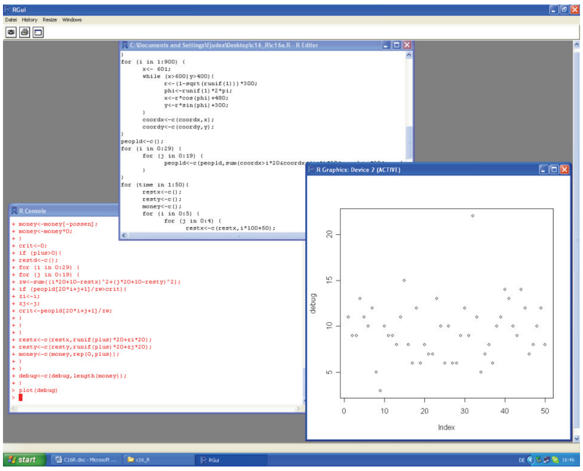
A Programmed Solution to ARGESIM Benchmark C16 'Restaurant Business Dynamics' using GNU R

Florian Judex, Vienna University of Technology, Austria, *florian.judex+E101@tuwien.ac.at*

Simulator: GNU R is an open source statistical software package, aimed to replace the popular S software. Due to a high level vector oriented scripting language akin to MathScript and the ability to handle huge amounts of data without slowing down, it can readily be adapted for Simulation Purposes. Like MATLAB, it distinguishes between the line oriented command shell, scripting, and windows for graphic output, as shown in Figure 1 below.


RGui
Datei History Resize Windows
C:\Documents and Settings\fjudex\Desktop\c16_R\c16a.R - R Editor
R Console
R Graphics: Device 2 (ACTIVE)
debug
Index
5
10
15
20
0
10
20
30
40
50
start
C16R.doc - Microsoft ...
c16_R
RGui

```
}
for (i in 1:900) {
        x<- 601;
        while (x>600|y>400){
                r<-(1-sqrt(runif(1)))*300;
                phi<-runif(1)*2*pi;
                x<-r*cos(phi)+480;
                y<-r*sin(phi)+300;
        }
        coordx<-c(coordx,x);
        coordy<-c(coordy,y);
}
peopld<-c();
for (i in 0:29) {
        for (j in 0:19) {
                peopld<-c(peopld,sum(coordx>i*20&coordx
        }
}
for (time in 1:50){
        restx<-c();
        resty<-c();
        money<-c();
        for (i in 0:5) {
                for (j in 0:4) {
                        restx<-c(restx,i*100+50);
```

```
+ money<-money[-possen];
+ money<-money*0;
+ }
+ crit<-0;
+ if (plus>0){
+ restd<-c();
+ for (i in 0:29) {
+ for (j in 0:19) {
+ zw<-sum((i*20+10-restx)^2+(j*20+10-resty)^2);
+ if (peopld[20*i+j+1]/zw>crit){
+ zi<-i;
+ zj<-j;
+ crit<-peopld[20*i+j+1]/zw;
+ }
+ }
+ }
+ restx<-c(restx,runif(plus)*20+zi*20);
+ resty<-c(resty,runif(plus)*20+zj*20);
+ money<-c(money,rep(0,plus));
+ }
+ }
+ debug<-c(debug,length(money));
+ }
> plot(debug)
>
```

Figure 1. The Gnu R Interface.

Model: Although C16 deaks with a discrete model, there is no need for event handling, as all decision are only based on the current state of the system, and there are no time delays which would eventually result in the cancellation of events.

Therefore the activity scanning approach is feasible.

Customers and restaurants are modelled as list containing, adding time in case of the customers and money in case of the restaurants. For the placement of the customers, the inverse cumulative density function of the given triangular distribution, which is not implemented in R, was derived by hand. The random values where then simulated using this formula and a [0,1] continuous uniform probability distribution.

To simplify computation, the continuous uniform probability distribution on [0,8] of the intervals between consummations was changed to a discrete uniform probability distribution on [1,7], preserving the mean of 4 and slightly lowering the standard deviation from 2.3 to 2.

This allows an easy advancement of time, using nested loops for weeks and days, e.g.

```
for (week in (1:years*52))
    for (day in (1:7) {
        time <- time - 1;
    }}
```

Using the vector oriented notation in R, many computations or evaluations of logical constructs can be done simultaneously, further speeding up the simulation. For example, determining the indices of the persons using a restaurant on a certain day can be determined by the command

```
active <- (1:3000) [time==0]
    for (counter in active) {...}
```

which creates a vector using those elements of the vector (1,2, … 3000) satisfying the condition the time counter of corresponding person equals zero, and uses this list to loop through these entries. A similar construct can be used to determine which restaurant is visited.

On the other hand, entries can also be removed from vectors, by slightly changing the arguments. The lines

```
restx<-restx[-positions];
resty<-resty[-positions];
money<-money[-positions];
```

will remove of (x,y) coordinates and the money counters for those restaurants identified in the vector called 'positions', and adjusts the length of the vector accordingly.

It should be duly noted at this point that R defines a matrix as an array with dimensions, which can be converted back to an array at any time. Just using one index when accessing a matrix will implicitly revert it into an array immediately, leading to propagating and hard-to-find errors in the script. Therefore one is on the save side using separate array like for x and y above when there is no real need for a matrix.

Finally, a combination command can be used to concatenate vectors. This used to place new restaurants. For example

```
money<-c(money,rep(0,plus));
```

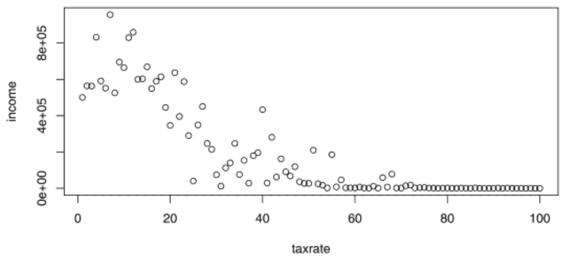


Figure 2. Overall Tax Income

adds as many new money counters as specified by the variable 'plus' behind the last existing one.

Using those tools, the whole model is implemented in less then hundred lines of code.

A-Task: Simulation of the one, five and ten year period is done easily by varying the variable for lentght of the outer loop. Plotting is done using one of the various routines offered by R for the graphical representation of data. Likewise, encapsulating the whole simulation routine into another loop allows 50 runs of the simulation, and one can use R statistical capabilities for analysis. This showed a mean value of 7.84 restaurants after 5 years, with a range between 3 and 14 surviving restaurants.

B-Task: If the tax rate climbs above 33 percent, the expected income from the two smaller towns gets too small to support a single restaurant. Therefore those 20 percent of the population nearly stop contributing to the overall tax income. In the simulation, the same starting Population was used to do five five-year runs for every tax rate between 1 and 100 percent. The average income per year depending on the tax rate is shown on Figure with a maximum at 7 percent, and the highest values between 4 and 12 percent, where the big cities are able to support 3 respectively 4 restaurants with a suitable safety margin, which cover most of their population.

C-Task: To keep track of the income of new restaurants lead new parameter vector has to be introduced. It is set to one in the first thirty restaurants, and zero in the newer ones. Therefore, just by computing the sum of this vector, the program keeps track of the old restaurants still existing. Together with the practice of adding the new restaurants after the last one already existing, the offset of the first new restaurant is calculated.

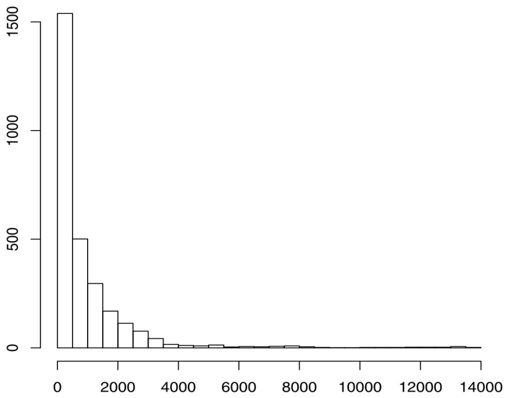


Figure 3. :Histogram of incomes

To create comparable data, the same starting population was used for each parameter, and the simulation time was set to 10 years, to even out the effects of the starting population.

The (rounded) summary in Table 1 shows the exponent of the distance has little effect on the overall expectation of income. This is because all new restaurants will start in the same square as the density is not updated after each restaurant, competing with each other for the customers available there, and quiet often closing down after one week. This leads to the data shown in the histogram of the income of the restaurants in their entire lifetime for d=6, with a mean of over 900, but a median of about 400, which equals less then two weeks of income about the profit threshold. A few survive, but most close down after just one week.

Resumé: As R is a statistical software with the aim of handling huge amounts of data, even quite a lot of operations on vectors of the length 3000 do not slow down the programm significantly compared to a test run with fewer people. As the timescale was quite large, simplification of the probability function did not alter the results in an unpredictable way.

Corresponding author: Florian Judex
Vienna University of Technology
Department of Analysis and Scientific Computing
Wiedner Hauptstraße 8-10, 1040 Vienna, Austria
florian.judex+E101@tuwien.ac.at

Received: January 20, 2007
Revised: February 25, 2007
Accepted: February 26, 2007