Statistical Decision Theory By Example

Adam Fleischhacker

June 25, 2019

We now continue our search for Robo. However, to simplify the problem, we assume Robo is hidden, and will remain so without moving until found, in one of seven caves as shown in Figure @ref(fig:sevenCaves).

We can encode our decision scenario, call it D, which is a set of variables defined by four elements:

1. **Outcomes**, X, representing all potential future scenarios. If there are N possibilities, we might label outcomes $X = \{x_1, x_2, \ldots, x_N\}$
2. **Actions**, A, that can be taken. If there are K potential actions, then we can label them $A = \{a_1, a_2, \ldots, a_K\}$.
3. **Probabilistic outcome model**, P, that for any action A provides a probability distribution $p(X|A)$.
4. **Utility function** (or loss function), $U(A \times X)$, which can take any particular action a and a subsesequent discovery of particular outcome x and map the combination, (a, x), to a value (e.g. profit, revenue, etc.).

Outcomes

Some of the variables are *random variables* and these are used to represent a decision maker's uncertainty. This uncertainty is often called a *state of nature* and the random variables might also be called *chance variables*. Mathematically, we define our random variable:

$$X \equiv \text{Cave number of Robo's location}$$

and we assign its *support* - the set of all possible realizations (i.e. values) that have non-zero probability - aas follows:

$$X \in \{1, 2, 3, 4, 5, 6, 7\}$$

with each value corresponding to a potential cave that Robo is hiding in.

Probability Model for Outcomes

We will use p_i to indicate the probability with which we think Robo is hiding in Cave i. Hence, $\sum_i p_i = 1$. Distributions such as this, where potential outcomes are limited to one of a countable number of possible outcomes and each outcome has a specified probability, occur so frequently that this class of distributions is given a name. Namely, X is described by a *categorical distribution*.

Within R, we can represent a categorical distribution by two vectors:

```
## initial probability distribution
possibleOutcomes = c(1,2,3,4,5,6,7) ##potential locations for Robo
probabilities = c(1/32, 2/32, 8/32, 7/32, 6/32, 3/32, 5/32) #arbitrary values
sum(probabilities) ##check they sum to 1
```

```
## [1] 1
```

```
##make dataframe for plotting
plotDF = data.frame(possibleOutcomes,probabilities)

##plot distribution
theme_set(theme_bw(18)) ##create a default theme for ggplot graphs
```

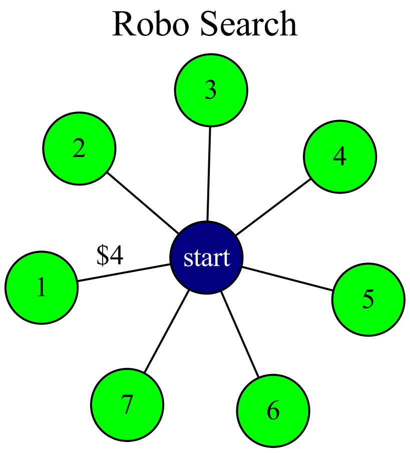


Figure 1: The search for Robo.

```r
ggplot(data = plotDF, aes(x=possibleOutcomes, y=probabilities)) +
  geom_col(fill = "DarkGreen") +
  scale_x_continuous(breaks = possibleOutcomes)
```

```r
##save the plot
probPlot = last_plot()
```

and as visualized in Figure @ref(fig:initProbModel). From this plot, we can easily see that Robo is most likely in *Cave 3*, but may also be in any of the other six caves. It is worth noting that in more general scenarios, the probabilistic outcome model can be dependent on action A. Fortunately, our choice of where to search does not change the likelihood of where robo is. Hence, $P(X|A) = P(X)$ due to independece between action and outcome.

Actions

For simplicity, let's assume we can only search one cave, say due to time constaints. If we find Robo, then we save his $20 replacement cost. However, to search each cave, it costs us $4 in time and equipment. Thus, we define the potential courses of action, and use A to represent these options where:

$$A \in \{1, 2, 3, 4, 5, 6, 7\}$$

and the i^{th} element, a_i, corresponds to the decision to search cave i.

Utility

$U(x, a)$, takes the joint assignment of both X and A and maps that assignment to a real number. In our Robo example, we will use monetary gain as our measure of utility and the larger the gain, the better. Assuming $A = 1$, the mapping from $A \times X \to U(a = 1, x)$ can be accomplished as below:

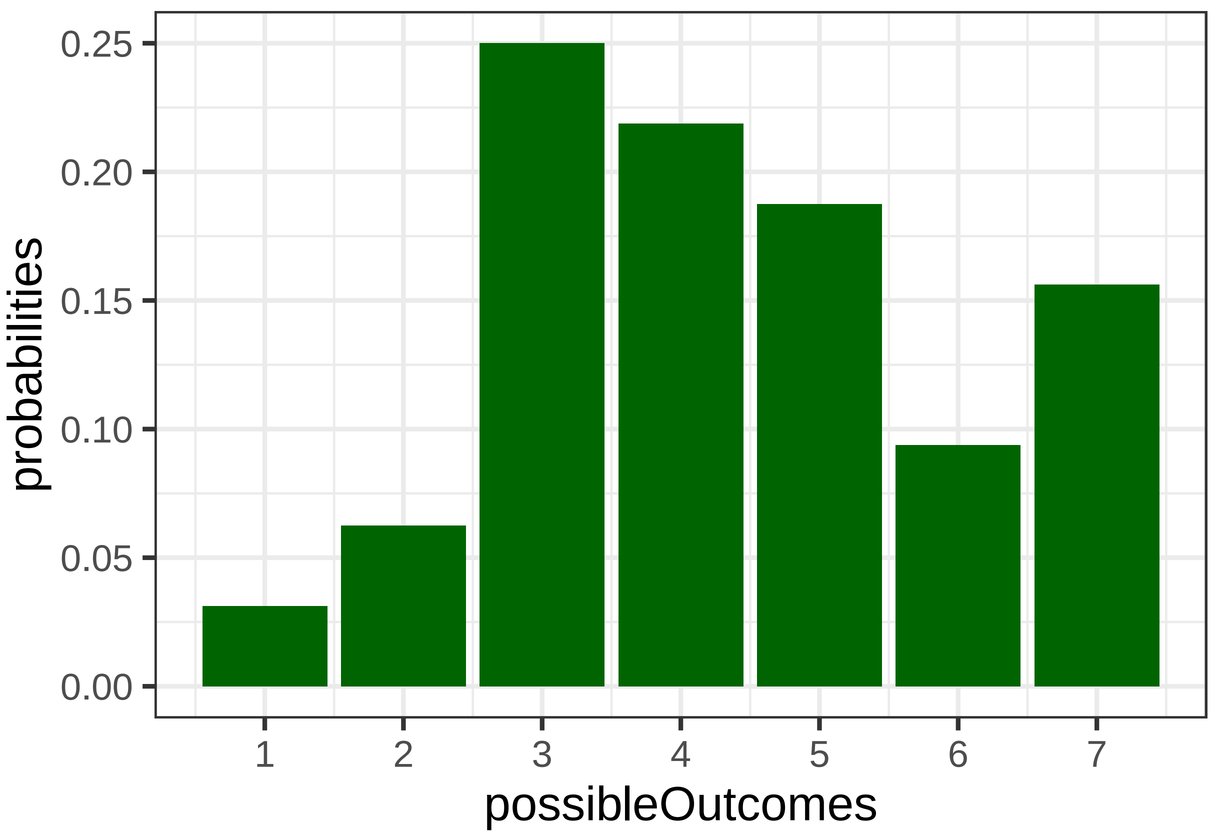


Figure 2: The probability of finding Robo in each cave.

```r
utilityAction1 = c(16, rep(-4,6))
UtilityAction1 = utilityAction1 * probabilities
UtilityAction1 ##shows contribution of each outcome to the expected utility
```

```
## [1]  0.500 -0.250 -1.000 -0.875 -0.750 -0.375 -0.625
```

```r
expUtilityAction1 = sum(UtilityAction1)
expUtilityAction1 ## summarizes all possible outcomes into an expected utility
```

```
## [1] -3.375
```

Interpreting the above expectation requires us to imagine that we can repeat this search for Robo over and over again; then, on average we expect our gain to be -3.375. In reality, our gain is always either $16 or -$4, but in expectation the average search yields a loss of -3.375. This is so close to -$4 because Robo is so unlikely to be in Cave 1. In fact, if faced with a choice of searching $Cave1$ or not searching at all, you are better off not searching which has an expected utility of $0.

To look at the expected utility for all of the actions, we can create functions in R to automate all of the calculations. The first function is used to more generally create the mapping from $A \times X \rightarrow U(a,x)$:

```r
utility = function(a,x) {
  ifelse(a == x, 20-4, -4)
  }
```

Then, create a probability function which given a value of x returns the associated probability:

```r
prob = function(x) {
  probabilities[x] ##extract the x element of probabilities
  }
```

Lastly create an expected utility function:

```r
expUtility = function(a) {
  sum(utility(a,possibleOutcomes) * prob(possibleOutcomes))
  }
```

From Problem Representation to Decision

Let's now assume there are tunnels connecting all of the adjacent cave and thus, for an additional $1 per adjacent cave, that cave can be searched. For example, the search costs to search caves 1 and 7 would be $5 and to search all seven caves would be $11. In this case, what would the optimal policy be? how many caves would be searched when using the optimal policy? (assume Robo will not be broken and his value when found is $20.

Let's assume, without rigorous proof of optimality, that the optimal ("best") policy of searching the seven caves satisfies the following criteria: 1) the policy will continue the search until Robo is found, 2) after a cave is searched and if Robo is not found, then the search party will move to an adjacent unsearched cave, and 3) the search stops after Robo is found to avoid the additional search costs.

With a little bit of thinking, you should now convince yourself that there are 14 possible search policies. You can start your search at any one of the seven caves and then continue your search in either a clockwise direction or a counter-clockwise direction. The fourteen policies are enumerated below:

With all four elements of the decision problem specified, we can then create a data frame representing action and expected utility:

```r
## Expected Utility by Possible Action
utilityDF = data.frame(action = 1:7) %>% tbl_df() %>% rowwise() %>%
  mutate(expUtil = expUtility(a = action))
utilityDF
```

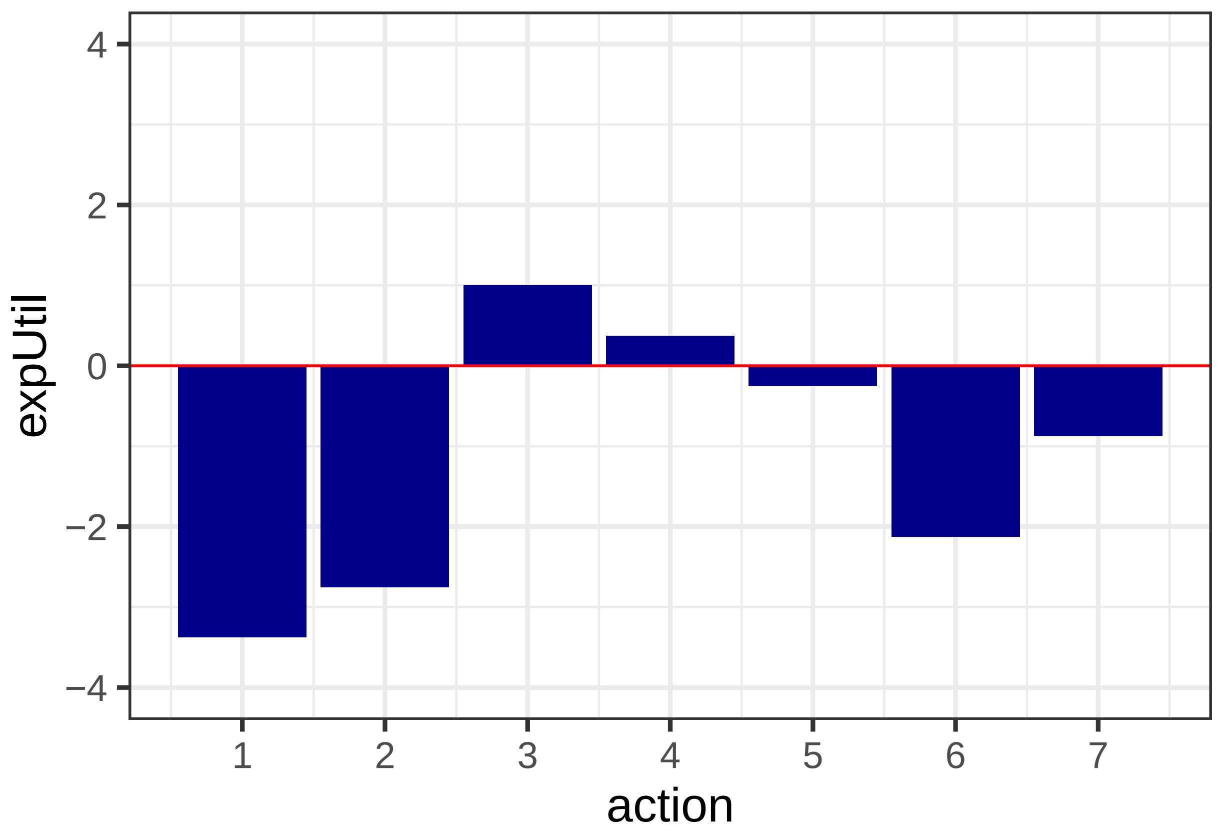


Figure 3: Expected utility from searching each cave.

```
## Source: local data frame [7 x 2]
## Groups: <by row>
##
## # A tibble: 7 x 2
##   action expUtil
##    <int>   <dbl>
## 1      1  -3.38
## 2      2  -2.75
## 3      3   1
## 4      4   0.375
## 5      5  -0.25
## 6      6  -2.12
## 7      7  -0.875
```

```
##plot the results
ggplot(utilityDF, aes(x = action, y =expUtil)) +
  geom_col(fill = "DarkBlue") +
  scale_x_continuous(breaks = possibleOutcomes) +
  scale_y_continuous(limits = c(-4,4)) +
  geom_hline(yintercept = 0, color = "Red")
```

Figure @ref(fig:expUtil) shows

, the only positive expected utilities are derived from searching either $Cave3$ or $Cave4$. Based on the principle

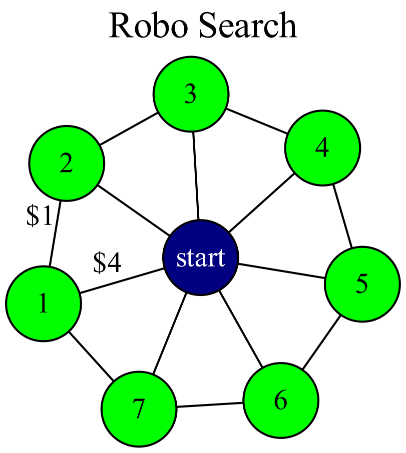


Figure 4: The search for Robo.

of maximum expected utility, we would therefore search $Cave3$ which has an expected utility of $1. This makes sense as there is a $\frac{8}{32} = 25\%$ chance of finding Robo in $Cave3$ ($16 gain) and 75% chance of not finding him (-$4 loss).

Do the below exercises in a separate R-script so that our line #'s stay synchronized:

> **CLASS EXERCISE:** What is the expected utility (i.e. monetary gain) of searching $Cave2$?
>
> **CLASS EXERCISE:** Suppose there is a 50% chance of Robo being broken when he is found. If he is broken, you can at least reuse some of his parts and you estimate his value when broken to be $5. What is the expected utility (i.e. monetary gain) of searching $Cave4$ in this case (round to three decimal places)?

A More Complicated Action Policy

Let's now assume there are tunnels connecting all of the adjacent caves as shown in Figure @ref(fig:sevenCaves2). Thus, for an additional $1 per adjacent cave, that cave can be searched. For example, the search costs to search caves 1 and 7 would be $5 and to search all seven caves would be $11. In this case, what would the optimal policy be? how many caves would be searched when using the optimal policy? (assume Robo will not be broken and his value when found is $20.

Let's also assume, without rigorous proof of optimality, that the optimal ("best") policy of searching the seven caves satisfies the following criteria:

1. The policy will continue the search until Robo is found,
2. after a cave is searched and if Robo is not found, then the search party will move to an adjacent unsearched cave, and
3. the search stops after Robo is found to avoid the additional search costs.

With a little bit of thinking, you should now convince yourself that there are 14 possible search policies. You can start your search at any one of the seven caves and then continue your search in either a clockwise direction or a counter-clockwise direction. The fourteen policies are enumerated below:

```
##use expand.grid to create a dataframe which
##enumerates all possible combinations of supplied vectors
actionDF = cbind(paste0("a",1:14),expand.grid(start = 1:7, direction = c("clockwise","counterClock")))
names(actionDF)[1] = "action"
actionDF
```

```
##    action start    direction
## 1      a1     1    clockwise
## 2      a2     2    clockwise
## 3      a3     3    clockwise
## 4      a4     4    clockwise
## 5      a5     5    clockwise
## 6      a6     6    clockwise
## 7      a7     7    clockwise
## 8      a8     1 counterClock
## 9      a9     2 counterClock
## 10    a10     3 counterClock
## 11    a11     4 counterClock
## 12    a12     5 counterClock
## 13    a13     6 counterClock
## 14    a14     7 counterClock
```

The above listing of search policies addresses one of the four critical elements in a decision scenario, namely a set of actions. The four elements applied to this updated space of potential search policies are discussed below:

1. Outcomes: X still represents Robo's location.

2. Actions: A can now be one of the 14 potential actions listed above, $A = \{a_1, a_2, \ldots, a_{14}\}$.
3. Probabilistic outcome model: $p(X|A) = p(x)$ represents the probability of finding Robo in any given location and remains equal to $\{\frac{1}{32}, \frac{2}{32}, \frac{8}{32}, \frac{7}{32}, \frac{6}{32}, \frac{3}{32}, \frac{5}{32}\}$.
4. Utility function: $U(A \times X)$ still represents monetary gain, but now must be updated to reflect the new cost structure of the additional search capability.

The updated utility function is specified below:

```
utility = function(a,x) {
  case_when(
    # Robo found going clockwise BEFORE going from cave #7 to cave#1
    actionDF$start[a] <= x &  actionDF$direction[a] == "clockwise" ~
      20 - 4 - 1*(x-actionDF$start[a]),
    # Robo found going clockwise AFTER going from cave #7 to cave #1
    actionDF$start[a] > x & actionDF$direction[a] == "clockwise" ~
      20 - 4 - 1*((7 - actionDF$start[a]) + x),
    # Robo found going counterclockwise BEFORE going from cave #1 to cave #7
    actionDF$start[a] < x ~
      20 - 4 - 1*(actionDF$start[a] + (7 - x + 1)),
    # Robo found going counterclockwise AFTER going from cave #1 to cave #7
    TRUE ~ 20 - 4 - 1*(actionDF$start[a] - x))  ### actionDF$start[a] >= x
}
## test it
utility(a=5,x=5)  ###start at 5 going clockwise... Robo in 5
```

```
## [1] 16
```

```
utility(a=12,x=5)  ###start at 5 going counter-clockwise... Robo in 5
```

```
## [1] 16
```

```
utility(a=5,x=6)  ###start at 5 going clockwise... Robo in 6
```

```
## [1] 15
```

```
utility(a=12,x=6) ###start at 5 going counter-clockwise... Robo in 6
```

```
## [1] 9
```

```
utility(a=5,x=3)  ###start at 5 going clockwise... Robo in 3
```

```
## [1] 11
```

```
utility(a=12,x=3) ###start at 5 going counter-clockwise... Robo in 3
```

```
## [1] 14
```

Now, if we know both our strategy and Robo's location, we can calculate utility. However, since Robo's location is uncertain, we need to calculate expected utility. Our decision criteria will be to *maximize expected utility* (MEU). Fortunately, the expected utility formula remains unchanged.

$E\left[U(a)\right] = \sum_{x \in X} U(a,x) \times p(x|a)\}$

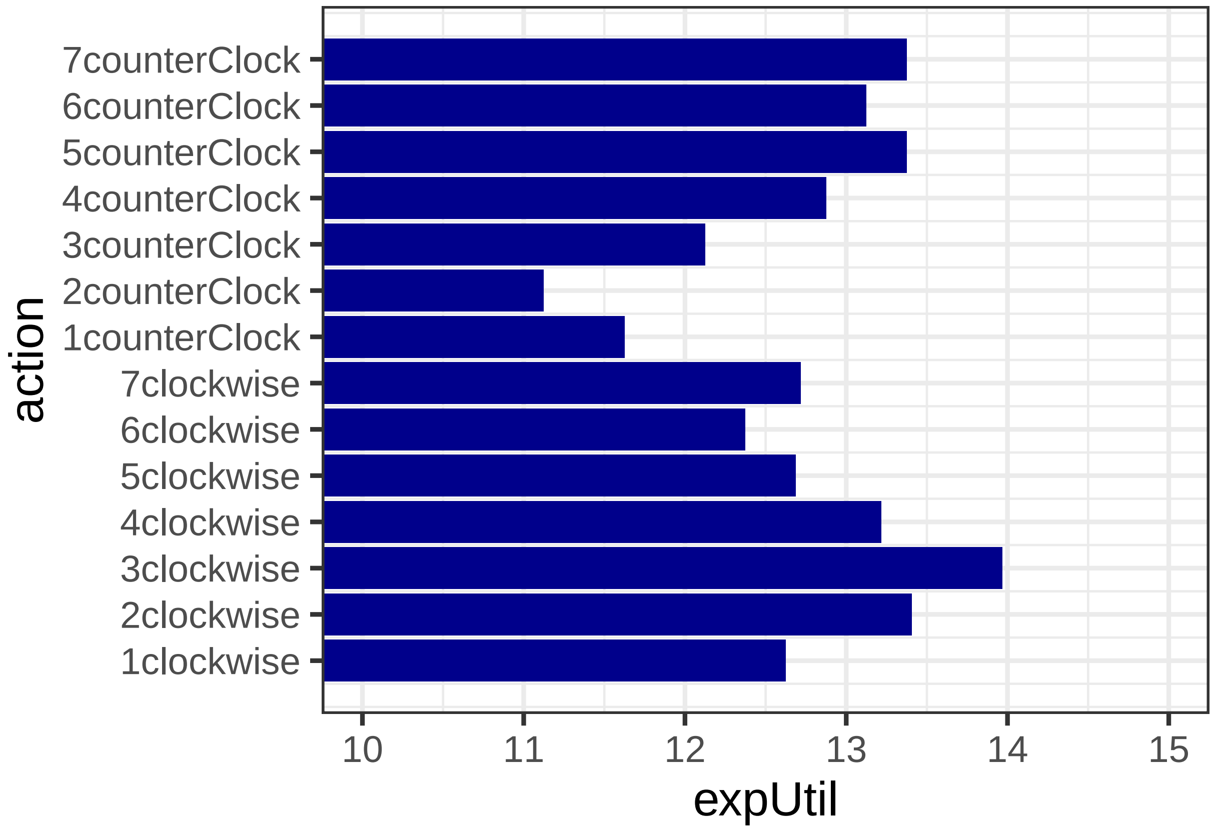
```
expUtility = function(a) {
  sum(utility(a,possibleOutcomes) * prob(possibleOutcomes))
  }
```

We can again create a data frame representing action and expected utility.

```
## Expected Utility by Possible Action Searching All Caves
utilityDF = data.frame(action = 1:14) %>%
  tbl_df() %>% ## make a tibble for printing
  rowwise() %>%  ## make each row a group, so mutate function acts on each row
  mutate(expUtil = expUtility(a = action))   ##get expected utility
utilityDF
```

```
## Source: local data frame [14 x 2]
## Groups: <by row>
##
## # A tibble: 14 x 2
##    action expUtil
##     <int>   <dbl>
##  1      1    12.6
##  2      2    13.4
##  3      3    14.0
##  4      4    13.2
##  5      5    12.7
##  6      6    12.4
##  7      7    12.7
##  8      8    11.6
##  9      9    11.1
## 10     10    12.1
## 11     11    12.9
## 12     12    13.4
## 13     13    13.1
## 14     14    13.4
```

```r
##plot the results
ggplot(utilityDF, aes(x = action, y =expUtil)) +
  geom_col(fill = "DarkBlue") +
  scale_x_continuous(breaks = utilityDF$action,
                     labels = paste0(actionDF$start,actionDF$direction)) +
  coord_flip(ylim = c(10,15))
```



```r
##save the plot
utilPlot = last_plot()
```

Then, we can plot the original prior distribution for Robo's location and the expected utility plot together (see Figure @ref(fig:combinePlots)) in order to make sense of the results.

```r
## analyze results by comparing Strategies and prob distribution of Robo's location
grid.arrange(probPlot,utilPlot)
```

Do the below exercise in a separate R-script so that our line #'s stay synchronized:

> **Class Exercise:** Assume that you only have time to search six of the seven caves. Modify the utility calculation and create a plot as in @ref(fig:combinePlots)?

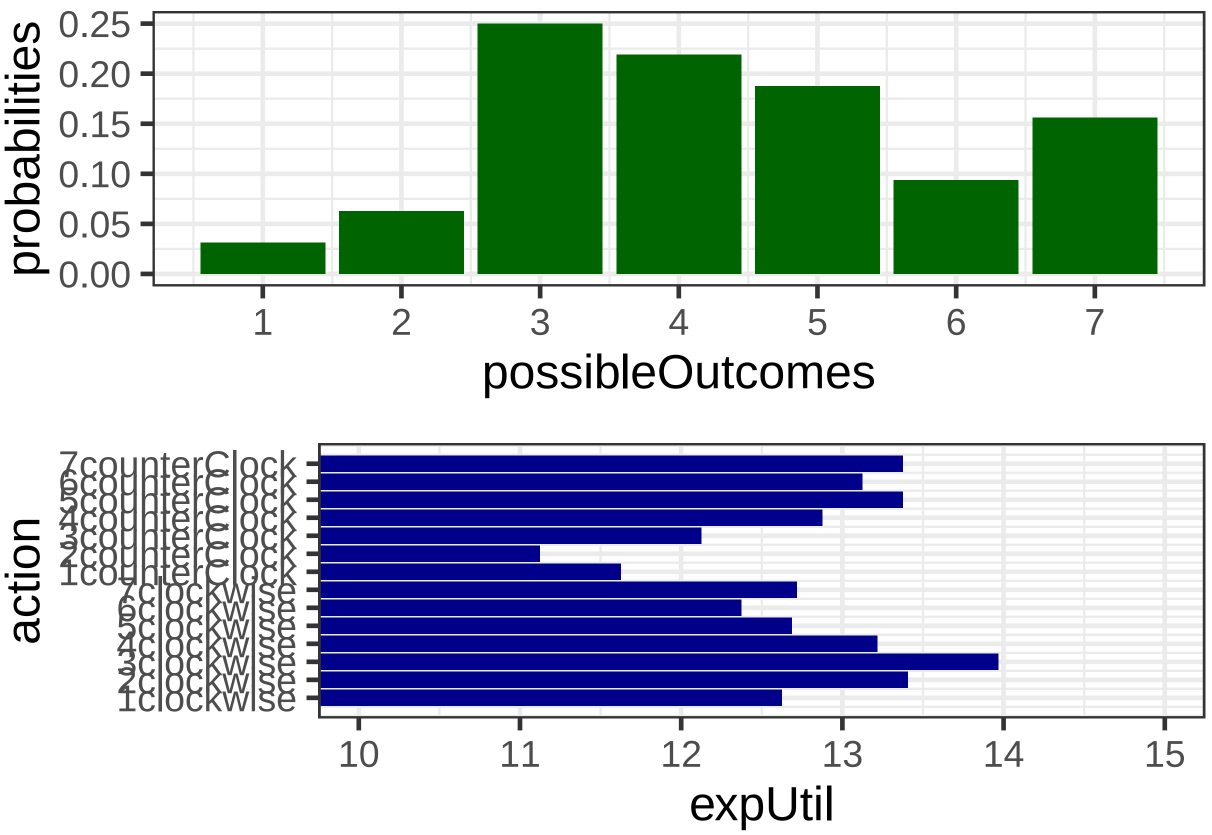


Figure 5: Plots of the potential outcomes and the utility of each possible action plan.