SMART in R

Therese Donovan

2020-04-14

Overview

This script is a translation of the SMART analysis introduced by Tony in class. SMART stands for "Simple Multi-Attribute Rating Technique". It is a very flexible decision approach where the decision problem must consider multiple (and sometimes conflicting) objectives.

The problem introduced involved buying a car for your teenager. Who are the stakeholders for this problem? A stakeholder is a a person, group or organization that has interest or concern in the decision at hand. The stakeholders often define the problem's goals and objectives. As Tony mentions, goals are lofty, but objectives are outcomes, with an emphasis on the word "measureable". For this problem, the objectives are:

1. miminize purchase price
2. maximize safety
3. maximize coolness
4. maximize fuel economy

Let's turn this into a vector in R

```
obj <- c("price", "safety", "efficiency", "coolness")

# set the directionality for each objective
dir <- c("minimize", rep("maximize", times = 3))
names(dir) <- obj
```

Next, we need to assign weights for each objective. Tony said that these range from 0 to 100, where 100 is the objective you value the most. Once the weights are entered, we can standardize them so that their sum is 1 with the `prop.table()` function:

```
# enter raw weights;
weights <- c(50, 100, 100, 25)
names(weights) <- obj

# normalize the weights so their sum is 1
norm.weights <- prop.table(weights)
norm.weights
```

```
##      price     safety efficiency   coolness
## 0.18181818 0.36363636 0.36363636 0.09090909
```

The next step is to compile a list of imaginative alternatives. A good decision analysis is only as good as the options that are considered. The alternatives for this problem include:

1. Honda civic (2000)
2. Mustang (1965)
3. Ford pick up (1992)
4. Ford focus (2004)

Let's turn these into a vector as well:

```
alternatives <- c("Honda", "Mustang", "Pickup", "Focus")
```

The next step is to score each option with respect to each objective. This is where *science* comes in: you need to predict, in some way, how each alternative scores. In the class exercise, Tony gave us these numbers and said they could be looked up in the latest issue of Car Buyers Blue Book. But in reality, these numbers often come from *models* that can be used to predict the *consequence* for each alternative under each objective.

Let's make an R object of the consequence table. Here, we will create a matrix to store our numbers

```
consequence.table <- matrix(data = c(10000, 15000, 5000, 7000,
                                     8, 3.2, 5.1, 7.6,
                                     30, 17, 15, 28,
                                     50, 100, 75, 30),
                            nrow = 4,
                            ncol = 4,
                            byrow = F)

# look at the table
consequence.table
```

```
##       [,1] [,2] [,3] [,4]
## [1,] 10000  8.0   30   50
## [2,] 15000  3.2   17  100
## [3,]  5000  5.1   15   75
## [4,]  7000  7.6   28   30
```

```
# add row names
row.names(consequence.table) <- alternatives

# add column names
colnames(consequence.table) <- obj

# look at the table again
consequence.table
```

```
##         price safety efficiency coolness
## Honda   10000    8.0         30       50
## Mustang 15000    3.2         17      100
## Pickup   5000    5.1         15       75
## Focus    7000    7.6         28       30
```

Next, we need to "standardize" the consequence table so that the worst score for any given objective is assigned a 0, and the best score for that objective is assigned 100. All other alternatives are scored relative to these. These are the "thermometers" that Tony talked about. To begin, we need to find the minimum and maximum score for each objective.

Here, we will use the `apply()` function in R. Apply functions are a broad family of functions in R that allow one to quickly apply some operation over rows or columns. The first argument is a dataframe or matrix to work on. The second argument is called "MARGIN", and here you enter a 1 if you want to apply some operation by row, and 2 if you want to apply the operation over columns. The third argument is called FUN, and here you specify the function that will do the operations.

```
# get the minimum
mins <- apply(consequence.table, MARGIN = 2, FUN = min)
mins
```

```
##      price     safety efficiency   coolness
```

```
##     5000.0        3.2       15.0       30.0
```

```
# get the maximum for each objective
maxs <- apply(consequence.table, MARGIN = 2, FUN = max)
maxs
```

```
##      price     safety efficiency   coolness
##      15000          8         30        100
```

Next, we need to turn these into "thermometers". How we do this depends entirely on whether we wish to "miminize" or "maximize" an objective.

If we wish **maximize** an objective, we can compute these scores as:

$$\frac{value - min}{max - min}$$

And if we wish to **minimize** an objective, this formula will do the trick:

$$1 - \frac{value - min}{max - min}$$

Let's compute these in R using the bracket-subsetting to extract the correct pieces of information:

```
# standard cost: here we wish to normalize for an objective that is minimized
cost <- 1 - (consequence.table[,'price'] - mins[1])/(maxs[1] - mins[1])
cost
```

```
##   Honda Mustang  Pickup   Focus
##     0.5     0.0     1.0     0.8
```

We could copy and paste this style of equation for all columns in our consequence table. But, instead, we'll use a loop to do this. Why? Because when copying and pasting, it is very easy to make a mistake, and a loop is more succinct. It may have many tiny steps, but it will give us more practice with looping and also allow us to look at the `switch()` function:

```
# set up an empty matrix to hold results
ct <- matrix(data = NA, nrow = 4, ncol = 4)

# loop through the four columns
for (i in 1:4) {

  # get the column names (objective)
  column.name <- colnames(consequence.table)[i]

  # extract the data to be normalized
  data <- consequence.table[,column.name]

  # find the direction
  direction <- dir[column.name]

  # get the minimum
  min <- min(data)

  # get the maximum
  max <- max(data)

  # compute the standardization
  values <- switch(direction,
```

```
                    minimize = (1 - (data - min)/(max - min)) * 100,
                    maximize = (data - min)/(max - min) * 100)

  # add the result to our ct table
  ct[,i] <- round(values)
}

# add row names
row.names(ct) <- alternatives

# add column names
colnames(ct) <- obj

# look at the results
ct
```

```
##         price safety efficiency coolness
## Honda      50    100        100       29
## Mustang     0      0         13      100
## Pickup    100     40          0       64
## Focus      80     92         87        0
```

Before we go further, it is important to do two things:

1. Remove any irrelevant objectives. These are objectives that don't really discriminate among the alternatives. For example, if safety was 5 for all alternatives, it won't help us at all to separate the alternatives because there is no variation among them.
2. Remove any dominated alternatives. These are alternatives that always are outperformed by other alternatives – if they are dominated, they really don't belong in the decision analysis.

One quick way to test for irrelant objectives it to use get their variances: if a column doesn't vary, it is irrelevant. Let's test this now:

```
vars <- apply(ct, MARGIN = 2, FUN = var)
vars
```

```
##      price     safety efficiency   coolness
##   1891.667   2202.667   2579.333   1874.917
```

Now, lets figure out which columns have 0 variance, and remove if when true.

```
# find the column indices that have no variance
indices <- which(vars == 0)

# if there are columns with no variance, omit them.
if (length(indices) > 0) {
  ct <- ct[,-indices]
}
```

One quick way to test for dominated alternatives is to rank them. Here, we'll use the `apply()` function to get the ranks for each column. Here, we'll rank the values in each column of our standardized consequence table:

```
ranks <- apply(X = ct, MARGIN = 2, FUN = rank)
ranks
```

```
##         price safety efficiency coolness
## Honda       2      4          4        2
## Mustang     1      1          2        4
```

```
## Pickup      4      2          1        3
## Focus       3      3          3        1
```

You will want to explore how the rank function handles ties though! Next, let's find out if any alternative is ranked last across all four objectives, and remove it if that is the case.

```
# find which rows have a total summed rank of 16
indices <- which(rowSums(ranks) == length(obj) * length(alternatives))

# if there are rows with lowest ranks across all objectives, omit them.
if (length(indices) > 0) {
  ct <- ct[-indices,]
}
```

Note: if your standardized consequence tables shrinks in rows or columns, you'll want to repeat this exercise until no more rows or columns can be eliminated.

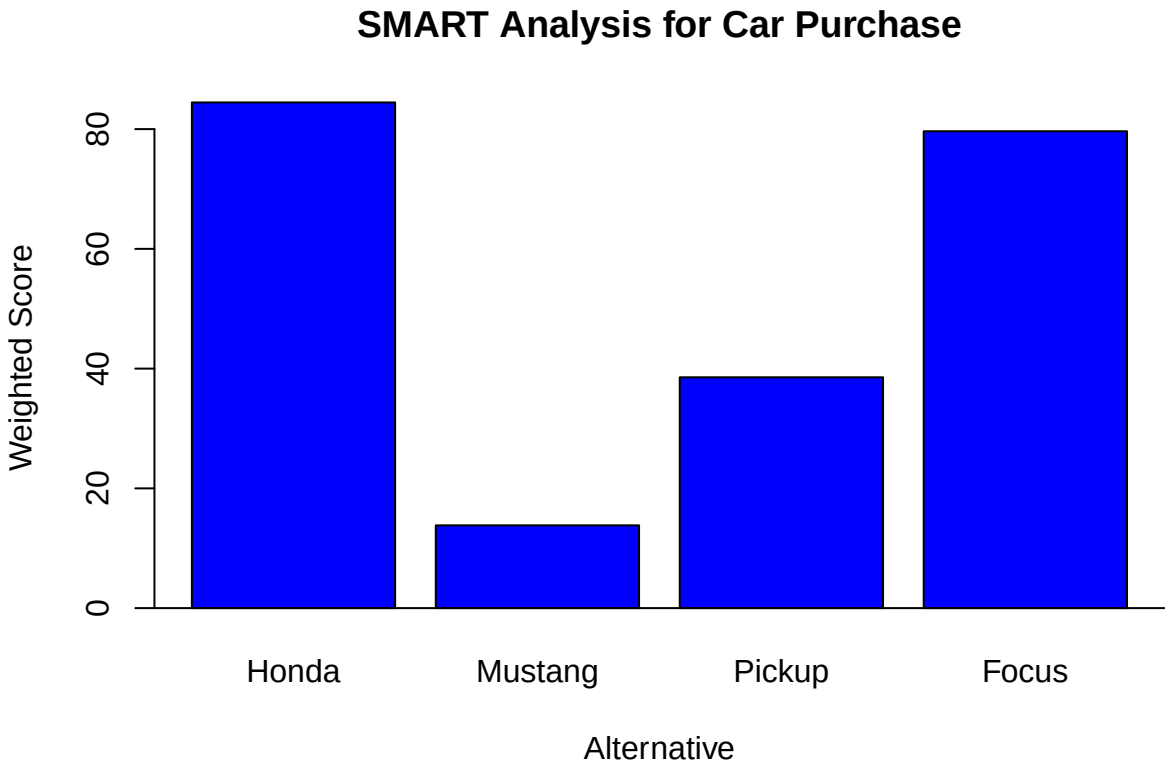
Finally, we can compute the final SMART table by multiplying the standardized table by the objective weights, which if you recall are stored in the vector called "norm.weights". In Excel, we used the SUMPRODUCT function to do this calculation. In R, a similar function is called `crossprod()`. We'll use the `apply()` function again, this time working our way by row instead of column, so that we take each alternative's standardized score and multiply it by the standardized weights to give just one score per alternative

```
scores <- apply(ct, MARGIN = 1, FUN = crossprod, y = norm.weights)
scores
```

```
##    Honda  Mustang   Pickup    Focus
## 84.45455 13.81818 38.54545 79.63636
```

Next, we can plot the results with as a simple barplot:

```
barplot(scores,
        main = "SMART Analysis for Car Purchase",
        xlab = "Alternative",
        ylab = "Weighted Score",
        col = "blue")
abline(h = 0)
```



At this point, you can do a sensitivity analysis by changing the weights. Tony indicated that the consequence table represents factual information, and in some cases that's true. But if the consequence table is filled in by *estimated* values (often estimated with some kind of model), you may very well want to see how robust your results are to changes in the table as well. As Tony emphasized, what you are most interested in knowing is what values would *change* the final rankings, which of course the decision makers are focused on.