MATLAB PROJECT
ECE 131A

NEVIN LIANG
705575353
14 MAR 2021

(1a) Code prints out 5 probabilities, each representing the probability of obtaining an odd number, calculated by runnning `t` iterations of a die roll

```
for j = [10,50,100,500,1000]
    disp(length(find(mod(mod(floor(rand(j,1)*1000),5)+1,2)==1))/j);
end
```

```
0.4000

0.6000

0.5700

0.5760

0.6180
```

(1b) Through mathematical analysis, it appears that the supposed probability should be 0.600 because there are 3 odd numbers and 5 total numbers. 3/5 = 0.600.

(1c) Yes, as `t` gets bigger, the output of my program gets closer and closer to 0.6. 0.5980 is very close to 0.6.

(1d) In this case, the probabilities for each number 1, 2, 3, 4, 5 are, respectively, 2x, 2x, x, x, x.

Let the integer output of our rand() function mod 7 represent the possible outcomes of our die. (0, 2) -> 1; (1, 3) -> 2, (4) -> 3, (5) -> 4, (6) -> 5. Splitting it up like this makes it so that if the outcome mod 7 is even, then the die roll is odd and vice versa.

```
for j = [10, 50, 100, 500, 1000]
    disp(length(find(mod(mod(floor(rand(j,1)*1000),7),2)==0))/j);
end
```

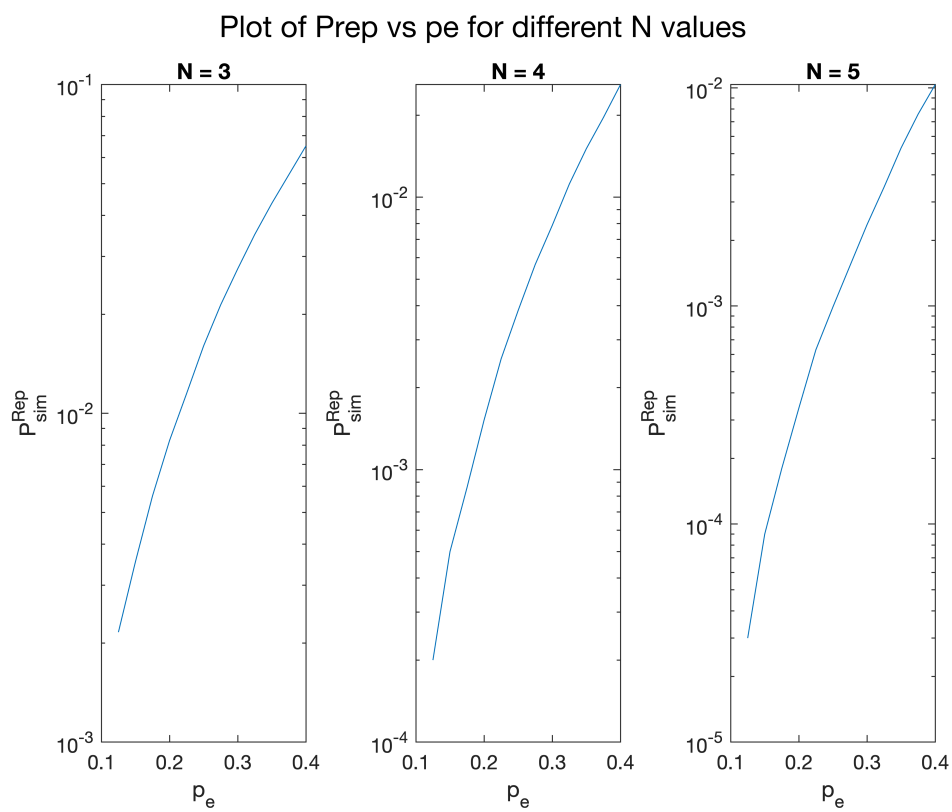
```
0.7000

0.5200

0.5100

0.5900

0.5720
```

Through mathematical analysis, it appears as if the probability the outcome is odd is 4/7 because (2x + x + x) / (2x + 2x + x + x + x) = 4/7.

Our output actually does show this value. 4/7 is approximately 0.5714, and our output is 0.5770 which is fairly close.

(2a) My method of calculating probabilities do not need to implement encode and decode. However, for reference, I have coded versions of these functions in the Appendix.

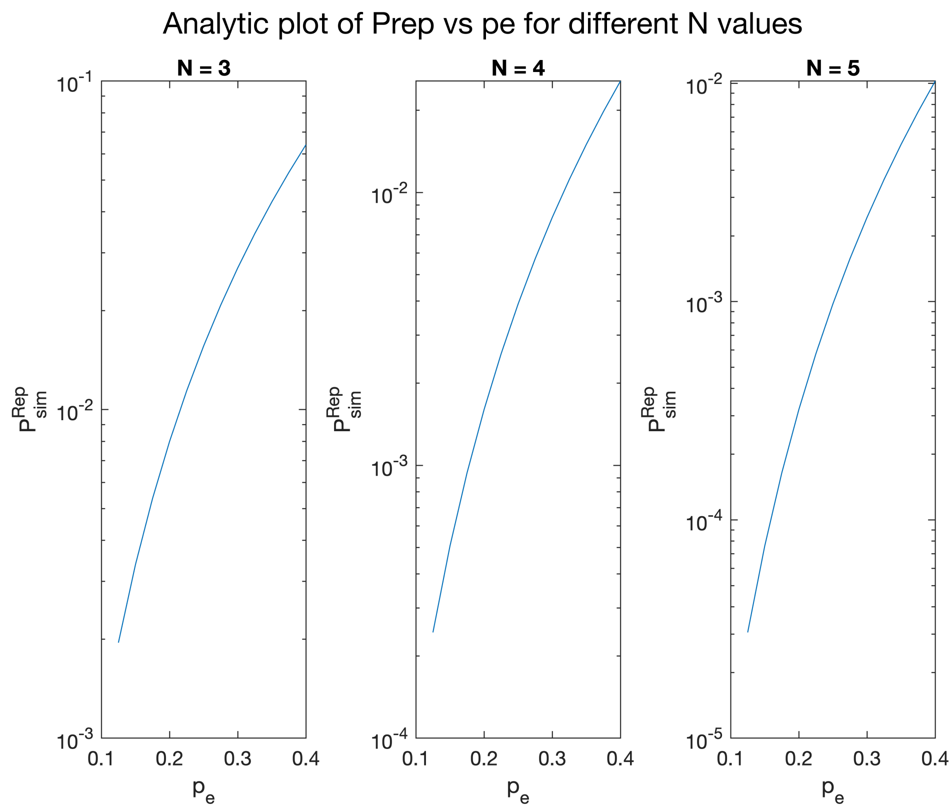
```
figure(1);
sgtitle("Plot of Prep vs pe for different N values");
N = [3, 4, 5];
pe = (0.125:0.025:0.4);
for n = N
    rvals = rand(100000, n);
    prep = zeros(length(pe));
    for x = 1:length(pe)
        prep(x) = sum(sum(rvals < pe(x), 2) == n) / 100000;
    end
    subplot(1, 3, n - 2);
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Rep}_{sim}");
end
```



(2b) For all to fail, we need probability of pe every single time. Thus, for N = n, the probability of a failed transmissions is pe^n.

```
figure(2);
sgtitle("Analytic plot of Prep vs pe for different N values");
for n = N
    prep = power(pe, n);
    subplot(1, 3, n - 2);
```

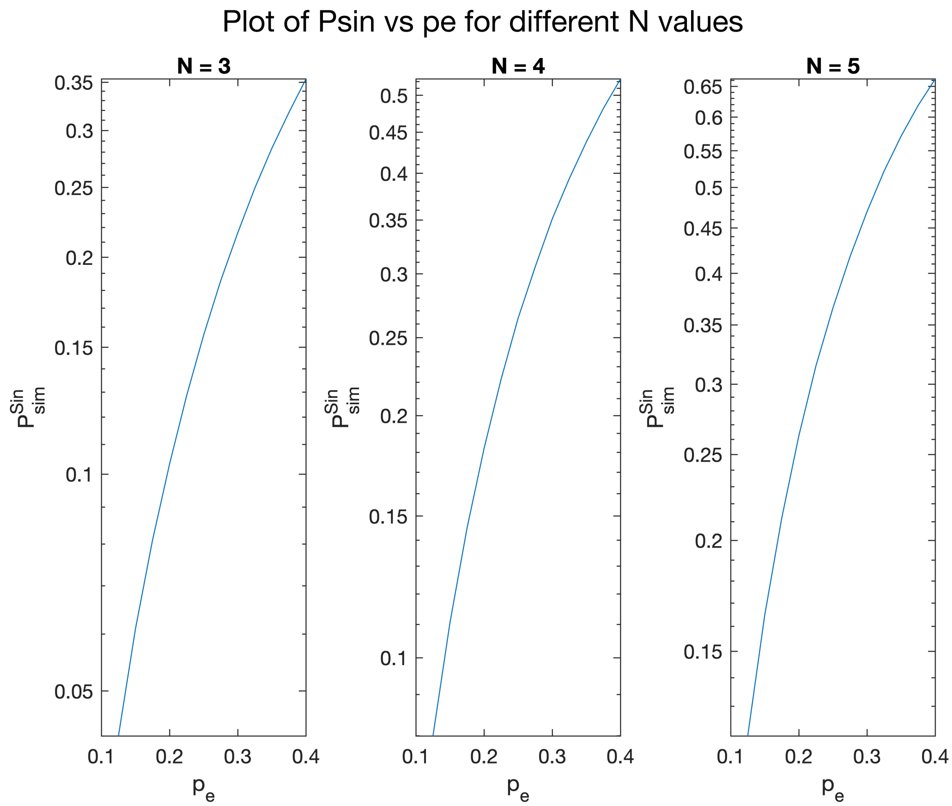
```
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Rep}_{sim}");
end
```



(2c) In this case, we are not allowed to have two failed bits because otherwise the final string is not recoverable

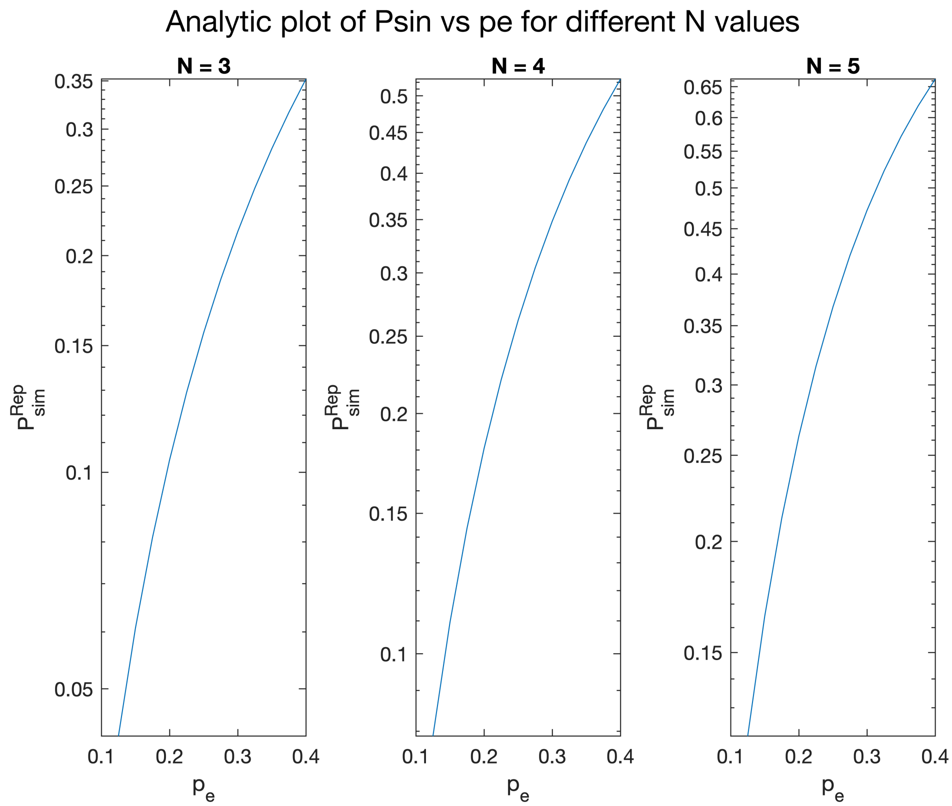
Thus, we can make a slight modification to our code to accomplish this: instead of testing whether the number of failed bits is equal to N, we make sure that the number of successful bits is greater than or equal to N - 1. Thus, the failed attempts are where the number of successful bits is less than N - 1.

```
figure(3);
sgtitle("Plot of Psin vs pe for different N values");
N = [3, 4, 5];
pe = (0.125:0.025:0.4);
for n = N
    rvals = rand(100000, n);
    prep = zeros(length(pe));
    for x = 1:length(pe)
        prep(x) = sum(sum(rvals > pe(x), 2) < n - 1) / 100000;
    end
    subplot(1, 3, n - 2);
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Sin}_{sim}");
end
```



And for the analytic version:

```
figure(4);
sgtitle("Analytic plot of Psin vs pe for different N values");
for n = N
    prep = 1 - power(1 - pe, n) - n .* pe .* power(1 - pe, n - 1);
    subplot(1, 3, n - 2);
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Rep}_{sim}");
end
```

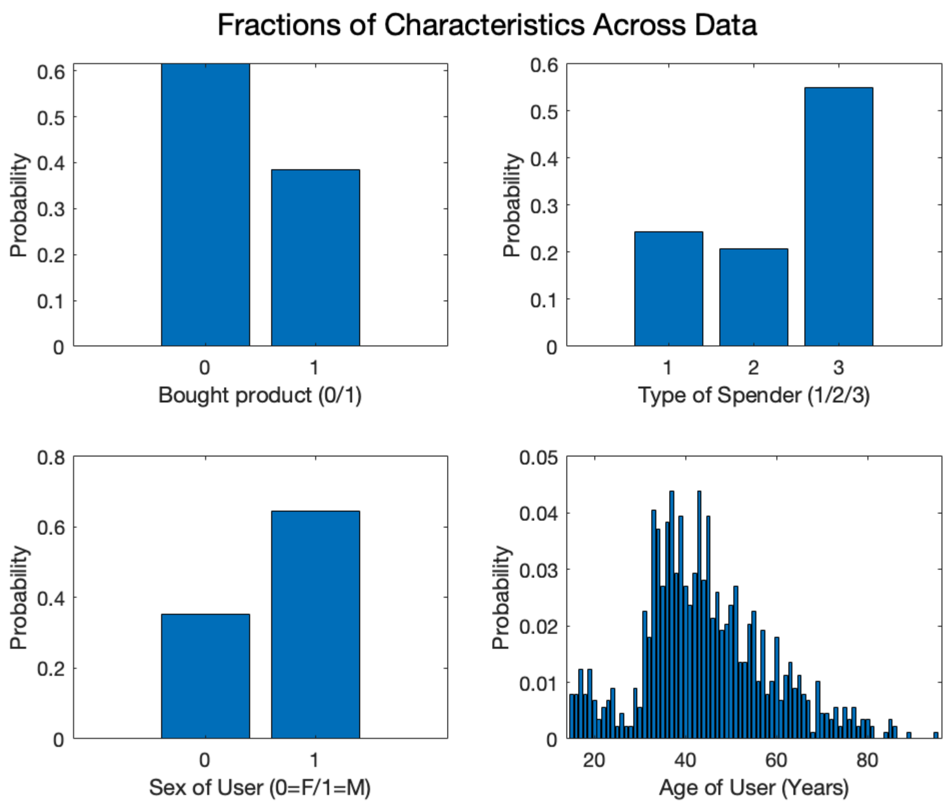


(2d) The plot of P_Sin is much, much larger than that of P_Rep. Which shows that it is much more likely that the transmission fails for Single-bit parity code.

(2e) Repetition code is useful when you want your transmission to be very perfect and have absolutely no loss of data. On the other hand, single-bit parity code transmission is much faster (n times as fast), but has a lot higher chance of dying. For example, transferring passwords across lines must be accurate to the bit. However, if you are in need of fast transmission and not entirely perfect signal transmission, like in a phone call, single-bit parity code would be better. Note that 0.5 error is too high haha but if we use double-bit or triple-bit I think it would be more feasible yet still not that slow.

(3a)

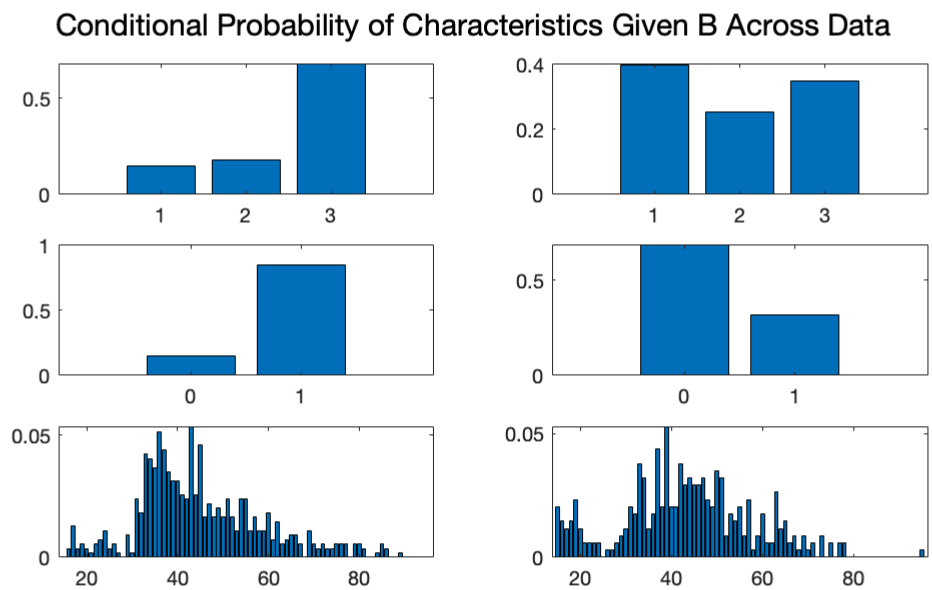
```
figure(1);
sgtitle("Fractions of Characteristics Across Data")
M = readmatrix("user_data.csv");
xlabs = ["Bought product (0/1)", "Type of Spender (1/2/3)", ...
           "Sex of User (0=F/1=M)", "Age of User (Years)"];
for c = 1:4
    subplot(2, 2, c);
    x = [unique(floor(M(:,c))); intmax];
    y = histcounts(M(:,c),x) / 887;
    x(end) = [];
    bar(x, y);
    xlabel(xlabs(c));
    ylabel("Probability");
end
```



(3b)

```
figure(2);
sgtitle("Conditional Probability of Characteristics Given B Across Data")
pb = histcounts(M(:,1),[0 1 intmax]) / 887;
for c = 2:4
    x = [unique(floor(M(:,c))); intmax];
    M1 = M(M(:,1)==0,:);
    M2 = M(M(:,1)==1,:);
    y1 = histcounts(M1(:,c),x) / (887 * pb(1));
    y2 = histcounts(M2(:,c),x) / (887 * pb(2));
```

```
    x(end) = [];

    subplot(4, 2, 2 * c - 3);
    bar(x, y1);
    subplot(4, 2, 2 * c - 2);
    bar(x, y2);
end
```



(3c) We would like to compute P(B = 0,T = 1,S = 0,A <= 55). We will proceed by using the conditional independence assumption:

P(B = 0,T = 1,S = 0,A <= 55) = P(B = 0) * P(T = 1, S = 0, A <= 55 | B = 0)

= P(B = 0) * P(T = 1 | B = 0) * P(S = 0 | B = 0) * P(A <= 55 | B = 0)

Now, to get each of these terms we have to look at our code from part (b).

P(B = 0) is just `sum(M(:,1)==0)/887` . P(T = 1 | B = 0) is just `sum(M1(:,2)==1)/887` divided by P(B = 0).

```
pb0 = pb(1)
```

```
pb0 = 0.6144
```

```
pt1b0 = sum(M1(:,2)==1)/887/pb0
```

```
pt1b0 = 0.1468
```

```
ps0b0 = sum(M1(:,3)==0)/887/pb0
```

```
ps0b0 = 0.1486
```

```
pa55b0 = sum(M1(:,4)<=55)/887/pb0
```

```
pa55b0 = 0.7927
```

```
finalp1 = pb0 * pt1b0 * ps0b0 * pa55b0;
disp(finalp1)
```

```
    0.0106
```

We see that this probability is 0.0106.

As for P(B = 1, T = 1, S = 0, A <= 55) we do the same thing except with B = 1.

```
pb1 = pb(2)
```

```
pb1 = 0.3856
```

```
pt1b1 = sum(M2(:,2)==1)/887/pb1
```

```
pt1b1 = 0.3977
```

```
ps0b1 = sum(M2(:,3)==0)/887/pb1
```

```
ps0b1 = 0.6813
```

```
pa55b1 = sum(M2(:,4)<=55)/887/pb1
```

```
pa55b1 = 0.8099
```

```
finalp2 = pb1 * pt1b1 * ps0b1 * pa55b1;
disp(finalp2)
```

```
    0.0846
```

We see that this probability is 0.0846 or 8.46%.

(3d) We use Bayes Theorem to calculate the answer to this problem:

P(B = 0 | T = 1, S = 0, A <= 55) = P(T = 1, S = 0, A <= 55 | B = 0) * P(B = 0) / P(T = 1, S = 0, A <= 55)

P(T = 1, S = 0, A <= 55 | B = 0) * P(B = 0) = the answer to the first part in part (c).

P(T = 1, S = 0, A <= 55) is just the sum of B = 0 and B = 1 of the previous part.

Thus, by Bayes Theorem,

```
pd1 = finalp1 / (finalp1 + finalp2)
```

```
pd1 = 0.1116
```

Similarly, P(B = 1 | T = 1, S = 0, A <= 55) is calculated in similar fashion:

```
pd2 = finalp2 / (finalp1 + finalp2)
```

```
pd2 = 0.8884
```

The two values for not buying and buying are 0.1116 and 0.8884. Thus, it is more likely that this demographic will buy the product.

(4a)

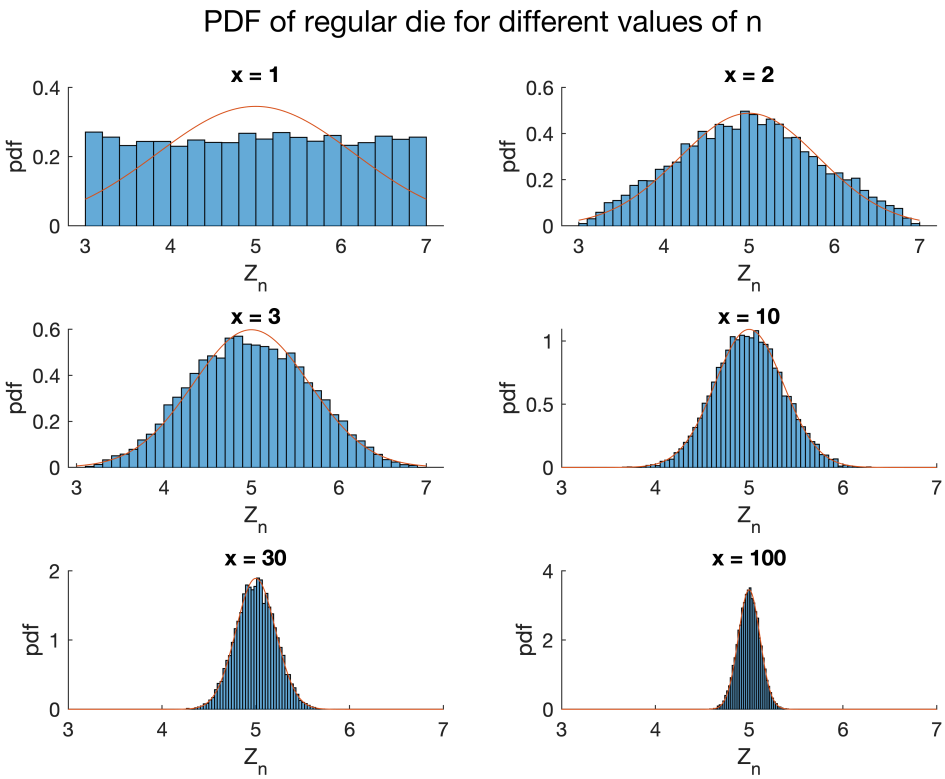
```
figure(1);
sgtitle("PDF of regular die for different values of n");
N = [1, 2, 3, 10, 30, 100];
S = cumsum(rand(100, 10000) * 4 + 3);
for i = 1:6
    s = subplot(3, 2, i);
    title("x = " + N(i)); xlabel("Z_n"); ylabel("pdf");
    hold on;
    histogram(S(N(i), :) / N(i), 'Normalization', 'pdf');
end
```

(4b) Xi is a uniform continuous random variable. In class, we derived formulas for mean and standard deviation of these types of RVs. As the RV falls between 3 and 7, the mean of Xi is (3 + 7) / 2 = 5. The mean of the mean should therefore also be 5. Thus, the mean of Zn is 5.

The variance formula is (b - a)^2 / 12 = 4^2 / 12 = 4 / 3, which is the variance of Xi. To find the variance of Zn, we use properties of variance. VAR(Zn) = VAR(sum(Xi) / n) = 1/n^2 * VAR(sum(Xi)) = 1/n^2 * n * VAR(Xi) since Xi are all i.i.d. This ends up being VAR(Xi) / n = 4/(3n).

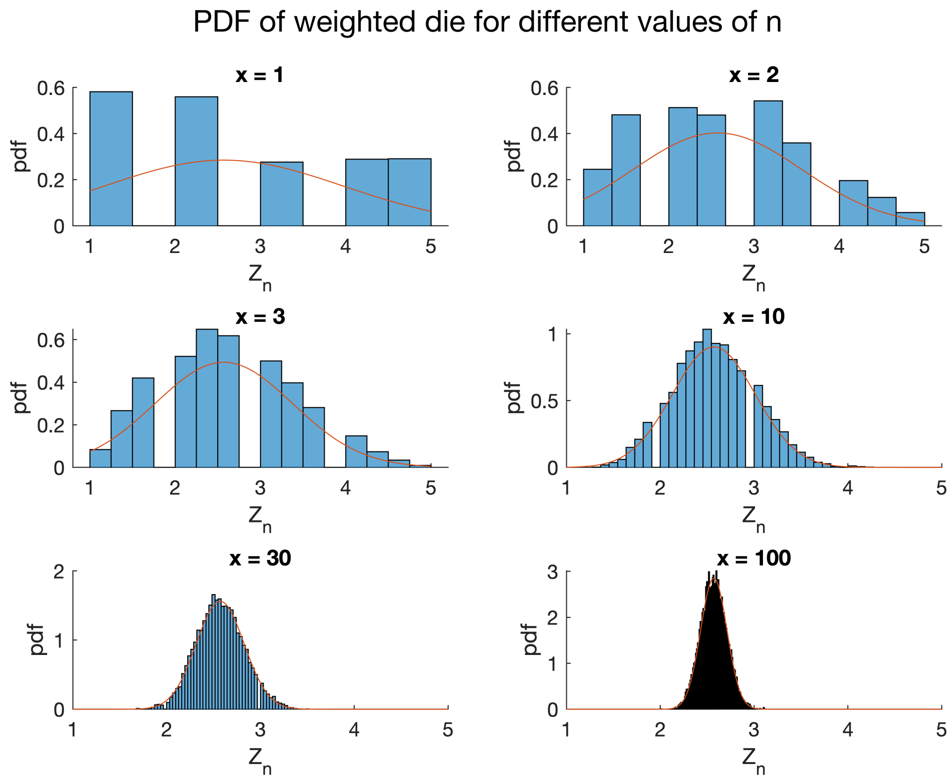
(4c)

```
for i = 1:6
    x = (3:0.01:7);
    y = normpdf(x, 5, sqrt(4 / (3 * N(i))));
    s = subplot(3, 2, i);
    plot(x, y);
    hold off;
end
```



(4d)

```
figure(2);
sgtitle("PDF of weighted die for different values of n");
N = [1, 2, 3, 10, 30, 100];
X = abs(floor(rand(100, 10000) * 7) - 2);
X(X == 0) = 5;
S = cumsum(X);
for i = 1:6
    s = subplot(3, 2, i);
    title("x = " + N(i)); xlabel("Z_n"); ylabel("pdf");
    hold on;
    histogram(S(N(i), :) / N(i), 'Normalization', 'pdf', 'BinWidth', 1 / (N(i) + 1));
    x = (1:0.01:5);
    y = normpdf(x, 2.5714, sqrt(1.9592 / N(i)));
    plot(x, y);
    hold off;
end
```



Note that in this problem, we have discrete RV instead of continuous. The calculation for mean is still straightforward 1 * 2/7 + 2 * 2/7 + 3 * 1/7 + 4 * 1/7 + 5 * 1/7 = 18/7 = 2.5714. The mean of both Xi and Zn are 2.571. However, the variance is different this time. We directly sum all the squared distances multiplied by probability: 1.5714^2 * 2/7 + 0.5714^2 * 2/7 + 0.4286^2 * 1/7 + 1.4286^2 * 1/7 + 2.4286^2 * 1/7 = 1.9592, the variance of Xi. The variance of Zn is just the same equation, though, by property of variance. This is 1.959 / n.

APPENDIX

```
% 1a
for j = [10,50,100,500,1000]
    disp(length(find(mod(mod(floor(rand(j,1)*1000),5)+1,2)==1))/j);
end

% 1d
for j = [10, 50, 100, 500, 1000]
    disp(length(find(mod(mod(floor(rand(j,1)*1000),7),2)==0))/j);
end

%2encode1
function s = encode1(bit)
    for i = 1:n
        s = strcat(s, '' + bit);
    end
end

%2decode1
function bit = decode1(s)
    for i = 1:n
        if s(i) == '0' || s(i) == '1'
            bit = s(i);
            break
        end
    end
end

%2encode2
function s = encode2(inp)
    t = 0;
    for i = 1:length(inp)
        t = t + str2double(inp(i));
    end
    s = inp + mod(t, 2);
end

%2decode2
function s = decode2(inp)
    t = 0;
    e = 0;
    for i = 1:length(inp)
        if (inp(i) ~= 'e')
            t = t + str2double(inp(i));
        else
            if e ~= 0
                s = 'IMPOSSIBLE';
                return
            else
                e = i;
            end
        end
    end
    inp(e) = mod(t, 2);
```

```
    s = inp;
end

%2a
figure(1);
sgtitle("Plot of Prep vs pe for different N values");
N = [3, 4, 5];
pe = (0.125:0.025:0.4);
for n = N
    rvals = rand(100000, n);
    prep = zeros(length(pe));
    for x = 1:length(pe)
        prep(x) = sum(sum(rvals < pe(x), 2) == n) / 100000;
    end
    subplot(1, 3, n - 2);
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Rep}_{sim}");
end

%2b
figure(2);
sgtitle("Analytic plot of Prep vs pe for different N values");
for n = N
    prep = power(pe, n);
    subplot(1, 3, n - 2);
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Rep}_{sim}");
end

%2c
figure(3);
sgtitle("Plot of Psin vs pe for different N values");
N = [3, 4, 5];
pe = (0.125:0.025:0.4);
for n = N
    rvals = rand(100000, n);
    prep = zeros(length(pe));
    for x = 1:length(pe)
        prep(x) = sum(sum(rvals > pe(x), 2) < n - 1) / 100000;
    end
    subplot(1, 3, n - 2);
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Sin}_{sim}");
end

figure(4);
sgtitle("Analytic plot of Psin vs pe for different N values");
for n = N
```

```
    prep = 1 - power(1 - pe, n) - n .* pe .* power(1 - pe, n - 1);
    subplot(1, 3, n - 2);
    semilogy(pe, prep);
    title("N = " + n);
    xlabel("p_e");
    ylabel("P^{Rep}_{sim}");
end

%3a
figure(1);
sgtitle("Fractions of Characteristics Across Data")
M = readmatrix("user_data.csv");
xlabs = ["Bought product (0/1)", "Type of Spender (1/2/3)", ...
            "Sex of User (0=F/1=M)", "Age of User (Years)"];
for c = 1:4
    subplot(2, 2, c);
    x = [unique(floor(M(:,c))); intmax];
    y = histcounts(M(:,c),x) / 887;
    x(end) = [];
    bar(x, y);
    xlabel(xlabs(c));
    ylabel("Probability");
end

%3b
figure(2);
sgtitle("Conditional Probability of Characteristics Given B Across Data")
pb = histcounts(M(:,1),[0 1 intmax]) / 887;
for c = 2:4
    x = [unique(floor(M(:,c))); intmax];
    M1 = M(M(:,1)==0,:);
    M2 = M(M(:,1)==1,:);
    y1 = histcounts(M1(:,c),x) / (887 * pb(1));
    y2 = histcounts(M2(:,c),x) / (887 * pb(2));
    x(end) = [];

    subplot(4, 2, 2 * c - 3);
    bar(x, y1);
    subplot(4, 2, 2 * c - 2);
    bar(x, y2);
end

%3c
pb0 = pb(1)
pt1b0 = sum(M1(:,2)==1)/887/pb0
ps0b0 = sum(M1(:,3)==0)/887/pb0
pa55b0 = sum(M1(:,4)<=55)/887/pb0
finalp1 = pb0 * pt1b0 * ps0b0 * pa55b0;
disp(finalp1)

pb1 = pb(2)
pt1b1 = sum(M2(:,2)==1)/887/pb1
ps0b1 = sum(M2(:,3)==0)/887/pb1
pa55b1 = sum(M2(:,4)<=55)/887/pb1
```

```
finalp2 = pb1 * pt1b1 * ps0b1 * pa55b1;
disp(finalp2)

%3d
pd1 = finalp1 / (finalp1 + finalp2)
pd2 = finalp2 / (finalp1 + finalp2)

%4a
figure(1);
sgtitle("PDF of regular die for different values of n");
N = [1, 2, 3, 10, 30, 100];
S = cumsum(rand(100, 10000) * 4 + 3);
for i = 1:6
    s = subplot(3, 2, i);
    title("x = " + N(i)); xlabel("Z_n"); ylabel("pdf");
    hold on;
    histogram(S(N(i), :) / N(i), 'Normalization', 'pdf');
end

%4c
for i = 1:6
    x = (3:0.01:7);
    y = normpdf(x, 5, sqrt(4 / (3 * N(i))));
    s = subplot(3, 2, i);
    plot(x, y);
    hold off;
end

%4d
figure(2);
sgtitle("PDF of weighted die for different values of n");
N = [1, 2, 3, 10, 30, 100];
X = abs(floor(rand(100, 10000) * 7) - 2);
X(X == 0) = 5;
S = cumsum(X);
for i = 1:6
    s = subplot(3, 2, i);
    title("x = " + N(i)); xlabel("Z_n"); ylabel("pdf");
    hold on;
    histogram(S(N(i), :) / N(i), 'Normalization', 'pdf', 'BinWidth', 1 / (N(i) + 1));
    x = (1:0.01:5);
    y = normpdf(x, 2.5714, sqrt(1.9592 / N(i)));
    plot(x, y);
    hold off;
end
```