Running Head: SMS-based research

Using SMS text messaging for teaching and data collection in the behavioral sciences

Stian Reimers[1] and Neil Stewart[2]

[1]University College London, United Kingdom

[2]University of Warwick, United Kingdom

Stian Reimers
Department of Psychology
University College London
26 Bedford Way
London WC1H 0AP
England
Tel: +44 207 679 5317
Fax: +44 207 436 4276
s.reimers@ucl.ac.uk

Reimers, S., & Stewart, N. (2009). Using SMS text messaging for teaching and data collection in the behavioral sciences. *Behavior Research Methods, 41,* 675-681.

Abstract

Recent interest in university teaching has focused on interactivity in lectures and practical classes, and teachers in several fields have set up systems in which students can interact with the lecturer using mobile-phone based SMS text messaging. This approach has particular potential in psychology, where students could use SMS messaging as a way of responding in simple psychology experiments or demonstrations. We describe a simple architecture for an SMS-based responding, using an SMS-to-HTTP message relay service, and a php/mySQL input-output handler. We describe briefly two experiments we have run using the system. The first experiment examined anchoring effects in an SMS-based auction. The second experiment examined delay discounting, with participants indicating their intertemporal preferences using SMS. Finally, we evaluate the feedback we obtained from students, about the practical and conceptual issues surrounding text-message based responding.

Using SMS text messaging for teaching and data collection in the behavioral sciences

Mobile phones – also known as cellphones – are increasingly considered as more than simply devices that allow people to talk to one another. Handsets themselves have evolved to have many capabilities that traditional wired phones do not: SMS text messaging, Wireless Application Protocol (WAP) mobile Internet, camera, Bluetooth, Global Positioning System (GPS), along with the capacity for running third-party applications, tools and games. Concurrently, applications have been developed to exploit these capabilities, allowing users to check email or read news headlines over WAP; use the GPS system to locate nearby shops and restaurants, and receive SMS-based alerts for sporting or financial breaking news.

Recently, mobile phones have started to be used for education and research. For example, Motiwalla (2007) discusses a framework for 'm-learning' (mobile phone-based learning), a subset of e-learning, as a way of enhancing traditional classroom-based teaching, noting that there are several ways in which mobile phones can be used to enhance teaching. These include housekeeping – reminding students of upcoming lectures or deadlines, and aiding scheduling and time management; forums – allowing students to interact with each other to discuss aspects of the course; and tutorials – where students can either be reminded of principles discussed in lectures, broaden their knowledge beyond lecture material, or conduct animated 'virtual practical classes', for example in engineering, using an interactive simulation of a particular system or principle to build intuition. These applications share the properties of delivering content to students outside normal teaching situations. During lectures and practical classes there is the possibility of allowing students to interact with the teacher in real time to ask questions, to give feedback on whether, say, the difficulty of the class is

appropriate, or to vote on, say, which topic to move onto next. It is this final application that we concentrate on, because in the behavioral sciences there is a unique scope for not just using mobile devices for m-learning, but for 'm-research': running mobile-based psychology experiments on students, either to gather new data, or to demonstrate some of the psychological principles being taught.

We have recently demonstrated that reaction time data can be collected from experiments run on mobile devices using Flash Lite (Reimers & Stewart, 2008), in much the same way as using Flash in a standard web browser (Reimers & Stewart, 2007). Others (e.g., Shephard, Kho, Chen, & Kosslyn, 2006) have used PDAs as input devices for behavioral research. Although this approach has the scope for running a sizeable subset of experiments that are traditionally run on computers, the technology – i.e., having a mobile phone that can run the appropriate software – and skills required to use it is not yet ubiquitous enough to run experiments in class without excluding a large number of participants.

So here, we focus on a much more ubiquitous capability of a mobile phone – SMS text messaging. All mobile phones have text messaging capabilities, and we have not yet met a student who does not know how to send a text message. SMS text messaging has been used in the behavioral sciences in the past, to prompt people at random times to sample their environment (Hogarth, Portell, & Cuxart, 2007), or to pair participants to allow them to play game-theoretical games in economics classes (Cheung, 2008).

Thus, in other disciplines, teachers can use SMS messaging to get feedback about students' enjoyment or understanding of the course; in psychology classes teachers can not only do this, they can also run novel experiments on students, who respond to stimuli presented in class using text messaging, and can even randomly allocate participants to conditions by sending different students different information on their mobile phones, as we

do below. This means that the teacher can both collect novel data, giving students the sense of being involved in real research; and show, rather than simply tell, students the setup and results from ‘classic’ psychology experiments.

We now describe the architecture of a system we have put together to process and respond to input from participants. Later, we give two concrete examples of experiments run using this architecture All the computer code used for these systems is available from the lead author. From the outset, we want to make it clear we are not claiming that this is the definitive way of running SMS-based experiments. We are primarily psychologists, not developers, and this may be reflected in the approach we took. However, this is what we found to be the simplest, most time- and cost- effective way to set up SMS-based responding, and we hope it will allow others to improve on it. Where we are aware of other alternatives to the architecture we used, we have briefly described them, which we hope will allow the reader to decide which approach is likely to work best for them.

SMS Architecture

At its most simple, what we need is a way to process SMS messages sent by participants using our own code, to store and collate responses, and finally to send SMS messages back to the participant. We therefore break this into three components, getting messages in, processing, and sending messages out. The overall architecture we discuss is shown in Figure 1.

Getting Messages In

The first stage is to take SMS input and convert it into a format that allows it to be processed by the code we have written. There are several ways in which this can be done. One way is to purchase a mobile phone and attach it to one’s own computer using a wired connection (e.g., Elliman, 2006). Participants send a message to the number associated with

the sim card, messages are delivered to the phone, and then uploaded from the phone to the computer. This means that there are no third parties involved, and the computer doing the processing does not require internet access. Using a direct connection between the cellular network and a computer than can process incoming SMS messages may become an increasingly viable way of linking computers to mobile phone input as the blurring of boundaries between computers and mobile devices develops. Already many mobile devices run windows software, and some notebooks come with the built-in ability to connect to 3G (mobile phone) networks.

One alternative approach might be to use a service like O2's Bluebook, a free web-based interface for accessing all messages sent to an O2 user's number. This is designed – among other things – as a back-up in case the user loses their phone, but in theory it could be possible to strip out messages automatically by repeatedly accessing and parsing the web-page as messages come in.

The approach we use is probably the easiest for psychologists who are relatively proficient programmers of web-based applications, and who want to avoid dealing with new software or purchasing hardware. There are hundreds of companies that run *message relay services*, which forward all messages sent to a specific allocated number to a user-specified script – generally written by the researcher in php or perl – using HTTP GET or POST methods. We discuss UK-based services, but a web search for 'SMS to HTTP' will give many options for most countries.

Thus, the researcher signs up for an account, and specifies the URL to which messages are to be relayed, e.g., http://www.reimers.co.uk/sms.php. We are not going to describe the GET or POST methods in detail now – they are widely-used methods that are used for passing data between web pages and applications, and are most commonly used for submitting HTML

forms. It is not necessary to understand the methods themselves for the purposes of running SMS experiments, except to be aware that the methods transfer variable names and their values to the data-handling script, and the variable values can be directly addressed by the script. The three different message relay services we have tried each transferred slightly different variables in different ways, but all, obviously, forward variables that hold the number from which the message originated, and the body of the message. Some also relay time and date variables, however, we will only consider the originating number and the message body. It should be noted that different services use different variable names for a given variable (thus, the variable containing the number from which the message originated might be 'orig' for one service, and 'incoming' for another, so code must be specifically tailored for a particular message relay service). There is also a difference in cost: one service we have used (sms2email.com) allows one to specify a keyword, and all messages sent to a common number that begin with the keyword are automatically, free-of-charge, delivered via HTTP POST to ones own server. The other two systems we have tried offer a dedicated number costing between $15 and $60 a month, and all messages sent to that number are delivered via HTTP GET or POST methods, without additional cost. We have found no correlation between cost and quality of service in the three systems we have tried.

Processing data

We next move to the content of the data-handling script that residing on our server, which handles GET or POST requests. We used php, which is a popular dynamic scripting language, which interacts easily with mySQL databases. For an introduction to php and mySQL, we recommend the training videos available on lynda.com, which have helped us. In our architecture, the data-processing stage can be broken down into three components. As this is not intended to be a tutorial, we do not discuss the precise coding in detail. Full versions of

the code we have used are available from the authors. A sample of php code we used to process input, interact with a database, and prepare output can be found in the Appendix.

Read variables. As we mentioned above, different relay services use different variable labels. In our system the variable containing the incoming number was Originator, sent using the GET method, so at the start of the php script we could access the incoming number by reading the value of the GET superglobal: $_GET['Originator']. Similarly, the body of the message is held in $_GET['Body'].

Interact with database. Next, a mySQL database is queried. The exact query will depend on the experimental design, and there are many sites on the web that offer introductions to using php to interact with databases. Generally, the first query should be an attempt to SELECT, based on the originating phone number, to see whether the participant has already registered or submitted a response from that phone. Next, if no record exists, an INSERT command is used to create a new record based on the originating phone number, or if a record exists, UPDATE is used to change values inside it.

Prepare output. Once the database query is complete, and the connection closed, the body of a reply message needs to be generated. This may be an error message based on the incoming message ‘Could not parse your input’, or on server-side technical problems ‘Could not query database’. More likely, it may simply be a message confirming safe receipt and successful processing of the incoming message ‘Successfully registered’. Finally, it may be something more complex that takes output from a mySQL query to generate bespoke text for a particular user ‘You are participant number 38’. We give an example below.

Sending messages out

Once the desired output text is held in a string, a message needs to be sent containing that text. Here, very little needs to be done – all the message relay services we have tried

supply their own php code to handle sending a message, and the experimenter just has to specify the number to which they wish to send the message, and the message content, as well as the experimenter's username and password for billing purposes. The procedure normally used is either an HTTP GET/POST method, or a SOAP-based protocol, but as before it is not necessary to understand precisely how it works, except at the highest level to understand that the message content specified will be sent as an SMS message to the number specified.

Two Classroom Experiments

We now give two concrete example of how the architecture we introduced here can be used to run experiments in an introductory class that focused on decision making. Both experiments were run with four classes, each of around 25 students. The experiment, when submissions were processed properly (see below) took around 10 minutes for the class to complete. In both these experiments participants were reimbursed for the cost of the text messages they sent.

Second price auction and anchoring.

We wanted to demonstrate the phenomenon of anchoring, where participants' estimates of a particular value are affected by initial irrelevant 'anchors'. In a classic example, Ariely, Prelec and Loewenstein (2003) ran an experiment in which participants indicated the amount they were willing to pay for items like computer equipment and bottles of wine. He first asked participants to write down the last two digits of their social security number (a random anchor), and then asked them to indicate whether they would be willing to pay that two-digit amount in dollars for the item under consideration. Finally, participants were asked the maximum they would really be willing to pay for the item. Participants stood a real chance of buying the item, a probability which increased with their offer size. Willingness to pay was significantly correlated with the social security number anchors.

Clearly, this approach needed assistants to collect in participants' responses, and input the data from all participants to look for correlations. This seemed like a perfect experiment to adapt for using mobile phones. The system we set up was as follows. Participants were introduced to the notion of a second price auction, in which they would all participate. The design was similar to ebay auctions: Bids were kept secret, and the winner paid the price of the second highest bid. An item – in our case a University of Warwick branded scarf – was brought out for inspection. Participants were asked to compose a text message that started 'BID' followed by the last three digits of their phone number. Then we asked participants if they would be willing to pay that amount in pence for the scarf, and add 'y' or 'n' to their message to indicate whether they would. Finally we asked them to finish the text message with the amount in pence that they wished to bid for the item, and send it to a number on the projector screen. This may appear complicated; however, we needed the keyword 'BID' at the start to indicate which experiment participants were completing. Several experiments, connected to different databases, were active at the same time, using the same phone number, so it was important that submissions could initially be identified and send to the appropriate piece of code for processing. Although we also automatically recorded the participant's mobile number, we made them type in the last digits in their message to maximize the chances of their using the digits as an anchor for their actual bid.

The message relay service forwarded all the sent messages to our php code, which split the messages into keyword – 'BID', hopefully – last 3 digits of phone number, y/n willingness to pay, and actual bid. We also recorded the last digits of the participants' phone numbers using $_GET['Originator'] to check they knew their own number. The php opened a mySQL database and checked whether a record with an ID matching the last digits of the participant's phone number was already present. If it was, participants received the message

'You appear to have already placed a bid. If you haven't, please ask for help.'. Next, our code checked that the last three digits of the phone number was indeed an integer, and that y/n was actually a 'y'or an 'n' (case insensitive), and finally that the bid was also an integer. If these were correct, it created a new entry in the database, with id number based on the last digits of the phone number, and fields for 'y/n' and, crucially, the actual bid. It then generated a message saying 'Bid successful, good luck.' which was sent back to the participant.

In parallel, we had created a dynamic php webpage, which directly pulled the contents of the bid database and displayed it as a standard html table, ordered by bid value, which also contained a column for the last three digits of participants' mobile. Thus, at any point we were able to see the highest bids, and once we closed the auction, by adding a record to the database as a marker for our php code to reject any further bids, we could immediately display the auction results to the class, and ask the winner to come forward and collect – and pay for – their scarf. We could also copy and paste the table into a ready-made excel sheet, which created a scatter plot of the bids against last digits of phone number. Unfortunately, our data were not as clear-cut as those of Ariely and colleagues, but the students were at least able to see very quickly that the experiment had failed.

Context and delay discounting

The second experiment used a more conventional between-subjects manipulation to investigate the effects of context on time preference. We, and others, have found that people's risk and time preferences are, rather than being stable as many economists would predict, malleable, and strongly influenced by what should be irrelevant contextual factors (e.g., Stewart, Chater, Stott & Reimers, 2003; Stewart & Reimers, in preparation; Robles & Vargas, 2007). Previous research has manipulated choice set or inter-trial context. Our aim was to investigate whether time preference could be manipulated by simply getting participants to

think about different delays in an irrelevant task.

Participants were told they would make a series of 6 binary choice of the form 'Would you rather have £15 in 4 days or £40 in 120 days?', which would appear on the screen at the front of the classroom, and that one participant would receive one of their choices selected at random for real. The winner would be chosen by a tie-breaker question – whoever was nearest to the correct answer would win. Unknown to the participants there were two versions of the tie-breaker question, one which asked about a time that fell between the delay for the immediate and delayed options ('How many days is it until the start of the Christmas holidays?') and one which occurred after the delayed option ('How many days is it until the start of the summer holidays').

The experimental setup required participants to send the message 'REGISTER' to the number we displayed on the projector screen. If the message was indeed 'REGISTER', the php script took the last digits of the incoming phone number and checked if a record existed already. If not, it added a new record. If the total number of records in the database was odd, then participants received a message 'Registration successful. The tie-breaker is: How many days is it until the start of the Christmas holidays?'. If it was even, participants received the message 'Registration successful. The tie-breaker is: How many days is it until the start of the summer holidays?'. Participants were instructed to reply to that text with 'SUBMIT' followed by their answer to the tie-breaker, followed by their choice for six questions, responding 'A' if they wanted the smaller-sooner sum, and 'B' if they wanted the larger-later sum. Thus, the message they sent was of the form 'SUBMIT 55 A A B B B B'. If the message received by the php code started with 'SUBMIT', the script first queried a mySQL database to check that the incoming number was already registered, then attempted to parse the rest of the message, and, if successful, wrote the responses to the database and sent a message confirming successful

submission back to the participant.

As with the auction experiment, we had a dynamic webpage in a browser that used php to query the database, and by refreshing the page, we could monitor participants' incoming responses. We also had a slightly more complex page which took participants' responses and collated them, displaying the proportion of participants in each condition who chose the smaller-sooner sum over the larger-later for each binary choice. Once the experiment was complete, we showed participants the webpage containing the collated results. This experiment also appeared to work, and participants could see that those who had been in the shorter-delay tiebreaker condition were more likely to take the smaller-sooner sum than those in the longer-delay tiebreaker condition.

Feedback

We ran a total of three practical class sessions using mobile phones, in which participants completed the second price auction and delay discounting experiments, along with an ultimatum game experiment, which was a simplified version of that used by Cheung (2008). We also had participants complete a short online survey about their mobile phone habits and their evaluation of the approach.

Our experience

Our main finding was that we had severe problems with the message relay system. Although it worked perfectly when testing the system architecture before running the experiment in class, we found that when tens of participants tried to submit responses at the same time, the message relay service would be unable to connect to our server's script. Although it is possible that our programming caused the problem, we have recently heard from another UK-based researcher, Terry McAndrew, who encountered similar problems with a message relay service processed using a .net framework (McAndrew, personal

communication). Like us, he found that the system worked well during development and testing, but when large numbers of participants responded, messages took up to half an hour to come through. This suggests that it may be a more general problem, and it is possible that spending more money on a more robust service may improve relay times.

In our case, the upshot was that messages were instead eventually relayed to our email inbox, and we had to click the URL link that was part of the message (containing the GET variables) manually for all but the first few participants. This meant that participants sometimes had to wait up to an hour to receive a submission.

The second finding is that it is important to be precise in specifying how messages are to be sent, and to use parsing algorithms that anticipated deviations from the instructed format. We found that participants often added units to the amounts they were sending '55days' rather than '55' in the discounting experiment, and '230p' rather than '230' in the auction. Similarly, participants often wrote 'YES' rather than 'Y' in the auction experiment. If the php script could not parse the input, it sent the participant a message back saying so, but for future research we suggest attempting to anticipate what participants may actually sent rather than what we tell them to send.

Participant feedback

The short survey that we asked participants to complete comprised two parts, one about participants' phone ownership and use, and one evaluating the use of text messaging in the class. Of the 21 participants who completed the survey, 100% owned a mobile phone, a total of 71% sent at least 100 messages a month, and only 5% sent fewer than 10 text messages a month. Thus, our student population appears to have the hardware and the skills to be able to participate in experiments using mobile phones.

We also asked participants how easy or hard they found responding using text

message, versus traditional paper-and-pencil experiments. The results are in Figure 2. The majority found it no harder to respond using text messaging (although there was no evidence that they found it easier either). We also asked how they would prefer to respond in future experiments (Figure 3): 38% preferred the idea of responding by text message, with a further 23% indifferent between the two. Thus, although the sample size prevents us from drawing strong conclusions, we found no evidence that participants actively dislike responding using text messaging. We also note that these results may underestimate real the perceived ease and preference for SMS messaging because of the technical problems we encountered here. We asked participants to ignore these, and imagine the system as it was intended to work, but people's free-text feedback suggested that they were thinking about the technical problems.

Wider applications

So far, we have concentrated on classroom-based scenarios for using text messaging in psychology, largely because our focus here has been on teaching. However, for pure research, there is wider potential outside the lab. It may be possible to run simple psychology experiments using, say, newspaper or magazine adverts in which people are asked to text their responses to a series of questions, choices or scenarios printed in the paper to a particular number, where the data are stored. It might even be possible to run different versions of tasks in either different editions of a newspaper or magazine, or in different editorial contexts. Similarly, it might also be possible to run short experiments just by putting up a poster prominently in a psychology department, with a number of questions to which participants have to respond using a text message string.

Finally, it is clear that there are many limitations of SMS text messaging – it takes time to enter and send text, and there is no precise control over the timings with which simuli are presented; it can only capture text; and it not very ergonomic for studies with more than a

handful of questions. However, it is possible to uses SMS messaging to effortlessly direct participants to, an online Flash Lite-based experiment, using a WAP Push response. We have recently set up a system where a participant sends the message 'GO' to our number. They are then sent a WAP Push message back, which contains a link to the Flash code. Agreeing to open the link automatically opens a WAP connection and downloads the Flash experiment, which can then save data across the web, like standard web-based experiments. At the moment, not enough students have phones that run Flash Lite, so we have not used it in classroom experiments, but we suspect that, within 2-3 years, Flash Lite will be ubiquitous enough for us to start running full reaction-time experiments using this architecture.

References

Ariely, D.,Loewenstein, G. & Prelec, D. (2003) Coherent arbitrariness: Stable demand Curves without stable preferences,*Quarterly Journal of Economics, 118,* 73-105.

Cheung, S. L. (2008). Using mobile phone messaging as a response medium in classroom experiments. *Journal of Economic Education, 39,* 51-67.

Elliman, D., G. (2006). *A System to support Interactive Teaching in the Lecture Theatre.* Retrieved November 25, 2008 from http://www.ics.heacademy.ac.uk/resources/rlos/elliman/FeedbackProj.pdf.

Hogarth, R. M., Portell, M., & Cuxart, A. (2007) What risks do people perceive in everyday life? A perspective gained from the experience sampling method (ESM). *Risk Analysis, 27,* 1427-1439.

Motiwalla, L. F. Mobile learning: A framework and evaluation. *Computers and Education, 49,* 581-596.

Reimers, S., & Stewart, N. (2007). Adobe Flash as a medium for online experimentation: A test of RT measurement capabilities. *Behavior Research Methods, 39,* 365-370.

Reimers, S., & Stewart, N. (2008). Using Adobe Flash Lite on mobile phones for psychological research: Reaction time measurement reliability and interdevice variability. *Behavior Research Methods, 40,* 1170-1176.

Robles, E. & Vargas, P. A. (2007). Parameters of delay discounting assessment: Number of trials, effort, and sequential effects. *Behavioural Processes, 78,* 285-290.

Shephard, J. M., Kho, S., Chen, J., & Kosslyn, S. M. (2006). MiniCog: A method for administering psychological tests and experiments on a handheld personal digital assistant. *Behavior Research Methods, 38,* 648-655.

Stewart, N., Chater, N., Stott, H. P., & Reimers, S. (2003). Prospect relativity: How choice options influence decision under risk. *Journal of Experimental Psychology: General, 132,* 23-46.

Author Note

Stian Reimers, University College London, and Neil Stewart, Department of Psychology, University of Warwick, Coventry, UK.

This research and Stian Reimers were supported by a fellowship from the ESRC Centre for Economic Learning and Social Evolution (ELSE). Neil Stewart was supported by ESRC grants ESRC RES-062-23-0952, ESRC RES-000-22-2459, and ESRC RES-000-23-1372. Correspondence concerning this article should be addressed to Stian Reimers, Department of Psychology, University College London, WC1H 0AP, UK. E-mail may be sent to s.reimers@ucl.ac.uk.

Figure Caption

Figure 1. General architecture for our SMS-based response system.

Figure 2. Participants' responses on a 5-point scale for how easy it was to use the SMS-based system versus pencil and paper.

Figure 3. Participants' responses on a 5-point scale for how much they would prefer to use the SMS-based system versus pencil and paper.

Figure 1

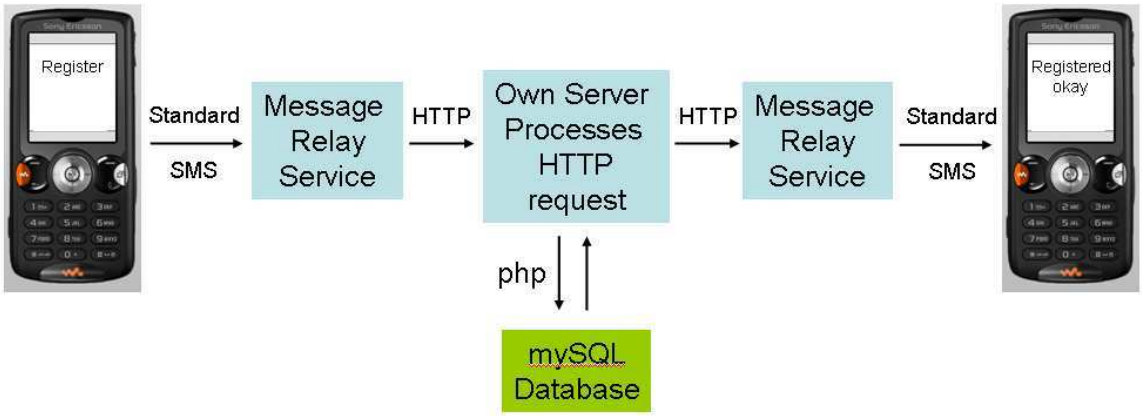

Register
Standard
SMS
Message Relay Service
HTTP
Own Server Processes HTTP request
HTTP
Message Relay Service
Standard
SMS
Registered okay
php
mySQL Database

Figure 2

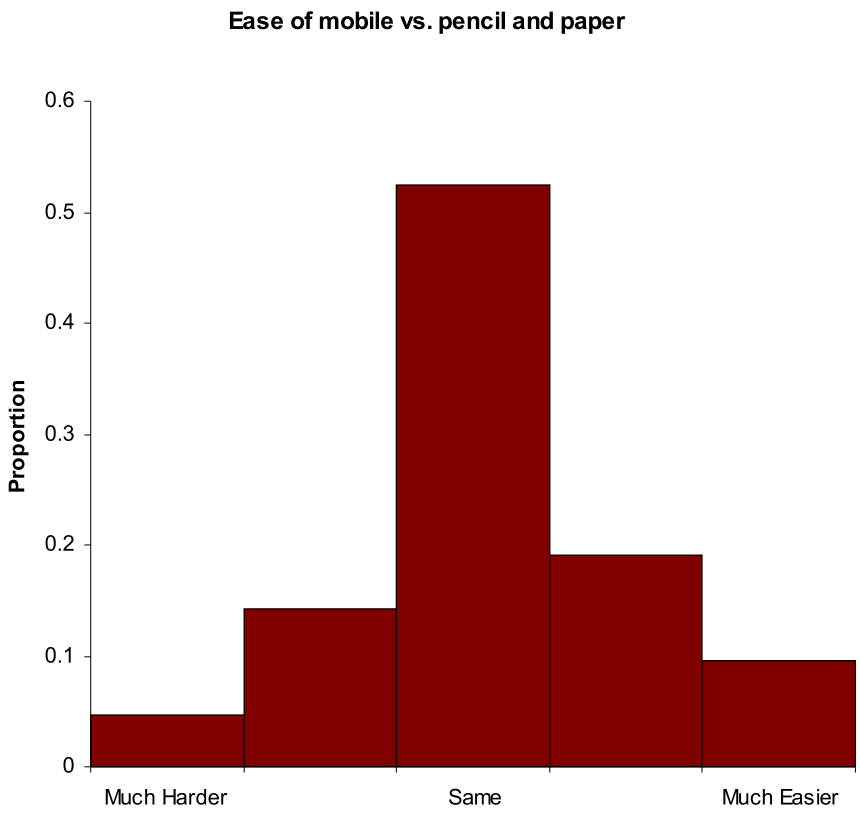
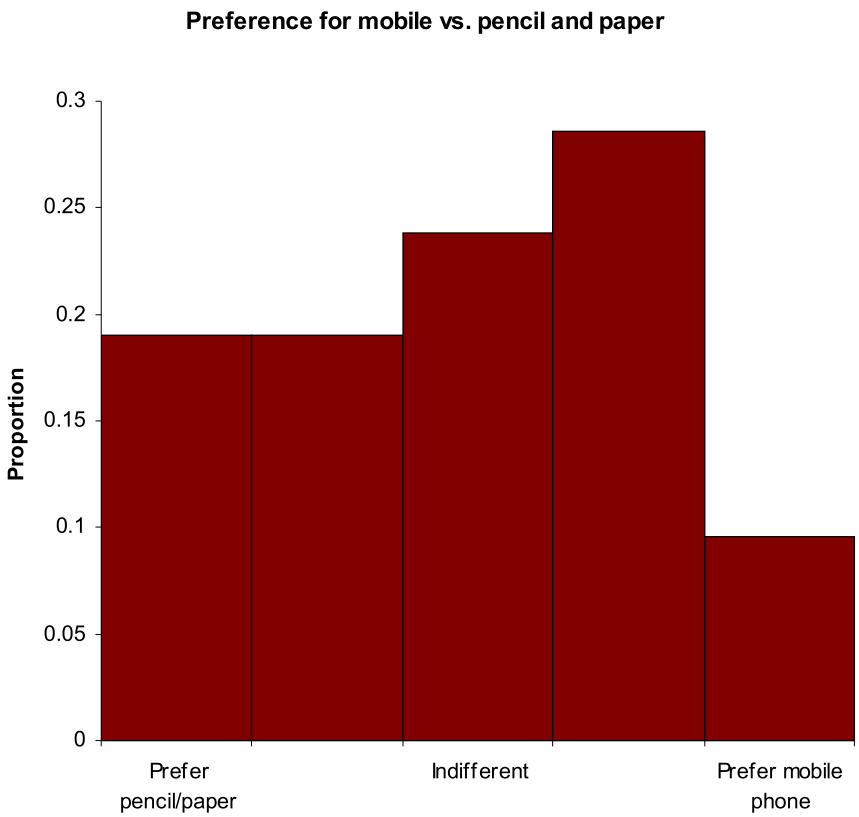

Ease of mobile vs. pencil and paper
Proportion
0.6
0.5
0.4
0.3
0.2
0.1
0
Much Harder
Same
Much Easier

Figure 3


Preference for mobile vs. pencil and paper
Proportion
0.3
0.25
0.2
0.15
0.1
0.05
0
Prefer pencil/paper
Indifferent
Prefer mobile phone

APPENDIX

Example php code for processing incoming SMS messages

```
<?php
//Here, we're processing data that has used the GET procedure
//to pass variables Originator=incoming mobile number, and Body=message text
…
//This is the message that we send back to the participant
$err="";
//Check if both are present, and if so, assign them to variables
if (isset($_GET['Originator'])) {$incoming=substr($_GET['Originator'],-5,5);} else {$err="No Number";}
if (isset($_GET['Body'])) {$conttemp=$_GET['Body'];} else {$err="No Message";}
//Make everything upper case
$cont=strtoupper($conttemp);
//Split message content variable, space delimited, into different variables.
list($procedure, $param1, $param2, $param3, $param4, $param5, $param6, $param7) = explode(" ", $cont);
//connect to database server
$connection = mysql_connect(DB_SERVER, DB_USER, DB_PASS);
//check whether connected okay
if(!$connection) {
    $err="Could not connect to database server. Please ask for assistance.";
} else { //connect to a particular database
    $db_select = mysql_select_db(DB_NAME, $connection);
     if(!$db_select) {
         $err="Could not connect to the database. Please ask for assistance.";
      } else {  //if we are here, then we've connected to the database okay
                 //now check what the first word in the message was
          if(!$procedure) {  //$procedure variable is not set
              $err=" Your message was incomplete, or did not contain the correct spaces.". $_GET['Body'];
          } else {
               if($procedure=="REGISTER") {  //randomly seed random number generator
                   srand(time());
                   //choose a random number 1, 2, 3, or 4
                   $random = (rand()%2)+1;
                   //Select all rows in database that have a num_id the same as the incoming phone number
                   $query= "SELECT * FROM SMS WHERE num_id = {$incoming}";
                   $existingrows = mysql_query($query);
                   //$numm is the number of rows selected by this procedure. Should be 0 if not already registered
                   $numm=mysql_num_rows($existingrows);
                   if(!$existingrows) { //means query did not execute
                       $err = "Could not query database called SMS. Please ask for assistance.";
                  } else if($numm==0) { //means query executed okay, no rows returned
                   //create a new row, with element num_id set to incoming phone number, and ran_condition to the
                   //random number just generated (so we know what condition the participant is assigned to
                       $query = "INSERT INTO SMS ( num_id, ran_cond) VALUES ('{$incoming}', $random )";
                  //check that it inserted the record okay, then set $err to one of four different statements.
                       if (mysql_query($query)) {
                          $err = "Successfully registered. Your question is:";
                          else if ($random==1) { $err.=" How many days is it until the official last day of this term?";}
                          else if ($random==2) { $err.=" How many days is it until the summer holidays begin?";}
                          else { $err.=" unavailable. Please ask for help.";}
                       } else {
                         $err = "Could not register you. Please ask for assistance.";
                       }
                  } else {
                       $err = "Number already registered, please ask for assistance.";
                  }
          }
...
?>
```