Developing An Advanced Preference Share Simulator

A Flexible and Scalable Approach using R and the Amazon Elastic Compute Cloud

Charles Carpenter (RSG), Jeffrey Dumont (RSG), Nelson Whipple (RSG)

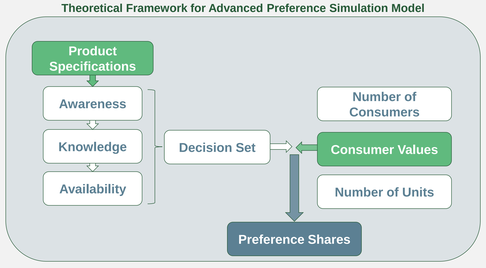
RESOURCE SYSTEMS GROUP, INC.

✓ Background

Recently, there has been a push towards developing more sophisticated simulation models to better aid in making marketing decisions. These more advanced simulation models bring together many ideas from a variety of fields – artificial intelligence, game theory, and probability theory to just name a few – and result in simulation models that have more realistic market behavior. The move towards complex simulations models comes at a cost in terms of computational resources. Being able to run a simulation in a practical amount of time is a concern of practitioners who are trying to meet client deadlines. One solution to overcome the computational challenges is to use recent developments in the R[1] programming language and cloud computing.

This poster provides practical guidance on developing and implementing an advanced simulation model. We combined preferences estimated from a hierarchical Bayesian choice model with additional survey questions about brand knowledge, awareness and familiarity. The computational challenges occur in the Monte Carlo simulation from both the model's posterior distributions and the distributions of knowledge, awareness and familiarity.

Theoretical Framework for Advanced Preference Simulation Model



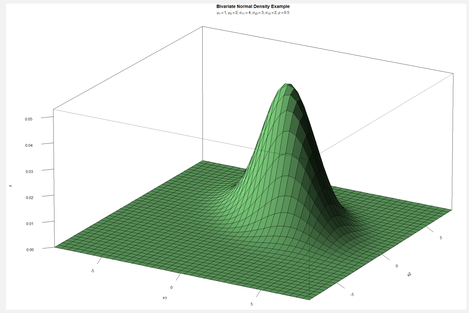
1 Model Implementation Framework

- Several software packages are capable of implementing the preference simulation model
- R was chosen for several reasons
  - Open source
  - Contains all of the necessary statistical functions
  - Easily set up to run "in the cloud"
- Our approach overcomes its potential drawbacks
  - Slow to execute code relative to other software
  - Limited by the amount of memory (RAM) on the computer (exceptions do exists, but were not explored)

2 R Has The Necessary Statistical Routines

- Key input to the preference simulation model is a set of posterior distributions of utility from a hierarchical Bayesian choice model
- Preference simulation model uses draws from these posterior distributions which can be generated effectively through the package MSBVAR[2] and function `rmvnorm()`
  - Function `mvrnorm()` from package MASS is also available

```
require(MSBVAR)
#Define variance-covariance matrix
sigma <- matrix(c(4,2,2,3), ncol = 2)
#Generate 500 random samples
x <- rmvnorm(n = 500, mean = c(1,2), sigma = sigma)
```



3 Speed Up R Code With data.table Package

- Once the draws have been simulated from the underlying posterior distributions, generating preference shares for each product in the model simply becomes a data aggregation task
- Aggregation tasks can prove very time intensive depending on the size of the sample and the number of draws simulated
- The data.table[3] package proved fastest for these tasks compared to base R functions and the plyr[4] package
- We compare timings from the base function `aggregate()`, the plyr function `ddply()`, and the `data.table()` package
- The test data contains 1000 draws for 8 products with 7 attributes for a dataset with 211 respondents

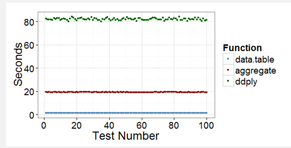
Structure of Draws from the Posterior Distribution

Respondent	Product	Attribute	Draw1	Draw2	Draw3
1	1	1	-2.19	2.90	-0.70
1	1	2	-0.16	1.46	-0.22
1	1	3	-0.43	1.47	-0.33
1	1	4	-2.29	2.20	-1.85

3 Speed Up R Code With data.table Package II

```
#Base R solution using aggregate() function
aggregate(theData[, !grepl("Respondent|Product", names(theData))],
          list(resp = theData$Respondent, prod = theData$Product),
          FUN = "sum")
#plyr solution using ddply
require(plyr)
ddply(theData, c("Respondent", "Product"), numcolwise(sum))
#data.table solution. Must convert data.frame into data.table
require(data.table)
theData <- data.table(theData, key = c("Respondent", "Product"))
theData[, lapply(.SD,sum), by = c("Respondent", "Product")]
```

- A set of function timings show that the `data.table()` function is nearly 15x faster than base function `aggregate()` and 55x faster than `ddply()`
- Our experience has shown that `data.table()` outperforms the other options as the size of the dataset increases



Function*	Mean (s)	SD (s)
data.table	1.45	0.03
aggregate	19.33	0.10
ddply	82.03	0.94

*Timing for computing utility sums for 1000 draws from posterior distribution

4 Make R Take Advantage of Multiple Cores

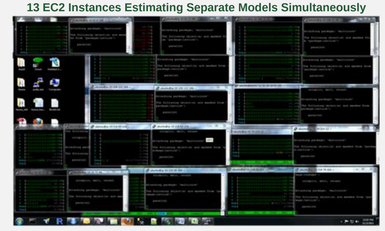
- Preference share calculation is perfect for parallel processing
- Individual level results do not depend on the results of other individuals or draws
- R has ample support for writing code to take advantage of multiple core computers, though specific implementations do vary by operating system
  - `parallel` package comes with R >= 2.14.0
  - `mclapply()` in Linux/Unix
  - `parLapply()` in Windows
- Writing the preference simulation code to take advantage of `mclapply()` provided a nearly 3.5x speed improvement using four cores
- In other applications, the speed improvement may be negated by the overhead in splitting tasks over multiple cores

```
require(parallel)
numDraws <- 5000
#Single core
lapply(seq_len(nrow(theData)),function(x)
  rmvnorm(numDraws, mean  = as.numeric(theData[x,2:39]),
          sigma = theCovMat))
#Multiple cores, Linux/Unix only
mclapply(seq_len(nrow(theData)),function(x)
  rmvnorm(numDraws, mean  = as.numeric(theData[x,2:39]),
          sigma = theCovMat))
#Windows variant. Must set up cluster manually, exporting needed
#packages and data to each core. Don't forget to stop the cluster
#when you're done!
cl <- makeCluster(getOption("cl.cores",4))
clusterEvalQ(cl, library(MSBVAR))
clusterExport(cl, c("theData", "theCovMat", "numDraws"))
parLapply(cl = cl, seq_len(nrow(theData)),function(x)
  rmvnorm(numDraws, mean  = as.numeric(theData[x,2:39]),
          sigma = theCovMat))
stopCluster(cl)
```

5 Run R in the Cloud

- Speed gains from using `data.table()` and multiple cores still may not provide enough computing capacity to estimate many models typical in practice
- Amazon Elastic Compute Cloud (EC2) is an effective way to add computing capacity
  - Web service that provides sizable computing capacity suitable for problems of almost any size
  - Charges per hour with rates ranging between $0.00 - $2.00
- The developers of the Bioconductor[5] project host a public Amazon Machine Image (AMI) with R and related packages preinstalled reducing time to start and configure for each project
- Additional open source tools were used to facilitate data transfer: PuTTY[6] and WinSCP[7]
- Possible to run 20 concurrent EC2 instances from a single Amazon Web Services (AWS) account
- Charges accrue even if not actively used, so plan carefully to avoid unneeded expenses

13 EC2 Instances Estimating Separate Models Simultaneously



6 Future Improvements

- Rewrite time intensive tasks in C++ using Rcpp[8]
- Leverage additional computing power through Amazon's MapReduce framework. The segue[9] package provides a basic interface.
- Develop tool to manage workflow to EC2 instances. Amazon Auto Scaling[10] may be an off-the-shelf solution.

7 For Further Reading

[1] R Development Core Team (2011). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.
[2] Patrick Brandt pbrandt@utdallas.edu (2011). MSBVAR: Markov-Switching, Bayesian, Vector Autoregression Models. R package version 0.6-0. http://CRAN.R-project.org/package=MSBVAR
[3] M Dowle, T Short and S Lianoglou (2011). data.table: Extension of data.frame for fast indexing, fast ordered joins and fast grouping. R package version 1.6.6. http://CRAN.R-project.org/package=data.table
[4] Hadley Wickham (2011). The Split-Apply-Combine Strategy for Data Analysis. Journal of Statistical Software, 40(1), 1-29. URL http://www.jstatsoft.org/v40/i01/
[5] Bioconductor: Open software development for computational biology and bioinformatics R.Gentleman, V. J. Carey, D. M. Bates, B.Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier,Y. Ge, and others 2004, Genome Biology, Vol. 5, R80.
[6] PuTTY: A Free Telnet/SSH Client. http://www.chiark.greenend.org.uk/~sgtatham/putty/
[7] WinSCP: Free SFTP, SCP, and FTP client for Windows: http://winscp.net/eng/index.php
[8] Dirk Eddelbuettel, Romain Francois (2011). Rcpp: Seamless R and C++ Integration. Journal of Statistical Software, 40(8), 1-18. URL http://www.jstatsoft.org/v40/i08/
[9] JD Long. Segue: Parallel processing on Amazon's Web Services. Includes a parallel lapply function for the Elastic Map Reduce (EMR) engine. http://code.google.com/p/segue/
[10] Amazon Auto Scaling: http://aws.amazon.com/autoscaling/