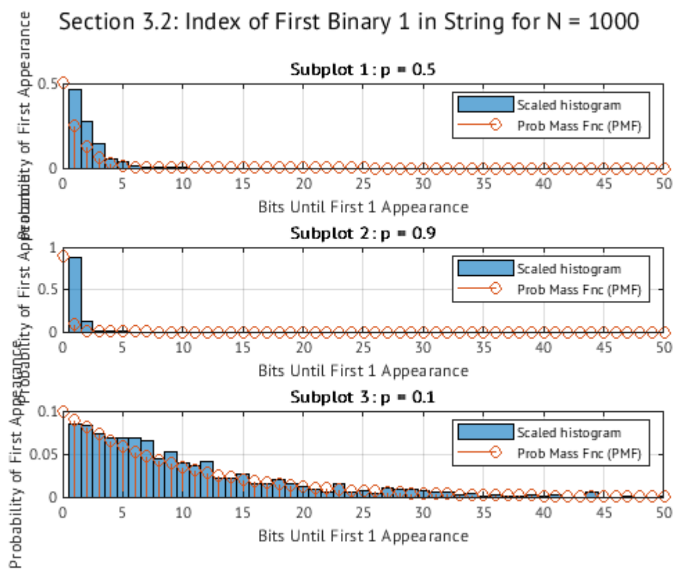

### Section 3.2 PMF for binary strings

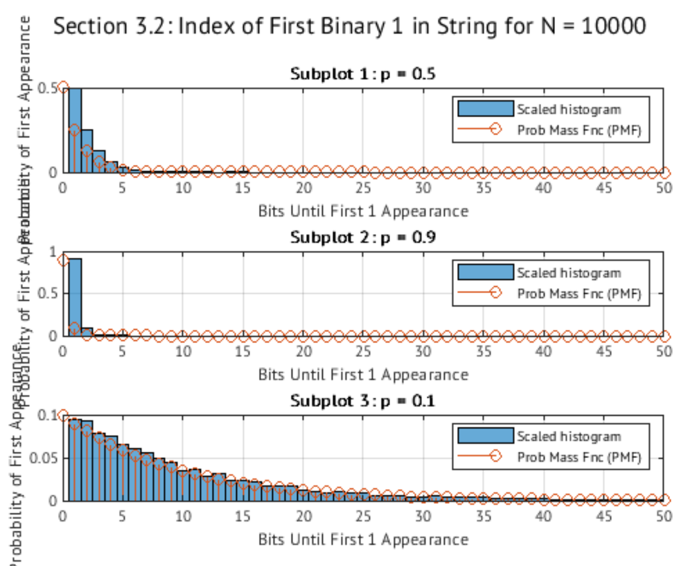
```
disp(' ');
disp('Section 3.2 PMF for Binary String');

% Don't forget new figures

n = 100;  % number of columns = length of binary string
Ntrials = [100 1000 10000];  % diffent number of trials, per assignment
p1 = [0.5 0.9 0.1];  % different values of p1 given in the assignement
bins_edges = [0.5:100.5]; % fill in the correct edges

% set up storage for random variable statistics (small "s")
outTable = zeros(length(Ntrials), length(p1));

for ktrials = 1:length(Ntrials)

    % Each trial consists of 100 digits
    random_numbers = rand(Ntrials(ktrials),100);  %

    % Do you need a new figure?
    figure();


    for kp = 1:length(p1) %loop on values of p1

        subplot(3,1,kp);
        work = (random_numbers<=p1(kp)); % set to 1's and zeros using p1
        % think about how this works! It's a useful trick that you will
        % need in later projects

        data = zeros(1,100); % initialize location of first 1 in each sample
        for kn = 1:Ntrials(ktrials)  % for each of the 100 element samples

            % This code finds the location of the first one, or, if there
            % are no ones, establishes 101 as the index
            i1 = find(work(kn,:)==1); % find all of the 1's
            if length(i1)==0  % if there are none
```

```
                data(kn)=n+1; %    indicate beyond end of sequence
            else
                data(kn)=min(i1);% otherwise take the first 1
            end
        end

        % Generate the raw histogram for this sequence
        hist_raw=histogram(data, 'BinEdges', bins_edges);
        % Generate the scaled histogram for this sequence
        %   Use 'Normalization','probability' again, because this is
        %   a PMF
        hist_norm=histogram(data, 'BinEdges', bins_edges, 'Normalization', 'probability');

        % Determine the theoretical geometric PMF and plot it on the same
        % axes as the normalized PMF
        k = (0:100);
        theoretical_pmf = (1 - p1(kp)).^k * p1(kp);

        hold on;
        stem(k, theoretical_pmf);
        hold off;

        xlim([0 50]);

        % Label your plots.  If you use subplots, and all subplots have the
        % same x-axis, you can label the x-axis of only the lowest subplot in each
        % column of plots
        xlabel("Bits Until First 1 Appearance");
        ylabel("Probability of First Appearance");
        legend('Scaled histogram', 'Prob Mass Fnc (PMF)');
        title(['Subplot ', int2str(kp), ': p = ', num2str(p1(kp))]);
        %title(['Section 3.2: ', int2str(Ntrials(ktrials)), ' index of first binary 1 in string for p: ', num2str(p1(kp))]);
        %Compute and the sample mean and variance
        sample_mean = mean(data);
        sample_var = var(data);

        %Compute and save the population mean and variance
        pop_mean = 1/p1(kp);
        pop_var = (1 - p1(kp)) / (p1(kp)^2);
        disp(['For ', int2str(Ntrials(ktrials)), ...
            ' with p: ', num2str(p1(kp)), ', sample mean: ', ...
            num2str(sample_mean), ' sample variance: ', num2str(sample_var), ...
            ' population mean: ', num2str(pop_mean), ...
            ' and population variance: ', num2str(pop_var)]);

        % Do whatever bookkeeping or housekeeping you need to do at the end of the loop
        % combine means and vars into single group
        values = [sample_mean, sample_var, pop_mean, pop_var];
        sgtitle(['Section 3.2: Index of First Binary 1 in String for N = ', int2str(Ntrials(ktrials))]);
        grid on;

    end % loop on p1

end

% Display the means and variances in a way that makes sense for you,
% because you have to report them in the Project report.


disp('-----------');
disp(' ');
```

```
Section 3.2 PMF for Binary String
For 100 with p: 0.5, sample mean: 1.99 sample variance: 2.1312 population mean: 2 and population variance: 2
For 100 with p: 0.9, sample mean: 1.07 sample variance: 0.065758 population mean: 1.1111 and population variance: 0.12346
For 100 with p: 0.1, sample mean: 10.6 sample variance: 110.7677 population mean: 10 and population variance: 90
For 1000 with p: 0.5, sample mean: 2.037 sample variance: 1.8415 population mean: 2 and population variance: 2
For 1000 with p: 0.9, sample mean: 1.14 sample variance: 0.15656 population mean: 1.1111 and population variance: 0.12346
For 1000 with p: 0.1, sample mean: 10.106 sample variance: 83.1539 population mean: 10 and population variance: 90
For 10000 with p: 0.5, sample mean: 1.9964 sample variance: 1.9974 population mean: 2 and population variance: 2
For 10000 with p: 0.9, sample mean: 1.1089 sample variance: 0.12505 population mean: 1.1111 and population variance: 0.12346
For 10000 with p: 0.1, sample mean: 9.9726 sample variance: 88.0689 population mean: 10 and population variance: 90
-----------
```



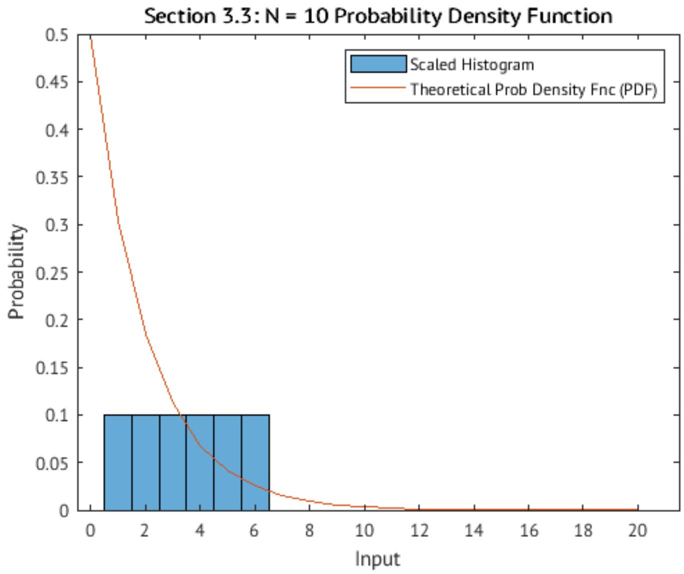



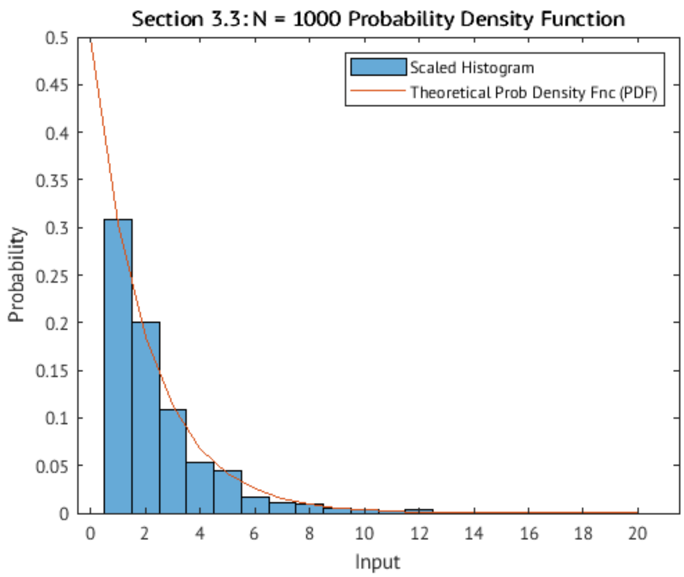

### Section 3.3 exponentially distributed

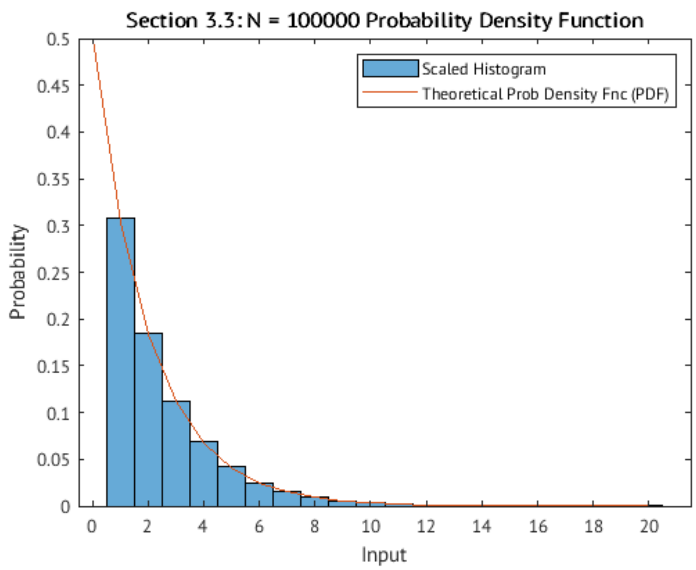
```
disp(' ');
disp('Section 3.3 Exponentially Distributed');
```

```
% Set up new plots as necessary.  Remember, you need ALL of the plots


Ntrials = [10 1000 100000]; %set according to the assignment
lambda = 0.5;  % as given in the assignment

theory_m =  1/lambda;   % analytical or population mean == mu
theory_v = 1/(lambda^2); % analytical or population variance == sigma^2
for ktrials = 1:length(Ntrials)   % repeat for each number of trials

      figure();
      % Set the bin edges
      bins_edges = [0.5: 20.5];

      % Generate the appropriate number of independent random trials
      data = randx(1,Ntrials(ktrials),lambda); % randx is given

      % Compute (and plot) the raw histogram
      hist_raw = histogram(data, 'BinEdges', bins_edges);

      % used to compute the scaling factor
      %hist_raw = histogram(data, 40, 'BinLimits', [0.5 20.5]);

      % Compute (and plot) the normalized histogram
      %    Use 'Normalization','pdf' because an exponential RV is a
      %    continuous random variable and has a pdf, not a pmf.
      hist_norm = histogram(data, 'BinEdges', bins_edges, 'Normalization', 'pdf');

      % used to compute the scaling factor with the line above
      %hist_norm = histogram(data, 40, 'BinLimits', [0.5 20.5], 'Normalization', 'pdf');

      % Compute and plot the true pdf on the same axes as the normalized
      % histogram
      k = (0:20);
      theoretical_pdf = lambda * exp(-lambda * k);
      hold on;
      plot(k, theoretical_pdf);
      hold off;

      % Compute the sample mean and variance and display them
      sample_mean = mean(data);
      sample_var = var(data);

      % labels and titles
      xlabel('Input');
      ylabel('Probability');
      title(['Section 3.3: N = ', int2str(Ntrials(ktrials)), ' Probability Density Function']);
      legend('Scaled Histogram', 'Theoretical Prob Density Fnc (PDF)');

      % Either compare the sample mean and variance with the population
      % mean and variance, or save the results to output later
      disp(['For N = ', int2str(Ntrials(ktrials)), ...
          ', sample and population mean are ', num2str(sample_mean), ...
          ' & ', num2str(theory_m), ...
          ' with sample and population variance ', num2str(sample_var), ...
          ' & ', num2str(theory_v)]);
end;

%Compare the sample mean and variance to the population mean and variance
%as requested

disp('-----------');
disp(' ');
```

```
Section 3.3 Exponentially Distributed
For N = 10, sample and population mean are 2.1958 & 2 with sample and population variance 4.6434 & 4
For N = 1000, sample and population mean are 1.8922 & 2 with sample and population variance 3.7739 & 4
For N = 100000, sample and population mean are 2.0025 & 2 with sample and population variance 4.0252 & 4
-----------
```



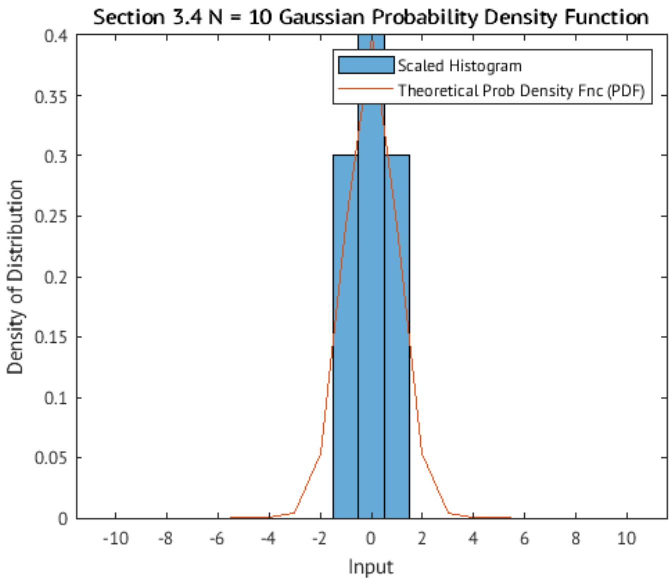



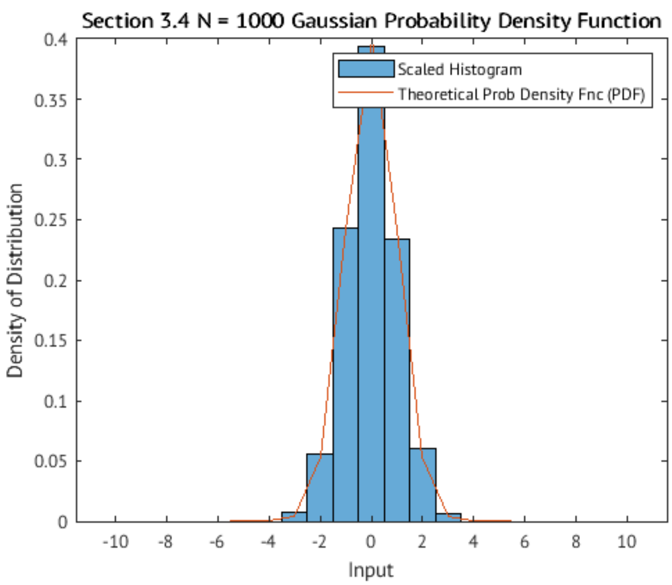

### Section 3.4 N(0,1)

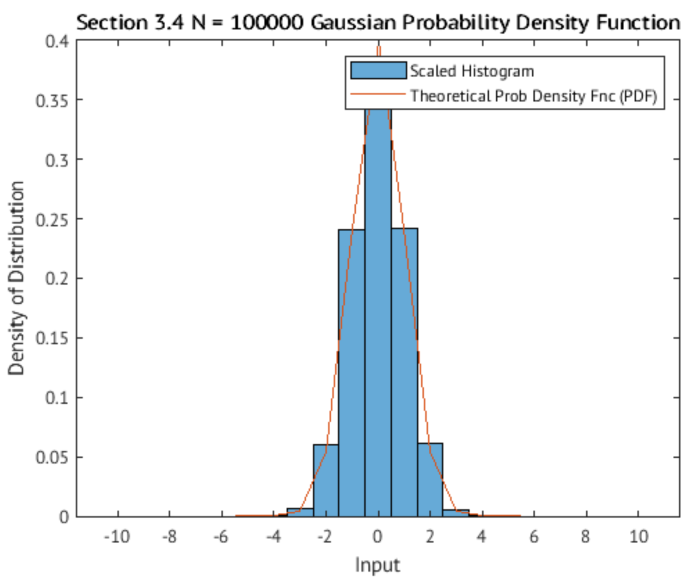
```
disp(' ');
disp('Section 3.4 Samples from N(0,1)');
```

```
Ntrials = [10 1000 100000];  % number of trials per assignment

% Theoretical (analytical) (population) mean and variance
theory_m = 0;  % given in assignment
theory_v = 1;  % given in assignement

for ktrials = 1:length(Ntrials)  % loop on the different number of trials
      figure();
      % Set the bin edges
      bin_edges = [-10.5:10.5];
      % Generate the random data
      data = randn(1,Ntrials(ktrials)); % function randn gives samples from N(0,1) by definition

      % Compute the raw histogram
      hist_raw = histogram(data, 'BinEdges', bin_edges);

      % Compute the normalized histogram with pdf normalization
      hist_norm = histogram(data, 'BinEdges', bin_edges, 'Normalization', 'pdf');

      % Compute the theoretical pdf and plot on same axes as normalized
      b = (-10:10);
      pdf = exp(-(b-theory_m).^2/(2*theory_v))/sqrt(2*pi*theory_v);
      hold on;
      plot(b, pdf);
      hold off;

      % label and title
      xlabel('Input');
      ylabel('Density of Distribution');
      title(['Section 3.4 N = ', num2str(Ntrials(ktrials)), ' Gaussian Probability Density Function']);
      legend('Scaled Histogram', 'Theoretical Prob Density Fnc (PDF)');

      % Compute the sample mean and variance and either save or print

      sample_m = mean(data);
      sample_v = var(data);
      disp(['For N = ', num2str(Ntrials(ktrials)), ...
          ' sample mean and variance are ', num2str(sample_m), ...
          ' & ' , num2str(sample_v)]);

end;
disp(['Theoretical mean and variance ', num2str(theory_m), ' & ', num2str(theory_v)]);

disp('-----------');
disp(' ');
```

```
Section 3.4 Samples from N(0,1)
For N = 10 sample mean and variance are 0.075549 & 0.56946
For N = 1000 sample mean and variance are 0.0096594 & 0.97414
For N = 100000 sample mean and variance are 0.0012593 & 0.99711
Theoretical mean and variance 0 & 1
-----------
```



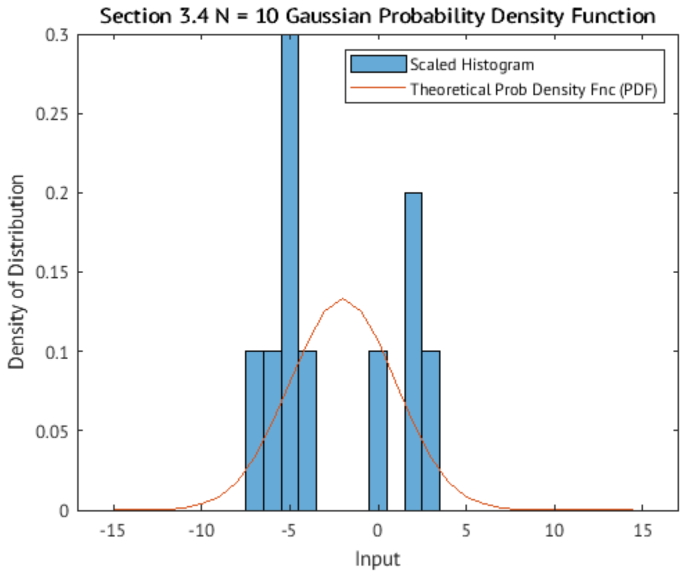



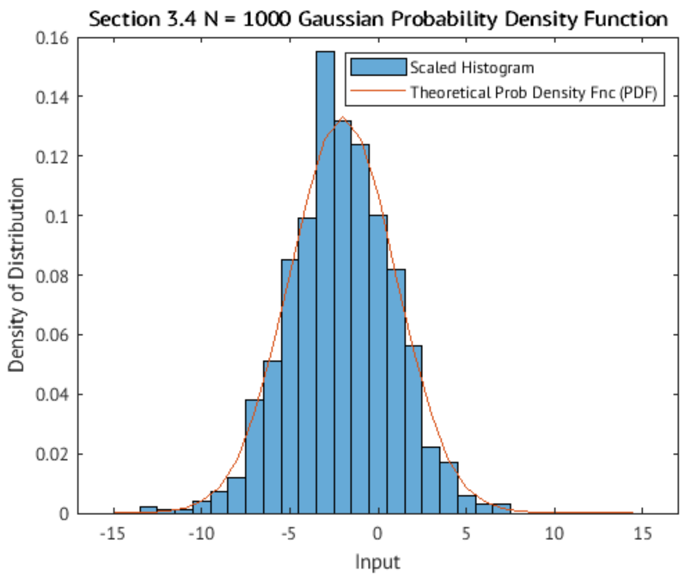



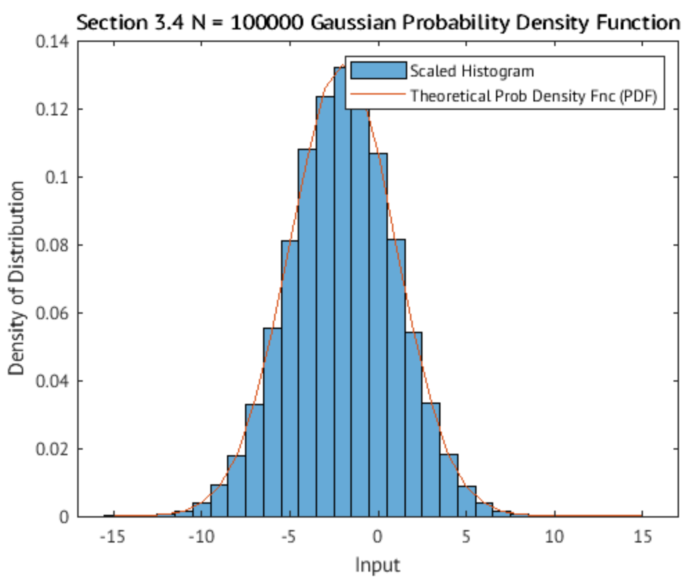

**Section 3.5 N(-2,9)**

```
disp(' ');
disp('Section 3.5 Samples from N(-2,9)');

% Set the number of trials and the theoretical mean and variance
Ntrials = [10 1000 100000];
theory_m = -2;
theory_v = 9;

for ktrials = 1:length(Ntrials) % loop on the number of trials

      figure();
      % Set the bin edges
      bin_edges = [-15.5:15.5];

      % Create the data. Note that for N(m,s2) instead of N(0,1),
      % we use  sqrt(s2)*randn + m.  You might ask yourself "why this
      % formula?"
      % I won't say this again, but you will need it in future projects

      data = sqrt(theory_v)*randn(1,Ntrials(ktrials))+theory_m; % samples from N(0,1) by definition

      % Compute and plot both histograms on different subplots or figures
      hist_raw = histogram(data, 'BinEdges', bin_edges);

      hist_norm = histogram(data, 'BinEdges', bin_edges, 'Normalization', 'pdf');

      % theoretical pdf function
      b = [-15:15];
      pdf = exp(-(b-theory_m).^2/(2*theory_v))/sqrt(2*pi*theory_v);
      hold on;
      plot(b, pdf);
      hold off;

      % label and title stuff
      xlabel('Input');
      ylabel('Density of Distribution');
      title(['Section 3.4 N = ', num2str(Ntrials(ktrials)), ' Gaussian Probability Density Function']);
      legend('Scaled Histogram', 'Theoretical Prob Density Fnc (PDF)');

      % Compute the sample mean and variance and display or save as
      % necessary
      sample_mean = mean(data);
      sample_var = var(data);
      disp(['For N = ', num2str(Ntrials(ktrials)), ...
          ' sample mean and variance are ', num2str(sample_mean), ...
          ' & ' , num2str(sample_var)]);

end;
disp(['Theoretical mean and variance ', num2str(theory_m), ' & ', num2str(theory_v)]);

% Display results as necessary

disp('-----------');
disp(' ');
```

```
Section 3.5 Samples from N(-2,9)
For N = 10 sample mean and variance are -2.476 & 13.6932
For N = 1000 sample mean and variance are -2.0988 & 8.3816
For N = 100000 sample mean and variance are -2.0027 & 8.9711
Theoretical mean and variance -2 & 9
-----------
```



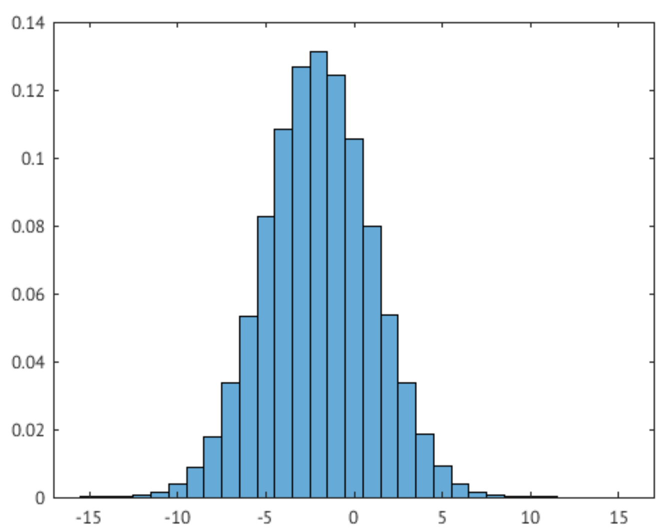




### Section 3.6

```
disp(' ');
disp(['Section 3.6 Computing Probabilities from PDF']);
```

```
Ntrials = [10 1000 100000];
theory_m = -2;
theory_v = 9;

data = sqrt(theory_v)*randn(1,Ntrials(3))+theory_m;
bin_edges = [-15.5:15.5];

hist_raw = histogram(data, 'BinEdges', bin_edges);
% Estimate the requested probability using the raw histogram from 3.5
%   This is a sum followed by a division
ix = find((hist_raw.BinEdges >= -5) & (hist_raw.BinEdges <= 1));
raw_sum = sum(hist_raw.Values(ix));
raw_prob = raw_sum / Ntrials(3);

% Estimate the required probability using the normalized histogram from 3.5
% above.  This is equivalent to a Riemann sum from Calc II
hist_norm = histogram(data, 'BinEdges', bin_edges, 'Normalization', 'pdf');
ix = find((hist_norm.BinEdges >= -5) & (hist_norm.BinEdges <= 1));
norm_prob = sum(hist_norm.Values(ix));
disp(['Section 3.6 N = ', int2str(Ntrials(3)), ' with probability of [-5, 1] (raw): ', num2str(raw_prob), ' and (norm): ', num2str(norm_prob)]);

% Estimate the required probability by integration.  Hint: you could use
% the Q function!
mu = theory_m;
var = theory_v;
func = @(x) (1/sqrt(2*pi*var)) * exp(-(x - mu).^2 / (2 * var));
q = integral(func, -5, 1);

disp(['Numerical integration of PDF: ', num2str(q)]);

% That's it, you're done!
```

```
Section 3.6 Computing Probabilities from PDF
Section 3.6 N = 100000 with probability of [-5, 1] (raw): 0.67639 and (norm): 0.67639
Numerical integration of PDF: 0.68269
```




*Published with MATLAB® R2021b*