Assignment 2

STAT 497-H | Reinforcement Learning

Matteo Esposito (40024121), William Ngo (40031586), Spyros Orfanos (40032280), Frederic Siino (40028348)

Concordia University

March 15th, 2019

Question 1

a) See `main.R` code.

b) See `main.R` code.

For parts (a) and (b), our state sweeping procedure is the following:

- Generate matrix SASRp for the specified policy π.
- Append column of state-values calculated using the Bellman equation for $v_\pi(s')$ to the matrix above.
- For the terminal states (initial states in the context of our calculation), i.e. states of the form (0,y,m) and (x,0,m) $\forall x \in [0, 30], y \in [0, 100], m \in [0, 3]$ we must initialize $v_\pi(s) = 0$.
- Our sweeping process will transition from state (0,0,0) $\longrightarrow$ (30,100,0) $\longrightarrow$ (0,0,1) $\longrightarrow$ (30,100,1) $\longrightarrow \ldots \longrightarrow$ (30,100,3). This transition is described below.

Bowser's HP and the number of remaining mushrooms are non-decreasing, thus, given an action a, the current state will transition into a state where Bowser's HP and the number of mushrooms are known. Hence, the random variable is Mario's HP, which we know in the next state will be of the form "Mario's current HP - Bowser's damage (+ health gained if Mario eats a mushroom)".

Define Sk to be the subset of all states that have the form (x,y,k) for k in 0,1,2,3. We see that S3 -> S2 -> S1 -> S0 when mushroom eaten. Otherwise, Sk -> Sk', i.e transition to a state in the same subset. This means Bowser's HP will decrease by 5, and Mario HP will decrease by i in 0:10.

So we can solve the Bellman Equation of V(s) as follows: Any state (0, j , k) or (I, 0, k) has value 0. How did we get to one of these states? Consider the state (1,0,0). So previous states of the form (1+i1,5,0), i1 in 0,. . . ,10. So we can solve the value function for these states exactly (and immediately). How did we get to one of these states? We fix i1, and so must have previously been in state (1 + i1 + i2, 10, 0), i2 in 0,. . . ,10, which we can solve the value function exactly. We repeat this for all remaining states in S0 (eg: next consider 2,0,0), and thus we have calculated V(s) for any s in S0.

Now, we sweep all the states in S1. Once in S1, we can either: (1) Win the game while staying in S1 (2) Lose the game while staying in S1 (3) Transition to some state in S0. In case 1 and 2, we take the approach to above (i.e. which state did we come from etc.). In case 3, we know all v(s) for states in S0 already, so we can again calculate it exactly. We repeat this for all remaining states in S1 , and thus we have calculated V(s) for any s in S1. We apply the same procedure for states S2 and S3 to find v(s) for all s in S.

c) For parts (1) and (2) we extract entry `(31,101,4)` of the `PolicyStateValueFunction` and `StateValueFunction` (optimal state-values) array respectively, whereas for part (3) we extract entry `(13,23,2)` from `OptimalActionMat`. For part (4) we take a sum of the product of the set of state-values of Bowser attacks from the `StateValueFunction` array and their probabilities.

(1) 0.4695039

(2) 0.5585385

(3) Action 0 (attack Bowser)

(4) 0.7966939

Question 2

a) See `main.R` code.

b) Output of `CalculatePolicyValueFunction`

n_{policy}	State 1 Value	State 2 Value
1	14.37500	12.81250
2	12.17391	10.00000
3	14.70588	13.23529
4	10.00000	10.00000
5	10.00000	10.00000
6	10.00000	9.00000

Therefore, clearly policy 3 is the optimal policy as it yields the gratest state-value average when considering states 1 and 2.

c) Calling `PolicyEvaluation(c(3,2), c(1,1), SASRp, 0.9, 1)`, we get the following estimate:

$$\begin{bmatrix} V_0(1) = 3 & V_0(2) = 2 \end{bmatrix} = \begin{bmatrix} 4.34 & 2.98 \end{bmatrix}$$

d) Calling `PolicyImprovement(c(3,2), SASRp, 0.9)`, we get the following policies:

$$\begin{bmatrix} \pi(1) & \pi(2) \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

e) After 20 generalized policy improvement (GPI) cycles, our final estimate of the optimal policy and optimal value function is:

Value Function Estimates	Optimal Policy Estimates
14.70588, 13.23529	1, 3

f) Solution:

$n_{iterations}$	Optimal Value Function (State 1, State 2)	Optimal Policy (State 1, State 2)
2	3.44, 2.08	1, 1
5	6.521606, 5.048554	1, 3
25	13.71078, 12.24019	1, 3
500	14.70588, 13.23529	1, 3

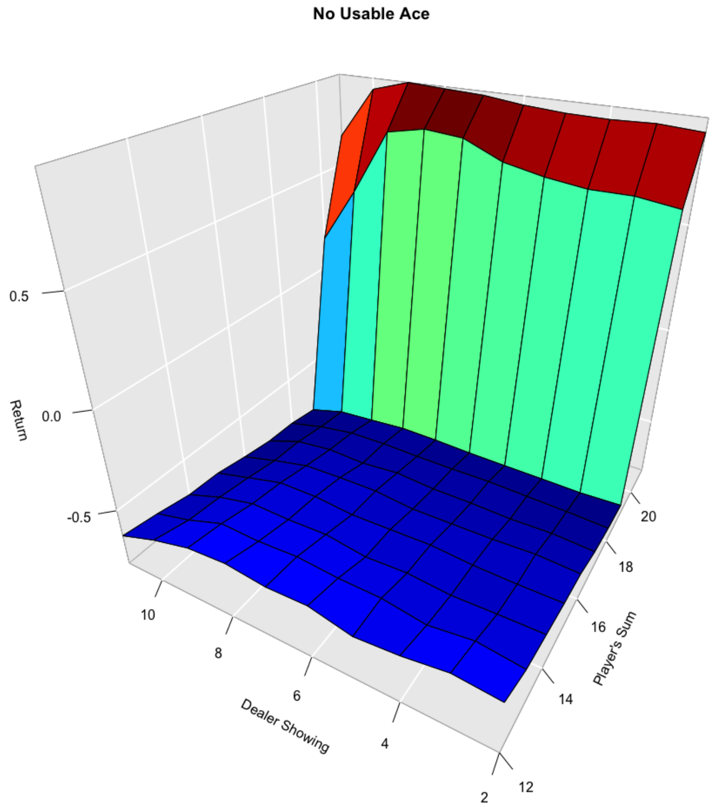
Question 3

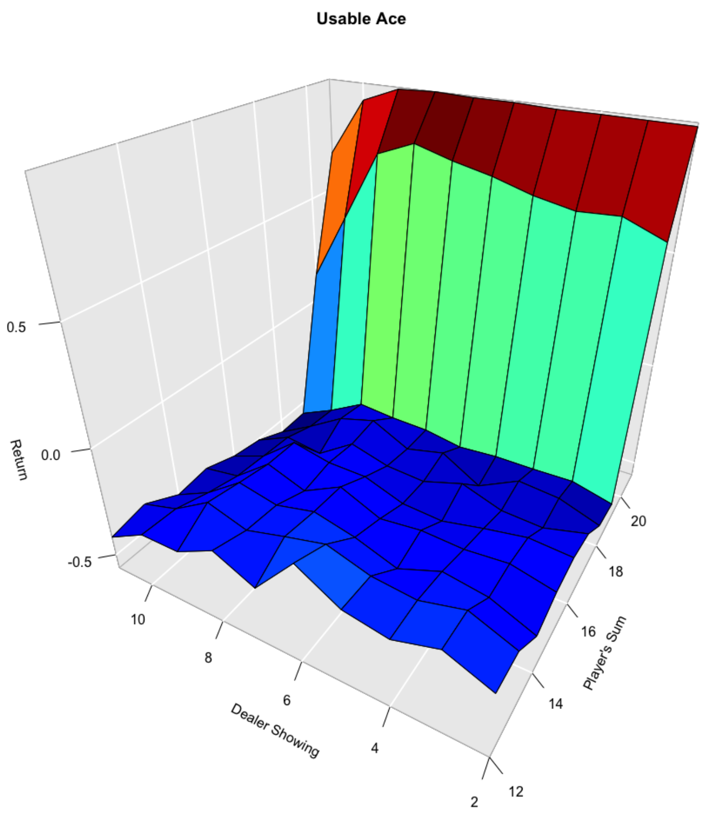
a) See `main.R` code.

b) Solutions:

Question	Function Called	Output
1	`ValueFunctionEstim[7-1,14-11,0+1]`	-0.5828246
2	`ValueFunctionEstim[4-1,16-11,1+1]`	-0.362916
3	`ValueFunctionEstim[4-1,20-11,0+1]`	0.6448043
4	`Nsamplebystate[4-1,18-11,0+1]`	5413

c) Graphical representation of value function:




Usable Ace
Return
0.5
0.0
-0.5
Dealer Showing
10
8
6
4
2
Player's Sum
12
14
16
18
20