Evolving a learning analytics platform

Ari Bader-Natal
Grockit
San Francisco, CA USA
ari@grockit.com

Thomas Lotze
Grockit
San Francisco, CA USA
thomas@grockit.com

ABSTRACT

Web-based learning systems offer researchers the ability to collect and analyze fine-grained educational data on the performance and activity of students, as a basis for better understanding and supporting learning among those students. The availability of this data enables stakeholders to pose a variety of interesting questions, often specifically focused on some subset of students. As a system matures, the number of stakeholders, the number of interesting questions, and the number of relevant sub-populations of students also grow, adding complexity to the data analysis task. In this work, we describe an internal analytics system designed and developed to address this challenge, adding flexibility and scalability. Here we present several examples of typical examples of analysis, discuss a few uncommon but powerful use-cases, and share lessons learned from the first two years of iteratively developing the platform.

Categories and Subject Descriptors

K.3 [**Computers and Education**]: Computer Uses in Education

1. INTRODUCTION

The purpose of learning analytics is to better understand and more effectively foster learning. The approach is heavily data-driven, as it is based on the collection, analysis, and interpretation of collected educational data. Software-based systems have the ability to easily record and analyze very fine-grained data, and web-based software systems have the added benefit of compiling this disparate data into a single centralized location. This allows for analysis of data in the aggregate, allowing population-wide patterns to be observed and leveraged. Two years ago, we began building one such system for Grockit's web-based collaborative learning platform. Since then, the growth of the platform, the size of the datasets, and the evolving needs of our learners and decision-makers have demanded and driven the continued evolution of this system. In this work, we discuss the iterative development of the tool, with a particular focus on techniques we have used to make it more flexible, powerful, and scalable.

Grockit aims to build a learning system that is simultaneously scalable, effective, and engaging, by leveraging synchronous collaborative learning dynamics and game dynamics [2, 3]. Behind the development of the social and game-based aspects of the system is a strong reliance on data analysis, allowing us to understand and improve on the effectiveness of the platform: Do students who work in groups spend more time on task than those who choose to work alone? Which of the various interventions available within Grockit lead to the largest learning gains? Do peer-awarded points motivate more discussion in small group settings? Are video explanations more effective than written explanations? How does an item response theory model compare to a knowledge tracing model, in terms of predictive response accuracy? How does frequency and duration of discussions during group study sessions differ between high school students and post-college learners? By being able to rapidly ask, answer, visualize, and disseminate findings from these questions, we can inform decisions around the design of an increasingly effective learning environment.

The challenges in creating an analytics platform for a web application – including learner analytics platforms – is in balancing the power and flexibility of an approach with the efficiency and scalability limitations of that approach. As the size of a dataset grows, this balance often shifts.

2. CREATING AN ANALYTICS PIPELINE

After a few weeks of treating each question that we wished to answer as a entirely new analysis project, patterns in the process began to emerge, and we began to create infrastructure to support the most common parts of the workflow. At a high level, the Grockit analytics system is now based on a standard pipeline consisting of (a.) *data collection*, (b.) *selection*, (c.) *analysis*, (d.) *visualization*, and (e.) *distribution*. The goal of *collection* is to instrument the system to record relevant data points for later reference, generally done with application code. The goal of *selection* is to draw together all of the relevant data to answer a particular question, generally done with SQL queries. The goal of *analysis* is to use that compiled data to answer a specific question, most frequently done using the R statistical package [8]. The goal of *visualization* is to create an effective way to convey that analysis, also generally done using R. The

goal of *distribution* is to organize and disseminate specific analyses to only those stakeholders to whom each is relevant, which is done via an internal web-based system in which different stakeholders are subscribed to different self-updating reports.

By creating a common framework for analytics, the ease of adding a new analysis is greatly improved. One can create a new analysis simply by creating two files: an SQL file to retrieve the data and an R file to analyze and display it. Because the infrastructure takes care of setting up the environment, retrieving the data, passing it to the appropriate steps, and collecting and distributing the results, the only thing which needs to be created for each analysis is the part which is different: the actual data retrieval and analysis. The overhead is minimal, allowing us to quickly look at many different analyses.

We can illustrate this workflow with a specific question: Are the difficulty levels of our questions appropriate for the ability levels of our students? Based on what we might find, we would choose to ask our content authors to focus on creating more difficult questions or more easy questions. To answer this, we start with a SQL database query to list each question identifiers, grouped by subject (e.g. Algebra I), along with the primary parameter that we use to characterize the difficulty of that item (it's IRT location parameter, β.) Similarly, we collect a list of all student unique identifiers, along with the ability estimate (IRT person parameter, θ) for the student in that subject. The resulting database tables are accessed through an R script, which is used to calculate and display back-to-back histograms to create a Wright Map of the data [5]. This analysis is then automatically disseminated to the content authoring team and the student modeling team. When we first ran this analysis last year, we found that of the ten domains for which we had developed IRT models, the quantitative questions that we had in place for the GMAT group could better reflect the ability level of the student population. Based on this, our content authoring team subsequently focused on developing additional content at the upper end of the difficulty scale.

The value in creating a common analytics pipeline is that new analyses can be performed rapidly, past analyses can be archived for future access, and common usage patterns can be easily identified and codified.

3. ASKING THE SAME QUESTIONS USING DIFFERENT DATASETS

When Grockit expanded from a single student population (post-college learners studying for a business school entrance exam) to other groups of learners, including high school students, we found ourselves wanting to ask the same questions of different subsets of learners. Beyond simply segmenting students in our database, we started wanting to ask questions based on a variety of other criteria: specific time windows, a particular teacher's class, all students in the treatment group of some controlled experiment, only the questions most recently added to the system, only students who have acted as peer tutors, only students who have worked with a specific instructor, excluding response data from teachers/tutors/administrators/authors, or any number of other data restrictions. In order to avoid constantly

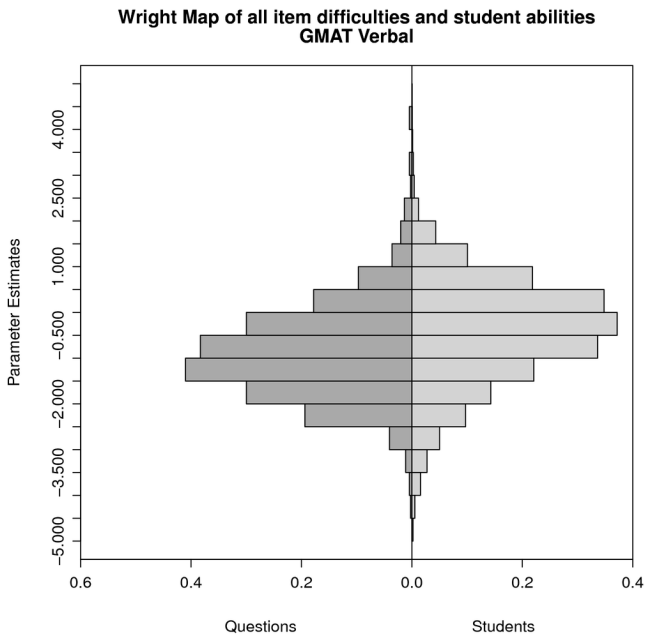

Figure 1: A Wright Map compares the distribution of IRT person parameters of all students to item parameters of all questions in the GMAT Verbal section in Grockit. This plot suggests that the student abilities are slightly above question difficulties, which informed our decision to author more challenging questions for this domain.

modifying (or duplicating) queries, we chose to add support for on-the-fly redefinitions of these sorts of constraints. As the data that we work with is highly relational, the goal was for these restrictions to cascade seamlessly through the model (e.g. if we exclude a particular game, we necessarily wish to exclude all questions answered in that game, any reviews of that game, all explanations read during that game, etc.) Online analytic processing (OLAP) was designed to address this sort of challenge, and the solution that we built shares several qualities with Relational OLAP (ROLAP) systems [4].

We chose to use a lightweight view-based solution, with our MySQL database. Data selection queries were modified to work with (non-materialized) views of the database tables rather than with the tables themselves. These view definitions form a directed acyclic graph (DAG), so most views are defined based on other views. Based on this DAG of non-materialized views, a single view redefinition (such as students who answered questions yesterday) effectively redefines all other views dependant on that one. Records in tables that are dependent on more than one other table are therefore only included in the associated view if all upstream records are considered.[1] In addition to this cascading intersection, a question can be asked of a composition of multiple

[1]For example, a database row recording a student having answering a question is only included in the view of that table if records for both that student and that question are included in views of their tables, respectively.

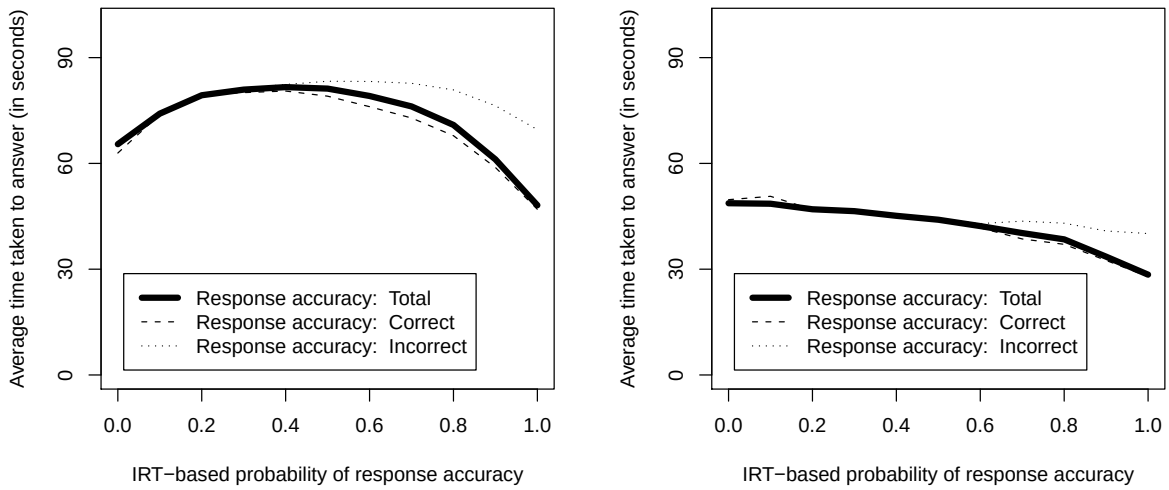

Figure 2: Average time to answer a question, as a function of the IRT-based probability of response accuracy (used as a proxy for subjective difficulty). GMAT Quantitative data on left, ACT English data on right.

non-overlapping view definitions. This means that we can easily ask questions about item responses to Geometry questions among students who have been active during the past 30 days. As these analyses are run on replicas of the application's production database rather than the database itself, long-running queries can process near-realtime data without degrading the performance of the application server.

One question we were interested to understand earlier this year: Do students generally spend more time or less time to answer a question that they find easy? That they find difficult? Does this vary by subject matter? To understand the relationship between subjective item difficulty, accuracy, and time taken, we used our item difficulty parameters to estimate the probability that each item response would be correct. We view this probability as a subjective difficulty metric, based on the assumption that students who have a very low probability of answering a question correctly will find that question difficult, and students who have a very high probability of answering a question correctly will find it to be easy. One notable benefit of separating the question definition from the data definition is that stakeholders can often access the results of existing queries for new subsets of students without requiring any new code to be written. New entries just need to be added to the report-scheduling queue:

```
QUERY=probability_time_accuracy
    VIEW=GameGMATQuantitative+UserStudyingForGMAT

QUERY=probability_time_accuracy
    VIEW=GameACTEnglish+UserStudyingForACT
```

Fig. 2 displays the output of this query for different subsets of data. For GMAT Quantitative questions, students are, on average, spending less time answering questions that are subjectively very easy or very difficult than they spend on questions with are in between, a finding that supports Koster's theory of the connection between challenge appropriateness and learner engagement [7]. Interestingly, a different trend is seen in ACT English responses, where students generally spend less time the easier they find the questions.

With this flexibility, a common use-case for the analytics platform, asking the same question of a different set of data, no longer requires any code changes, and could be made accessible to non-technical stake-holders.

4. RECORDING EVENTS TO ENABLE EXPERIENCE ANALYSIS

We found that some of the questions that we sought to answer required knowledge about events that were not already being recorded in the database. Rather than create new models and relational tables for each such event type, we opted to add support for a simple experience logging facility. These are currently written to the database to allow analytics queries to join them with other relational data, but they could theoretically be logged elsewhere initially and only merged into the analytics environment at analysis-time.

Questions that we have been very interested in become possible to answer by annotating adding a few such event experiences. Did a student ever see their study plan? Did they start viewing an available video explanation? Did a student ever access their performance analytics page? These event experiences played a role in a bigger-picture question that we've been exploring recently: Of all of the many opportunities for learning within Grockit – individual problem solving, small peer-group study, instructor-led lessons, skill-based video explanations, private tutoring sessions, and

InitialQuestions
Should new students be given easier questions when playing alone, until they answer at least 5 questions correctly?
This test is being assigned and evaluated for new users only.

Subsequent activity	*EasyFirst* (1657 persons)	*Normal* (1721 persons)	(difference)	
Participated in further study	**73.2% (1213)**	67.5% (1161)	(2.6% to 8.9%)	**
Participated in group study	**31.4% (520)**	27.0% (464)	(1.3% to 7.5%)	*
Logged in again later	30.8% (510)	28.4% (489)		
Participated in group discussion	**22.1% (366)**	18.4% (316)	(1% to 6.5%)	*
Reviewed past questions	13.9% (230)	12.8% (221)		

Table 1: A reformatted portion of the automatically-produced output of a controlled experiment testing the effect of the difficulty of the first few questions presented to a new student. "Difference" indicates a confidence interval around the difference between the percentages from the two groups. A single * indicates a p-value that is significant at the $\alpha = 0.05$ level, and a double ** indicates significance at the $\alpha = 0.01$ level (two-tailed).

skill-customized practice, among others – which of these is most effective? Understanding this could inform decisions ranging from the study plans that we offer students, the specific activities that we encourage at various points in time, and even decisions about removing certain activities altogether. When we first sought to ask this question, we found that we hadn't recorded all of the data that we were interested in examining (including information on partial or entire video explanations watched). Instrumenting the system with a few logging records was sufficient to collect the additional information. Details can be found in Bader-Natal, Lotze, and Furr [1].

An event-logging facility within a web application allows data analysts to record data necessary for analysis that are not otherwise persisted, often in a single line of code.

5. TESTING NEW HYPOTHESES WITH RANDOMIZED CONTROLLED EXPERIMENTS

Where OLAP-style analysis allows us to understand and describe trends that we see in collected student data, this approach does not directly let us test out new hypotheses. As with other types of web applications, web-based educational software provides an ideal environment for running randomized controlled experiments [6]. Our goal with Grockit's implementation of a framework for doing so was to (a.) minimize the amount of work necessary to introduce a new experiment, and (b.) minimize the amount of work necessary to analyze an experiment. We illustrate below how just a few lines of application code automatically generate an analysis of the differences among treatment and control groups over a fixed set of outcome measurements (used for all such analyses). For a recent experiment, we sought to understand the affect that the difficulty of the first few questions presented to a student had on subsequent participation and retention rates. This code randomly assigns students (with equal probabilities) to the treatment or control groups, and provide a different user experience based on that assignment:

```
experiment = { "InitialQuestions" => ["EasyFirst", "Normal"] }
if (SplitTest.find_or_create_assignment(self, "InitialQuestions")
      == "EasyFirst")
  show_easy_first
else
  show_regular
end
```

The first line defines a unique identifier for the new experiment, which, combined with the unique identifier for a particular student (based on *self*, in the second line), is the basis for a random assignment to one of the groups. By adding this application code, the analysis in Table 1 is automatically generated and distributed daily. Here, students were assigned to the treatment group with a $p = 0.5$ probability. The null hypothesis is that the two populations of students have the same true proportions, and the alternative is that the proportion is different in one of the populations. In the case of the analysis in Table 1, we found that the *EasyFirst* question selection strategy resulting in an increase in the rate of subsequent study, group study, and participation in group discussions.

A built-in infrastructure for introducing and evaluating randomized controlled experiments allows for a powerful and valuable new class of analysis: hypothesis testing.

6. DISTRIBUTION: FROM PUSH TO PULL

Originally, the completed reports were emailed as PDF files to specific recipients. As the number of reports and students increased, emailing many large files became infeasible as a way of distributing the results of the reports. In place of this, we now have a centralized, web-based reporting system. All past reports are archived, and the source data and intermediate data files are persisted to allow for later analysis, comparisons over time, or replications of past results. Furthermore, by using the same user authentication as the main Grockit site, we can also provide teachers with access to reports on their classes. Finally, by providing a common way of accessing the reports, it allows stakeholders to discuss individual reports simply by sharing a link to the report under discussion.

Access control, search, and data archiving can all be simplified by means of a centralized, web-based repository for the output of an analytics reporting system.

7. FOCUSING ON PERFORMANCE, SCALABILITY, AND STABILITY

As the number of students in the system has increased, the size of data to be analyzed has become much larger, forcing us to consider how to scale the workflow and analyses to deal with more reports on more data. The first and simplest

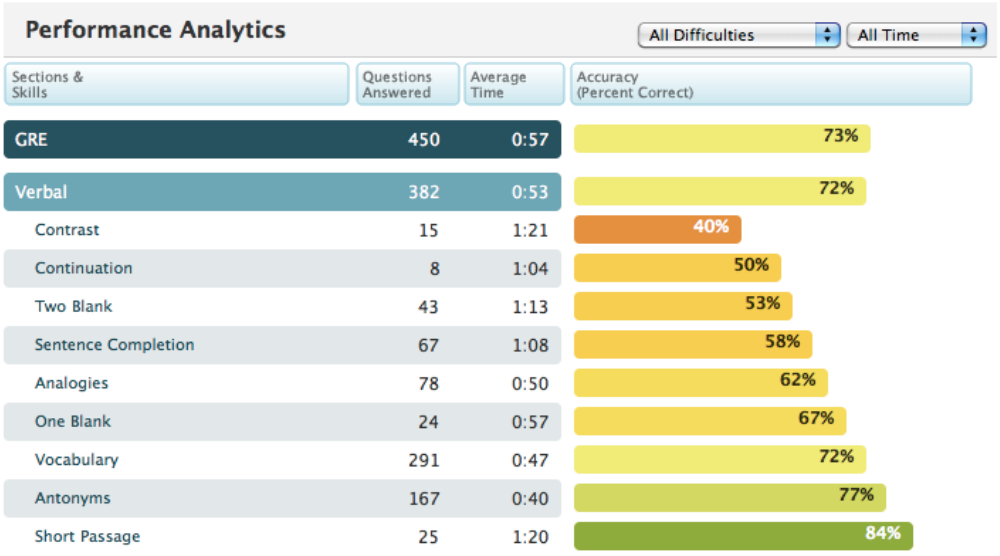

Figure 3: For each student, we provide access to a simple report showing their accuracy in each of several skill areas. This allows them to visualize their strengths and focus on the areas they are weak in. The student can select various difficulty ranges and time ranges to view their analytics over.

answer is increasing the power of the computers running the analyses, in our case by performing the analyses on Amazon EC2 instances with additional RAM and faster processors. One useful tool for making use of these powerful remote machines in developing analyses is the ability to run analyses as though they were local, by sending the request to the remote machine, having it perform the data selection, analysis, and visualization, and then retrieving the results for display as though they had been run locally.

Additionally, since we do not want to slow down our production web-server or database with intensive analysis, we replicate the databases. Originally, this was done daily at the beginning of the set of analyses to be run; but as the data size increased, copying over all of the data became infeasible. Instead, we continuously keep the reporting database near-realtime by using MySQL replication against the production database, ensuring that our reports are run on up-to-date data. This automatic replication process also allowed us to begin to distribute reporting across multiple machines; instead of having a single machine run all reports and waiting for each report to finish before running the next one, we have multiple machines, each with an up-to-date copy of the database and workflow code, which can run through the entire reporting process in parallel. The next step in the process it to have a pool of ready reporting machines, along with a central job distribution system, such that any report request, whether run periodically or by an analyst trying to answer a new question, can be sent directly to the next available machine. Providing a central job distribution process information about priority and the state of running jobs would allow all job requests to be integrated, and also allow for any user viewing a report on the web-based system to initiate report jobs on-demand.

As the analyses become valuable on a recurring basis, rather than simply being a one-time answer to a question, we need to ensure that they are stable in the face of continual changes to the data and application, so that the stakeholders relying on the analysis can rely on their being ready and available. While we have automated tests that ensure that reports are successfully generated, we wish to also have a simple capacity to ensure that the results of the analysis are still reliable and accurately reflect the answer to the question. This additional capacity can be provided by having a single test file for each report, that verifies correct results from a prespecified testing dataset. This can be done without slowing down the process for rapid one-off analyses, which can be performed without creating or providing automated tests.

As the dataset grows and results become mission-critical, issues of performance, scalability, and reliability become increasingly important.

8. PROVIDING ANALYTICS DIRECTLY TO LEARNER AND TEACHERS

As teachers and students become more data-oriented about their learning process, several analyses initially prepared using the analytics pipeline have since been moved to a user-facing location within the application itself. Analytics reports that were distributed weekly to teachers in classroom pilots were subsequently incorporated into the teacher's dashboard in Grockit. Skill-grained student performance reporting, illustrated in Fig. 3, was also first created within the analytics pipeline, which allowed for rapid iteration and refinement before it was transitioned into the production application. Once this transition is made, computation time becomes critical. In order to minimize page load speeds, we proceed by precomputing and caching, leveraging a Hadoop infrastructure.

However, as the number of games and questions in the system increases, directly computing this when the student re-

quests it began to take several seconds, especially for more active students. This is particularly true given that the student can select not just to view their all-time analytics results, but also select from 35 different combinations of problem difficulty and response recency to analyze their performance. To solve this, we began precomputing portions of the students' analytics information. For each day the student was active, for each skill to be analyzed, we compute their total number of questions answered, number correct, and total time taken. This can be done using Hadoop for all users in just a few minutes, allowing for frequent updates of the precomputed data. Generating the percentage correct and average time for each individual skill or track becomes a simple matter from this data. This can then quickly be retrieved as a single row from an indexed table, and sent to the client browser for rendering. Overall, we saw between a 10-25x improvement in response time. As we provide more analytics within the application, increasing the amount of precomputing and cloud-based parallelization will allow us to provide feedback without sacrificing performance.

Students and teachers may benefit from more direct access to the results of a learning analytics system. Doing so may require aggressive performance optimizations to provide immediate access to near real-time analysis.

9. CONCLUSION

To take full advantage of the rich data available from computer-based learning systems, creating a pipeline for processing and presenting advanced analysis can be a significant boon for learning about students' behavior and performance. We have described many of the advantages to be gained by doing so, as well as methods which we have used to achieve them. We hope that our system may serve as an example of what is possible by automating this pipeline. In making the analytics more readily available, scalability and stability must be addressed; we have described these challenges as well as approaches we are developing to address them. Finally, the future of analytics is one where the results are available not only to researchers and system designers, but also directly to students and teachers. In order to do this, we must not only make the process of developing new analytics as easy as possible, but also reliable and accessible to the teachers and students that is meant to inform and help.

10. REFERENCES

[1] Ari Bader-Natal and Thomas Lotze and Daniel Furr. A Comparison of the Effects of Nine Activities within a Self-Directed Learning Environment on Skill-Grained Learning. In Biswas, G. and Bull, S. and Kay, J. and Mitrovic, A., editors, *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, Lecture Notes in Computer Science, 2011, Volume 6738/2011, Springer.

[2] Ari Bader-Natal. Incorporating game mechanics into a network of online study groups. In Scotty D. Craig and Darina Dicheva, editors, *Supplementary Proceedings of the 14th International Conference on Artificial Intelligence in Education*, volume 3, Intelligent Educational Games workshop, pages 109–112, July 2009. IOS Press.

[3] Ari Bader-Natal. Interaction synchronicity in web-based collaborative learning systems. In Theo Bastiaens, Jon Dron, and Cindy Xin, editors, *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2009*, pages 1121–1129, Vancouver, Canada, October 2009. AACE.

[4] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[5] Gerhard H. Fischer and Ivo W. Molenaar, editors. *Rasch Models: Foundations, Recent Developments, and Applications*. Springer-Verlag, New York, 1995.

[6] R. Kohavi, R.M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 959–967. ACM, 2007.

[7] Raph Koster. *A Theory of Fun for Game Design*. Paraglyph Press, 2004.

[8] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0.