Table of Contents

Problem 1 - RNA-seq 1
Section (1A): 1
Section (1B) 3
Section (1C) 4
Section (1D) 17
Section(1E) 17
Section (1F) 21
Section (1G) 22
Problem 2 22
Section (2A) 23
Section (2B) 23
Section (2D) 25
Section (2E) 26
Section (2F) 28
Section (2g) 30
Problem 3 Section (3A) 33
Section (3B) 33
Section (3C) 34
Section (3D) 37

Problem 1 - RNA-seq

Gabriel Ketron 916571733 Bim 154 Hw3 (Sorry for length, much of the code is copied over instead of functions.)

```
%Initialization (initial E step)
Y = [1 0 1 1 1;1 1 0 0 1;1 1 1 0 0];
readA= Y(:,1)/sum(Y(:,1));
readB= Y(:,2)/sum(Y(:,2));
readC= Y(:,3)/sum(Y(:,3));
readD= Y(:,4)/sum(Y(:,4));
readE= Y(:,5)/sum(Y(:,5));
INTreadA= Y(:,1)/sum(Y(:,1));
INTreadB= Y(:,2)/sum(Y(:,2));
INTreadC= Y(:,3)/sum(Y(:,3));
INTreadD= Y(:,4)/sum(Y(:,4));
INTreadE= Y(:,5)/sum(Y(:,5));
count = 0;
dParameter1 = 1;
```

Section (1A):

```
while dParameter1 > 0.01
    iterA = readA;
    iterB = readB;
    iterC = readC;
    iterD = readD;
    iterE = readE;
    %Single iteration prep (M step)
```

```
    totalChance =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))+(readA(2,:)+readB(2,:)+r
    pred = (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))/
totalChance;
    pgreen = (readA(2,:)+readB(2,:)+readC(2,:)+readD(2,:)+readE(2,:))/
totalChance;
    pblue = (readA(3,:)+readB(3,:)+readC(3,:)+readD(3,:)+readE(3,:))/
totalChance;
    %Set new read percentag
    readA(1,:) = INTreadA(1,:)*pred;
    readB(1,:) = INTreadB(1,:)*pred;
    readC(1,:) = INTreadC(1,:)*pred;
    readD(1,:) = INTreadD(1,:)*pred;
    readE(1,:) = INTreadE(1,:)*pred;
    readA(2,:) = INTreadA(2,:)*pgreen;
    readB(2,:) = INTreadB(2,:)*pgreen;
    readC(2,:) = INTreadC(2,:)*pgreen;
    readD(2,:) = INTreadD(2,:)*pgreen;
    readE(2,:) = INTreadE(2,:)*pgreen;
    readA(3,:) = INTreadA(3,:)*pblue;
    readB(3,:) = INTreadB(3,:)*pblue;
    readC(3,:) = INTreadC(3,:)*pblue;
    readD(3,:) = INTreadD(3,:)*pblue;
    readE(3,:) = INTreadE(3,:)*pblue;
    %Normalize read percentages
    readA = readA/sum(readA);
    readB = readB/sum(readB);
    readC = readC/sum(readC);
    readD = readD/sum(readD);
    readE = readE/sum(readE);
    %change in parameter
    dParaMatrix = zeros(3,5);
    dParaMatrix(1,1) = abs(readA(1,:)-iterA(1,:));
    dParaMatrix(2,1) = abs(readA(2,:)-iterA(2,:));
    dParaMatrix(3,1) = abs(readA(3,:)-iterA(3,:));
    dParaMatrix(1,2) = abs(readB(1,:)-iterB(1,:));
    dParaMatrix(2,2) = abs(readB(2,:)-iterB(2,:));
    dParaMatrix(3,2) = abs(readB(3,:)-iterB(3,:));
    dParaMatrix(1,3) = abs(readC(1,:)-iterC(1,:));
    dParaMatrix(2,3) = abs(readC(2,:)-iterC(2,:));
    dParaMatrix(3,3) = abs(readC(3,:)-iterC(3,:));
    dParaMatrix(1,4) = abs(readD(1,:)-iterD(1,:));
    dParaMatrix(2,4) = abs(readD(2,:)-iterD(2,:));
    dParaMatrix(3,4) = abs(readD(3,:)-iterD(3,:));
    dParaMatrix(1,5) = abs(readE(1,:)-iterE(1,:));
    dParaMatrix(2,5) = abs(readE(2,:)-iterE(2,:));
    dParaMatrix(3,5) = abs(readE(3,:)-iterE(3,:));
    dParameterRow = max(dParaMatrix);
    dParameter1 = max(dParameterRow);
    count = count + 1;
end
dParameter1;
pred;
pgreen;
```

```
pblue;
count;
% This algorithm requires 6 iteratons to convergence. This is the same
% solution from class.
```

Section (1B)

```
Y = [1 0 1 1 1;1 1 0 0 1;1 1 1 0 0];
readA= Y(:,1)/sum(Y(:,1));
readB= Y(:,2)/sum(Y(:,2));
readC= Y(:,3)/sum(Y(:,3));
readD= Y(:,4)/sum(Y(:,4));
readE= Y(:,5)/sum(Y(:,5));
INTreadA= Y(:,1)/sum(Y(:,1));
INTreadB= Y(:,2)/sum(Y(:,2));
INTreadC= Y(:,3)/sum(Y(:,3));
INTreadD= Y(:,4)/sum(Y(:,4));
INTreadE= Y(:,5)/sum(Y(:,5));
count = 0;
dParameter1 = 1;
while dParameter1 > 0.001
    iterA = readA;
    iterB = readB;
    iterC = readC;
    iterD = readD;
    iterE = readE;
    %Single iteration prep (M step)
    totalChance =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))+(readA(2,:)+readB(2,:)+r
    pred = (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))/
totalChance;
    pgreen = (readA(2,:)+readB(2,:)+readC(2,:)+readD(2,:)+readE(2,:))/
totalChance;
    pblue = (readA(3,:)+readB(3,:)+readC(3,:)+readD(3,:)+readE(3,:))/
totalChance;
    %Set new read percentag
    readA(1,:) = INTreadA(1,:)*pred;
    readB(1,:) = INTreadB(1,:)*pred;
    readC(1,:) = INTreadC(1,:)*pred;
    readD(1,:) = INTreadD(1,:)*pred;
    readE(1,:) = INTreadE(1,:)*pred;
    readA(2,:) = INTreadA(2,:)*pgreen;
    readB(2,:) = INTreadB(2,:)*pgreen;
    readC(2,:) = INTreadC(2,:)*pgreen;
    readD(2,:) = INTreadD(2,:)*pgreen;
    readE(2,:) = INTreadE(2,:)*pgreen;
    readA(3,:) = INTreadA(3,:)*pblue;
    readB(3,:) = INTreadB(3,:)*pblue;
    readC(3,:) = INTreadC(3,:)*pblue;
    readD(3,:) = INTreadD(3,:)*pblue;
    readE(3,:) = INTreadE(3,:)*pblue;
    %Normalize read percentages
    readA = readA/sum(readA);
```

```
    readB = readB/sum(readB);
    readC = readC/sum(readC);
    readD = readD/sum(readD);
    readE = readE/sum(readE);
    %change in parameter
    dParaMatrix = zeros(3,5);
    dParaMatrix(1,1) = abs(readA(1,:)-iterA(1,:));
    dParaMatrix(2,1) = abs(readA(2,:)-iterA(2,:));
    dParaMatrix(3,1) = abs(readA(3,:)-iterA(3,:));
    dParaMatrix(1,2) = abs(readB(1,:)-iterB(1,:));
    dParaMatrix(2,2) = abs(readB(2,:)-iterB(2,:));
    dParaMatrix(3,2) = abs(readB(3,:)-iterB(3,:));
    dParaMatrix(1,3) = abs(readC(1,:)-iterC(1,:));
    dParaMatrix(2,3) = abs(readC(2,:)-iterC(2,:));
    dParaMatrix(3,3) = abs(readC(3,:)-iterC(3,:));
    dParaMatrix(1,4) = abs(readD(1,:)-iterD(1,:));
    dParaMatrix(2,4) = abs(readD(2,:)-iterD(2,:));
    dParaMatrix(3,4) = abs(readD(3,:)-iterD(3,:));
    dParaMatrix(1,5) = abs(readE(1,:)-iterE(1,:));
    dParaMatrix(2,5) = abs(readE(2,:)-iterE(2,:));
    dParaMatrix(3,5) = abs(readE(3,:)-iterE(3,:));
    dParameterRow = max(dParaMatrix);
    dParameter1 = max(dParameterRow);
    count = count + 1;
end
dParameter1;
pred;
pgreen;
pblue;
count;
% 9 iterations are required.
```

Section (1C)

```
%New Initialization 1:
Y = [0 0 1 1 1;1 1 0 0 1;1 1 1 0 0];
readA= Y(:,1)/sum(Y(:,1));
readB= Y(:,2)/sum(Y(:,2));
readC= Y(:,3)/sum(Y(:,3));
readD= Y(:,4)/sum(Y(:,4));
readE= Y(:,5)/sum(Y(:,5));
INTreadA= Y(:,1)/sum(Y(:,1));
INTreadB= Y(:,2)/sum(Y(:,2));
INTreadC= Y(:,3)/sum(Y(:,3));
INTreadD= Y(:,4)/sum(Y(:,4));
INTreadE= Y(:,5)/sum(Y(:,5));
count = 0;
dParameter1 = 1;
while dParameter1 > 0.01
    iterA = readA;
    iterB = readB;
    iterC = readC;
    iterD = readD;
```

```
    iterE = readE;
    %Single iteration prep (M step)
    totalChance =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))+(readA(2,:)+readB(2,:)+r
    pred = (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))/
totalChance;
    pgreen = (readA(2,:)+readB(2,:)+readC(2,:)+readD(2,:)+readE(2,:))/
totalChance;
    pblue = (readA(3,:)+readB(3,:)+readC(3,:)+readD(3,:)+readE(3,:))/
totalChance;
    %Set new read percentag
    readA(1,:) = INTreadA(1,:)*pred;
    readB(1,:) = INTreadB(1,:)*pred;
    readC(1,:) = INTreadC(1,:)*pred;
    readD(1,:) = INTreadD(1,:)*pred;
    readE(1,:) = INTreadE(1,:)*pred;
    readA(2,:) = INTreadA(2,:)*pgreen;
    readB(2,:) = INTreadB(2,:)*pgreen;
    readC(2,:) = INTreadC(2,:)*pgreen;
    readD(2,:) = INTreadD(2,:)*pgreen;
    readE(2,:) = INTreadE(2,:)*pgreen;
    readA(3,:) = INTreadA(3,:)*pblue;
    readB(3,:) = INTreadB(3,:)*pblue;
    readC(3,:) = INTreadC(3,:)*pblue;
    readD(3,:) = INTreadD(3,:)*pblue;
    readE(3,:) = INTreadE(3,:)*pblue;
    %Normalize read percentages
    readA = readA/sum(readA);
    readB = readB/sum(readB);
    readC = readC/sum(readC);
    readD = readD/sum(readD);
    readE = readE/sum(readE);
    %change in parameter
    dParaMatrix = zeros(3,5);
    dParaMatrix(1,1) = abs(readA(1,:)-iterA(1,:));
    dParaMatrix(2,1) = abs(readA(2,:)-iterA(2,:));
    dParaMatrix(3,1) = abs(readA(3,:)-iterA(3,:));
    dParaMatrix(1,2) = abs(readB(1,:)-iterB(1,:));
    dParaMatrix(2,2) = abs(readB(2,:)-iterB(2,:));
    dParaMatrix(3,2) = abs(readB(3,:)-iterB(3,:));
    dParaMatrix(1,3) = abs(readC(1,:)-iterC(1,:));
    dParaMatrix(2,3) = abs(readC(2,:)-iterC(2,:));
    dParaMatrix(3,3) = abs(readC(3,:)-iterC(3,:));
    dParaMatrix(1,4) = abs(readD(1,:)-iterD(1,:));
    dParaMatrix(2,4) = abs(readD(2,:)-iterD(2,:));
    dParaMatrix(3,4) = abs(readD(3,:)-iterD(3,:));
    dParaMatrix(1,5) = abs(readE(1,:)-iterE(1,:));
    dParaMatrix(2,5) = abs(readE(2,:)-iterE(2,:));
    dParaMatrix(3,5) = abs(readE(3,:)-iterE(3,:));
    dParameterRow = max(dParaMatrix);
    dParameter1 = max(dParameterRow);
    count = count + 1;
end
dParameter1;
```

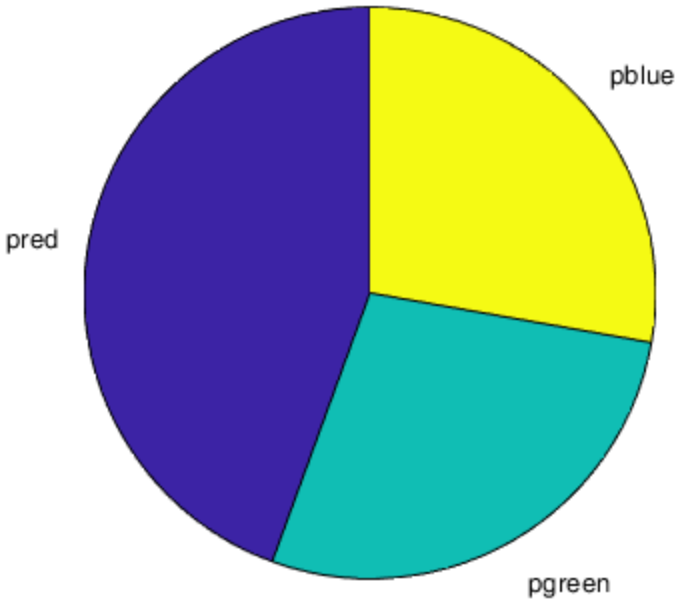
```
pred;
pgreen;
pblue;
X = [pred pgreen pblue];
labels = {'pred','pgreen','pblue'};
figure
pie(X,labels)
snapnow
count;
Y = [0 0 1 1 1;1 1 0 0 1;1 1 1 0 0];
readA= Y(:,1)/sum(Y(:,1));
readB= Y(:,2)/sum(Y(:,2));
readC= Y(:,3)/sum(Y(:,3));
readD= Y(:,4)/sum(Y(:,4));
readE= Y(:,5)/sum(Y(:,5));
INTreadA= Y(:,1)/sum(Y(:,1));
INTreadB= Y(:,2)/sum(Y(:,2));
INTreadC= Y(:,3)/sum(Y(:,3));
INTreadD= Y(:,4)/sum(Y(:,4));
INTreadE= Y(:,5)/sum(Y(:,5));
count = 0;
dParameter1 = 1;
while dParameter1 > 0.001
    iterA = readA;
    iterB = readB;
    iterC = readC;
    iterD = readD;
    iterE = readE;
    %Single iteration prep (M step)
    totalChance =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))+(readA(2,:)+readB(2,:)+r
    pred = (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))/
totalChance;
    pgreen = (readA(2,:)+readB(2,:)+readC(2,:)+readD(2,:)+readE(2,:))/
totalChance;
    pblue = (readA(3,:)+readB(3,:)+readC(3,:)+readD(3,:)+readE(3,:))/
totalChance;
    %Set new read percentag
    readA(1,:) = INTreadA(1,:)*pred;
    readB(1,:) = INTreadB(1,:)*pred;
    readC(1,:) = INTreadC(1,:)*pred;
    readD(1,:) = INTreadD(1,:)*pred;
    readE(1,:) = INTreadE(1,:)*pred;
    readA(2,:) = INTreadA(2,:)*pgreen;
    readB(2,:) = INTreadB(2,:)*pgreen;
    readC(2,:) = INTreadC(2,:)*pgreen;
    readD(2,:) = INTreadD(2,:)*pgreen;
    readE(2,:) = INTreadE(2,:)*pgreen;
    readA(3,:) = INTreadA(3,:)*pblue;
    readB(3,:) = INTreadB(3,:)*pblue;
    readC(3,:) = INTreadC(3,:)*pblue;
    readD(3,:) = INTreadD(3,:)*pblue;
    readE(3,:) = INTreadE(3,:)*pblue;
    %Normalize read percentages
```

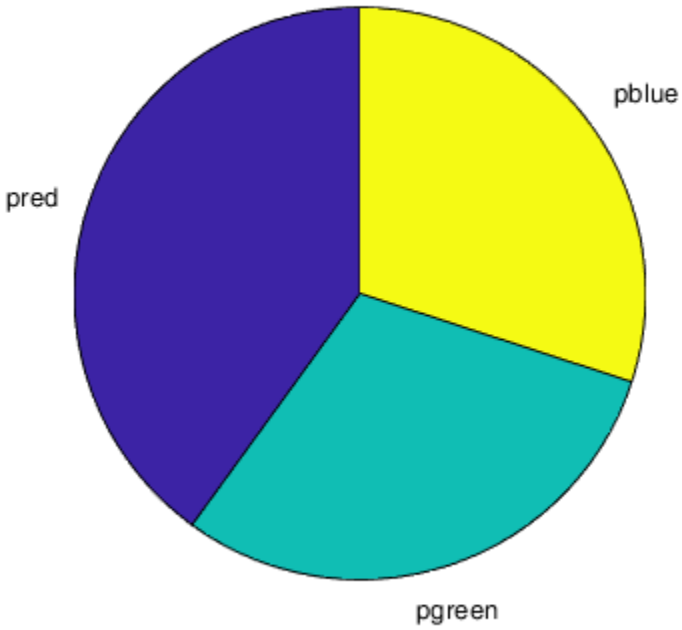
```
    readA = readA/sum(readA);
    readB = readB/sum(readB);
    readC = readC/sum(readC);
    readD = readD/sum(readD);
    readE = readE/sum(readE);
    %change in parameter
    dParaMatrix = zeros(3,5);
    dParaMatrix(1,1) = abs(readA(1,:)-iterA(1,:));
    dParaMatrix(2,1) = abs(readA(2,:)-iterA(2,:));
    dParaMatrix(3,1) = abs(readA(3,:)-iterA(3,:));
    dParaMatrix(1,2) = abs(readB(1,:)-iterB(1,:));
    dParaMatrix(2,2) = abs(readB(2,:)-iterB(2,:));
    dParaMatrix(3,2) = abs(readB(3,:)-iterB(3,:));
    dParaMatrix(1,3) = abs(readC(1,:)-iterC(1,:));
    dParaMatrix(2,3) = abs(readC(2,:)-iterC(2,:));
    dParaMatrix(3,3) = abs(readC(3,:)-iterC(3,:));
    dParaMatrix(1,4) = abs(readD(1,:)-iterD(1,:));
    dParaMatrix(2,4) = abs(readD(2,:)-iterD(2,:));
    dParaMatrix(3,4) = abs(readD(3,:)-iterD(3,:));
    dParaMatrix(1,5) = abs(readE(1,:)-iterE(1,:));
    dParaMatrix(2,5) = abs(readE(2,:)-iterE(2,:));
    dParaMatrix(3,5) = abs(readE(3,:)-iterE(3,:));
    dParameterRow = max(dParaMatrix);
    dParameter1 = max(dParameterRow);
    count = count + 1;
    if count < 5
        X = [pred pgreen pblue];
        labels = {'pred','pgreen','pblue'};
        figure
        pie(X,labels)
    end
end
dParameter1;
pred;
pgreen;
pblue;
X = [pred pgreen pblue];
labels = {'pred','pgreen','pblue'};
figure
pie(X,labels)
snapnow
count;
%This set converges for both 0.01 and 0.001 cutoffs.
Y = [0 0 1 1 1;0 1 0 0 1;1 1 1 0 0];
readA= Y(:,1)/sum(Y(:,1));
readB= Y(:,2)/sum(Y(:,2));
readC= Y(:,3)/sum(Y(:,3));
readD= Y(:,4)/sum(Y(:,4));
readE= Y(:,5)/sum(Y(:,5));
INTreadA= Y(:,1)/sum(Y(:,1));
INTreadB= Y(:,2)/sum(Y(:,2));
INTreadC= Y(:,3)/sum(Y(:,3));
INTreadD= Y(:,4)/sum(Y(:,4));
INTreadE= Y(:,5)/sum(Y(:,5));
```

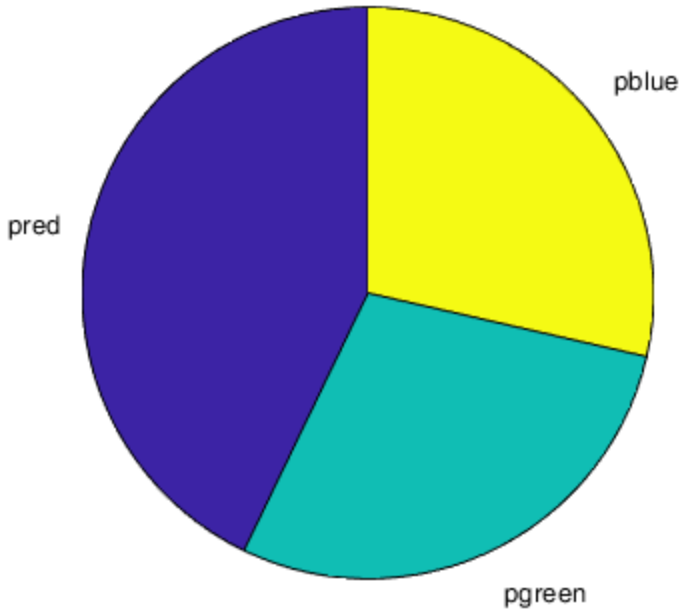
```
count = 0;
dParameter1 = 1;
while dParameter1 > 0.01
    iterA = readA;
    iterB = readB;
    iterC = readC;
    iterD = readD;
    iterE = readE;
    %Single iteration prep (M step)
    totalChance =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))+(readA(2,:)+readB(2,:)+r
    pred = (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))/
totalChance;
    pgreen = (readA(2,:)+readB(2,:)+readC(2,:)+readD(2,:)+readE(2,:))/
totalChance;
    pblue = (readA(3,:)+readB(3,:)+readC(3,:)+readD(3,:)+readE(3,:))/
totalChance;
    %Set new read percentag
    readA(1,:) = INTreadA(1,:)*pred;
    readB(1,:) = INTreadB(1,:)*pred;
    readC(1,:) = INTreadC(1,:)*pred;
    readD(1,:) = INTreadD(1,:)*pred;
    readE(1,:) = INTreadE(1,:)*pred;
    readA(2,:) = INTreadA(2,:)*pgreen;
    readB(2,:) = INTreadB(2,:)*pgreen;
    readC(2,:) = INTreadC(2,:)*pgreen;
    readD(2,:) = INTreadD(2,:)*pgreen;
    readE(2,:) = INTreadE(2,:)*pgreen;
    readA(3,:) = INTreadA(3,:)*pblue;
    readB(3,:) = INTreadB(3,:)*pblue;
    readC(3,:) = INTreadC(3,:)*pblue;
    readD(3,:) = INTreadD(3,:)*pblue;
    readE(3,:) = INTreadE(3,:)*pblue;
    %Normalize read percentages
    readA = readA/sum(readA);
    readB = readB/sum(readB);
    readC = readC/sum(readC);
    readD = readD/sum(readD);
    readE = readE/sum(readE);
    %change in parameter
    dParaMatrix = zeros(3,5);
    dParaMatrix(1,1) = abs(readA(1,:)-iterA(1,:));
    dParaMatrix(2,1) = abs(readA(2,:)-iterA(2,:));
    dParaMatrix(3,1) = abs(readA(3,:)-iterA(3,:));
    dParaMatrix(1,2) = abs(readB(1,:)-iterB(1,:));
    dParaMatrix(2,2) = abs(readB(2,:)-iterB(2,:));
    dParaMatrix(3,2) = abs(readB(3,:)-iterB(3,:));
    dParaMatrix(1,3) = abs(readC(1,:)-iterC(1,:));
    dParaMatrix(2,3) = abs(readC(2,:)-iterC(2,:));
    dParaMatrix(3,3) = abs(readC(3,:)-iterC(3,:));
    dParaMatrix(1,4) = abs(readD(1,:)-iterD(1,:));
    dParaMatrix(2,4) = abs(readD(2,:)-iterD(2,:));
    dParaMatrix(3,4) = abs(readD(3,:)-iterD(3,:));
    dParaMatrix(1,5) = abs(readE(1,:)-iterE(1,:));
```

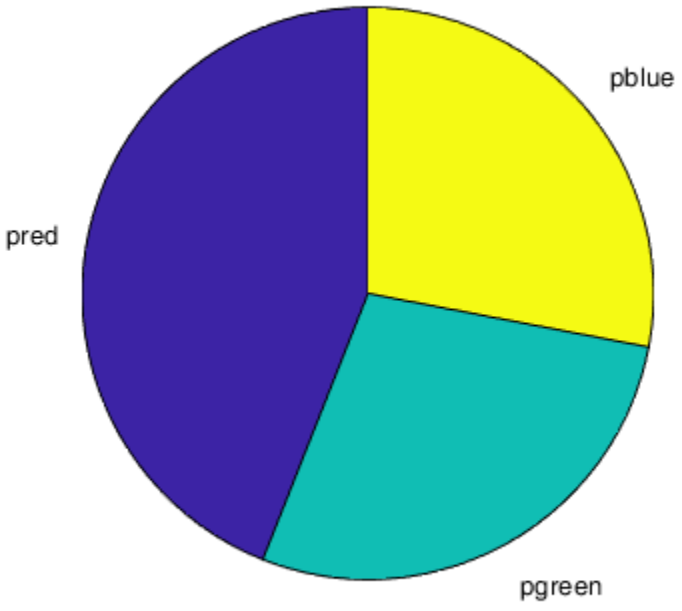
```
    dParaMatrix(2,5) = abs(readE(2,:)-iterE(2,:));
    dParaMatrix(3,5) = abs(readE(3,:)-iterE(3,:));
    dParameterRow = max(dParaMatrix);
    dParameter1 = max(dParameterRow);
    count = count + 1;
end
dParameter1;
pred;
pgreen;
pblue;
X = [pred pgreen pblue];
labels = {'pred','pgreen','pblue'};
figure
pie(X,labels)
snapnow
count;
Y = [0 0 1 1 1;0 1 0 0 1;1 1 1 0 0];
readA= Y(:,1)/sum(Y(:,1));
readB= Y(:,2)/sum(Y(:,2));
readC= Y(:,3)/sum(Y(:,3));
readD= Y(:,4)/sum(Y(:,4));
readE= Y(:,5)/sum(Y(:,5));
INTreadA= Y(:,1)/sum(Y(:,1));
INTreadB= Y(:,2)/sum(Y(:,2));
INTreadC= Y(:,3)/sum(Y(:,3));
INTreadD= Y(:,4)/sum(Y(:,4));
INTreadE= Y(:,5)/sum(Y(:,5));
count = 0;
dParameter1 = 1;
while dParameter1 > 0.001
    iterA = readA;
    iterB = readB;
    iterC = readC;
    iterD = readD;
    iterE = readE;
    %Single iteration prep (M step)
    totalChance =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))+(readA(2,:)+readB(2,:)+r
    pred = (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:))/
totalChance;
    pgreen = (readA(2,:)+readB(2,:)+readC(2,:)+readD(2,:)+readE(2,:))/
totalChance;
    pblue = (readA(3,:)+readB(3,:)+readC(3,:)+readD(3,:)+readE(3,:))/
totalChance;
    %Set new read percentag
    readA(1,:) = INTreadA(1,:)*pred;
    readB(1,:) = INTreadB(1,:)*pred;
    readC(1,:) = INTreadC(1,:)*pred;
    readD(1,:) = INTreadD(1,:)*pred;
    readE(1,:) = INTreadE(1,:)*pred;
    readA(2,:) = INTreadA(2,:)*pgreen;
    readB(2,:) = INTreadB(2,:)*pgreen;
    readC(2,:) = INTreadC(2,:)*pgreen;
    readD(2,:) = INTreadD(2,:)*pgreen;
```

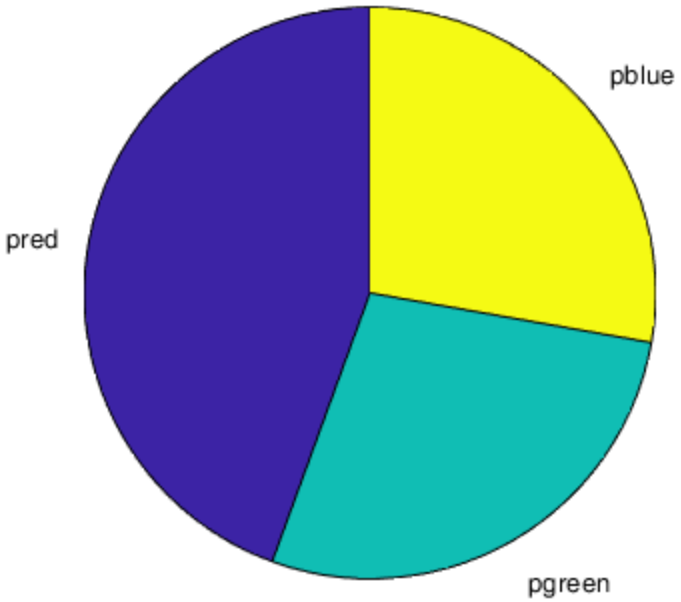
```
        readE(2,:) = INTreadE(2,:)*pgreen;
        readA(3,:) = INTreadA(3,:)*pblue;
        readB(3,:) = INTreadB(3,:)*pblue;
        readC(3,:) = INTreadC(3,:)*pblue;
        readD(3,:) = INTreadD(3,:)*pblue;
        readE(3,:) = INTreadE(3,:)*pblue;
        %Normalize read percentages
        readA = readA/sum(readA);
        readB = readB/sum(readB);
        readC = readC/sum(readC);
        readD = readD/sum(readD);
        readE = readE/sum(readE);
        %change in parameter
        dParaMatrix = zeros(3,5);
        dParaMatrix(1,1) = abs(readA(1,:)-iterA(1,:));
        dParaMatrix(2,1) = abs(readA(2,:)-iterA(2,:));
        dParaMatrix(3,1) = abs(readA(3,:)-iterA(3,:));
        dParaMatrix(1,2) = abs(readB(1,:)-iterB(1,:));
        dParaMatrix(2,2) = abs(readB(2,:)-iterB(2,:));
        dParaMatrix(3,2) = abs(readB(3,:)-iterB(3,:));
        dParaMatrix(1,3) = abs(readC(1,:)-iterC(1,:));
        dParaMatrix(2,3) = abs(readC(2,:)-iterC(2,:));
        dParaMatrix(3,3) = abs(readC(3,:)-iterC(3,:));
        dParaMatrix(1,4) = abs(readD(1,:)-iterD(1,:));
        dParaMatrix(2,4) = abs(readD(2,:)-iterD(2,:));
        dParaMatrix(3,4) = abs(readD(3,:)-iterD(3,:));
        dParaMatrix(1,5) = abs(readE(1,:)-iterE(1,:));
        dParaMatrix(2,5) = abs(readE(2,:)-iterE(2,:));
        dParaMatrix(3,5) = abs(readE(3,:)-iterE(3,:));
        dParameterRow = max(dParaMatrix);
        dParameter1 = max(dParameterRow);
        count = count + 1;
        if count < 5
            X = [pred pgreen pblue];
            labels = {'pred','pgreen','pblue'};
            figure
            pie(X,labels)
            snapnow
        end
    end
    dParameter1;
    pred;
    pgreen;
    pblue;
    X = [pred pgreen pblue];
    labels = {'pred','pgreen','pblue'};
    figure
    pie(X,labels)
    snapnow
    count;
    %This initialization converges to be without any green.
```

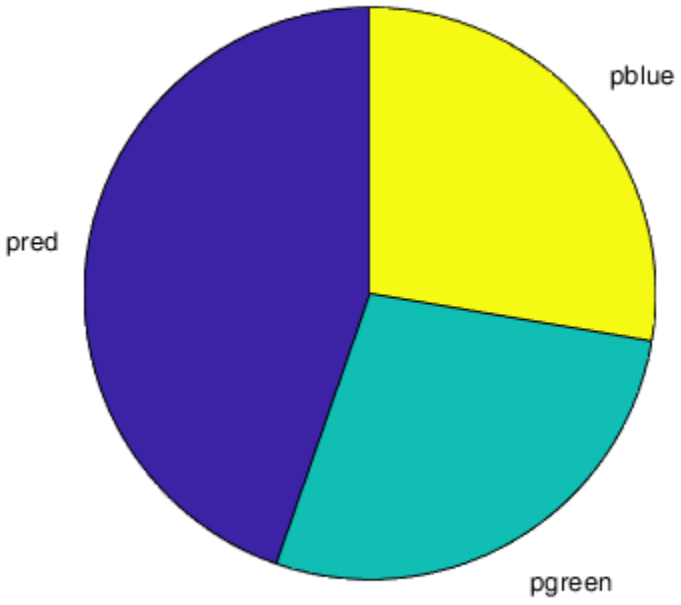

pblue
pred
pgreen

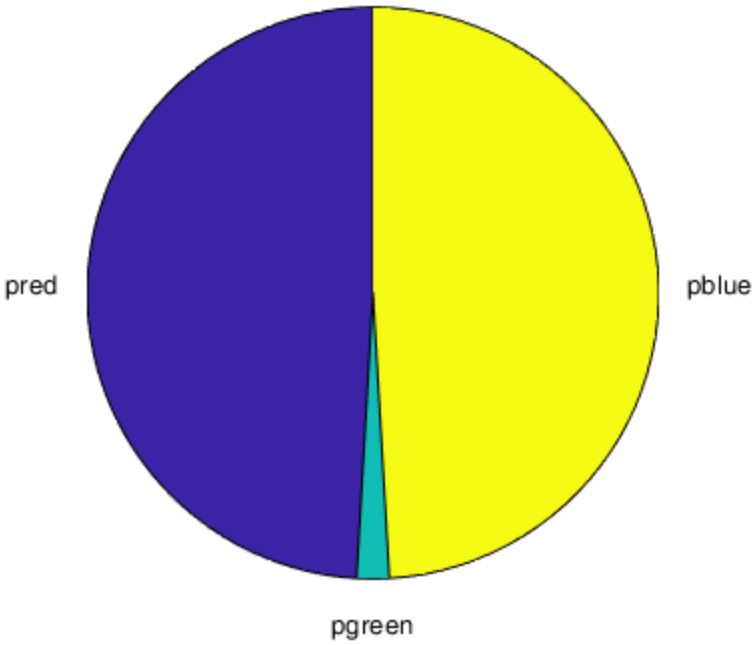

pblue
pred
pgreen

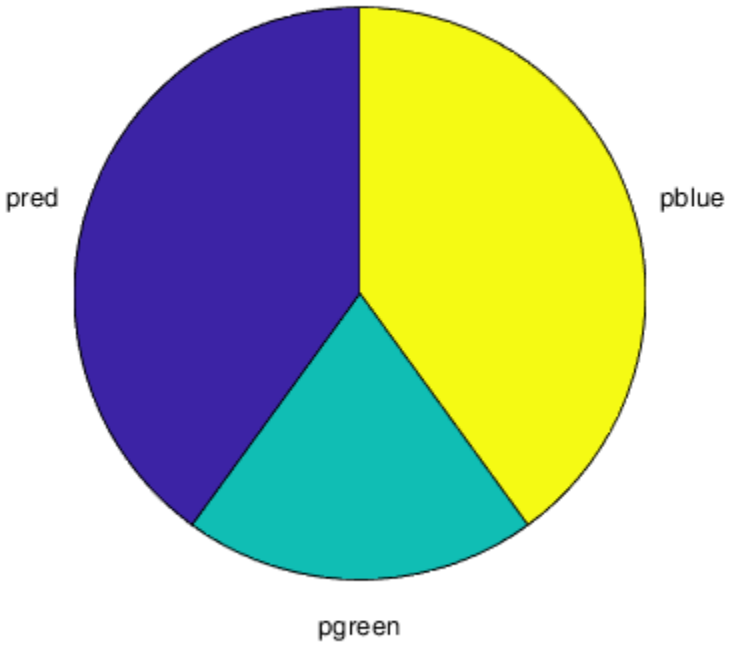

pblue
pred
pgreen

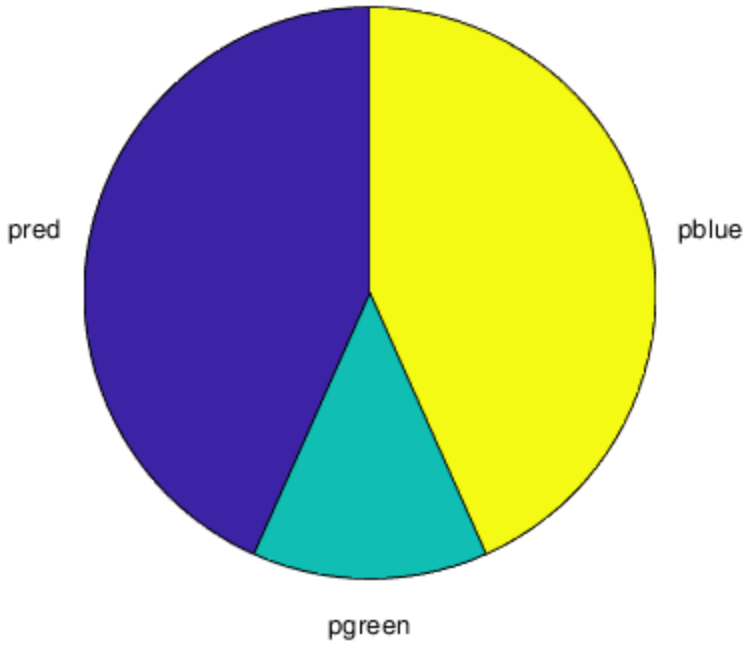

pblue
pred
pgreen

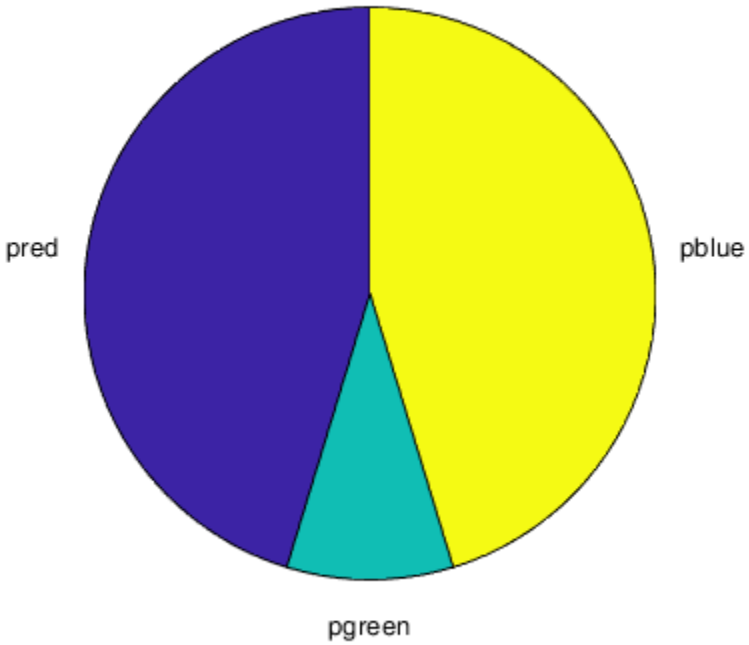

pblue
pred
pgreen

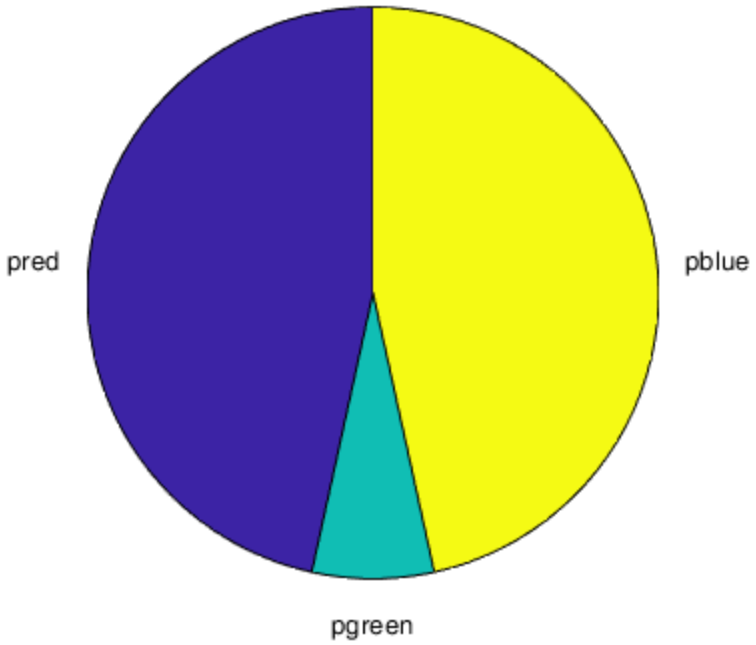

pblue
pred
pgreen

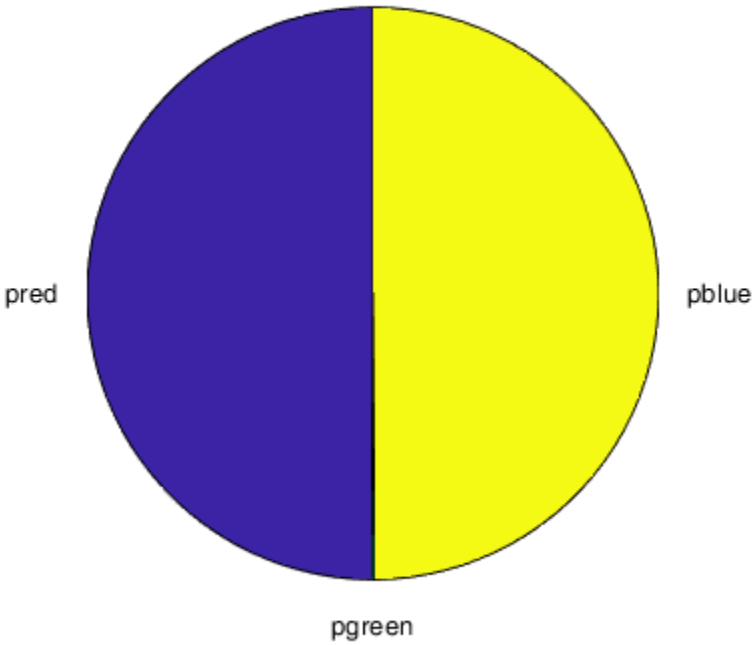

pred
pblue
pgreen


pred
pblue
pgreen


pred
pblue
pgreen


pred
pblue
pgreen


pred
pblue
pgreen


pred
pblue
pgreen

Section (1D)

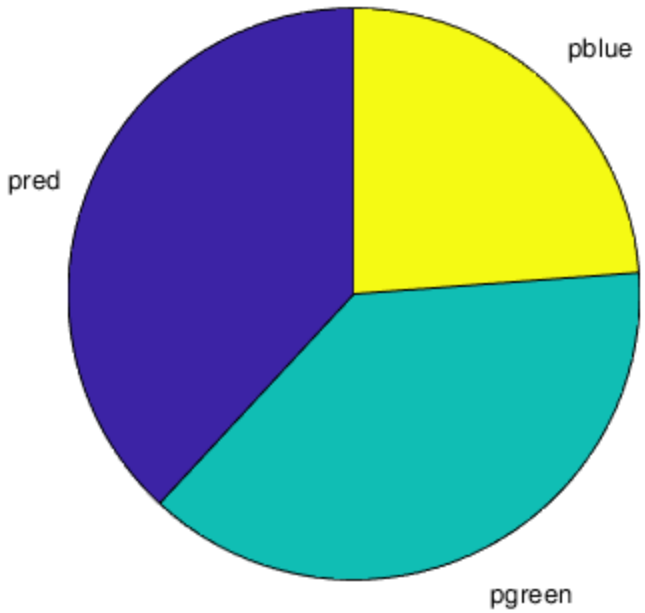
New likelihood function: L(P) = (pred/sum(p)+pgreen/sum(p)+pblue/sum(p))^2*(pgreen/sum(p)+p-blue/sum(p))*((pred/sum(p)+pblue/sum(p))*(pred/sum(p))*(pred/sum(p)+pgreen/sum(p))*(pgreen/sum(p)) log L(P) = log(pgreen + pblue) + log(pred + pblue)+ log(pred) + log(pred+pgreen)+log(pgreen). The new likelihood function is still concave since it converges to a local maximum.

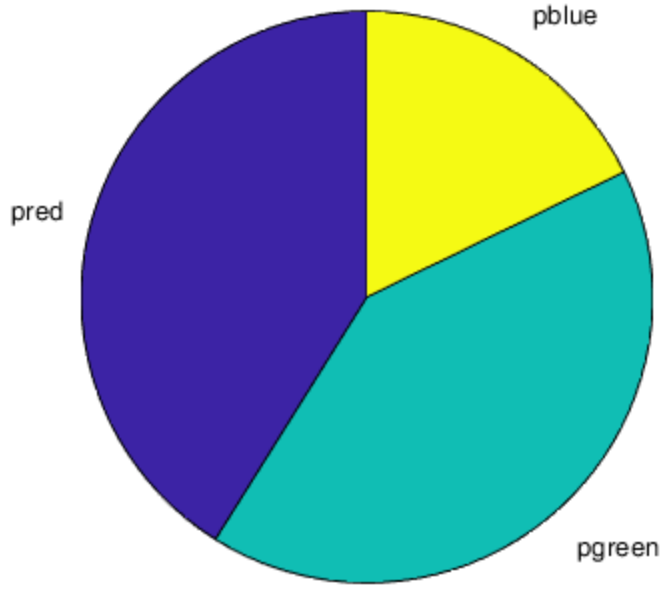
Section(1E)

```
Y = [1 0 1 1 1 1 0;1 1 0 0 1 1 1;1 1 1 0 0 1 0];
readA= Y(:,1)/sum(Y(:,1));
readB= Y(:,2)/sum(Y(:,2));
readC= Y(:,3)/sum(Y(:,3));
readD= Y(:,4)/sum(Y(:,4));
readE= Y(:,5)/sum(Y(:,5));
readF= Y(:,6)/sum(Y(:,6));
readG= Y(:,7)/sum(Y(:,7));
INTreadA= Y(:,1)/sum(Y(:,1));
INTreadB= Y(:,2)/sum(Y(:,2));
INTreadC= Y(:,3)/sum(Y(:,3));
INTreadD= Y(:,4)/sum(Y(:,4));
INTreadE= Y(:,5)/sum(Y(:,5));
INTreadF= Y(:,6)/sum(Y(:,6));
INTreadG= Y(:,7)/sum(Y(:,7));
count = 0;
dParameter1 = 1;
while dParameter1 > 0.001
    iterA = readA;
    iterB = readB;
    iterC = readC;
    iterD = readD;
    iterE = readE;
    iterF = readF;
    iterG = readG;
    %Single iteration prep (M step)
    totalChance =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:)+readF(1,:)+readG(1,:))+(r
    pred =
 (readA(1,:)+readB(1,:)+readC(1,:)+readD(1,:)+readE(1,:)+readF(1,:)+readG(1,:))/
totalChance;
    pgreen =
 (readA(2,:)+readB(2,:)+readC(2,:)+readD(2,:)+readE(2,:)+readF(2,:)+readG(2,:))/
totalChance;
    pblue =
 (readA(3,:)+readB(3,:)+readC(3,:)+readD(3,:)+readE(3,:)+readF(3,:)+readG(3,:))/
totalChance;
    %Set new read percentag
    readA(1,:) = INTreadA(1,:)*pred;
    readB(1,:) = INTreadB(1,:)*pred;
    readC(1,:) = INTreadC(1,:)*pred;
    readD(1,:) = INTreadD(1,:)*pred;
```

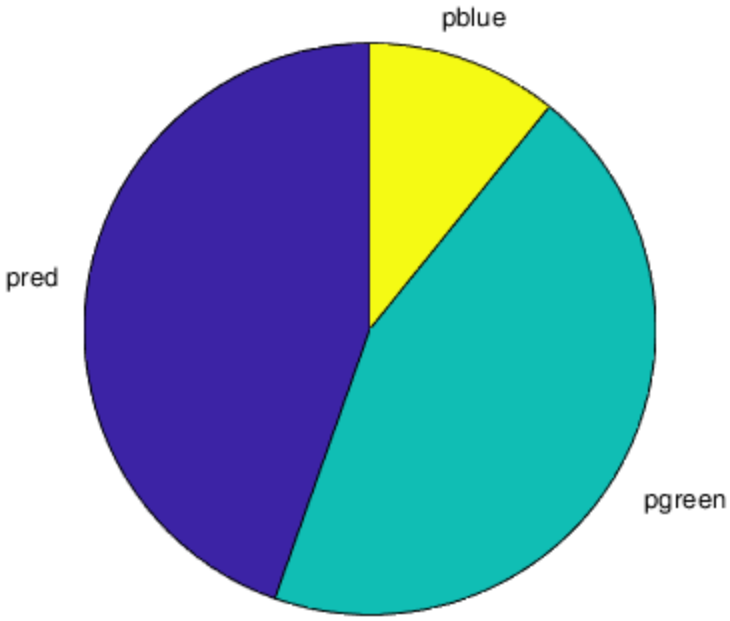
```
readE(1,:) = INTreadE(1,:)*pred;
readF(1,:) = INTreadF(1,:)*pred;
readG(1,:) = INTreadG(1,:)*pred;
readA(2,:) = INTreadA(2,:)*pgreen;
readB(2,:) = INTreadB(2,:)*pgreen;
readC(2,:) = INTreadC(2,:)*pgreen;
readD(2,:) = INTreadD(2,:)*pgreen;
readE(2,:) = INTreadE(2,:)*pgreen;
readF(2,:) = INTreadF(2,:)*pgreen;
readG(2,:) = INTreadG(2,:)*pgreen;
readA(3,:) = INTreadA(3,:)*pblue;
readB(3,:) = INTreadB(3,:)*pblue;
readC(3,:) = INTreadC(3,:)*pblue;
readD(3,:) = INTreadD(3,:)*pblue;
readE(3,:) = INTreadE(3,:)*pblue;
readF(3,:) = INTreadF(3,:)*pblue;
readG(3,:) = INTreadG(3,:)*pblue;
%Normalize read percentages
readA = readA/sum(readA);
readB = readB/sum(readB);
readC = readC/sum(readC);
readD = readD/sum(readD);
readE = readE/sum(readE);
readF = readF/sum(readF);
readG = readG/sum(readG);
%change in parameter
dParaMatrix = zeros(3,7);
dParaMatrix(1,1) = abs(readA(1,:)-iterA(1,:));
dParaMatrix(2,1) = abs(readA(2,:)-iterA(2,:));
dParaMatrix(3,1) = abs(readA(3,:)-iterA(3,:));
dParaMatrix(1,2) = abs(readB(1,:)-iterB(1,:));
dParaMatrix(2,2) = abs(readB(2,:)-iterB(2,:));
dParaMatrix(3,2) = abs(readB(3,:)-iterB(3,:));
dParaMatrix(1,3) = abs(readC(1,:)-iterC(1,:));
dParaMatrix(2,3) = abs(readC(2,:)-iterC(2,:));
dParaMatrix(3,3) = abs(readC(3,:)-iterC(3,:));
dParaMatrix(1,4) = abs(readD(1,:)-iterD(1,:));
dParaMatrix(2,4) = abs(readD(2,:)-iterD(2,:));
dParaMatrix(3,4) = abs(readD(3,:)-iterD(3,:));
dParaMatrix(1,5) = abs(readE(1,:)-iterE(1,:));
dParaMatrix(2,5) = abs(readE(2,:)-iterE(2,:));
dParaMatrix(3,5) = abs(readE(3,:)-iterE(3,:));
dParaMatrix(1,6) = abs(readF(1,:)-iterF(1,:));
dParaMatrix(2,6) = abs(readF(2,:)-iterF(2,:));
dParaMatrix(3,6) = abs(readF(3,:)-iterF(3,:));
dParaMatrix(1,7) = abs(readG(1,:)-iterG(1,:));
dParaMatrix(2,7) = abs(readG(2,:)-iterG(2,:));
dParaMatrix(3,7) = abs(readG(3,:)-iterG(3,:));
dParameterRow = max(dParaMatrix);
dParameter1 = max(dParameterRow);
count = count + 1;
if count < 5
    X = [pred pgreen pblue];
    labels = {'pred','pgreen','pblue'};
```

```
        figure
        pie(X,labels)
        snapnow
    end
end
dParameter1;
readA;
count;
%system converges from multiple intializations to minimize pblue.
 starting
%with green and blue at 50/50 instead of 3/3rds causes convergence to
 10%
%blue instead of 0.2%. red/blue 50/50 similarly converges blue to only
 1%.
```




pblue
pred
pgreen


pblue
pred
pgreen



Section (1F)

i.Likelihood: L(P) = (pred/sum(p)+pgreen/sum(p)+pblue/sum(p))^100*(pgreen/sum(p)+p-blue/sum(p))^150*((pred/sum(p)+pblue/sum(p))^200*(pred/sum(p))^250*(pred/ sum(p)+pgreen/sum(p))^300*(pgreen/sum(p))^400

```
%Log Likelihood:
%150log(pgreen/sum(p)+pblue/sum(p))+200log(pred/sum(p)+pblue/
sum(p))+250log(pred/sum(p))+300log(pred/sum(p)+pgreen/
sum(p))+400log(pgreen/sum(p)).
aReads = [1; 1; 1];
aReads = repmat(aReads,1,100);
bReads = [0; 1; 1];
bReads = repmat(bReads,1,150);
cReads = [1; 0; 1];
cReads = repmat(cReads,1,200);
dReads = [1; 0; 0];
dReads = repmat(dReads,1,250);
eReads = [1; 1; 0];
eReads = repmat(eReads,1,300);
gReads = [0; 1; 0];
gReads = repmat(gReads,1,400);

Y = [aReads bReads cReads dReads eReads gReads];
dY = Y;
count = 0;
```

```
dP = 1;
while dP > 0.001
    iterY = dY;
    totalchance = sum(dY,'all');
    pred = sum(dY(1,:),'all')./totalchance;
    pgreen = sum(dY(2,:),'all')./totalchance;
    pblue = sum(dY(3,:),'all')./totalchance;
    pChart = [pred pgreen pblue];
    for k = 1:3
        dY(k,:) = Y(k,:).*pChart(k);
    end
    for i = 1:1400
            dY(:,i) = dY(:,i)./sum(dY(:,i),'all');
    end
    dParaMatrix = zeros(3,1400);
    for k = 1:3
        for i = 1:1400
            dParaMatrix(k,i) = abs(dY(k,i)-iterY(k,i));
        end
    end
    dParameterRow = max(dParaMatrix);
    dP = max(dParameterRow);
    count = count + 1;
end
dY(:,1);
dP;
count;
pSave = pChart;
pSabund = pChart.*1400;
% ii. The results are similar as the single read sample. This likely
 comes about from a lack of blue only reads, so the likelihood
 function is drawn over times to more prevalent readings.
% iii. The number of reads does not increase operational complexity.
 While
% the structure of the code would give the appearance of going from
 O(n) to
% O(n^2), both were O(n^2) in order to make the parameter difference
% matrix.
```

Section (1G)

The derivation of E and M would change. The relative length effects the likelihood function by dividing the sampling probability. Since new locations exist for the read to attach to on the same transcript, these differences apply by substituting At for p(t)*l(t)/sum(p(r)l(r)) where t is the investigated transcript and r is a summation across all existing transcripts. This would effect the E step as the relative lengths would need to be multiplied into the top and bottom.

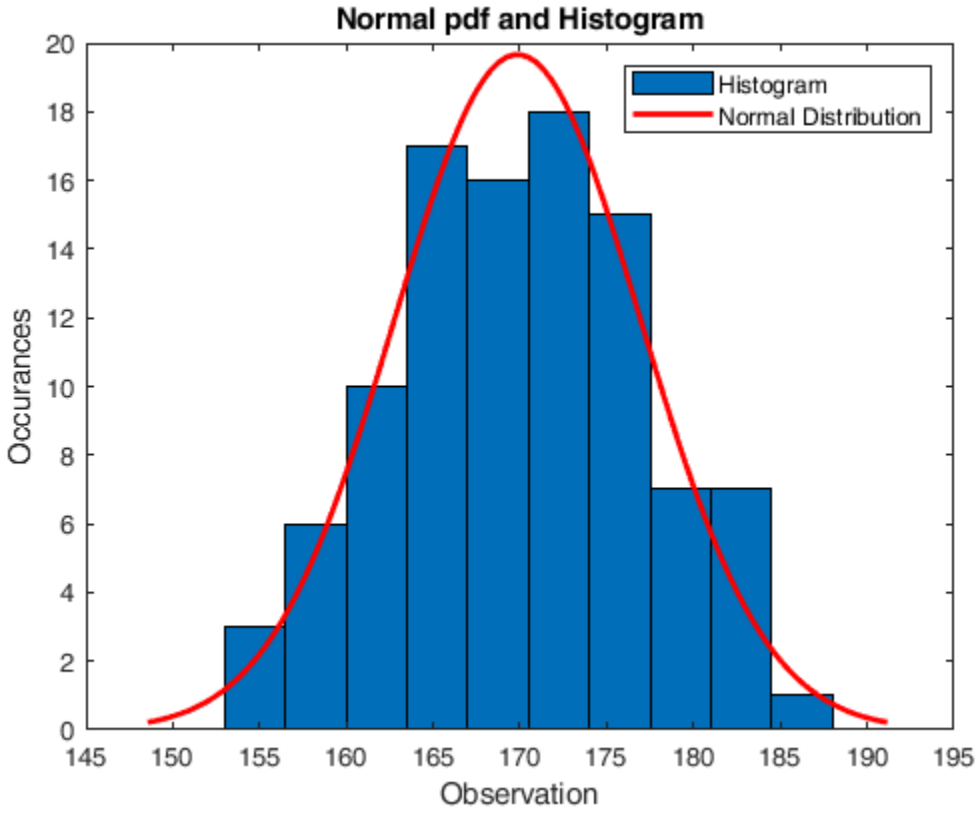
Problem 2

```
DUMBHO = 170;
OOF = 7;
```

Section (2A)

```
r = normrnd(170,7,100,1);
heightAVG= mean(r,'all')
histfit(r)
title('Normal pdf and Histogram')
xlabel('Observation')
ylabel('Occurances')
legend('Histogram','Normal Distribution')
```

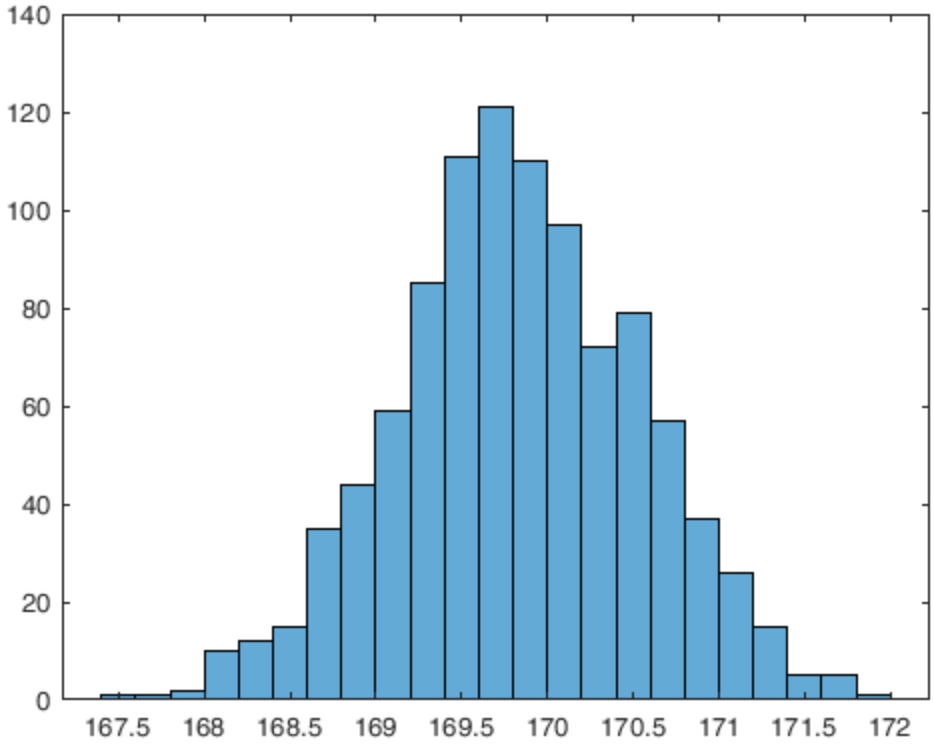
```
heightAVG =

  169.8559
```



Section (2B)

```
y = randsample(r,100,true);

%i.Probability of selecting same dataset: (1/100)^100
%ii. Probabiltiy of finding the same dataset of n observations in x
 attempts is
%P(X=<x) = 1-(1-(1/n)^n)^x
%iii. P(X=<5) = (1-(1-(1/1000)^1000)^5 = 0 (effectively).
```

```
%Z% Section (2c)

x = [];
for index = 1:1000
    omniset = datasample(r,100);
    omnisetavg = mean(omniset);
    x(end+1) = omnisetavg;
end
x;
xavg = mean(x);
figure
hist2= histogram(x)


hist2 =

  Histogram with properties:

            Data: [1×1000 double]
          Values: [1×23 double]
         NumBins: 23
        BinEdges: [1×24 double]
        BinWidth: 0.2000
       BinLimits: [167.4000 172]
   Normalization: 'count'
       FaceColor: 'auto'
       EdgeColor: [0 0 0]

  Use GET to show all properties
```

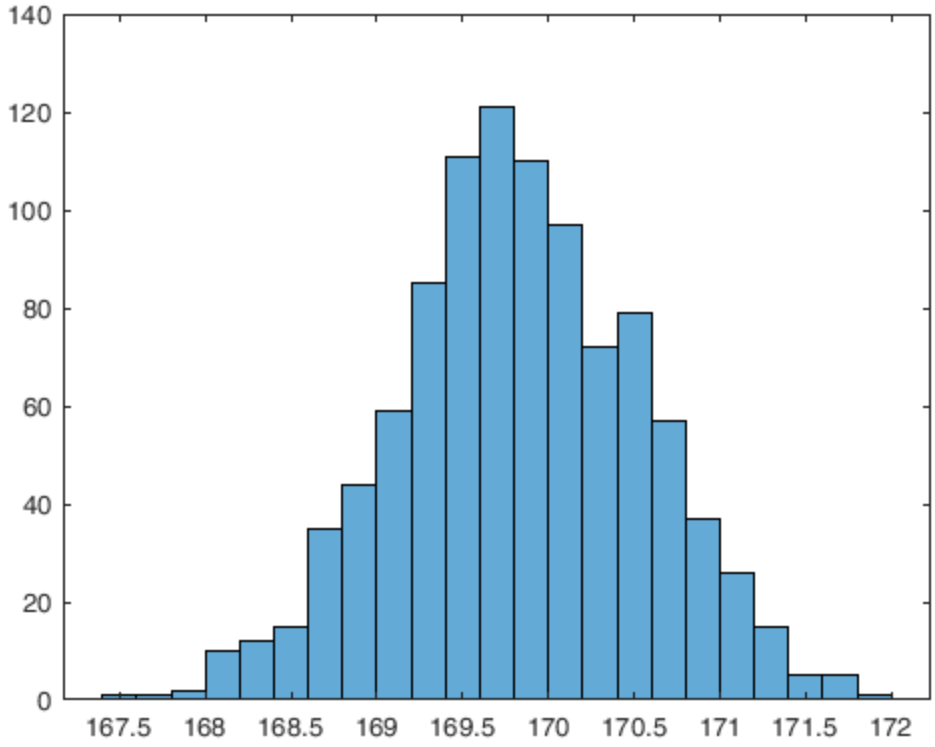


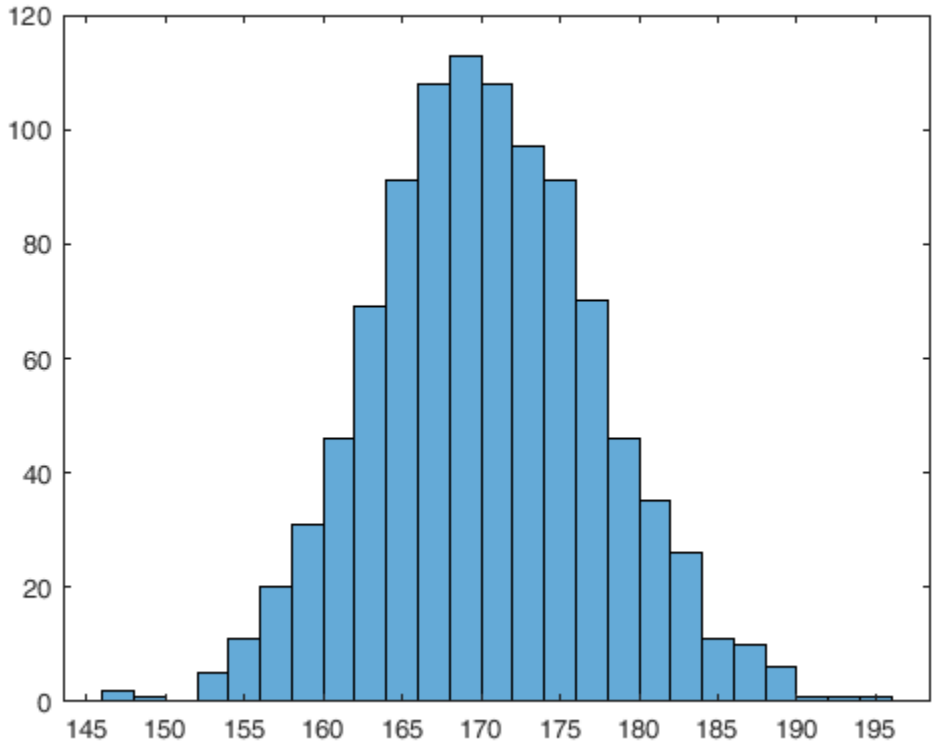
Section (2D)

```
Xstd = std(x)
StandardErrorM = Xstd/sqrt(length(x));
ts = tinv([0.025 0.975],length(x)-1);
CI = xavg + ts*StandardErrorM
xprime = x';
pd = fitdist(xprime,'Normal');
ci = paramci(pd)
%CI values agree


Xstd =

    0.7135


CI =

  169.7865  169.8751


ci =

  169.7865    0.6836
  169.8751    0.7462
```

Section (2E)

```
x = normrnd(170,7,1000,1);
x;
xavg = mean(x);
figure
hist3= histogram(x)
hold off
Xstd = std(x);
StandardErrorM = Xstd/sqrt(length(x));
ts = tinv([0.025 0.975],length(x)-1);
CI = xavg + ts*StandardErrorM
xprime = x';
pd = fitdist(x,'Normal');
ci = paramci(pd)


hist3 =

  Histogram with properties:

            Data: [1000×1 double]
          Values: [1×25 double]
         NumBins: 25
        BinEdges: [1×26 double]
        BinWidth: 2
       BinLimits: [146 196]
   Normalization: 'count'
       FaceColor: 'auto'
       EdgeColor: [0 0 0]

  Use GET to show all properties


CI =

  169.8500  170.7323


ci =

  169.8500    6.8104
  170.7323    7.4349
```

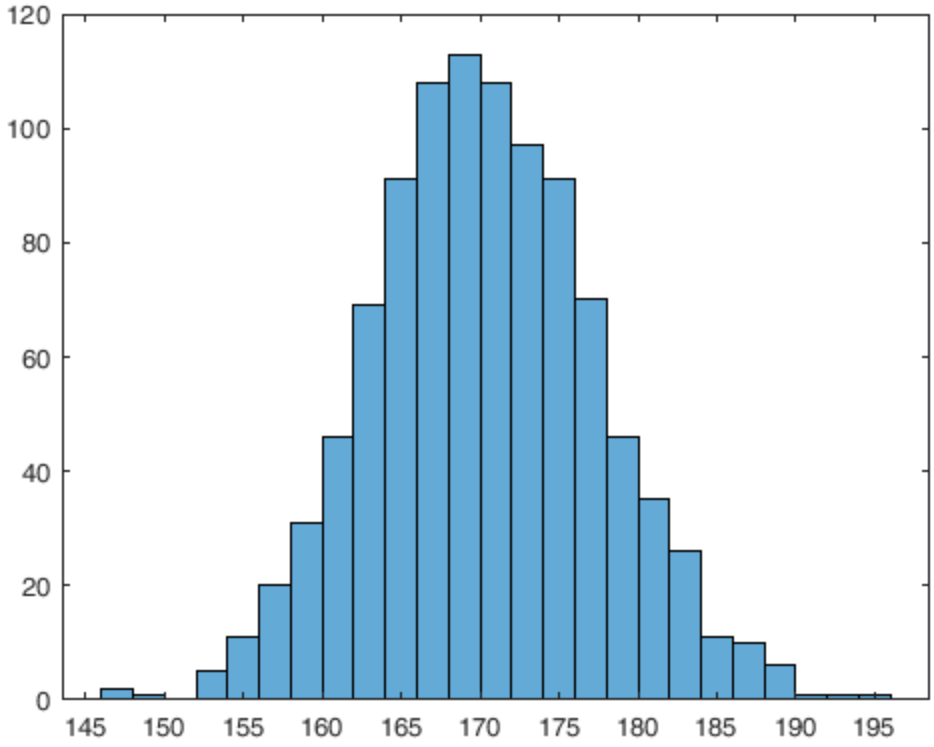

140
120
100
80
60
40
20
0
167.5
168
168.5
169
169.5
170
170.5
171
171.5
172

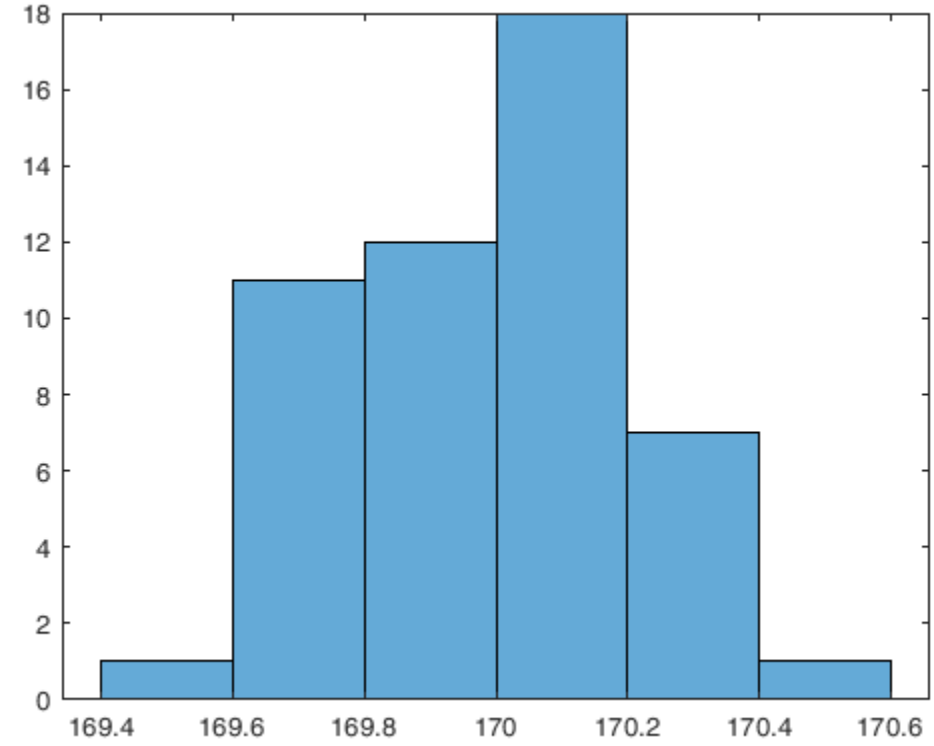

120
100
80
60
40
20
0
145
150
155
160
165
170
175
180
185
190
195

Section (2F)

```
r = normrnd(170,7,1000,1);
x = [];
for index = 1:50
    omniset = datasample(r,1000);
    omnisetavg = mean(omniset);
    x(end+1) = omnisetavg;
end
x;
xavg = mean(x);
figure
hist4= histogram(x)
Xstd = std(x);
StandardErrorM = Xstd/sqrt(length(x));
ts = tinv([0.025 0.975],length(x)-1);
CI = xavg + ts*StandardErrorM
xprime = x';
pd = fitdist(xprime,'Normal');
ci = paramci(pd)
%the CI for the 50 iteration version is much tighter then the 1000
%original samples/
```

```
hist4 =

  Histogram with properties:

            Data: [1×50 double]
          Values: [1 11 12 18 7 1]
         NumBins: 6
        BinEdges: [169.4000 169.6000 169.8000 170 170.2000 170.4000
 170.6000]
         BinWidth: 0.2000
        BinLimits: [169.4000 170.6000]
    Normalization: 'count'
        FaceColor: 'auto'
        EdgeColor: [0 0 0]

  Use GET to show all properties


CI =

  169.9308  170.0580


ci =

  169.9308    0.1869
  170.0580    0.2788
```

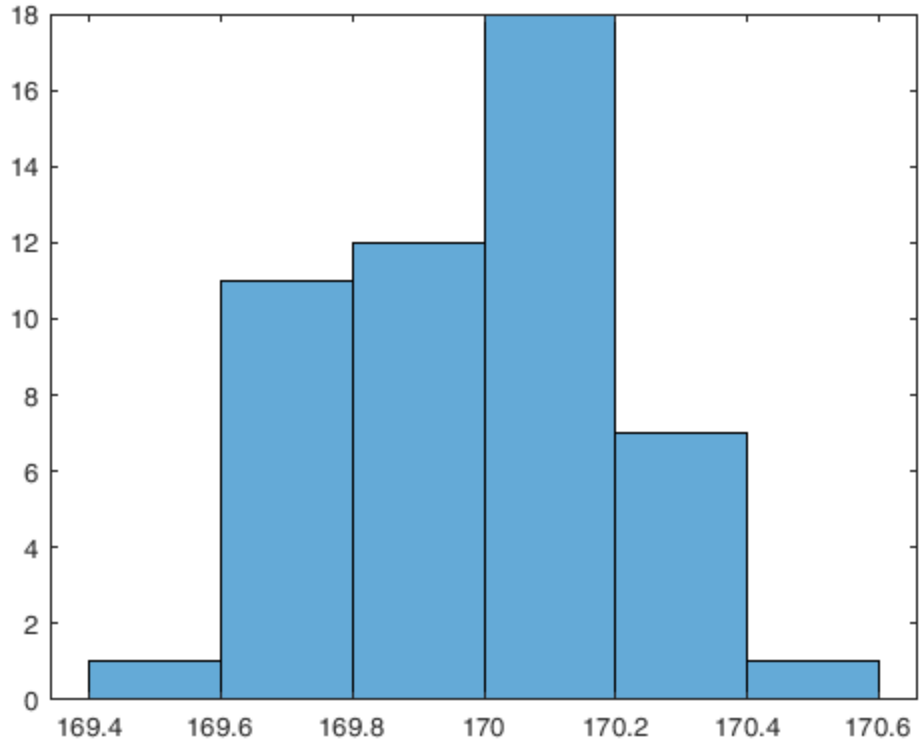

120
100
80
60
40
20
0
145
150
155
160
165
170
175
180
185
190
195

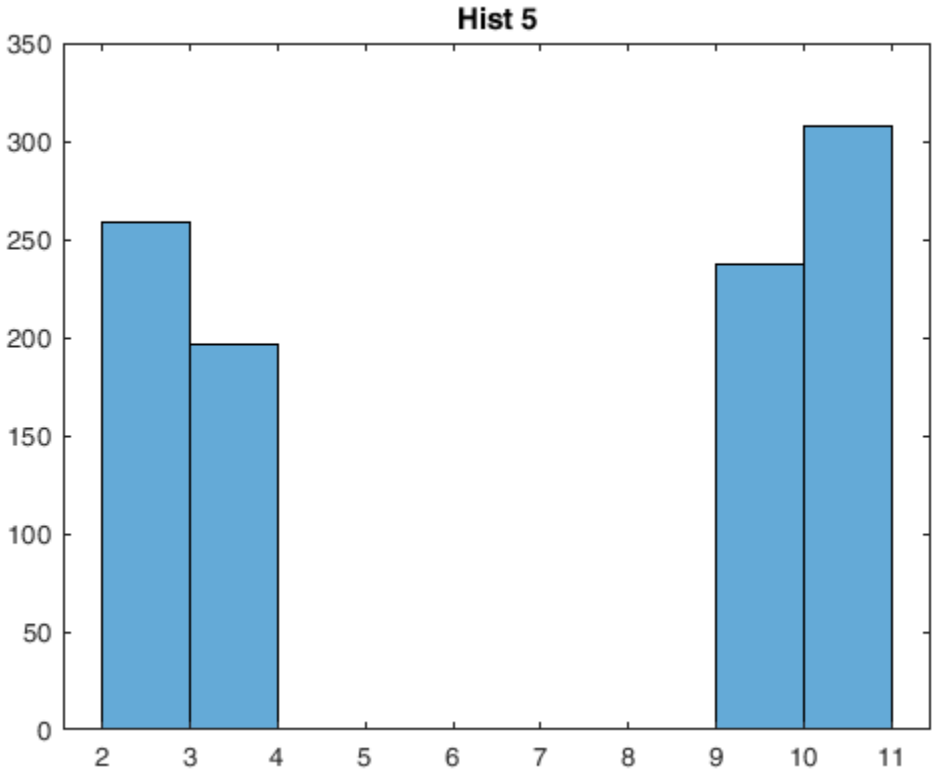

18
16
14
12
10
8
6
4
2
0
169.4
169.6
169.8
170
170.2
170.4
170.6

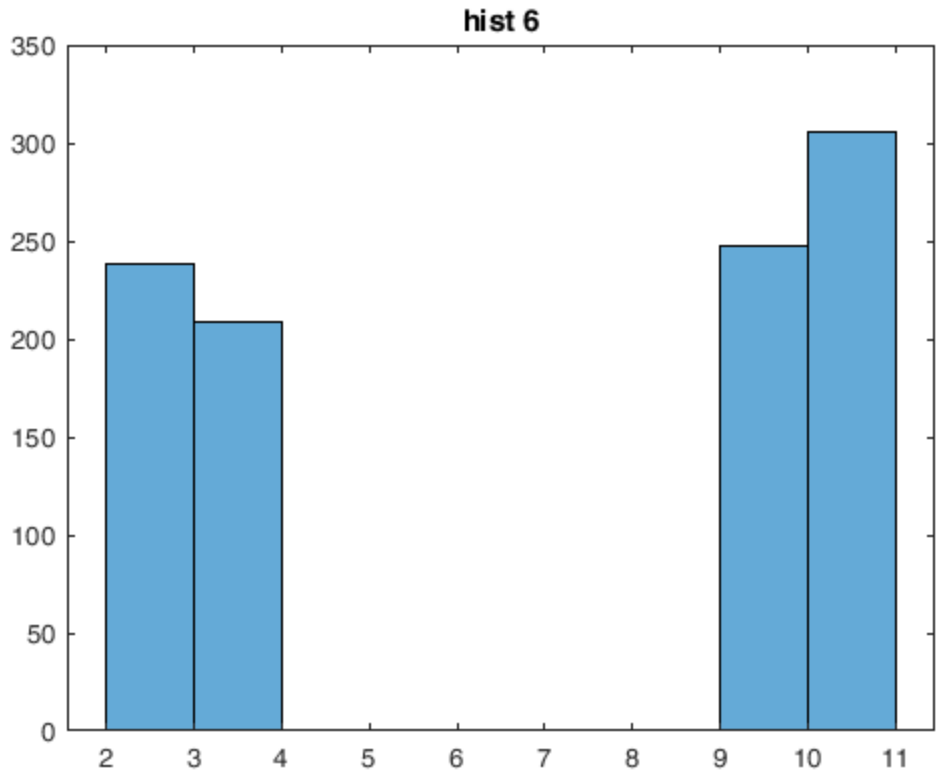
Section (2g)

```
%i.
x = [];
for index = 1:1000
    r = randsample(10,1);
    if r == 1|2|3
        val = normrnd(0,1,100,1);
    end
    if r == 4|5
        val = normrnd(3,0.1,100,1);
    end
    if r > 5
        val = normrnd(10,1,100,1);
    end
    x = [x val];
end
xi = [];
for index = 1:1000
    r = randsample(100,1);
    r = x(:, r);
    omniset = datasample(r,100);
    omnisetavg = mean(omniset);
    xi(end+1) = omnisetavg;
end
xavg = mean(xi);
figure
hist5= histogram(xi)
title('Hist 5')
hold on
Xstd = std(xi);
StandardErrorM = Xstd/sqrt(length(xi));
ts = tinv([0.025 0.975],length(xi)-1);
CI = xavg + ts*StandardErrorM
xprime = xi';
pd = fitdist(xprime,'Normal');
ci = paramci(pd)
%ii.
xi = [];
for index = 1:1000
    r = randsample(100,1);
    r = x(:, r);
    r = r';
    omniset = datasample(r,100);
    omnisetavg = mean(omniset);
    xi(end+1) = omnisetavg;
end
xi;
xavg = mean(xi);
figure
hist6= histogram(xi)
title('hist 6')
hold off
```

```
%iii. Since the bootstrap is based of of the different percentage
%allocations of the mixng coefficients, it also reflects the random
%selection of distribution found in the original sample.


hist5 =

  Histogram with properties:

            Data: [1×1000 double]
          Values: [259 196 0 0 0 0 0 237 308]
         NumBins: 9
        BinEdges: [2 3 4 5 6 7 8 9 10 11]
        BinWidth: 1
       BinLimits: [2 11]
    Normalization: 'count'
        FaceColor: 'auto'
        EdgeColor: [0 0 0]

  Use GET to show all properties


CI =

    6.6119    7.0467


ci =

    6.6119    3.3567
    7.0467    3.6646


hist6 =

  Histogram with properties:

            Data: [1×1000 double]
          Values: [238 209 0 0 0 0 0 247 306]
         NumBins: 9
        BinEdges: [2 3 4 5 6 7 8 9 10 11]
        BinWidth: 1
       BinLimits: [2 11]
    Normalization: 'count'
        FaceColor: 'auto'
        EdgeColor: [0 0 0]

  Use GET to show all properties
```


18
16
14
12
10
8
6
4
2
0
169.4
169.6
169.8
170
170.2
170.4
170.6


Hist 5
350
300
250
200
150
100
50
0
2
3
4
5
6
7
8
9
10
11



Problem 3 Section (3A)

```
%Starting with the read matrix Y, we can generated simulated samples
 of
%1400 reads by randomly selecting with replacement 1400 new reads.
 These
%will form new datasets that could then be used to create a set of EM
%aquired abundances.
```

Section (3B)

```
pRuns = [];
for j = 1:10
    sampledY = Y;
    for i = 1:1400
        r = randsample(1400,1);
        sampledY(:,i)=Y(:,r);
    end
    dY = sampledY;
    count = 0;
    dP = 1;
    while dP > 0.001
        iterY = dY;
        totalchance = sum(dY,'all');
        pred = sum(dY(1,:),'all')./totalchance;
```

```
        pgreen = sum(dY(2,:),'all')./totalchance;
        pblue = sum(dY(3,:),'all')./totalchance;
        pChart = [pred; pgreen; pblue];
        for k = 1:3
            dY(k,:) = Y(k,:).*pChart(k);
        end
        for i = 1:1400
                dY(:,i) = dY(:,i)./sum(dY(:,i),'all');
        end
        dParaMatrix = zeros(3,1400);
        for k = 1:3
            for i = 1:1400
                dParaMatrix(k,i) = abs(dY(k,i)-iterY(k,i));
            end
        end
        dParameterRow = max(dParaMatrix);
        dP = max(dParameterRow);
        count = count + 1;
    end
    pRuns = [pRuns pChart];
end
pRuns
```

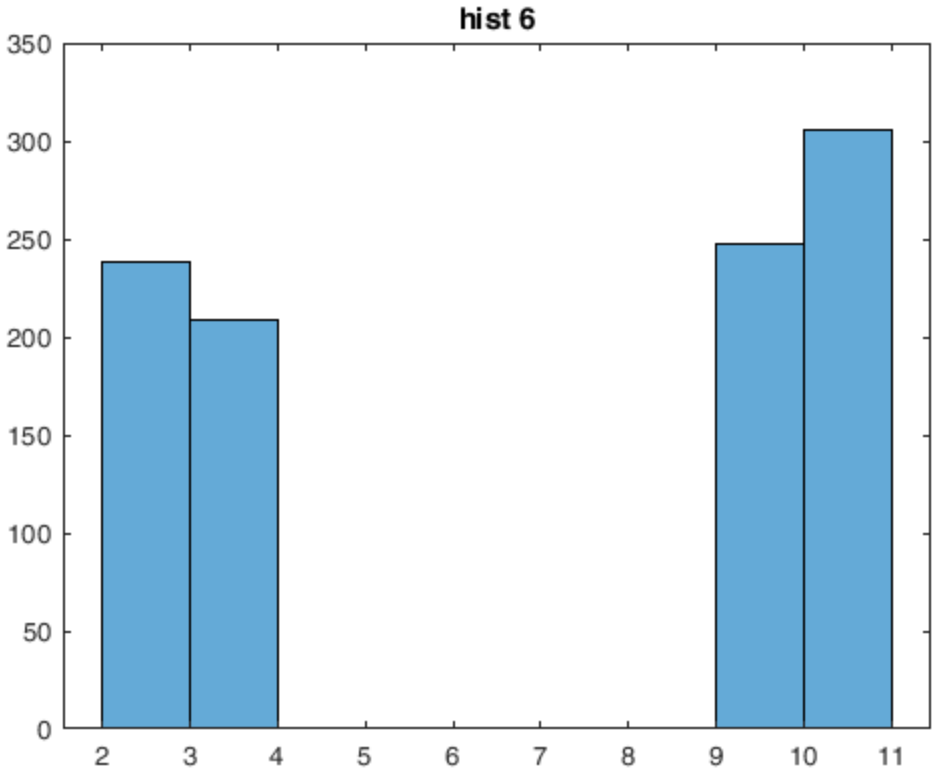
```
pRuns =

  Columns 1 through 7

    0.4497    0.4497    0.4497    0.4497    0.4497    0.4497    0.4497
    0.5499    0.5499    0.5499    0.5499    0.5499    0.5499    0.5499
    0.0004    0.0004    0.0004    0.0004    0.0004    0.0004    0.0004

  Columns 8 through 10

    0.4495    0.4497    0.4497
    0.5498    0.5499    0.5499
    0.0006    0.0004    0.0004
```

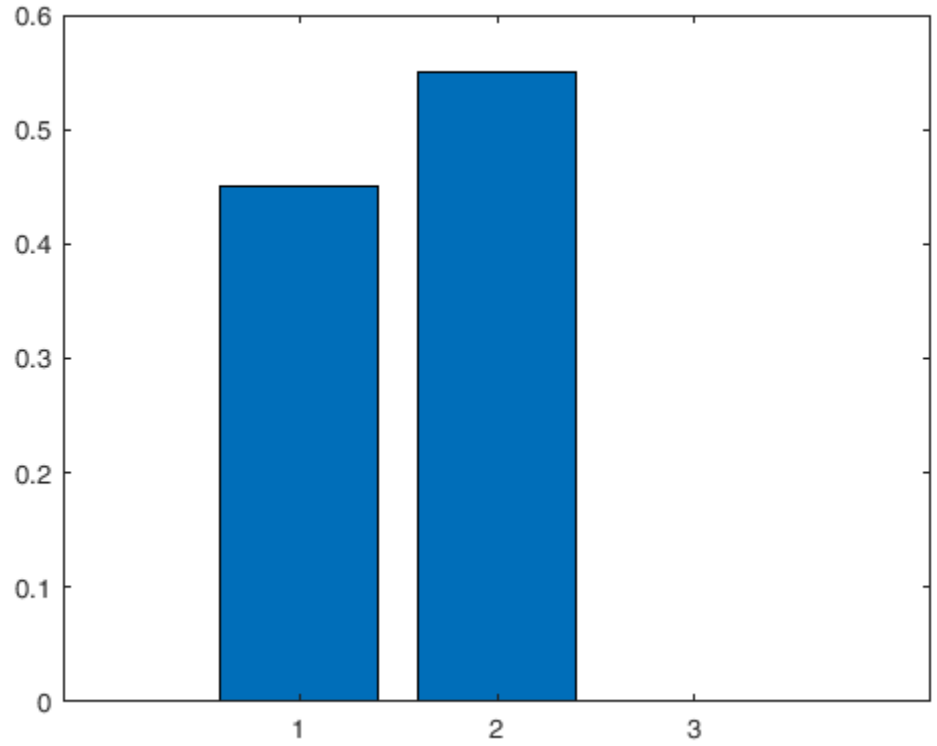
Section (3C)

```
pSave
pRunR =(pRuns(1,:));
pRunG =(pRuns(2,:));
pRunB =(pRuns(3,:));

diffR = pRunR - pSave(1);
diffG = pRunG - pSave(2);
diffB = pRunB - pSave(3);

dSqR = diffR.^2;
dSqG = diffG.^2;
dSqB = diffB.^2;
```

```
sqSumR = sum(dSqR,'all')/9;
sqSumG = sum(dSqG,'all')/9;
sqSumB = sum(dSqB,'all')/9;

stdRed = sqrt(sqSumR);
stdGreen = sqrt(sqSumG);
stdBlue = sqrt(sqSumB);
stderror = [stdRed, stdGreen, stdBlue];
l = 1:3;
figure
bar(l,pSave)

hold on

er = errorbar(l,pSave,stderror);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off
%Error bars are small because there is almost no variation.
```

```
pSave =

    0.4497    0.5499    0.0004
```

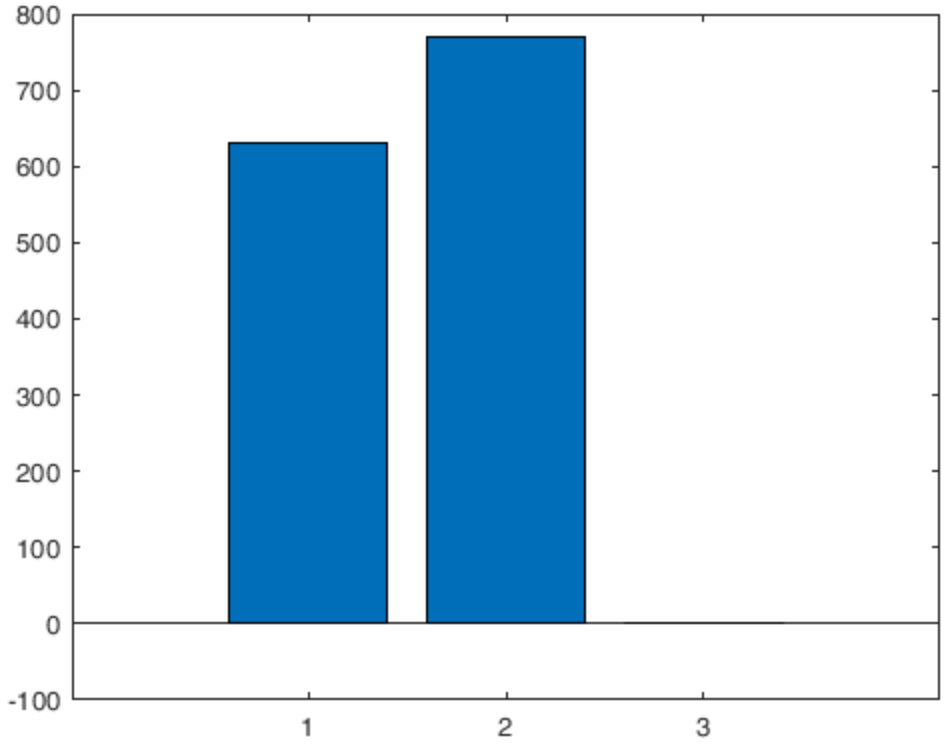

hist 6
350
300
250
200
150
100
50
0
2
3
4
5
6
7
8
9
10
11


0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3

Section (3D)

```
pAbund = [];
for j = 1:10
    sampledY = Y;
    for i = 1:1400
        r = randsample(1400,1);
        sampledY(:,i)=Y(:,r);
    end
    dY = sampledY;
    count = 0;
    dP = 1;
    while dP > 0.001
        iterY = dY;
        totalchance = sum(dY,'all');
        pred = sum(dY(1,:),'all')./totalchance;
        pgreen = sum(dY(2,:),'all')./totalchance;
        pblue = sum(dY(3,:),'all')./totalchance;
        pChart = [pred; pgreen; pblue];
        for k = 1:3
            dY(k,:) = Y(k,:).*pChart(k);
        end
        for i = 1:1400
                dY(:,i) = dY(:,i)./sum(dY(:,i),'all');
        end
        dParaMatrix = zeros(3,1400);
        for k = 1:3
            for i = 1:1400
                dParaMatrix(k,i) = abs(dY(k,i)-iterY(k,i));
            end
        end
        dParameterRow = max(dParaMatrix);
        dP = max(dParameterRow);
        count = count + 1;
    end
    pCount = (pChart.*1400);
    pAbund = [pAbund pCount];
end
pAbund

pRunR =(pAbund(1,:));
pRunG =(pAbund(2,:));
pRunB =(pAbund(3,:));

diffR = pRunR - pSabund(1);
diffG = pRunG - pSabund(2);
diffB = pRunB - pSabund(3);

dSqR = diffR.^2;
dSqG = diffG.^2;
dSqB = diffB.^2;

sqSumR = sum(dSqR,'all')/9;
```

```
sqSumG = sum(dSqG,'all')/9;
sqSumB = sum(dSqB,'all')/9;

stdRed = sqrt(sqSumR);
stdGreen = sqrt(sqSumG);
stdBlue = sqrt(sqSumB);
stderror = [stdRed, stdGreen, stdBlue];

l = 1:3;
figure
bar(l,pSabund)

hold on

er = errorbar(l,pSave,stderror);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off
%Error bars are small because there is almost no variation.


pAbund =

  Columns 1 through 7

  629.3496  629.6052  629.3591  629.6172  629.6139  629.6058  629.6145
  769.7752  769.8632  769.7785  769.8674  769.8662  769.8635  769.8665
    0.8752    0.5315    0.8624    0.5154    0.5200    0.5307    0.5190

  Columns 8 through 10

  629.6163  629.6145  629.6173
  769.8671  769.8664  769.8674
    0.5166    0.5191    0.5152
```



Published with MATLAB® R2020a