Contents


```
% Ivan Chowdhury
% Stochastics
% Spring 2019
% Dungeons and Dragons Simulations
```

Problem 1

```
clc, clear;
numIter = 100000;
AbilityScore = zeros(1,numIter);

for i = 1:numIter
    roll3d6 = randi(6,3,1); % Roll a 6 sided dice, 3 times. Record results in column vector
    AbilityScore(i) = sum(roll3d6);  % Sum up rolls to get 1 ability score
end

maxabilitycount = nnz(AbilityScore == 18); % Count number of times an ability score of 18 occ
urs
P_maxroll = maxabilitycount/numIter
```

```
P_maxroll =

    0.0050
```

```
clc, clear;
numIter = 100000;
funAbilityScore = zeros(1, numIter);

for i = 1:numIter
    funRoll3d6 = sum(randi(6,3,3));
    funAbilityScore(i) = max(funRoll3d6);
end

maxAbilityCount = nnz(funAbilityScore == 18); % Count number of perfect ability scores in sim
ulation.
P_funmaxroll = maxAbilityCount/numIter
```

```
P_funmaxroll =

    0.0132
```

Very high number of iterations needed to generate a perfect char

```
clc, clear;
numIter = 100000;
charStats = zeros(numIter,6);
FontaineCtr = 0;

for i = 1:numIter
    for j = 1:6
        funRoll3d6 = sum(randi(6,3));
        charStats(i,j) = max(funRoll3d6); % Assign 6 ability scores per character (row)
    end

    numStat9(i) = nnz(charStats(i,:)==18); % Counts how many of each character's stats is equ
al to 18
    if numStat9(i) == 6
        FontaineCtr = FontaineCtr + 1;
    end
end

P_perfectChar = FontaineCtr/numIter
```

```
P_perfectChar =

     0
```

Very high number of iterations needed to generate a character with all 9's

```
clc, clear;
numIter = 100000;
charStats = zeros(numIter,6);
KeeneCtr = 0;

for i = 1:numIter
    for j = 1:6
        funRoll3d6 = sum(randi(6,3));
        charStats(i,j) = max(funRoll3d6); % Assign 6 ability scores per character (row)
    end

    numStat9(i) = nnz(charStats(i,:)==9); % Counts how many of each character's stats is equa
l to 9
    if numStat9(i) == 6
        KeeneCtr = KeeneCtr + 1;
    end
end

P_AverageChar = KeeneCtr/numIter
```

```
P_AverageChar =
```

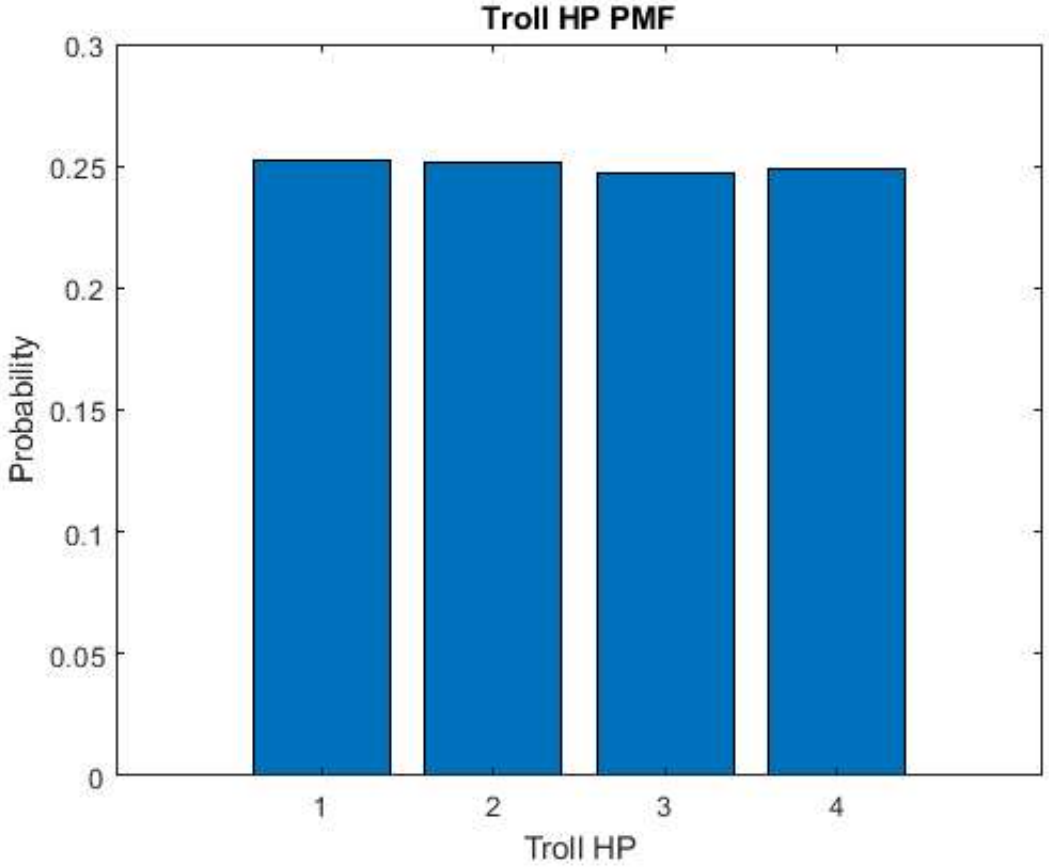
```
     0
```

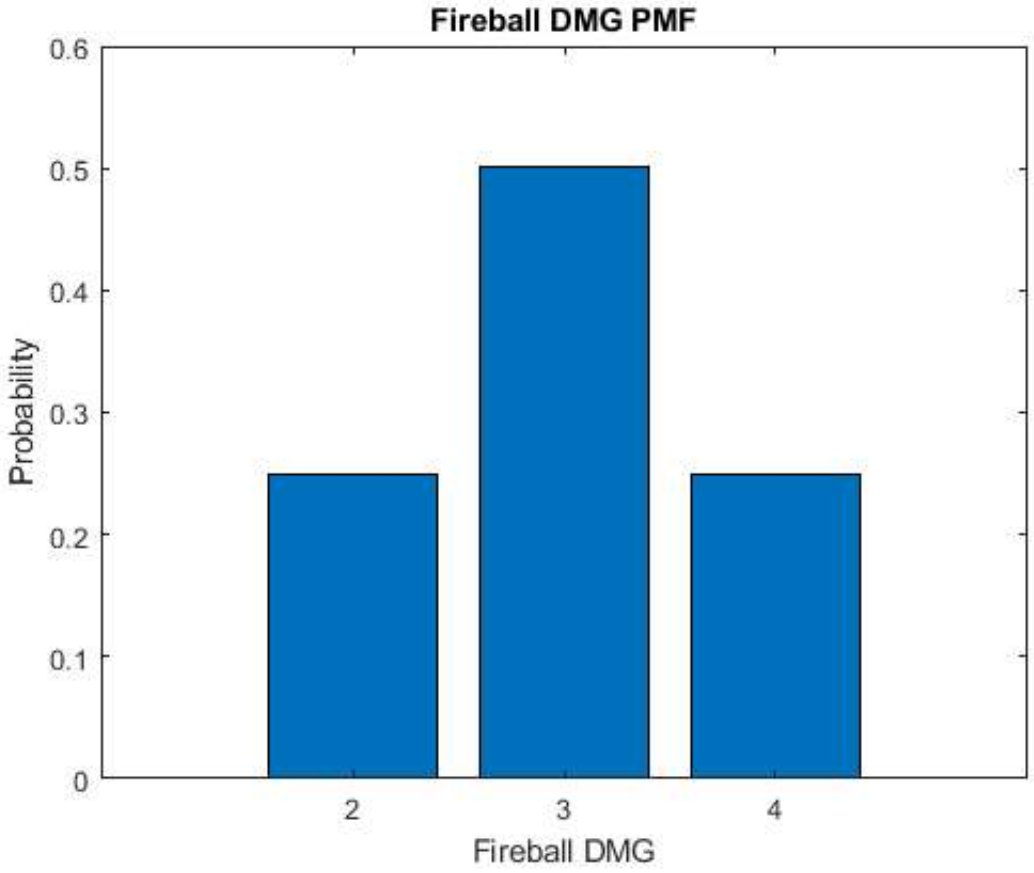
Problem 2

```
clc, clear;
numIter = 100000;

trollHP = zeros(1,numIter);
FIREBALL = zeros(1,numIter);

for i = 1:numIter
    trollHP(i) = randi(4,1,1); % Roll a 4 sided dice, 1 time
    FIREBALL(i) = sum(randi(2,2,1)); % Roll a 2-sided dice, twice, and sum results.
end

% Averages
avgHP = mean(trollHP)
avgFIREBALLdmg = mean(FIREBALL)

% Calculate PMF
P_trollHP1 = nnz(trollHP == 1)/numIter % P(Troll HP = 1)
P_trollHP2 = nnz(trollHP == 2)/numIter % P(Troll HP = 2)
P_trollHP3 = nnz(trollHP == 3)/numIter % P(Troll HP = 3)
P_trollHP4 = nnz(trollHP == 4)/numIter % P(Troll HP = 4)

P_fireball2 = nnz(FIREBALL == 2)/numIter % P(FIREBALL DMG = 2)
P_fireball3 = nnz(FIREBALL == 3)/numIter % P(FIREBALL DMG = 3)
P_fireball4 = nnz(FIREBALL == 4)/numIter % P(FIREBALL DMG = 4)

HP = [1 2 3 4];
FireBallDMG = [2 3 4];
P_TrollHP = [P_trollHP1 P_trollHP2 P_trollHP3 P_trollHP4];
P_fireball = [P_fireball2 P_fireball3 P_fireball4];

figure;
bar(HP,P_TrollHP);
title("Troll HP PMF");
xlabel("Troll HP");
ylabel("Probability");

figure;
bar(FireBallDMG,P_fireball);
title("Fireball DMG PMF");
xlabel("Fireball DMG");
ylabel("Probability");
```

```
avgHP =

    2.4927


avgFIREBALLdmg =
```

```
    2.9991

P_trollHP1 =

    0.2524

P_trollHP2 =

    0.2516

P_trollHP3 =

    0.2469

P_trollHP4 =

    0.2491

P_fireball2 =

    0.2494

P_fireball3 =

    0.5020

P_fireball4 =

    0.2486
```


Troll HP PMF
Probability
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
Troll HP


Fireball DMG PMF
Probability
0.6
0.5
0.4
0.3
0.2
0.1
0
2
3
4
Fireball DMG

```
clc, clear;
```

```
numIter = 100000;
numTrolls = 6;

trollPack = zeros(1,numTrolls);
remainingHP = zeros(numIter,numTrolls);
numAlive = zeros(numIter,1);

for i = 1:numIter
    for j = 1:6
        trollPack(j) = randi(4,1,1);    % Randomly generate troll pack
    end

    FIREBALL = sum(randi(2,2,1)); % Randomly generate fireball

    remainingHP(i,:) = trollPack - FIREBALL; % Find remaining troll HP
    numAlive(i,1) = nnz(remainingHP(i,:) > 0); % Counts how many trolls from each pack is sti
ll alive
end

numDeadPacks = nnz(numAlive == 0); % Count how many troll packs were fully killed
P_slayall = numDeadPacks/numIter % P(all trolls slayed)
```

```
P_slayall =

    0.3414
```

```
clc, clear;
numIter = 100000;
numTrolls = 6;

trollPack = zeros(1,numTrolls);
remainingHP = zeros(numIter,numTrolls);
numAlive = zeros(numIter,1);
survivorHP = zeros(1,numIter);

for i = 1:numIter
    for j = 1:6
        trollPack(j) = randi(4,1,1);    % Randomly generate troll pack
    end

    FIREBALL = sum(randi(2,2,1)); % Randomly generate fireball

    remainingHP(i,:) = trollPack - FIREBALL; % Find remaining troll HP
    numAlive(i) = nnz(remainingHP(i,:) >= 1); % Counts how many trolls from each pack is stil
l alive
    if (numAlive(i) == 1)
        survivorIndex = find(remainingHP(i,:) >= 1); % Find HP of survivor troll
        survivorHP(i) = remainingHP(i,survivorIndex); % skipped indices in matrix will have z
ero values, these will be ignored
    end
end

expectedSurvivorHP = mean(nonzeros(survivorHP))
```

```
expectedSurvivorHP =

    1.0607
```

```
clc, clear;
numIter = 100000;
totalDMG = zeros(1, numIter);

for i = 1:numIter
    attackRoll = randi(20);
    if (attackRoll >= 11)
       SwordDMG = sum(randi(6,1,2));
       totalDMG(i) = SwordDMG;

       attackRoll2 = randi(20);
       if (attackRoll2 >= 11)
          HammerDMG = randi(4);
          totalDMG(i) = SwordDMG + HammerDMG;
       end
    end
end

expectedDMG = mean(totalDMG)
```

```
expectedDMG =

    4.1204
```

Published with MATLAB® R2017b