Daniel Schwartz
CS 123: Rogers
6-12-14

WHO STAYS?
UNDERSTANDING USER RETENTION ON KHAN ACADEMY

Design

My goal for this project was to explore the process of user retention and attrition on Khan Academy and to ideally identify any "low-hanging fruit" strategies that Khan Academy might take to improve its user retention.

Some users "drop out" because even a perfect version of Khan Academy (whatever that would mean) would have little to offer them (they know all the available curriculum, or they are not interested in or ready to learn), and we shouldn't try to keep them. Some users drop out because the content itself (i.e. videos and problems) isn't clear or engaging enough to hold their attention, but improving content is probably not a "low-hanging fruit". In other cases, though, the users are interested and ready to learn, and find the learning materials adequate, but they drop out because they're interacting with the content ineffectively — maybe studying the wrong topics at the wrong time, or not watching the videos thoroughly, or spending too little time on problems, or working in a less efficient mode (e.g. mastery vs. normal). And these are the users we can do something about!

In order to understand retention from the data, most of which is problem-solving and video-watching logs, I needed to aggregate user activity over different time periods, which I planned to do through MapReduce. There are on the order of 10 million users (non-parent, non-teacher, age 13+), so the full time-aggregated data would be roughly small enough to analyze in memory in R on a single machine (10 observations on 10 million users = 100 million rows in a data frame, for ~ 5GB). In other words, the analysis phase could reasonably take advantage of multiple threads/processers as necessary but not multiple machines. Because I spent a lot of time failing to set up a Hadoop cluster on Google Compute Engine, in the end I only had time to conduct the analysis on a subsample of users — slightly over 30,000. While I have good longitudinal data for these users, I am not necessarily able to track them from their first time as Khan Academy users so the sample analysis is more perhaps more appropriate to "current" user retention than new user retention.

Implementation

Because the raw log data (~400GB) are stored in Google Cloud Storage, I wanted to use Google Compute Engine (an IaaS, or Infrastructure as a Service, product) to implement my MapReduce code. Although Google provides public code to set up a Hadoop cluster on Compute Engine, the details were too involved for me to figure out in time, and it would have made more sense to move the data to RCC Midway. However, because I will likely continue with this project in the future, I will discuss how my MapReduce code functions.

In particular, my MapReduce program (kaMapReduce.java) takes the problem logs and outputs different time-interval summaries for each user, using an implementation somewhat similar to Task 4 of our first problem set this quarter. The mapper reads in each problem log, parses and simplifies the log, and outputs the user ID as a key and the relevant data (timestamps, correct or not, number of attempts, number of hints used) as a value. The reducer obtains all of the problem logs and sorts them by timestamp to find the user's join date and then based on that computes the aggregate data for each time interval and outputs this as the value to each user ID key. The time-trend analysis in R requires "stacked" data, where each row is a user by time-interval observation, but it is much more efficient to process the raw data into a "flat" data set, where each line is a user with all the different time-interval data (so the user ID, a fairly long string, is not repeated 10 times per user in the output file). Flat data can be fairly easily reconfigured into flat data in R using the melt and cast methods of the package reshape2. As a side note, this MapReduce task unfortunately cannot use a combiner because the reducer need to have all problem logs in order to determine the true join date and categorize logs into time intervals accordingly.

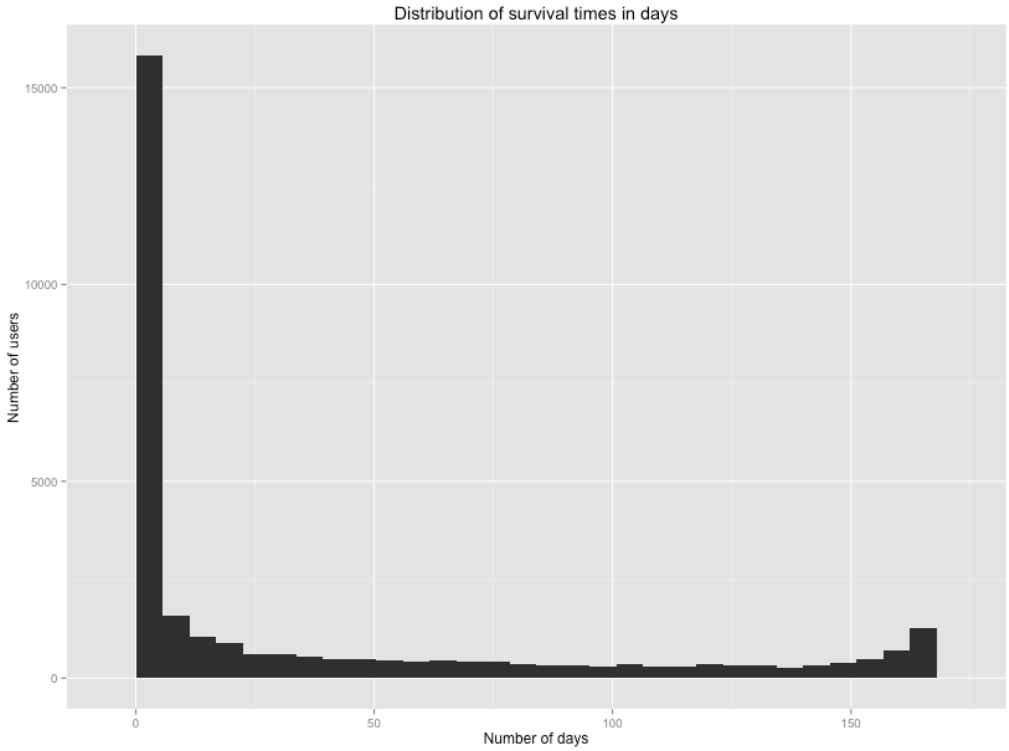
To let the sample data in R be reasonably large I used the data.table class, a powerful extension of R's canonical data.frame allowing for significantly faster joins and queries. In fact data.table uses syntax analogous to SQL, but it can handle any R expression in its select statement so there is great opportunity to do make complex queries in relatively short coding and processing time. R data.tables are also sorted on their keys so that rows with the same key are contiguous in memory, which considerably speeds up the sort of aggregation I needed to do. I processed slightly over 2 GB of raw data in my R-based sample analysis, but my code performed quickly enough that I probably could have scaled the data up much closer to memory capacity without really being inconvenienced. And though unnecessary for the scale I decided to work at,

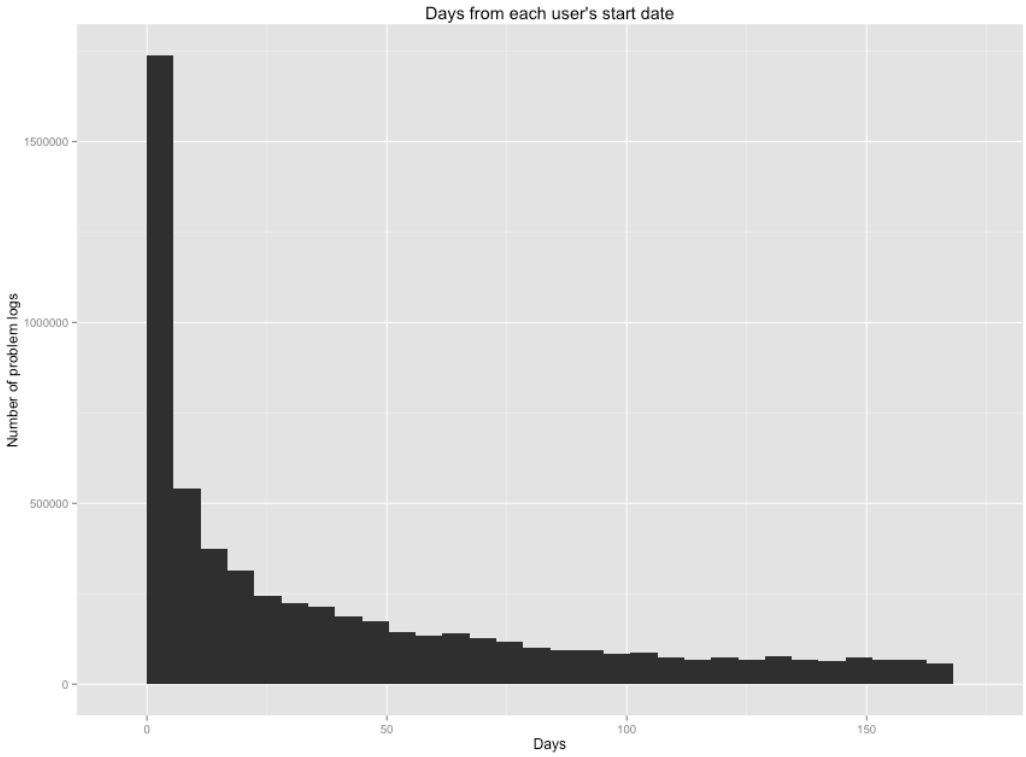
I also wrote a compression script in R (compress.R) to reduce the size of the data by approximately two-thirds.

In terms of the statistics, I used a small set of discrete-time survival analysis models, which can be implemented as generalized linear models easy to compute using base R. These are models that estimate the probability of someone not surviving past a certain time interval. If I knew about an ecology of users (for instance, if I was only looking at kids who use Khan Academy in their classroom and I knew who their teachers were) then slightly more complex models including random effects would be appropriate, but I have no such data. I used the package ggplot2 to implement graphics and the convenient gridExtra package to produce tables.

Results

As I mentioned earlier, although my analysis is based just on a proportionately small subset of the data I believe I have been able to produce legitimate insights relevant to the "population" of Khan Academy users at large. In particular, I was worried that my sample of the data (which just consists of the first three problem log files) would only contain brief periods of user activity. In fact, the median number of problems per user in this sample is 48, the mean 220, and the maximum 23,770. The histogram below suggests that this distribution is skewed appropriately as it might be in the population, where would expect many users to contribute a small number of problems each and gradually fewer users contributing a greater and greater number of problems each. Additionally the distribution of the length of time that users remain in the sample is also reasonable (median 4 days, mean 38), as we see below. The slight pooling at the end of the distribution is because of censoring at the time that data was collected (these are people who will probably drop out eventually, but haven't dropped out yet).





Moving along, we can the model results. In the main model of this preliminary analysis, I estimate effects of each tie interval, doing twnety or more problems on the first day, getting more

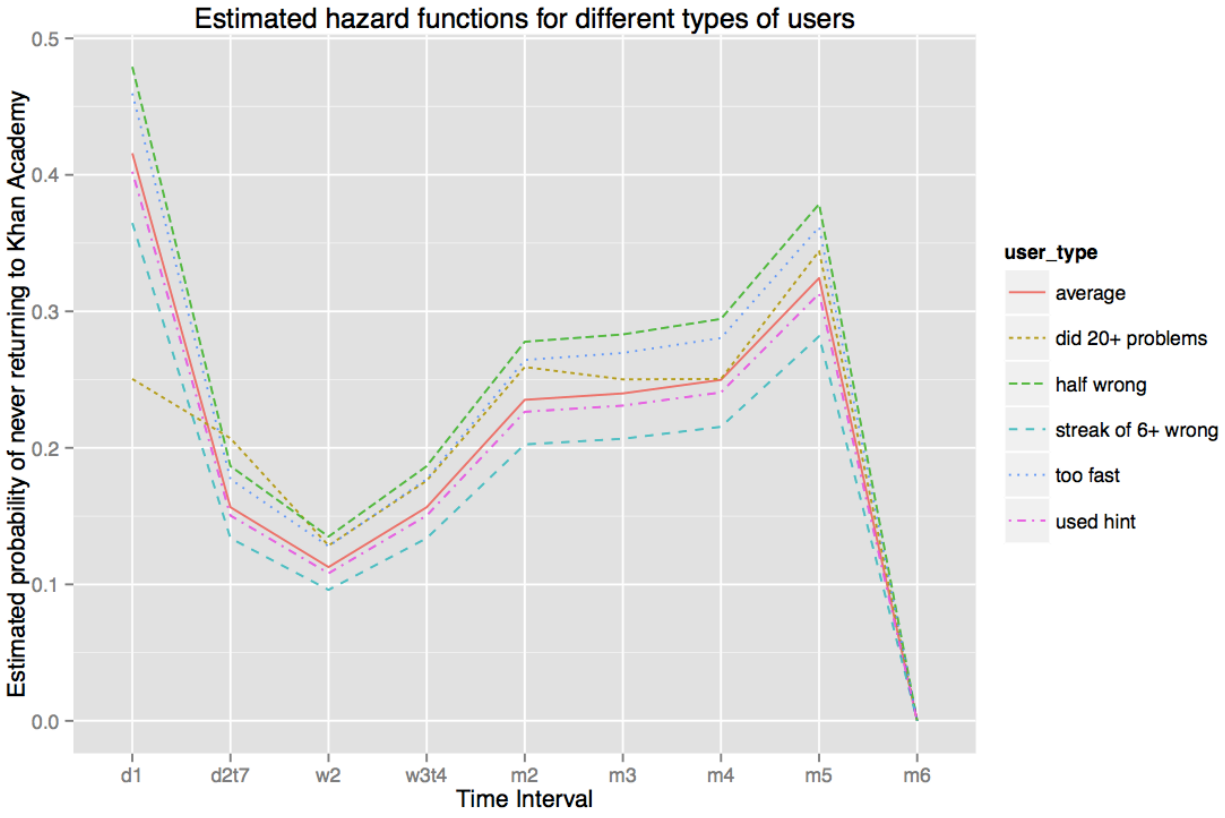
than half the problems wrong on the first day, having a streak of six or more wrong answers on the first day, answering questions in under 15 seconds on average on the first day, and using a hint on the first day. Here all parameter estimates are on a complementary log-log scale (necessary as a link function from the binary outcome to a vector of linear effects that can take on any value in the real line) so we can transform to see more intuitive results. Exponentiating once for the coefficients on our binary predictors, we have the hazard ratios in the table below. A hazard ratio describes the relative risk that a member of the first group (coded as 1) experiences the outcome, as compared to someone in the second group (coded as 0).

For instance, my model estimates that solving problems on your first day in under 15 seconds on average makes you 15% more likely to leave Khan Academy within six months than those who take more the 15 seconds per question on average on their first day. In other words, we see rather large negative effects on retention of working too quickly and getting more than half your problems wrong. Somewhat surprisingly, having a wrong streak of six or more reputedly decreases the chances of attrition by 16%, but this effect is likely confounded with the fact that most users will do a fair number of problems before getting six wrong. Using a hint on the first unexcitingly reduces the rates of attrition by 4%. Last we have doing twenty or more problems on the first day, which notably my model allows to have different effects at different time intervals.

Hazard Ratios for Dropping out of Khan Academy

	Did 20+ problems	More than half wrong	Streak of 6+ wrong	Too fast	Used a hint
Days 2–7	1.36	1.21	0.84	1.15	0.96
Week 2	1.15	1.21	0.84	1.15	0.96
Week 3–4	1.14	1.21	0.84	1.15	0.96
Month 2	1.12	1.21	0.84	1.15	0.96
Month 3	1.05	1.21	0.84	1.15	0.96
Month 4	1.08	1.21	0.84	1.15	0.96
Month 5	1.08	1.21	0.84	1.15	0.96
Month 6	1.05	1.21	0.84	1.15	0.96

If we take a look at a plot of the actual hazard rates (below), we can see that users have the highest probability of attrition very early on, but that this probability rises again to the Month 5 mark. This suggests that user attrition in this sample is actually two separate processes — first, many new users decide that Khan Academy is not interesting or useful or relevant to their lives, and second, more "veteran" users find that it *no longer* is useful (perhaps when they complete a course or finish studying a particular subject).



Looking to the Future

This analysis has identified a number of possible ways Khan Academy might try new strategies for user retention — in particular, making students spend a minimum of 15 seconds on a problem before submitting an answer seems especially promising and simple to implement. The system could also try to make it harder for students to get a large proportion of their problems wrong — for example by suggesting that users with a growing proportion of wrong answers try an easier topic, or by adaptively giving students problems targeted to their estimated

ability level as determined by some model perhaps borrowing from item response theory (though this last alternative is admittedly less of a "low-hanging fruit").

In future work I am especially eager to extend this analysis to data about lecture-watching and different content areas (I would be surprised if users studying all subjects drop out in the same patterns). When I replicate this analysis on the fully reduced data, it will also be interesting to see if Khan Academy has done a better or worse job at retaining users as time has passed. Ultimately I am more concerned with providing litmus tests for new features aimed at improving retention that Khan Academy can experimentally implement and analyze more systematically than I am at making a model that will predict with greatest possible accuracy if a user will drop out at any given time — ultimately I believe that while interesting, this direction is less actionable.