tme

Teaching Marketing Engineering with R

Christian Buchta and Stefan Theußl

April 13, 2012

Abstract

Marketing Engineering is important for analysis and decision making in the context of Marketing. The R package~**tme** provides tools for marketing engineering. It's purpose is mainly for teaching.

1 Introduction

This manual is divided into several chapters. In each chapter, one field of marketing engineering is going to be described.

2 Market Response Models

Marketing response models in their simple form try to describe the relationship between advertising effort and sales. There are many different types of response models. They differ not only in their shape but also in other dimensions like linearity vs. nonlinearity, certainty vs. uncertainty, or determinstic vs. stochastic and so forth. In the following section a brief survey of the available models in our package will be presented.

2.1 Theory

Implemented Response Models (see Lilien, Kotler, and Moorthy (1992)):

Linear $Q = a + bX, b > 0$

Power Series $Q = a + bX + cX^2 + \ldots$

2.2 Examples

```
> x <- c(1:1000)
> par(mfrow=c(2,2))
> plot(x,response(x,c(5,0.004),model="exponential"),xlab="advertising effort",ylab="sales",type="
> plot(x,response(x,c(800,0.004,100),model="modexp"),xlab="advertising effort",ylab="sales",type=
> plot(x,response(x,c(800,-1,0.005, 100),model="logistic"),xlab="advertising effort",ylab="sales"
> plot(x,response(x,c(900,100,2,50000),model="adbudg"),xlab="advertising effort",ylab="sales",typ
>
```

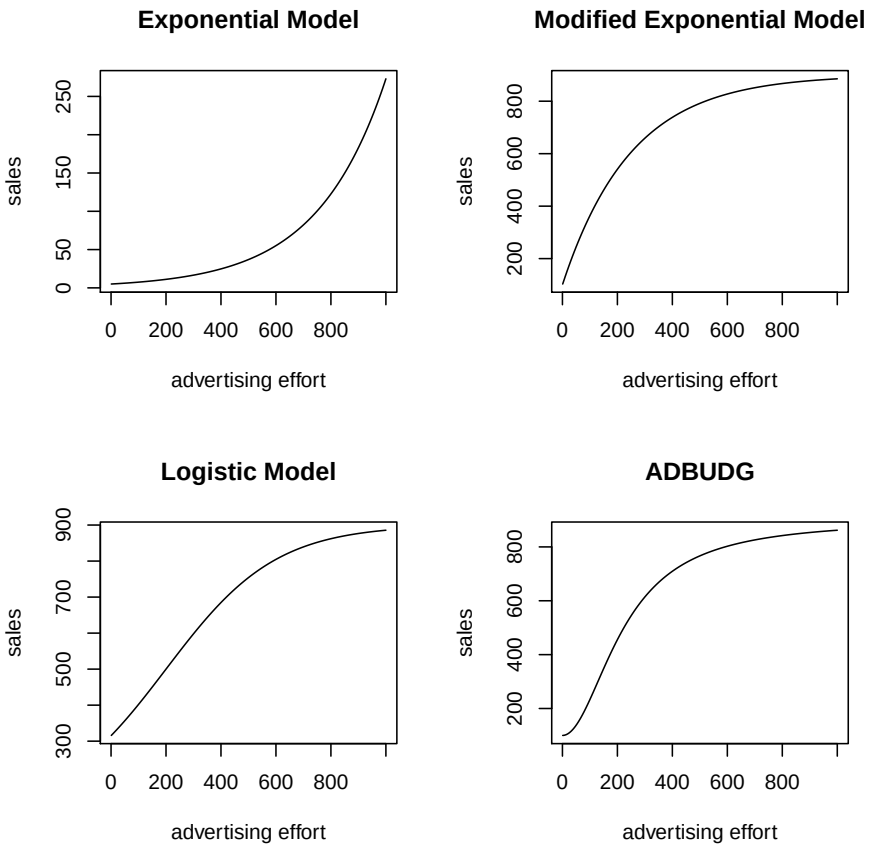

Figure 1: A sample of response models.

3 New Product Decisions

3.1 Theory

3.2 Computational Infrastructure

3.3 Examples

4 Salesforce and Channel Decision

4.1 Theory

Selling is one of the most important elements in the marketing mix. Many people in a company are engaged in sales and related activities. So firms should manage their salesforce appropriately. Furthermore salesforce sizing and allocation are fundamental issues in this context. Fortunately, there exist many models to support decision making in this area. Some of this models are going to be presented in this chapter.

4.1.1 Linear Sum Assignment Problem

The assignment problem is famous in both combinatorial optimization and linear programming. In a *Linear Sum Assignment Problem* a matrix $C = (c_i j)$ is given and we want to find a solution in which each row matches a column in such a way that the sum of the corresponding entries is minimized. The matrix C in the LSAP represents the costs. In the marketing sense we want to maximize our profit. Therefore we have to find the correspondig columns to each row, so that the sum is a maximum.

For more details regarding the Implementation see **clue**.

4.1.2 Salesforce Allocation

The best way of setting salesforce size is to increase selling effort in each control unit until the marginal cost of increasing selling effort is equal to the marginal return from increased sales (Beswick (1977)). It is clear that this ideal situation is not often met in practice since there are restrictions one has to face. For example there are budget restrictions. This Problem can be formulated as follows:

$$\text{Maximize Profits } = Sp - Nv - F, \tag{1}$$

$$\text{subject to } Nv \leq L, (N \text{ an integer}) \tag{2}$$

$$S \leq L_2 \tag{3}$$

For a given Salesforce N, maximizing profits (1) is equivalent to maximizing total sales:

$$S = \sum_{i=1}^{n} r_i \tag{4}$$

where r_i is the response to the selling effort in control unit i. The Algorithm works as follows

Step 1 maximize (4) and check if the constraints (2) and (3) are satisfied.

Step 2 If constraints are satisified and profits (1) have increased, add 1 to N and repeat step 2. When either of this two conditions is violated return to the previous step and stop.

The response function (r_i) should be a function which describes sales response to selling effort as a function of performance indicators (potential, workload, company effort and experience). Or it can be a subjective estimate of the response (considering these differnces).

Beswick (1977) use a response function of the following form:

$$r_i = z_i t_i^a + C_i \tag{5}$$

where r_i respectively z_i can be estimated with non-linear regression. t_i is the time a salesman spends in area i (which is to be calculated). The following function was found to give good results:

$$z_i = 0.3258(w_i/\bar{w})^{0.172}(c_i/\bar{c})^{0.646}p_i^{0.694}(m_i/\bar{m})^{0.105} \tag{6}$$

$$C_i = 0$$

$$a = 0.217$$

4.1.3 Planning Sales Calls

Companies with a large salesforce can by improving efficiency and effectiveness of managing sales calls increase their profits. Lodish (1971))˜proposed to use an interactive salesman's call planning system called *CALLPLAN* to aid salesman or salesmanagement in allocating sales call time more efficiently. The aim is to maximize the returns from their calls. The model is based on the assumption that the expected sales to each client (or prospect) is a function of the average number of calls per effort period during a response period (which should be long enough to accomodate carryover effects from each effort period)

CALLPLAN depends on the formulation of a response function (ie. *ADBUDG*, for details see chapter 2)

The mathematical formulation of this problem can be stated as follows:
Find the set of x_i to maximize profits z

$$z = \sum_{i=i}^{I} a_i r_i(x_i) - e\sum_{j=1}^{J} NT_j c_j \tag{7}$$

There are some constraints which have to be hold:

- The amount of time spent on selling and travelling must be less then the amount of time available during an average effort period.

$$\sum_{i=i}^{I} t_i x_i + \sum_{j=1}^{J} NT_j u_j \leq T \tag{8}$$

- The number of trips to an area is a function of the number of calls made to each account in the area.

$$NT_j = \text{Max } \{x_i \text{ such that } g_i = j\} \text{ for } j = 1, \ldots, J \tag{9}$$

- Furthermore the number of calls in and effort period must lie within the stated bounds.

$$Min_i \leq x_i \leq Max_i \text{ for } i = 1, \ldots, I \tag{10}$$

4.2 Computational Infrastructure

Function `solveAP` calculates the assignments and the maximum of a given assignment problem. It takes a nxn matrix as an argument.

Function `beswick()` takes a vector of performance indicators for the control units and certain parameters like the budget restriction or the maximum sale one can achieve. Furthermore you can set a verbose flag.

Function `performanceIndicators()` calculates the performance indicator for each control unit. See equation (6) for details.

There is a method for `coef`. This method returns the selling effort one should give to each control unit.

4.3 Examples

- Linear Sum Assignemt Problem

```
> ## Linear Sum Assignment Problem
>
> data("ap")
> ap

   OE WD PR
A 100 90 80
B  92 92 85
C 102 95 75

> solveAP(ap)

Optimal Assignment with maximum of 280 :
A => OE, B => PR, C => WD
```

- Allocating Selling Effort

```
> ## Allocating Selling Effort via Dynamic Programming
>
> ## some performance indicators
> g <- c(75.36, 28.76, 36.21, 43.28, 56.65, 69.27, 63.04)
> names(g) <- LETTERS[seq(g)]
> bm <- beswick(g)
> bm

maximized profit: 184.564287933158
salesforce size: 10

> coef(bm)

         B          C          D          E          G          F
0.06520882 0.08696728 0.10868742 0.15216687 0.17391638 0.19565985
         A
0.21739339

> ## you can do nonlinear regression or you calculate the
> ## performance for each control unit as follows
>
> ## workload for each area i
> w <- c(20,25,14,38)
> ## prior company experience (market share)
> c <- c(0.35,0.3,0.12,0.32)
> ## potential sales in area i
> p <- c(100,150,130,120)
> ## regional managers experience (time with the company)
> m <- c(1.5,3,2.5,4)
> I <- performanceIndicators(w,c,p,m)
> besw <- beswick(I)
> besw

maximized profit: 54.4320243402315
salesforce size: 10
```

- Callplan

```
> ## CALLPLAN
> data("callplan")
> CPEX

      Calls Hours Sales Margin none  -50  +50 saturation
north     4     2    60    0.4  0.2 0.62 1.30        1.5
south     4     2    50    0.3  0.3 0.70 1.15        1.4

> cpm <- as.callplan(CPEX)
> cpm

A callplan for  2 territories

> ## current plan
> x <- do.call(eval.callplan,c(list(cpm$e), r=2, cpm,
+              details=TRUE))
> x

$profit
[1] 39

$slack
[1] 0

$effort
north south
    4     4

$sales
north.b south.b
     60      50

> ## optimize plan with gom
> gcp <- gom(eval.callplan, evalArgs=c(r=2, cpm), evalVector=TRUE,
+            bounds=data.frame(Lower=rep(0,length(cpm$h)),
+                              Upper=rep(8,length(cpm$h)),
+                              row.names=names(cpm$p)),
+            numVar=1:2, intVar=1:2, trace=TRUE)

 0 39
  1
 2 40.90114
  3
 4 41.05269
  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28

> xx <- do.call(eval.callplan,c(list(gcp$bestVar), r=2, cpm, details=TRUE))
> xx

$profit
[1] 41.05269

$slack
[1] 0
```

```
$effort
north south
    5     3

$sales
 north.b  south.b
69.82283 43.74520
```

```
> ## some plots
> op <- par(mfrow = c(1,2), pty="s")
> barplot(cpm$e, ylab="Calls",
+         main=paste("Current:",formatC(x$profit,digits=3)))
> barplot(gcp$bestVar, ylab="Calls",
+         main=paste("Recommended:",formatC(gcp$bestFit,digits=3)))
> par(op)
>
```

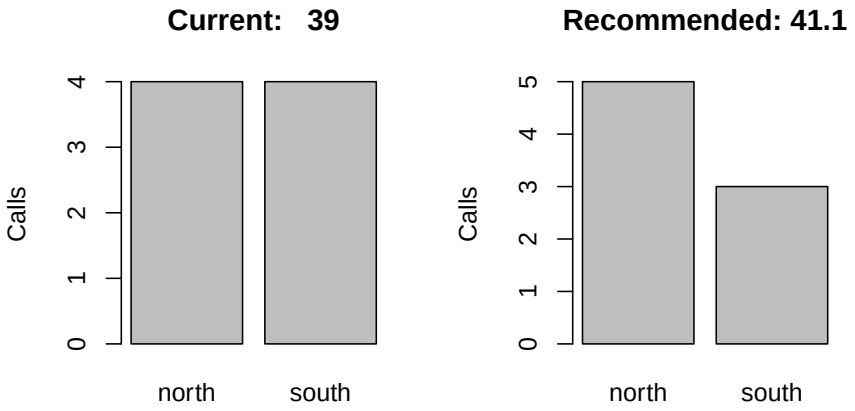

Figure 2: Optimizing Callplan

References

C.˜A. Beswick. Allocating selling effort via dynamic programming. *Management Science*, 23(7): 667–678, March 1977.

G.˜L. Lilien, P.˜Kotler, and K.˜Moorthy. *Marketing Models*. Prentice Hall, 1992.

L.˜M. Lodish. Callplan: An interactive salesman's call planning system. *Management Science*, 18 (4):P25–P40, December 1971.