Market Research using SAS, Ruby-on-Rails, and Sawtooth

Tom Taylor
Palladian Analysis & Consulting, LLC
Houston, TX
ttaylor@computer.org

Abstract

Market research over the web seems common, but most commercial surveys require robust systems that can be custom designed for the nuances of the specific customer. Do-it-yourself survey websites such as Zoomerang are not adequate. The client wants complete control that only a custom survey site can offer. These commercial surveys are for product design optimization, quality of service, and client satisfaction.   Over the past year, the author has created and executed several customer surveys with surprising efficiency. He has found Ruby-on-Rails to be an ideal development environment for the web interface. Back-end analysis is still best done with SAS and sometimes Excel pivot tables. For product optimization studies, the conjoint model from Sawtooth Software has proved to be very cost effective. Some of these studies have been done in four languages simultaneously. Hosting has moved from rented web-farm servers to geographically separated Mac-Mini's. This paper discusses the techniques the author considers best practice in these surveys.

Market Studies

Thanks to the web, for better or worse, surveys have proliferated. Most of us are at the point of feeling annoyed by surveys. However, surveys continue to be a handy tool for collecting systematic data about products, employees, or customers/patients/clients.

In reality, surveys are used to tune market or personnel systems. Engineers can tune modern cars because they have dozens of sensors on them all hooked back through a central data system. Surveys are the marketers and personnel managers poor version of the sensors. We gain a few insights into the nature of the client or employee as to what is good and bad, efficient or slow. These insights may be good for a week or a lifetime and will never be perfect. On the other hand they have an excellent track record of giving warning of impending problems. For new product studies, surveys are invaluable for pining down the priorities for product features.

Trending Surveys

We have two types of trending surveys we work with, Customer Satisfaction and Client Satisfaction. The objectives of these surveys are two. First is to find areas for immediate improvement, the second is to monitor trends year to year. Our most successful surveys have been incorporated into the objectives and bonus criteria of managers and sales personnel. These surveys normally serve two user groups, executive management and front line management. For executive management we like to use SAS clustering as a tool to focus on areas of interest. Clusters allow us to spot anomalies that would otherwise be overlooked. In one study, clusters showed that workers in politically unstable countries were less satisfied with their job than workers in stable countries such as the US. For front line management we like to use Excel pivot tables to report the results. This allows everyone to drill down and see the specific results for their individual area of interest. These pivot tables are a pain to put together, but they really pull the interest of front line managers.

In customer satisfaction studies we typically work with multiple languages and several divisions of the client company. The question set we use is based on extensive research that has identified robust questions. However, one of our main goals is to design the survey to be very efficient for the respondent. This is where we have found Ruby-on-Rails to be so useful. In combination with HTML tables we can place comparison questions side-by-side and precisely layout the questionnaire for speed of comparison by respondent. We also make it very apparent that they are working in a secure website. This further enhances their belief that this is an important survey and that we value their opinion.

To facilitate multiple languages, we place most questions in a database table with one field being language type. Then each question presentation is just a call to a data record and field. Otherwise the code is the same for all languages. It is fairly amazing what one can put in the memo field of a modern database. Images, computer code, markup languages, all seem to work. Occasionally we have problems because the world had not yet completely standardized on utf-8 Unicode for character display and storage. Currently if one can find all the defaults settings in advance of coding and change them to utf-8 then every major language displays properly on first try. Most software seems to be set to Latin1 as the language standard. This starts failing with Spanish and seems only to do well with core European languages like English and French.

Product Optimization Studies

Many of our new product studies include conjoint analysis. Conjoint is an ingenious method developed in the 70's at Wharton School of Business
(ref: http://knowledge.wharton.upenn.edu/paper.cfm?paperID=779). Conjoint does a good job of revealing insights into respondents decision making process. It boils down to this; conjoint reveals the trade-off process we all go through to make a selection or for products, a purchase decision. It does a good job of modeling at what point we stop focusing on price and start looking at features Conjoint allows us to understand the tradeoff process. Once the process is understood, then desirability of features can be modeled.

How We Use Software for Market Studies

Ruby-on-Rails (RoR)

Ruby-on-Rails (RoR) is what we use for web pages and to connect to the database. A little background for you. Ruby is a programming language. Rails is a directory structure, that includes log files, and a pre-formed data base structure that really facilitate the rapid development of web services for databases. Many call Rails a framework. To the SAS user it means RoR has many characteristics one likes in SAS, examples are log files, an easy facility to produce output, and a scripting, no compiler process. The Ruby language is object-oriented. Object oriented has value in that it encourages code reuse and organization. A big feature of Ruby-on-Rails is ease of accessing back end data. RoR is most commonly used in conjunction with the mySQL database. There are several databases than can readily be used with RoR. SAS is not one of the direct linked data bases but there are tools to allow the connection to be made. These will be explained later.

Ruby itself has been around as long as Java, i.e., the mid-90's. The Rails addition is a more recent development that caused interest to soar. One ranking of software interest, the "TIOBE Programming Community Index for June 2007" (http://www.tiobe.com/index.htm?tiobe_index) Gives RoR as the 10th most popular programming language and rapidly rising from the 19th spot one year ago. SAS is listed as 12th on the same list.

So why is RoR so popular? RoR is just very well optimized to put a web services front end on a database. To get started in the RoR world requires a more extended learning period than say Visual Basic or SAS. However, once there life becomes much easier. Specifically, it is probably easier to start a web project in something like Visual Basic or a .NET tool. However in most cases, it will be easier to finish and maintain a project in RoR. The reason for this statement is the rails structure and object nature of Ruby restrain one from corrupting structure in the last few days of the project. Many times those last minute add-ons and fixes cause code volume to double as slight variations of paths and variables cause one to just repeat blocks of code to avoid hours of making the code more adaptable. The object world tends to be naturally more adaptable and thus less likely to cause coding to turn into a massive cut and paste project. Once cut and paste gets started then future changes require global searches of directories to find the hard coded variables. A year or two later, developers become fearful of changes. So RoR is gaining popularity because its web services framework makes it relatively quick to get some bare bones pages up. Then the object structure makes it much more maintainable over time.

SAS

When we started using SAS for market studies in the 80's, studies were paper and pencil. All data was hand coded and analyzed. The key SAS tools were clustering and regression. Today we use SAS mostly for clustering. Regression has moved to Sawtooth Software. This is not a bad reflection of SAS, Sawtooth just has very specialized tools for doing conjoint studies. These tools serve our needs for regression.

For the web services user interface, avid SAS fans will point out that SAS has tools to replace everything we do with RoR, so why not use these? It really boils down to licensing, remote servers, and current trends in computer science. Our current environment is to develop on a laptop and move production code to two Mac-mini doing web-server duty (more on these servers later). This would require three licenses and SAS would not like us using Mac-mini licenses as public web servers. Also, for some clients we rent space on remote servers, that is yet another SAS license issue. So to maintain flexibility and for ease of deployment we moved into the RoR world for data collection.

Thus driven by the forbidding constraint of SAS licensing, we have generally moved to less constrained products for small and quick projects. Additionally, RoR skills are widely applicable to many database systems. We can easily adapt to existing client databases. Most code improvements can be kept in RoR while the back-end database remains unchanged.

Sawtooth Software

In the 80's, we started doing Conjoint with SAS, and 3x5 cards. but as things moved to the web, the card scheme could not transfer. Sawtooth Software came up with a scheme that allowed the cards to be replaced with a presentation that would work on computer screens. This solution was a specialized niche and SAS never did not copy but seem instead to support Sawtooth efforts. Thus tiny Sawtooth Software (http://www.sawtoothsoftware.com/) has become the standard for this niche. The Sawtooth process does a substitute for the regression process, but Sawtooth does not cluster responses. So SAS's great clustering tools are still used there.

SAS and RoR Comunication

If you are still cognizant after the above discourses, you may be asking why not at least eliminate mySQL and just use SAS as the back-end database connected straight to RoR. After all SAS in essence is a database with excellent statistical tools.

It is clearly possible to put SAS behind RoR, we have just not had a requirement to force us to take the time to do so. (We plan to have an example running before the presentation.) There are four ways for this to be accomplished, three of which are available today. The preferred way is to use the ADO (ActiveX Data Objects) interface that was developed by Microsoft and supported by both SAS and RoR For SAS, in the 9.1 documentation refer to “SAS Data Providers: ADO/OLE DB Cookbook “ and study the IOM (integrated object model). For RoR refer to
http://rubyonwindows.blogspot.com/2007/06/using-ruby-ado-to-work-with-ms-access.html
This example uses Ruby and ADO to communicate to MS Access but you will quickly see it is an exact parallel to the SAS material above. We have used this technique to connect PHP to SAS, so RoR to SAS using the same technique looks completely feasible.

A word of advice; when these systems fail, they seldom give a useful error message, they just do nothing. Usually permissions (authentication) or services that are not running are the problem. Tracing these issues down tends to be trial and error work. This can be extremely time consuming. When working on this avoid giving time commitments, you will likely be way off.

Back to technique alternatives: The second technique is to use ODBC communication. ODBC (Open DataBase Connectivity) is a predecessor to ADO. This is not as good as ADO as it is purely for communicating data. ADO will do both data and code blocks/stored procedures (SAS and Oracle like the stored procedures name). The SAS ODBC tools are in SAS docs, the Ruby tools are at http://rubyforge.org/projects/odbc-rails

The third technique is to have both SAS and ruby create XML (extensible markup language) files. Then both tools can be running and check for the existence of specific data sets, if the data exists, they execute. This sounds a bit crude, but it is actually used by many big systems. Intel runs billion dollar chip factories using this communication scheme as a standard. The only trick here is to contain your listening scheme so that it does not eat many of your CPU cycles by just repeatedly seeing if a file exists.

The forth technique is to wait for someone to create a Ruby database driver. We count 13 other Ruby database drivers at present. But give that both Ruby and SAS speak ADO, it is not likely a Ruby – SAS specific driver will ever be developed.

Issues With Servers

Growing tired of Microsoft Servers

I[1] started in the micro-computer world with an Apple II. In the decades since, I have gone between Apple and PC's on numerous occasions. For the last decade, applications had kept me in the PC-Microsoft world. Over the last few years I have become very dishearten by the difficulty of setting up new servers in the Microsoft world. I noticed that nearly everyone came up with basic server setups and then resist changes out of a great fear of pathing or permission problems. I was included in this fearful group. I had also grown tired of endless clicking to find obscure windows and having no easy way to remember where the obscure windows were hidden. My experience with PHP, RoR and Apache had lead me to conclude that configuration files were much easier to deal with than the endless point and clicking. These irritations and the rise of interest in Linux lead me to.... Apple.

Return to Apple

A colleague of mine who has several small business clients came by one day and showed how he had thrown out all the windows servers and moved these folks to Ubuntu Linux (ref: http://www.ubuntu.com/) He reported the clients were just as happy, but his administration life was much more straight-forward. His reasons were configuration file setups, easy imaging

1 Forgive me for switching tense from we to I, this section is one persons experience.

between machines and no licenses to keep straight. This sounded attractive, but I wanted to use nice small machines that were quiet and could set around an office without looking like a tangle of wires and tacky odd colored computer boxes. This lead me to the Apple Mac-mini. It turns out that Mac had a few years back switched its OS over to FreeBSD, http://www.freebsd.org (essentially UNIX but not legal to call it that). But this FreeBSD from Apple came packaged with a very nice GUI (graphical user interface) and preloaded in a pretty, cigar-box size, quiet running computer called the Mac-mini http://www.apple.com/macmini/ . So for $600 I get a world class OS, the Apache web server, more computing power than NASA probably had in the the moon landings, and it all sits quietly on the corner of my desk.

Apple versus Microsoft for Web Servers

Having been dedicated to Microsoft for the past decade, returning to the Apple Mac was like learning to walk all over, a struggle for a day or so. After overcoming the single button mouse and the slew of other details that were different, I got down to development work. What I discovered was the strength of the OS was what could be done in a terminal session. I concluded that Microsoft had spent the last decade trying to kill the DOS window terminal session. Whereas the FreeBSD/UNIX world had enhanced the terminal session to where it was far more flexible than the limitations imposed by pop-up windows. The catch was one needed the documentation much more than the take-a-guess point-and-click world.

The other benefit was the Microsoft registry disappeared. No longer was I dealing with obscure long codes to get components to work together.

Reliability and Redundancy

When working outside corporate server farms or rented rack space, the potential expense of redundancy does not disappear. To engineer a truly redundant system at a single site is very expensive, two paths to the web, backup power sources etc. We conclude that if we ran identical systems at two geographically separate locations and then just redirected the DNS (domain naming service, the http://...) when the primary system failed we could have excellent reliability with downtime limited to minutes anytime there was a failure. To accomplish this we redirected our URL to www.DNSmadeeasy.com This service will test your primary server every few minutes and should it fail to respond, immediately redirect all traffic to another server. In our case the servers are 40 miles apart, but they could be 1000 miles apart just as well. Admittedly we run the risk of losing some data during the fail-over, but for our clients this is not a big issue. With a little more effort we could add more features that would eliminate most of this loss as well.

Conclusion

So what should you take away from this overview? Here are the top seven:

1. Trending surveys of employees or customers insure your organization is top notch and nothing of significance, e.g. hidden problems, is being missed.
2. Product optimization studies using conjoint are robust and extremely valuable to product designers trying to determine feature and cost trade offs.
3. Ruby-on-Rails is a very efficient web services development environment. It popularity has grown rapidly over the past 18 months. This rapid growth is likely to continue.
4. By using a combination of SAS, Ruby-on-Rails, and Sawtooth software, market and personnel studies of any reasonable nature can be efficiently created and analyzed.
5. There are many ways to create inter-process communication. We listed four that might be use for SAS and Ruby. Three are available today.
6. The power of inexpensive computers and operating systems continues to amaze. The $600 Mac-mini can be use as a robust web server and has capacity to serve moderate size web applications.
7. There are ways to create redundant systems that are physically separated with automatic fail-over. These can offer excellent backup and fail-over capacity at a fraction of the cost of bullet-proof data centers.

Thank you for your interest.

About the Author:

Tom Taylor has been a SAS affiliated partner for fourteen years. His core interest are with Market Studies, Operations Research, and Inventory Optimization. His systems serve several clients in the Houston area, including several Fortune 1000 companies. He has extensive experience in developing GUI's for SAS. Most recently, he has turned his efforts to using Ruby-on-Rails for a robust GUI development environment. He also serves as adjunct professor of project management at University of Houston-Victoria.