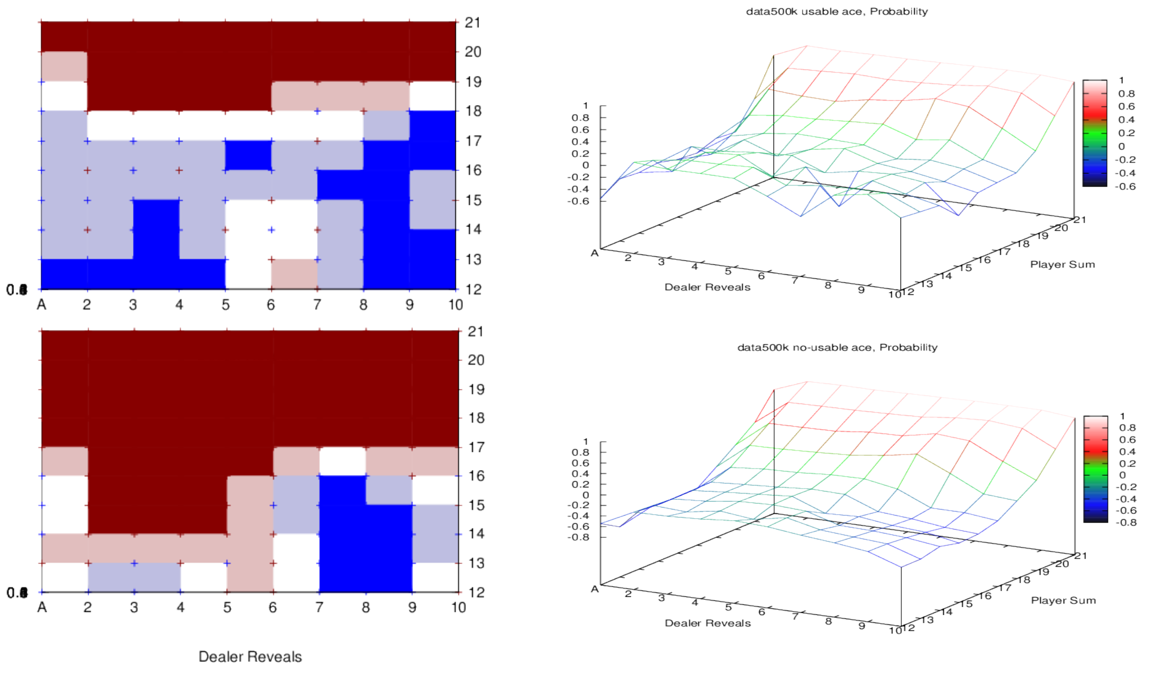
The program aimed to replicate the learning that was done by Sutton and Barto in "Reinforcement Learning" Example 5.3. Basic rules of blackjack were set up, an additional rule that was added was while the player's sum was less than 12 (with no ace present), the dealer would automatically hit the player as we recognize this as the only logical move to make. In addition, the only actions that were allowed were HIT and STAY. The only variable that the agent was allowed to make its decision based on was the player's sum, whether it had a usable ace, and the revealed card of the dealer.

Monte Carlo-ES off policy methods with 0 discounting was implemented as the learning of choice.

The initial policy given was to STAY if the player sum was 20 or 21, otherwise HIT.

In the primary run, it was discovered that when the Qval of the action was set to 0 as the starting value, the final policy, even after 500k iterations would be very different than the policy that was discovered by Sutton and Barto. This was particularly interesting because the graph of the Value function seemed almost identical in both cases.



The graphs show the results after 500k iterations. Left: Policy settled Red Points denote the action STAY, where blue denotes HIT. Right: Graph of Value function of the policy. Top: Usable Ace is Present. Bottom: No Usable Ace. The Value functions are almost identical to the graphs obtained in Reinforcement Learning (page 121). Other graphs can be found in the results/graphs directory.

However, when the Qval was initialized to 2, the policy seemed to converge to something very similar to the found policy, which included the behavior to favor hitting when the Dealer revealed an Ace with No Usable Ace present. The final policy after 500k iterations differed from the found policy by 12 points on the Usable Ace and 9 points with no Usable Ace. I speculate this may be caused by a combination of the agent picking the wrong action early on and winning by chance (probably due to the dealer busting) and losing when choosing the correct. However, I am skeptical of this, as the probability of winning by chance should be outweighed by time.

=> Some additional details from the -visit.dat file (which displays number of times you visited each action on the Usable Ace policy)

For 500k iterations

```
-Dealer has: 6, Player: 12
        -HIT - Visted: 3 , Qval = -0.33
        -STAY - Visited: 191, Qval = -0.13
-Dealer has: 6, Player: 13
        -HIT - Visted: 7 , Qval = -0.14
        -STAY - Visited: 451, Qval = -0.08
```

Now for these actions, we can see that even though the HIT action is appropriate, the agent simply does not visit the action enough times the gather a strong enough conclusion about it and the few times the STAY state was visited, it did not do as bad as it. We also notice that the more frequently visited states are much better gotten the policy for (i.e. player sum = 21, these are all visited greater than 2000 times each)

Following this approach, I am not fully convinced of the MC-ES approach, primarily due to the failure of the policy converging. (The policy occasionally varies significantly on some runs of the program, even for high number of iterations)

A possible better approach to this problem might be to have the agent learn for what sum it should hit, given if it has a Usable Ace and the card revealed by the dealer. This is since in blackjack if you were to hit with a higher value, then you are obviously going to hit for the lower value.