Volume 15, No. 3
May 2017
ISSN: 1545-679X

INFORMATION SYSTEMS EDUCATION JOURNAL

In this issue:

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **EDSIG**, the Education Special Interest Group of AITP, the Association of Information Technology Professionals (Chicago, Illinois). Publishing frequency is six times per year. The first year of publication was 2003.

ISEDJ is published online (http://isedj.org). Our sister publication, the Proceedings of EDSIGCon (http://www.edsigcon.org) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the conference. At that point papers are divided into award papers (top 15%), other journal papers (top 30%), unsettled papers, and non-journal papers. The unsettled papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the ISEDJ journal. Currently the target acceptance rate for the journal is under 40%.

Information Systems Education Journal is pleased to be listed in the 1st Edition of Cabell's Directory of Publishing Opportunities in Educational Technology and Library Science, in both the electronic and printed editions. Questions should be addressed to the editor at editor@isedj.org or the publisher at publisher@isedj.org. Special thanks to members of AITP-EDSIG who perform the editorial and review processes for ISEDJ.

Information Systems Education Journal

Java vs. Python Coverage of Introductory Programming Concepts: A Textbook Analysis

Kirby McMaster
kmcmaster@weber.edu
Computer Science - retired
Weber State University
Ogden, UT 84408 USA

Samuel Sambasivam
ssambasivam@apu.edu
Computer Science
Azusa Pacific University
Azusa, CA 91702 USA

Brian Rague
brague@weber.edu
Computer Science
Weber State University
Ogden, UT USA 84408

Stuart Wolthuis
stuartlw@byuh.edu
Computer and Information Sciences
Brigham Young University-Hawaii
Laie, HI 96762 USA

Abstract

In this research, we compare two languages, Java and Python, by performing a content analysis of words in textbooks that describe important programming concepts. Our goal is to determine which language has better textbook support for teaching introductory programming courses. We used the TextSTAT program to count how often our list of concept words appear in a sample of Java and Python textbooks. We summarize and compare the results, leading to several conclusions that relate to the choice of language for a CS0 or CS1 course.

Keywords: programming concepts, Java, Python, textbooks.

1. INTRODUCTION

In the early years of computing, the choice of a first language for programmers was often decided by the work environment, typically Information Technology divisions with specialized needs. Assembly language for a specific hardware system was the usual situation. Programming in a higher-level language such as Fortran or Cobol became common over time as more versatile computing platforms and elaborate computing problems emerged.

When Computer Science programs at universities began to develop, the choice of an introductory programming language was determined primarily by the curriculum designers, with an emphasis on the pedagogical value of the language rather than its popularity or practicality in developing real-world applications. As might be expected in the academic world, there was and still is a diversity of opinion on what the first language should be (Siegfried, Chays, & Herbert, 2008).

The most recent Computer Science Curriculum Guidelines (2013) published by ACM/IEEE state that "...advances in the field have led to an even more diverse set of approaches in introductory courses [and these] approaches employed in introductory courses are in a greater state of flux." Moreover, the report observes "...that rather than a particular paradigm or language coming to be favored over time, the past decade has only broadened the list of programming languages now successfully used in introductory courses".

In the 1970s and 1980s, Pascal became the language taught most often in introductory programming courses. Eventually, many schools moved to C for practical reasons, since graduates rarely used Pascal in their employment. As the benefits of object-oriented programming became evident, the first language evolved to C++ and later to Java, which provides a more managed development environment (de Raadt, Watson, & Tolman, 2002).

The tradeoffs of an object-first approach versus an imperative-first approach in introductory courses have been extensively and hotly debated (Lister, 2006). This decision about which programming paradigm to teach beginning students strongly influences the choice of introductory language. Alternatively, some early courses in CS emphasized broader computing concepts rather than the subtleties of programming syntax (Sooriamurthis, 2010). The paramount question regarding the delivery of an effective introductory CS course remains "What to teach?", followed immediately by "Which language best supports the concepts to be taught?".

In recent years, the increased demand for programming courses for liberal arts students has led to the development of what are termed CS0 courses (with CS1 courses aimed for CS majors). The preferred programming language for a CS0 course is often different from the language taught in CS1. CS0 languages trend toward predominantly visual environments such as Alice, or more dynamic popular choices such as Python.

Purpose of this Research
Much research has been performed over the last few decades on which language is best for an introductory programming course (Brilliant & Wiseman, 1996). In an effort to contribute to this discussion, our research focuses on two languages--Java and Python. These languages are increasing in popularity for introductory courses, especially Python (Guo, 2014). Rather than evaluate the usability or suitability of the languages within an introductory context, we performed a content analysis (Krippendorff, 2012) of Java and Python textbooks to determine how well they cover important CS0/CS1 programming concepts such as *class* and *algorithm*.

We developed a list of basic programming concepts that might be taught in an introductory course. Initial sources used for developing this concepts list were drawn from various instructional assessments, curriculum resources, and introductory course content that we designed ourselves or researched. We then counted how often each textbook mentioned each concept. We did not study the order in which the concepts were presented, nor did we judge how well the concepts were explained. We simply summarized frequencies for the words that represented each concept.

An instructor in a programming course usually chooses a textbook to guide how she/he will organize and present the material. Our main research assumption is that the framework of the author is reflected by the words used most often in the textbook. The framework we are evaluating is one that is appropriate for introductory programming. From the author's choice of words, we can judge how suitable the textbook will be for teaching the main concepts of the programming course.

2. METHODOLOGY

This section of the paper describes the methodology used to collect word frequency data from selected Java and Python textbooks. The words we examine represent important concepts for an introductory programming course.

Programming Concepts
We created a list of important programming concepts from several sources. We started with an initial list of programming terms taken from quizzes and exams we have given to CS1

students to measure their understanding of course topics. In earlier research, we performed a word frequency analysis of object-oriented programming (OOP) textbooks (representing a variety of languages) to empirically reveal frequent OOP concepts. We used the results of that study to form a list of OOP words.
In the current study, we created a list consisting of programming concepts mentioned in the Programming Fundamentals (PF) section of the Computing Curricula 2001 Computer Science Final Report (2001). We created an additional list of concepts based on the Software Development Fundamentals (SDF) section of the Computer Science Curricula 2013 Final Report.

In constructing our list, we attempted to avoid keywords from specific languages, such as *float* and *while*. However, a few keywords, such as *class*, were difficult to omit. From the above sources, we formed a combined list that grew to 100 programming concepts. This larger list evolved as we performed the actual word analysis in the selected Java and Python books.

Sample of Textbooks
We collected a sample of 10 Java textbooks and 10 Python textbooks. We wanted our sample to include popular books in both languages. Due to budget constraints (i.e. no budget), we chose textbooks that were available on the Internet and could be downloaded as PDF files. We obtained a reasonably diverse sample of books (see References), but some were older editions (e.g Zelle, 2002).

We later observed that the Java books tended to be larger (i.e. contained more words). The average size of the Java books was 222,953 words, whereas the average size for the Python books was 144,039 words. As a quick check to confirm that the sizes of our Java and Python books were representative, we compared 10 Java books and 9 Python books (not including very short books) listed on Amazon. For the Amazon books, the total number of words was not available, but the number of pages was given. The Amazon sample averages were 690 pages for the Java books and 514 pages for the Python books. So on Amazon, the Java books tend to be larger, which is consistent with our downloaded sample.

Convert PDF files to Text Files
Textbooks in PDF file format are not convenient for performing repeated word searching and counting. Fortunately, Adobe Reader has a "File/save As" menu choice to convert the contents of a PDF file to a text file. We used Adobe Reader to create a text file for each of the 20 textbooks in our study.

We noticed that the text file versions of the books included many character strings that contained digits, punctuation, and other non-alphabetic symbols. To simplify our counting of concept words, we wrote a short program (in Python) that removed all non-letter symbols and replaced them with blank characters. This program also converted all letters to lower-case. We used this program to obtain a filtered set of 20 text files which consisted of only letters and blanks. Note that none of the targeted word groups contains a numeric or special character.

Perform Word Counts
We used a popular program called TextSTAT (Huning, 2007) to obtain word counts for all words on our programming concept list. With TextSTAT, you first define a "Corpus", which holds a list of text files. We defined a corpus for each textbook and linked the corpus to the transformed textfile containing the textbook.

To perform a word search, a separate TextSTAT screen allows the user to specify search options. Most of the time, we used the option to include all words, with the words and frequencies presented in alphabetical order. We would then go through the concept list (also in alphabetical order) and record/total the frequencies for each word group. This was the most labor-intensive part of our methodology. Occasionally, we would enter a short string (e.g. *iterat*) to search for all words that contain the string (e.g. *iterate*, *iteration*, *iterator*).

3. ANALYSIS OF DATA

The number of programming concepts on our evolving list reached 100 by the end of our data analysis. Alphabetically, the concepts ranged from *abstraction* to *variable*. As mentioned in the methodology section, each concept was represented by a group of one or more words. For example, the word group for the OOP concept *object* contained two words--*object* (singular) and *objects* (plural).

For every concept, we counted the number of occurrences of each word group member in the Java and Python textbooks. As an example, in the Java book by Schildt (2007), the word *object* appears 1674 times, and the word *objects* appears 380 times. The total word count for the concept is 2054.

Convert Word Counts to Word Rates

Because each textbook contains a different number of words, the actual word counts for concepts are not comparable across books. Larger books tend to have larger word counts. To standardize the counts, we converted each word count for a concept to a *word rate*. The rate we chose was "per 100,000 words". That is, we divided the concept word count by the total number of words in the book and multiplied by 100,000.

For example, Schildt's book mentioned above contains a total of 325,991 words. The word count for the *object* concept is 2054. This count is rescaled to a word rate as shown below:

word rate = (2054/325,991)*100,000 = 630.1

This means that the object concept is mentioned 630.1 times per 100,000 words in Schildt's book. Word rates were calculated for each concept in each book.

Calculate Trimmed Means

After concept word rates were obtained in all Java and Python textbooks, averages were calculated separately for the Java and Python values. Because the word rates for concepts (Java or Python) often varied widely from book to book, we calculated trimmed means (instead of the usual untrimmed versions) to diminish the effect of outliers. To provide a conservative treatment for these outliers, our trimmed means include only the middle 6 out of 10 word rates. The top two and bottom two word rates are dropped.

For example, word rates for the *object* concept in all 10 Java textbooks are:

522.4	561.7	630.1	334.5	843.3
684.9	703.5	767.2	863.5	488.4

Removing the two highest rates (863.5 and 843.3) and two lowest rates (334.5 and 488.4), the trimmed mean for *object* in the Java books is 645.0. Two trimmed means were calculated for each concept, one for Java and the other for Python.

Distributions of Trimmed Means

Each set of books (Java and Python) provided a sample of 100 trimmed means, representing word rates for the 100 concepts. A statistical description of the Java and Python distributions is summarized in Table 1.

Many of the statistics are larger for the Java distribution than the Python distribution. The central tendency measures (mean and median) are higher, and the dispersion measure (IQR) is larger. This is primarily due to the greater number of concept words in the Java books.

Statistic	Java	Python
Sample N	100	100
Minimum	0.34	0.00
Centile 25	18.92	10.50
Median	58.00	38.05
Centile 75	134.27	116.68
Maximum	987.40	601.93
IQR	115.35	106.18
Mean	109.95	90.59

Table 1: Distributions of Trimmed Means

For the Java distribution, the *maximum* word rate is for the concept *class*, and the *minimum* word rate is for *decomposition*. For Python, the *maximum* word rate is for *function*, while the *minimum* word rate is (again) for *decomposition*. The Java *median* word rate is the midpoint between the word rates of the two middle concepts *stream* and *block*. For Python, the two middle concepts are *block* and *event*.

The *mean* of the Java word rates is almost twice the size of the median. This indicates that the distribution is positively *skewed*, mainly due to the presence of several high word rates (including the maximum value). The mean of the Python word rates is more than twice the size of the median, indicating another positively *skewed* distribution.

The variability of scores in a distribution is usually described by the *standard deviation*. However, this statistic is inflated when outliers are present. A more stable measure of variation is the interquartile range IQR (Upton & Cook, 1996), which is the difference between the 75th centile value and the 25th centile value. For Java, the 75th centile concept is *definition*, and the 25th centile concept is *link*. The corresponding concepts for Python are *set* (75th centile) and *literal* (25th centile).

The word rates for programming concepts tend to be higher in the Java books. Overall, 62 of the 100 concepts have a higher word rate in the Java books than in the Python books. The remaining 38 concepts appear more often in the Python books. Additional details and comparisons of these two word rate distributions are presented in the following sections.

Most Frequent Concepts

The fifteen programming concepts with the highest word rates for Java and Python are listed in Table 2.

Java Concept	Rate	Python Concept	Rate
class	987.4	**function**	601.9
method	949.8	**list**	487.0
object	645.0	program	462.1
value	477.5	value	451.1
program	460.6	string	410.4
string	399.8	file	372.0
type	369.5	object	336.7
variable	288.6	number	319.7
array	272.2	code	300.6
system	253.7	method	298.9
number	251.4	class	297.0
file	216.9	**line**	263.7
code	213.2	**module**	235.8
statement	212.1	type	204.0
thread	188.2	statement	203.1

Table 2: Most Frequent Concepts
(Differences in **bold**)

Eleven of the concepts appear on both lists, but in different ranked positions. This demonstrates substantial agreement by authors on which concepts are *most important* in both languages. Four concepts are on the Java list only, and four others are confined to the Python list. The concepts that are not on both lists are shown in **bold**.

Among the Java concepts, the top three--class, method, and object--describe features of object-oriented programming (OOP). These concepts are also on the Python list, but with lower word rates. Six of the Java concepts--value, string, type, variable, array, and number--describe data types and data structures. The Python list contains four of these concepts, but replaces array with list and excludes variable.

The I/O concept *file* is on both lists, but has a higher word rate in the Python books. The Java concept *thread* is rarely mentioned in the Python texts. *Function* and *module* are older terms used to describe modular programming. Python retains these terms, whereas the Java books prefer the OOP concepts *method* and *class*.

Least Frequent Concepts

The fifteen programming concepts with the lowest word rates for Java and Python are listed in Table 3. Again, eleven of the concepts appear on both lists, but in different ranked positions. This shows agreement by Java and Python authors on concepts they perceive to be *unimportant* in both languages. Concepts that appear on only one list are shown in **bold**.

Java Concept	Rate	Python Concept	Rate
encapsulation	9.3	**constant**	6.6
debug	8.1	maintainable	5.8
signature	7.9	**stream**	5.1
record	7.9	encapsulation	4.0
maintainable	7.1	reserved	3.9
abstraction	5.9	branch	3.1
polymorphism	5.5	pointer	2.8
relation	5.4	polymorphism	2.5
reserved	5.1	procedure	1.6
procedure	4.7	signature	1.5
pointer	4.2	quality	1.5
branch	3.3	**queue**	0.6
module	1.3	**thread**	0.6
quality	0.6	abstraction	0.6
decomposition	0.3	decomposition	0.0

Table 3: Least Frequent Concepts
(Differences in **bold**)

The concepts that appear on both least-frequent lists include a few surprises. Some of these concepts are often considered important by programming instructors. Certainly *abstraction* is a key programming topic. Of the three pillars of OOP (*encapsulation*, *inheritance*, and *polymorphism*), two are on both least-frequent lists. Thankfully, these textbooks spare *inheritance* from such neglect. The *signature* concept, relevant to *polymorphism*, is rarely mentioned.

Function and *procedure* were once distinct concepts in modular programming. Perhaps due to compromises made in the design of the C language (and perpetuated in C++ and Java), the *procedure* word has been replaced with "void" functions.

From the Software Engineering (SE) vocabulary, *quality* and *maintainable* are held in low regard by both Java and Python textbooks. The concept of *pointer* has low word rates, although the substitute term *reference* does appear more often in both sets of books. *Keyword* is more popular than *reserved* word. Finally, almost none of the books contain *decomposition*, which is the least frequent word on both lists. This concept embodies a core strategy in modular programming.

Middle Frequency Concepts

We have presented word rates for the top 15 and bottom 15 programming concepts, and now turn our attention to the 70 concepts with middle-level usage rates. This list of concepts is too long to include in a single table in the paper. Instead, in Table 4 we present 10 Software Engineering concepts that have middle-level word rates in the programming textbooks.

	Java	**Python**
Concept	**Rate**	**Rate**
problem	63.9	57.9
solution	32.1	48.1
requirement	29.9	42.8
specification	55.5	39.5
model	25.1	13.6
algorithm	34.9	22.5
design	49.2	12.3
test	85.5	138.2
style	21.1	17.7
document	40.5	44.0

Table 4: Middle Frequency Concepts
Software Engineering Words

For Java books, the SE word rates range from 21.1 (for *style*) to 85.5 (for *test*). The word rates in Python books range from 12.3 (for *design*) to 138.2 (again for *test*).

Concepts on the list include *problem* (Java/Python rates 63.9/57.9) and *solution* (Java/Python rates 32.1/48.1), reflecting the problem-solving focus in SE. The words *requirement*, *specification*, *model*, *algorithm*, *design*, and *document* are life cycle development activities. *Style* is a consideration to ensure source code is readable and maintainable. The relatively low word rates for *style* (Java/Python rates 21.1/17.7) and for *model* (Java/Python rates 25.1/13.6) are unfortunate.

As Table 4 indicates, all of these concepts appear with moderate word rates in both the Java and Python textbooks. Six of the concepts appear more often in Java books, while the other four are more frequent in Python books. There is no obvious single criterion for determining which language favors which SE concepts.

Word Rate Correlation

In this section, instead of examining the Java and Python word rate distributions separately, we consider the joint distribution of the two rates. If the focus on key introductory concepts is consistent across all examined textbooks, we would expect to find a positive relationship between the Java and Python word rates. For most programming concepts, a higher word rate in the Java books should suggest a higher word rate in the Python books, and vice versa.

To measure the degree of linearity in the relationship, we calculated the Pearson correlation coefficient. The correlation value we obtained for our 100 pairs of scores was 0.601, which is positive but far from 1.0.

We do not claim that the relationship should be linear, but it should be monotonic. A better statistic for monotonic relationships is the Spearman rank-order correlation (Maritz, 1995). Our result for the Spearman statistic was 0.726, which describes a fairly strong *increasing* relationship between Java and Python word *ranks*.

A scatter diagram of the word rate pairs, converted to ranks from 1 (highest rank) to 100 (lowest rank), is displayed as Figure 1.

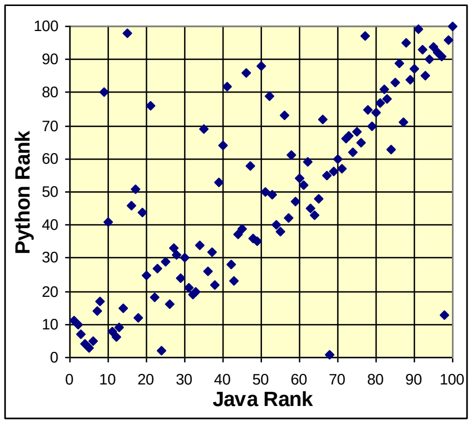


Figure 1: Java vs. Python Concept Ranks

In this figure, we can see that most of the pairs of ranks fall approximately along a line that runs from pair (1,1) to pair (100,100). Below the implied line, two obvious outliers are the pairs (98,13) for *module* and (68,1) for *function*. In these pairs, the Python rank is much higher (closer to 1) than the Java rank. Above the line, the two most noticeable outliers are (15,98) for *thread* and (9,80) for *array*. These concepts have a much higher Java rank (closer to 1).

A more complete list of outliers is presented in Table 5.

Concept	Java Rank	Python Rank	Diff
module	98	13	-85
function	68	1	-67
interface	16	46	30
system	10	41	31
component	35	69	34
event	17	51	34
stream	50	88	38
constant	46	86	40
declaration	41	82	41
constructor	21	76	55
array	9	80	71
thread	15	98	83

Table 5: Largest Differences in Ranks
("Highest" rank is 1)

The choice of how large the difference in ranks should be to consider a concept an outlier is subjective. In this table, we include all pairs in which the difference in ranks is 30 or larger. A negative difference occurs when Python has a higher rank. A positive difference favors Java. Note that all but two of the concepts in Table 5 have a higher Java rank.

We noted earlier that *function* and *module* are among the top fifteen concepts in word frequency in Python books. This table indicates that these two popular Python concepts appear much less often in Java books. Three OOP concepts--*constructor*, *component*, and *interface*--are favored by Java books.

The data concepts *array*, *declaration*, and *constant* appear less often in Python books for various reasons. Python prefers *lists* over *arrays*. Variables are not overtly *declared* in Python. *Stream* I/O, as a generalization of file I/O, is implemented in Java as stream classes. Real-time *events* and *threads* are common Java features, but not Python.

4. SUMMARY AND CONCLUSIONS

The choice of programming language for introductory Computer Science courses is a strong indicator of the concepts emphasized during course instruction. Ongoing discussion about what to teach and which language tool best supports learning objectives for introductory programming courses continues unabated among instructors, administrators, and accreditation organizations. A definitive "best practices" approach in this area remains unresolved. Our current work further informs this debate by correlating core programming concepts with specific textbooks that promote either Java or Python as the coding language.

The primary purpose of this study was to compare how well Java and Python textbooks provide coverage of important introductory programming topics. We developed a list of 100 programming concepts, and we collected a sample of 10 Java books and 10 Python books. We then counted how often words that represent the concepts appeared in the books. After standardizing the data, we computed trimmed means of word rates for all 100 concepts, with separate rates for Java and Python. From this data, we draw the following conclusions.

First, words that describe our 100 programming concepts have a greater density (higher word rates) in the Java books in our study. The word rate distribution for Java has a mean of 109.25, with a maximum value of 987.40. For Python, the mean is 90.59, with a maximum of 601.93.

Second, there is remarkable agreement between the programming concepts mentioned most often in the Java and Python books. Eleven of the top 15 Java concepts are also included in the top 15 Python concepts. Highly-used concepts for both languages include *class*, *object*, and *method*, each representing OOP.

Third, there is also agreement on which concepts are rarely mentioned in both sets of books. Eleven of the bottom 15 Java concepts are also in the list of 15 least-used Python concepts. Common neglected concepts include *encapsulation* and *polymorphism* for OOP, plus SE concepts *quality* and *maintainable*. It is disappointing that *abstraction* is on both bottom 15 lists.

Fourth, several concepts appear on only one of the top 15 or bottom 15 word lists for Java and Python. The top 15 Java-only concepts include *array* and *variable*. Among the top 15 Python-only concepts, *array* is replaced by *list*, and other concepts are added. The bottom 15 Java concepts include *module*, which is a top 15 concept for Python. The bottom 15 Python list includes *thread*, which is a top 15 concept for Java.

Fifth, a fairly strong increasing relationship exists between concept ranks for Java vs. Python, as indicated by a rank-order correlation of 0.726. There are a few clear exceptions to this relationship. *Thread*, *constructor*, and *declaration* have much higher Java ranks. *Module* and *function* have much higher Python ranks.

Sixth, Java and Python textbooks devote substantial time on practical concepts that describe how to write code. Discussion of Software Engineering concepts that deal with how to *think* like a programmer and write efficient, maintainable code receive less attention. This learning goal may be less important in an introductory programming course, but it becomes a major focus as students progress through a Computer Science degree program.

Overall, both Java and Python books provide reasonable levels of support for most of the programming concepts we considered. The choice of Java or Python (or other language) for an introductory class should be based on considerations beyond textbook support for important concepts. Whatever language and textbook are chosen, instructors must be prepared to provide additional material to achieve their desired course objectives.

Future Research
Planned future research activities include:

1. Perform a similar study comparing Java and C++ textbooks to determine how well they support important CS1 concepts.

2. Perform a similar study comparing textbooks for Python and another language (e.g. Ruby) to determine how well they support important CS0 concepts.

3. Perform research to provide empirical support to improve our list of important programming concepts. This is not a trivial task, in light of previous research by Hertz (2010) and Tew & Guzdial (2010).

Note: A complete list of our 100 programming concepts, along with Java and Python trimmed mean word rates, are presented in Table 6 in the APPENDIX.

5. REFERENCES

Brilliant, S. S., and Wiseman, T., “The First Programming Paradigm and Language Dilemma”, ACM SIGCSE Bulletin Vol. 28, No. 1 (1996), p. 338-342.

Computing Curricula 2001 Computer Science Final Report, Joint Task Force on Computing Curricula, Association of Computing Machinery, IEEE Computer Society, 2001.

Computer Science Curricula 2013, Joint Task Force on Computing Curricula, Association of Computing Machinery, IEEE Computer Society, 2013.

deRaadt, Michael, Watson, Richard, and Toleman, Mark, “Language Trends in Introductory Programming Courses,” InSITE, June 2002. proceedings.informingscience.org /IS2002Proceedings/papers /deRaa136Langu.pdf

Guo, Philip, "Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities." Communications of the ACM, Blogs, 2014.

Hertz, Matthew, "What do 'CS1' and 'CS2' Mean? Investigating Differences in the Early Courses." SIGCSE Proceesings, Milwaukee, 2010.

Huning, M, TextSTAT 2.7 User’s Guide. TextSTAT, created by Gena Bennett, 2007.

Krippendorff, Klaus H., Content Analysis: An Introduction to Its Methodology, 3rd Ed. SAGE Publications, 2012.

Lister, Raymond, E. A. Research perspectives on the objects-early debate. In ITiCSE proceedings (2006), pp. 146--165.

Maritz, J. S., Distribution-Free Statistical Methods (2nd ed). Chapman and Hall, 1995.

Siegfried, Robert M., Chays, David, and Herbert, Katherine G., “Will There Ever be Consensus on CS1?” In Proceedings of FECS. 2008, 18-23. home.adelphi.edu/~siegfried /Consensus.pdf

Sooriamurthi, Raja, The Essence Of Object Orientation For CS0: Concepts Without Code. Journal of Computing Sciences in Colleges, Vol. 25 (3), p 67-74, January, 2010.

Tew, Allison Elliott, & Guzdial, M., Developing a Validated Assessment of Fundamental CS1Concepts, SIGCSE Proceedings, Milwaukee, 2010.

Upton, Graham, and Cook, Ian, Understanding Statistics. Oxford University Press, 1996, p.55.

Java Textbooks:

Arnold, Ken, James Gosling, and David Holmes, THE Java Programming Language (4th ed).Addison Wesley Professional, 2005.

Deitel, Harvey, and Paul Deitel, Java How to Program (4th ed). Prentice Hall, 2002.

Downey, Allen B., Think Java: How to Think Like a Computer Scientist. Allen Downey, 2012.

Eck, David J., Introduction to Programming Using Java (Version 6.0.3). Hobart and William College, 2014 (PDF version of on-line book).

Lemay, Laura, and Charles L. Perkins, Teach Yourself JAVA in 21 Days. Sams.net Publishing, 1996.

Roberts, Eric. S., The Art and Science of Java (Preliminary Draft). Stanford University, 2006.

Schildt, Herbert, Java: The Complete Reference (7th ed). McGraw-Hill, 2007.

Sierra, Kathy, and Bert Bates, Head First Java (2nd ed). O'Reilly.

Stein, Lynn Andrea, Interactive Programming in Java. Lynn Andrea Stein, 1999.

Wu, C. Thomas, An Introduction to Object-Oriented Programming with Java (5th ed). McGraw-Hill, 2010.

Python Textbooks:

Downey, Allen, Think Python: How to Think Like a Computer Scientist (Version 2.0.15). Green Tea Press, 2015.

Halterman, Richard L., Learning to Program with Python. Richard L. Halterman, 2011.

Heinold, Brian, Introduction to Programming Using Python. Brian Heinold, 2012.

Jackson, Cody, Learning to Program Using Python. Cody Jackson, 2011.

Kuhlman, Dave, A Python Book: Beginning Python, Advanced Python, and Python Exercises. Dave Kuhlman, 2009.

Lutz, Mark, Programming Python (4th ed). O'Reilly, 2011.

Maruch, Stef, and Aahz Maruch, Python for Dummies. Wiley, 2006.

Payne, James, Beginning Python: Using Python 2.6 and Python 3.1. Wiley Publishing, 2010.

Pilgrim, Mark, Dive Into Python. Mark Pilgrim, 2004.

Zelle, John M., Python Programming: An Introduction to Computer Science (Version 1). Wartburg College, 2002.

APPENDIX

Table 6: Concept Word Rate Trimmed Means for Java and Python

	Concept	Java Rate	Python Rate			Concept	Java Rate	Python Rate
1	abstraction	5.9	0.6		51	literal	14.0	10.5
2	algorithm	34.9	22.5		52	local	36.2	36.0
3	argument	114.4	142.7		53	loop/looping	112.6	152.5
4	array	272.2	7.8		54	maintain/maintainable	7.1	5.8
5	assignment/assign	53.7	55.8		55	method	949.8	298.9
6	block	56.9	38.4		56	model/modeling	25.1	13.6
7	boolean	82.0	19.8		57	module	1.3	235.8
8	branch/branching	3.3	3.1		58	nest/nested	23.0	22.4
9	case	127.0	81.0		59	number/numeric	251.4	319.7
10	character	120.0	119.6		60	object	645.0	336.7
11	class	987.4	297.0		61	operation/operator	139.1	157.7
12	code	213.2	300.6		62	output	106.8	80.0
13	component	100.4	17.2		63	parameter	92.7	84.0
14	condition/conditional	49.1	53.1		64	pattern	37.1	32.5
15	constant	63.1	6.6		65	pointer	4.2	2.8
16	constructor	141.1	9.9		66	polymorphism	5.5	2.5
17	control	61.7	22.7		67	problem	63.9	57.9
18	correct/correctness	21.2	18.1		68	procedure	4.7	1.6
19	data	133.5	175.5		69	process/processing	61.7	74.0
20	debug/debugging	8.1	15.0		70	program	460.6	462.1
21	declaration/declare	80.9	7.6		71	quality	0.6	1.5
22	decomposition/decompose	0.3	0.0		72	queue	16.1	0.6
23	definition/define	134.3	95.1		73	record	7.9	6.9
24	design	49.2	12.3		74	recursion/recursive	25.0	28.0
25	development/develop	23.9	27.5		75	reference	84.2	34.4
26	documentation/document	40.5	44.0		76	relation/relational	5.4	6.6
27	dynamic/dynamically	9.3	7.6		77	requirement/require	29.9	42.8
28	efficient/efficiency	12.7	9.9		78	reserved	5.1	3.9
29	encapsulation/encapsulate	9.3	4.0		79	scope	12.5	7.7
30	error	77.9	102.9		80	selection	13.1	10.9
31	event	152.8	37.7		81	sequence	50.3	67.2
32	exception	125.3	89.7		82	set	142.4	116.7
33	expression	98.1	111.0		83	signature	7.9	1.5
34	file	216.9	372.0		84	software	20.2	21.1
35	floating/floating-point	13.5	16.7		85	solution/solve/solving	32.1	48.1
36	function	24.8	601.9		86	specification/specify	55.5	39.5
37	identifier	11.8	9.8		87	stack	56.2	9.7
38	implementation/implement	144.4	45.2		88	statement	212.1	203.1
39	index	60.5	74.2		89	stream	59.1	5.1
40	information	68.4	72.2		90	string	399.8	410.4
41	inheritance/inherit	44.1	21.1		91	structure	33.5	44.7
42	input	74.6	128.9		92	style	21.1	17.7
43	instance	137.3	110.4		93	system	253.7	55.5
44	integer	116.0	94.0		94	test/testing	85.5	138.2
45	interface	161.0	44.4		95	thread	188.2	0.6
46	iteration/iterate	11.7	20.5		96	tree	16.8	19.6
47	keyword	21.4	23.1		97	type	369.5	204.0
48	line	146.4	263.7		98	user	110.9	151.7
49	link/linked	18.9	17.4		99	value	477.5	451.1
50	list	137.1	487.0		100	variable	288.6	164.8

Agile Learning: Sprinting Through the Semester

Guido Lang
guido.lang@quinnipiac.edu
Quinnipiac University
Hamden, CT 06518

Abstract

This paper introduces agile learning, a novel pedagogical approach that applies the processes and principles of agile software development to the context of learning. Agile learning is characterized by short project cycles, called sprints, in which a usable deliverable is fully planned, designed, built, tested, reviewed, and launched. An undergraduate elective Computer Information Systems course on web development was redesigned to implement a semester-long agile learning experience. Results of a student survey conducted at the end of the semester reveal that agile learning combines learning and application of learning, while allowing students to fail more and fail faster. At the same time, agile learning takes longer than traditional project-based learning and makes it easier for students to fall behind. Nevertheless, students indicated a strong preference for agile learning over traditional project-based learning. Importantly, students' preference for and performance in agile learning was not influenced by their learning style. However, agile learning requires significant amount of planning, balancing the need to provide instructions with the need to provide explanations, as well as significant amount of one-on-one student support.

Keywords: agile learning, pedagogy, learning style

1. INTRODUCTION

The proliferation of massive open online courses (MOOCs) with the goal of "learn how to code" has spurred the development of innovative pedagogical approaches that have the potential to disrupt Information Systems (IS) education, as well as higher education in general (Drachsler & Kalz, 2016; Fox, 2016). For example, popular MOOCs offered by Code Academy (https://www.codecademy.com/), Treehouse (https://teamtreehouse.com/), and One Month (https://onemonth.com/) teach various aspects of coding by guiding students through the iterative development of multiple increasingly sophisticated software applications.

I term this pedagogical approach "agile learning." Agile learning applies the processes and principles of agile software development to the context of learning. It is characterized by short project cycles, called "sprints," in which a usable deliverable is fully planned, designed, built, tested, reviewed, and launched. Through several sprints, students iteratively expand and improve the deliverables. Agile learning stands in contrast to traditional project-based learning, which is often characterized by a linear process through which students develop deliverables (Lee, Huh, & Reigeluth, 2015; Melles et al., 2015).

The present work reports the results of a first implementation of agile learning in the context of undergraduate Computer Information Systems (CIS) education. In particular, this work addresses the following research questions:

(1) What are the advantages and disadvantages of agile learning, as perceived by students?

(2) Do students prefer agile learning to traditional project-based learning?

(3) Does learning style affect students' preference for and performance in agile learning?

(4) What are the challenges of designing and implementing an agile learning experience?

The following sections describe agile learning, the methodology, the results, as well as the contributions and limitations of this research.

2. AGILE LEARNING

Traditional project-based learning is often implemented in a linear process that begins with theoretical lectures before asking students to plan, design, build, test, review, and ultimately launch a useable deliverable (Lee, Huh, & Reigeluth, 2015; Melles et al., 2015). Similar activities are part of nearly all student projects – such as an English paper, a financial report, or a marketing presentation. Interestingly, traditional project-based learning was popularized in the early 2000s, a time when the traditional "waterfall" systems development methodology was prevailing (Condliffe et al., 2015; Matkovic & Tumbas, 2010). Just like project-based learning, traditional systems development involves executing the aforementioned activities in a linear fashion. Figure 1 depicts the traditional project-based learning process.

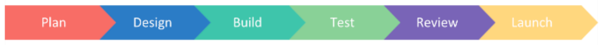


Figure 1: Traditional Project-Based Learning Process

I propose the term agile learning to refer to the application of the processes and principles of agile software development to the context of learning. Agile software development is characterized by short development cycles, called sprints, in which a working software application is fully planned, designed, built, tested, reviewed, and launched (Anand & Dinakaran, 2016; Matharu et al., 2015). Through several sprints, developers iteratively expand and improve the software application. In the context of learning, development cycles and working software applications are replaced by project cycles and useable deliverables, respectively. In other words, an agile learning experience consists of multiple short project cycles, called sprints, in which a useable deliverable is fully planned, designed, built, tested, reviewed, and launched. One of the defining features of agile software development – and by extension agile learning – is the fact that each sprint ends with a useable deliverable that is increasingly being expanded and improved upon. Although originally introduced in the early 2000s, agile software development only became widely adopted in the last few years (Anand & Dinakaran, 2016). Figure 2 depicts the agile learning process.

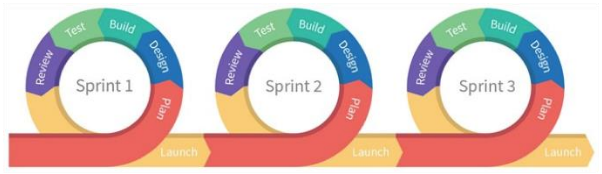


Figure 2: Agile Learning Process

In addition to the above-mentioned processes, agile learning also applies the four principles of agile software development to the context of learning (Jørgensen et al., 2015; Bustard & Keenan, 2009). The four principles of agile software development were first stated in the "Manifesto for Agile Software Development" (Beck et al., 2001), which was drafted by 17 leading software development experts that recognized the need for an alternative to documentation-driven, heavyweight software development processes.

The first agile principle is "individuals and interactions over processes and tools." Applied to the context of learning, it suggests for the instructor to focus on working with students one-on-one and to be flexible in adjusting the processes and tools used in the classroom. The second agile principle is "working software over comprehensive documentation," which suggests shifting the focus from students writing reports to students producing something that can be used in a professional environment. The third agile principle is "customer collaboration over contract negotiation." Applied to learning, it suggests for the instructor to collaborate with students instead of strictly enforcing assignments and associated rules. Lastly, the fourth agile principle is "responding to change over following a plan," which further emphasizes the need for the instructor to be willing to depart from the traditional semester-long course schedule and instead to adjust the schedule in response to students' needs as they arise. The goal of the agile learning principles is to improve the instructor's ability to facilitate learning in an agile learning experience.

4. METHODOLOGY

Course and Content Development

An undergraduate elective Computer Information Systems (CIS) course on Web Development (CIS 381) at Quinnipiac University was completely redesigned to implement a semester-long agile learning experience. CIS 381 guided students through the process of building web applications from idea to deployment, placing an equal emphasis on front and back end aspects of web development. Student learning objectives of the course included:

- Evaluate and justify choices in design patterns and technologies used in web development
- Explain and configure the fundamental structure of a web application
- Implement responsive design in a web application frontend using Bootstrap
- Develop a secure web application backend using Django/Python
- Understand and implement the basic principles of web services from the perspective of both the client and service provider

Students developed web applications that adhere to industry best practices and leverage professional tools, such as Django (web application framework), Github (version control system), Bootstrap (front end library), Nitrous (cloud-based IDE), and Heroku (deployment platform). The author of this work was the instructor for this course in Fall 2015 (N = 37).

The semester was divided into four sprints of increasing length (i.e. 1 week for sprint 1, 2 weeks for sprint 2, 3 weeks for sprint 3, and 4 weeks for sprint 4). Each sprint consisted of a web development project that required students to repeat and add to the work conducted in the previous sprint. For example, in sprint 1, students developed a simple splash page. In sprint 2, students developed a more advanced single-page website. In order to complete sprint 2, students had to repeat most of the steps from sprint 1 before being introduced to new content. The instructions in sprint 2 asked students to identify and repeat the necessary steps from sprint 1 on their own, before giving them step-by-step instructions for the new aspects of sprint 2. This continued until sprint 4, when students were asked to develop a complex web application with few instructions, thus requiring them to apply their learning from the previous sprints.

With each sprint, students were given increasing creative freedom over the actual content of their web development project. For example, while the first sprint required all students to implement the same project, the last sprint specified only functional requirements and gave students full control over the content domain. Requirements of sprint 4 included "the web app shall include user account management," "the web app shall allow users to view, add, edit, and delete objects and related objects," and "the web app shall include a search function." At the end of sprint 4, students had developed different web apps featuring e.g. restaurant reviews, college sports forums, and travel logs. Holding the functional requirements constant across all students reduced the complexity of potential technical problems and thus allowed the instructor to assist each student throughout the sprints (without the help of a teaching assistant). Table 1 provides an overview of the four sprints.

Sprint	Duration	Project
1	1 week	Splash page
2	2 weeks	Landing page
3	3 weeks	Web app
4	4 weeks	Final project

Table 1: Overview of the Sprints

The course also applied the agile learning principles. For example, students were provided instructions through video tutorials (that were recorded by the instructor), which gave the instructor time to respond to students' questions and work with them one-on-one. Students developed working websites of increasing sophistication, using professional tools, and industry best practices, as needed. The above-mentioned agile learning process was used as a guideline and not as a strict process – thus allowing the instructor to adjust the pace and deliverables to students' needs.

Data Collection and Analysis

At the end of the semester, students completed a survey, which measured perceived advantages and disadvantages of agile learning, preference for agile learning over traditional project-based learning, and learning style. The survey included definitions of agile learning and traditional project-based learning, thus allowing students to draw on their personal experience when comparing the two pedagogical approaches. A total of N_{Final} = 34 students completed the survey for extra credit (worth approximately 5% of their final grade), for a response rate of 92%. The open-ended questions were analyzed using qualitative content analysis (Mayring, 2000). Learning style was measured using the Learning Style Inventory (LSI; Kolb & Kolb, 2005), which asks participants to rank the endings of 12 sentences according to how well they think each one fits with how they would go about learning something. Detailed instructions on the LSI, including how to calculate the learning style dimensions, can be found in Kolb and Kolb (2005). The survey was not anonymous, thus allowing me to correlate students' responses with their performance in the course. The full survey instrument can be found in Appendix A.

The challenges regarding the design and implementation of the agile learning experience are the outcome of reflection-on-action performed by the instructor (Schön, 1983).

5. RESULTS

Advantages and Disadvantages

Four themes – two advantages and two disadvantages – emerged from the qualitative content analysis of students' responses to the open-ended questions. The first major advantage of agile learning, as perceived by the students, is that it combines learning and application of learning. By introducing new concepts, as they are needed, and immediately applying these concepts in practice, students are able to decrease the time lag between learning and the application of learning. As stated by one student: "I'm able to implement what I'm learning right away instead of waiting until I learn other material and then having to do everything at once."

The second major advantage of agile learning is that it allows students to fail more and fail faster. By going through multiple iterative projects, or sprints, students are able to recognize the shortcomings of their understanding more often and faster than in traditional project-based learning. One student observed: "I know exactly where my weak points are and can easily fix them because I know what portion or part I'm having trouble with." Likewise, another student stated "you can see your mistakes and areas that you can improve on while working on different projects."

The first major disadvantage is that agile learning takes longer than traditional project-based learning. As agile learning involves iteration and repetition, it is likely that traditional project-based learning conveys the same amount of learning material in a shorter amount of time. In line with this concern, one student remarked that "maybe it takes longer but that did not seem to be a problem here because each project led up to the big final project."

The second major disadvantage of agile learning is that it is easier for students to fall behind than in traditional project-based learning. Since students are working, hands-on, on projects in every single class, they are required to stay up-to-date – especially when they miss class. As one student put it, "[it] is very necessary to be on top of the work, it was very important to go to class and to follow along with the lessons and be able to ask questions." Similarly, another student noted that "if a student did not understand one concept taught early, they could fall behind. All of the concepts are built off of each other and if you miss one section you could end up very lost."

Preference for Agile Learning

The three items measuring students' preference for agile learning (i.e. "I prefer agile learning to project-based learning", "I believe agile learning helps me achieve my learning better than project-based learning", "I wish more classes would use agile learning") are highly correlated (all *r*s > .60, *p*s < .001), as is also evident in the aggregate responses shown in Figure 3.

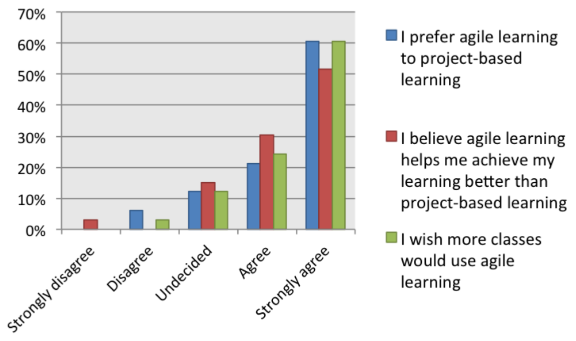


Figure 3: Preference for Agile Learning

Eighty-two percent of the participants agree or strongly agree with the statements "I prefer agile learning to project-based learning" (*M* = 4.30, *SD* = 1.10) and "I believe agile learning helps me achieve my learning better than project-based learning" (*M* = 4.27, *SD* = .94). Moreover, 85% of the participants agree or strongly agree with the statement "I wish more classes would use agile learning" (*M* = 4.42, *SD* = .83). Taken together, these responses indicate a strong preference for agile learning over project-based learning.

Influence of Learning Style

The students exhibited a diverging learning style, which is characterized by an emphasis of Concrete Experience (CE; *M* = 36.15, *SD* = 4.85) over Abstract Conceptualization (AC; *M* = 28.15, *SD* = 5.44) and Reflective Observation (RO; *M* = 30.73, *SD* = 5.60) over Active Experimentation (AE; *M* = 24.97, *SD* = 5.03). The participants' aggregate learning style profile is shown in Figure 4.

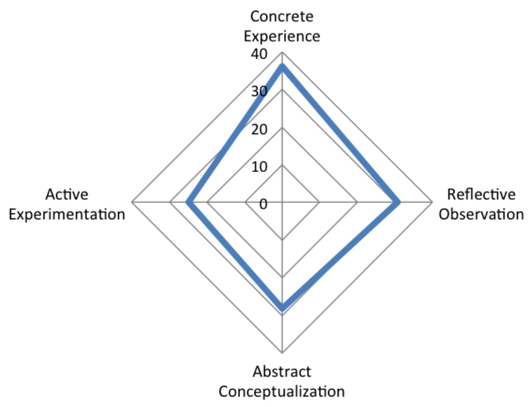


Figure 4: Learning Style Profile

Individuals with a diverging learning style are best at viewing concrete situations from many different points of view. They tend to perform better in situations that call for generation of ideas, such as brainstorming sessions (Kolb & Kolb, 2005). It is noteworthy that the diverging learning style is highly atypical of students in CIS/IS. As previous research has shown, the prevalent learning style among CIS/IS students is assimilating (Kolb & Kolb, 2005). Individuals with an assimilating learning style are best at understanding a wide range of information and putting it into concise, logical form. To better understand if and to what extent learning style might influence preference for and performance in agile learning, two multivariate regression analyses were performed.

The first multivariate regression analysis was used to test if the learning style dimensions (i.e. CE, AC, RO, AE) predict the preferences for agile learning (i.e. "I prefer agile learning to project-based learning", "I believe agile learning helps me achieve my learning better than project-based learning", "I wish more classes would use agile learning"). Results suggest that learning style does not affect preference for agile learning ($F(3, 29) = .66$, $p > .05$).

The second multivariate regression analysis was used to test if the learning style dimensions predict students' performance in the course (i.e. assignment grades, midterm grade, final grade). Results suggest that learning style does not affect performance in agile learning ($F(3, 29) = 2.15$, $p > .05$). Moreover, the student performance in this course (as measured by the assignment grades) suggests that the agile learning approach allowed students to achieve the stated learning objectives (M = 86.13%, SD = 21.43%). Taken together, these findings suggest that learning style, as measured by the LSI, does not influence preference for and performance in agile learning.

Challenges

Three challenges for the design and implementation of agile learning became apparent from the instructor's reflection-on-action: First, agile learning requires a significant amount of planning by the instructor. As each sprint repeats and builds upon the previous sprint, it is crucial that the projects are chosen and developed in a way that introduces increasingly complex concepts over time.

Second, agile learning requires balancing the need to provide students with step-by-step instructions on how to do something with the need to provide students with explanations on why to do something. As students are in the midst of a sprint, it is often easier to just give instructions on what to do next than to step back and explain why something needs to be done.

Third, agile learning requires significant amount of one-on-one student support from the instructor. Given that students work hands-on for almost the entire semester, many problems and questions arise that need to be addressed one-on-one with the instructor. Since this implementation of agile learning made extensive use of online videos, the instructor was able to address most of the problems and questions in class.

6. CONTRIBUTIONS AND LIMITATIONS

The present work contributes to IS education in two ways: First, it introduces the concept of agile learning, which has hitherto not been explored in the IS and education literatures. This has the potential to improve our understanding of teaching and learning and lays the groundwork for future research in this area. Second, it implemented and evaluated agile learning in an undergraduate CIS course. This, in turn, has the potential to improve the practice of teaching and learning in IS and beyond.

However, the present work is not without limitations. First, the design and implementation of the agile learning experience did not follow previously established guidelines. As such, it is possible that one could have designed and implemented a purer agile learning experience and thus conducted a better test of the viability of agile learning in IS education. Second, the quantitative and qualitative results must be seen in light of the relatively small sample size and students exhibiting a learning style that is unusual of CIS/IS students. Future research is clearly needed to replicate and deepen the insights derived from this work.

7. CONCLUSION

The present work introduces agile learning. Agile learning is a novel pedagogical approach that applies the processes and principles of agile software development to the context of learning. Agile learning was implemented and subsequently evaluated in an undergraduate CIS course. Results of a student survey suggest that agile learning combines learning and application of learning, while allowing students to fail more and fail faster. At the same time, agile learning takes longer than traditional project-based learning and makes it easier for students to fall behind. Nevertheless, students indicated a strong preference for agile learning over traditional project-based learning. Importantly, students' preference for and performance in agile learning was not influenced by their learning style, as measured by the LSI. From the instructor's point of view, agile learning requires significant amount of planning, balancing the need to provide instructions with the need to provide explanations, as well as significant amount of one-on-one student support. This work opens avenues for future research on the potential of agile learning in IS education and beyond.

8. REFERENCES

Anand, R., & Dinakaran, M. (2016). Popular Agile Methods in Software Development: Review and Analysis. *International Journal of Applied Engineering Research*, 11(5), 3433-3437.

Beck, K. et al. (2001). Agile Manifesto for Software Development. Retrieved 5/18/2016 from http://agilemanifesto.org

Bustard, D., & Keenan, F. (2009). Soft Systems Methodology: An Aid to Agile Development. In Barry, C., Conboy, K., Lang, M., Wojtkowski, G., Wojtkowski, W. (Eds.), *Information Systems Development: Challenges in Practice, Theory, and Education*, New York: Springer, 25-38.

Condliffe, B., Visher, M. G., Bangser, M. R., Drohojowska, S. & Saco, L. (2015). Project-Based Learning: A Literature Review. MDRC. Retrieved 5/18/2016 from https://s3-us-west-1.amazonaws.com/ler/MDRC+PBL+Literature+Review.pdf

Drachsler, H., & Kalz, M. (2016). The MOOC and Learning Analytics Innovation Cycle (MOLAC): A Reflective Summary of Ongoing Research and Its Challenges. *Journal of Computer Assisted Learning*, 32, 281-290.

Fox, R. (2016). MOOC Impact Beyond Innovation. In: Ng, Chi-hung Clarence, Fox, Robert, Nakano, Michiko (Eds.), *Reforming Learning and Teaching in Asia-Pacific Universities* (Education in the Asia-Pacific Region: Issues, Concerns and Prospects, Volume 33), Berlin: Springer, 159-172.

Jørgensen, M. T. N., Hovmøller, H., Nielsen, J. R., & Tambo, T. (2015). Improving offshoring of Low-Budget Agile Software Development Using the Dual-Shore Approach: An Autoethnographic Study. *Proceedings of 36th Information Systems Research in Scandinavia Seminar*, 2-17.

Kolb, A. Y., & Kolb, D. A. (2005). The Kolb Learning Style Inventory—Version 3.1 2005 Technical Specifications. Department of Organizational Behavior, Weatherhead School of Management, Case Western Reserve University, Cleveland, OH,

Lee, D., Huh, Y. & Regeluth, C. (2015). Collaboration, Intragroup Conflict, and Social Skills in Project-Based Learning. *Instructional Science*, 43(5), 561-590.

Matharu, G., Mishra, A., Singh, H., Upadhyay, P. (2015). Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. *ACM SIGSOFT Software Engineering Notes*, 40(1), 1-6.

Matkovic, P., & Tumbas, P. (2010). A Comparative Overview of the Evolution of Software Development Models. *Journal of Industrial Engineering and Management*, 1(4), 163-172.

Mayring, P. (2000). Qualitative Content Analysis. *Qualitative Social Research*, 1(2), Art. 20, Retrieved 5/18/2016 from http://nbn-resolving.de/urn:nbn:de:0114-fqs0002204.

Melles, G., Anderson, N., Barrett, T., Thompson-Whiteside, S. (2015). Problem Finding through Design Thinking in Education. In Patrick Blessinger, John M. Carfora (Eds.), *Inquiry-Based Learning for Multidisciplinary Programs: A Conceptual and Practical Resource for Educators* (Innovations in

Higher Education Teaching and Learning, Volume 3), Bingley, UK: Emerald, 191-209.

Schön, D. (1983) *The Reflective Practitioner. How Professionals Think in Action*. London: Temple Smith.

Editor's Note:

This paper was selected for inclusion in the journal as a EDSIGCon 2016 Distinguished Paper. The acceptance rate is typically 7% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a paper in 2016.

Appendix A: Survey Instrument

At the beginning of the survey, students were provided the following introduction to agile learning:

"This course used a novel pedagogical approach called agile learning. Agile learning contrasts traditional project-based learning, which is often implemented in a linear process that begins with theoretical lectures before asking students to plan, design, build, test, review, and ultimately launch a useable deliverable. An agile learning experience consists of multiple short project cycles, called sprints, in which a useable deliverable is fully planned, designed, built, tested, reviewed, and launched. Over the course of the semester, you completed four sprints: the splash page, the landing page, the web app, and the final project."

Advantages and Disadvantages:
The following were open-ended questions.

What would you say are the advantages of agile learning (compared to project-based learning)?

What would you say are the disadvantages of agile learning (compared to project-based learning)?

Preference for Agile Learning:
The following items were answered using a 5-point Likert scale, labeled 1 – Strongly disagree; 2 – Disagree; 3 – Undecided; 4 – Agree; 5 – Strongly agree.

I prefer agile learning to project-based learning.

I believe agile learning helps me achieve my learning better than project-based learning.

I wish more classes would use agile learning.

Kolb Learning Style Inventory:

(See Smith & Kolb, 1985)

Testing Frequency in an Introductory Computer Programming Course

Joni K. Adkins
jadkins@nwmissouri.edu

Diana R. Linville
dianar@nwmissouri.edu

School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468, USA

Abstract

This paper reports the findings of a study done to determine if increasing the number of exams in a course had an effect on student grades. Some studies have found that more frequent exams positively influence scores while other studies have found more frequent exams do not make a difference in student achievement. This study examines the impact of adding two additional exams to an introductory computer programming course taken by undergraduate computer science, information systems, and other STEM majors. The findings did not show any significant differences in student performance between the fall classes that took three exams and the spring classes that took five exams. In addition a survey was given to discover student attitudes and preferences regarding exam frequency and scheduling. The survey results revealed students want more exams in courses to reduce anxiety and increase confidence and motivation to study.

Keywords: exam frequency, testing frequency, number of exams, computer programming, information systems, computer science

1. INTRODUCTION

Educators often consider and modify the assessment plan in an attempt to improve student learning and achievement. Testing frequency and its impact on student performance have been studied for years. The number of exams and quizzes administered is generally limited due to faculty resources (Kuo & Simon, 2009). The additional work for an instructor to conduct frequent testing in their courses can be daunting. In addition, administering exams consumes valuable instruction time that could be used for classroom learning (Roediger & Karpicke, 2006; Mines, 2014; Leeming, 2002). Therefore, fewer exams, perhaps a midterm and final, are common in many college classrooms. Some believe that students will study less when given more exams because the overall weight of each exam on the overall class grade is lower (Mines, 2014).

Studies have provided conflicting results as some show support that frequent testing in the classroom improves student performance (Leeming, 2002; Kling, McCorkle, Miller, & Reardon, 2005; Gholami & Moghaddam, 2013) while others find that there is no statistical difference in student performance with less frequent testing (Murphy & Stanga, 1994; Mines, 2014). The relevant literature does not come to a definite consensus on the impact frequent testing has on student performance in a course.

2. TESTING FREQUENCY AND STUDENT PERFORMANCE

Faculty can look to the literature for evidence about the appropriate number and frequency of exams. A seminal article by Bangert-Drowns, Kulik, and Kulik (1991) analyzed several other studies and found positive effects of frequent testing in 29 of the 35 studies; 13 of the positive studies and 1 of the negative studies were statistically significant. Another meta-analysis revealed that more frequent exams and student performance were not correlated in a linear fashion (Kuo & Simon, 2009). In other words, adding another exam to a course becomes less significant as the number of exams increases.

More frequent exams means that there is less material for students to learn (Bangert-Drowns, et al., 1991), students may prepare better instead of procrastinating, and students are able to receive more feedback (Mines, 2014). Kling, et al. (2005) investigated the impact of frequent testing on student performance in a marketing course. Their study included 2 sections of a marketing course where 1 section was given 12 quizzes during the semester while the other section was given 3 exams. Both sections were given the same final exam at the end of the semester. Their findings suggested that students retain information better in frequent testing environments with high content overlap (Kling, et al., 2005).

Leeming (2002) conducted a study in a psychology course to determine if a student's performance improved when given an exam every day in class. He proposed that students who did poorly in the course had the ability to learn but just did not study enough (Leeming, 2002). The results showed that grades were significantly higher when students were given an exam every day. Leeming (2002) also found that students in the exam-a-day course outperformed students in the traditional course on a retention test and that fewer students withdrew from the course. A similar experiment to analyze the effect of weekly quizzes on final achievement tests in high school students was conducted by Gholami and Moghaddam (2013). The study included 70 students in different classes taught by the researchers. The classes were split up into an experimental group who received weekly quizzes, and a control group who only received a mid-term and a final. The results indicated the experimental group who took weekly quizzes did significantly better than the control group on the final achievement test (Gholami & Moghaddam, 2013).

The increased performance may be explained by the testing effect. The testing effect is the "phenomenon of improved performance from taking a test" contending that testing both measures and changes knowledge, leading to increased performance (Roediger & Karpicke, 2006, p. 181). In a study by Butler and Roediger (2007), participants watched a lecture and then either studied a lecture summary, took a multiple choice test, took a short answer test, or did nothing. One month later, the participants took a comprehensive exam. Butler and Roediger (2007) found that all review methods improved the participant's score on the final exam with the short answer exam having the most impact. An examination of several studies on testing memory concluded that "repeatedly studying material is beneficial for tests given soon after learning, but on delayed critical tests with retention intervals measured in days or weeks, prior testing can produce a greater performance than prior studying" (Roediger & Karpicke, 2006, p. 189). Thus the testing effect may play a role if the students have more exams in a class, leading them to study and learn from the exams.

College faculty realize that students often wait until right before an exam to begin their studying. These intense "cramming" sessions are encouraged by less frequent exams while more frequent exams may lead to more continuous studying (Roediger & Karpicke, 2006). Michael (1991) called this the procrastination scallop, where students wait until they have a test to begin their studying. Therefore, if more frequent exams were given, students may study more (Michael, 1991).

More frequent testing is not always found to have a significant effect on student performance. Murphy and Stanga (1994) examined the effects of frequent testing in an introductory income tax course. Their experiment used four sections of a single course taught by the same instructor where two sections were given six exams prior to the final and two sections were given three exams before the final. The questions on the exams and the final exam were exactly the same. There was no significant difference in final exam scores between the two groups (Murphy & Stanga, 1994). In another study, Mines (2014) examined the relationship between testing frequency and the final grade in an environmental engineering course. The study looked at data from ten course offerings between the years of 2001 and 2012. The statistical data showed that testing frequency had little effect on a student's final grade (Mines, 2014). Due to conflicting evidence, there is still a need for more research to confirm or refute the effect of frequent testing (Ramshe, 2014).

3. TESTING FREQUENCY AND STUDENT ATTITUDES

Another factor to consider when determining the right number of exams is student preference. While the literature reports inconsistent results about student achievement and frequent testing, studies regularly show that students prefer frequent testing (Leeming, 2002; Kling, et al, 2005; Kuo & Simon, 2009). Regardless of these findings, a common pedagogy in college courses remains a midterm exam and a final exam (Roediger & Karpicke, 2006).

The findings are fairly consistent across research areas regarding student's attitude towards the course and instructor (Kuo & Simon, 2009). Bangert-Drowns, et al. (1991) completed a meta-analysis on the effects of frequent classroom testing. Student attitude towards frequent testing was evaluated in four of the studies they examined, and the results showed that students who had more frequent exams rated their instruction more positively. They concluded that by frequently testing students there is a positive effect on the classroom environment (Bangert-Drowns, et al., 1991). Leeming (2002) had students complete a questionnaire at the end of the term regarding specifically the exam-a-day procedures. One question posed was "Given a choice, I would choose this procedure over just a few exams." with the overwhelming majority agreeing (Leeming, 2002). Students indicated they studied and learned more (Leeming, 2002). Attendance may also be positively affected by more frequent exams (Leeming, 2002) while also reducing test anxiety (Kuo & Simon, 2009; Kling, et al, 2005; Gholami & Moghaddam, 2013).

Student evaluations of the instructor is another way to access student attitude. Murphy and Stanga (1991) used the end-of-term student evaluations to assess if there was an adverse effect on instructor evaluations when frequent exams were given. Students in their experimental group who took more exams during the semester, felt stronger about the benefits of the course and the effectiveness of the instructor's teaching (Murphy & Stanga, 1994). While these students also indicated that they felt less anxiety before taking an exam, course evaluations for both groups were favorable. In addition, comments given as feedback on the evaluations supported their conclusion that students prefer frequent testing (Murphy & Stanga, 1994). Kling, et al., (2005) also used the instructor evaluations at the end of the term to assess student attitude. They hypothesized that frequent testing would improve the instructor evaluations at the end of the term. Their results revealed that instructor feedback was higher when more quizzes were given throughout the semester (Kling, et al., 2005).

The researchers in this study were interested in finding if adding exams to an introductory computer programming class would improve learning but were unable to find studies regarding testing frequency in computer programming courses. Another reason for questioning the exam frequency was growing enrollments have made finding empty classrooms for evening exams a challenge. Therefore, changes were made to the testing plan for the spring classes. Keeping in mind the research on performance and student attitudes in regards to testing frequency, the researchers tested to see if more exams in an introductory computer programming course would lead to improved average scores on individual exams, the final exam, and the overall course. In addition, student attitude towards frequency of exams was also assessed.

4. METHOD

Data from two instructors who taught Computer Programming I in both the spring and fall were used in the study. Course materials used in all sections in the study were consistent with the same assignments, projects, and the same or similar quizzes and exams. All students were given a textbook and had access to the same instructor-generated materials including notes, videos, and exercises. All sections of the course enforced the same attendance policy where students lost points after three absences. Students enrolled in the course were mostly Missouri Academy students or undergraduate freshmen with majors in computer science, management information systems, interactive digital media, or another STEM field.

All students in the fall introductory computer programming course sections met in the evening for three 90-minute exams, approximately five weeks apart. The students in the spring course sections took five exams during their regularly scheduled 50-minute class. The exams were given approximately every three weeks. The total number of exam points in the class did not change. The three exams in the fall were worth 100 points each, and the five exams in the spring were worth 60 points each. The students in the fall course took the same instructor- generated exams. The instructor-generated exams in the spring course were slightly changed so that a student taking the exam later in the day would not have an advantage from learning about exam questions. An example of a modification is shown below.

Question 1 Section 1:
```
        s = "hello"
        r = "world"
        What is the output of the following operation:
        print(s * 3)
```

Question 1 Section 2:
```
        s = "python"
        r = "programming"
        What is the output of the following operation:
        print(r * 2)
```

All fall students took the same 200-point comprehensive final exam while the spring students took similar versions of the final exam, again to prevent later sections from having an advantage of learning exam answers.

At the end of the spring term, students who took the course in either the fall or spring were invited to participate in a survey that asked them about their exam frequency preference.

5. RESULTS

The exam scores were averaged to get an overall exam percentage for comparison. The final exam percentage and the final course percentage were also compared. An independent-samples t-test was conducted to compare average exam score, final exam score, and final course grade for fall students who took three exams and spring students who took five exams. While the exam average and the final grade for the spring class with five exams was higher than the fall class, there were no significant differences found in average exam score, final exam score, or final grade between the fall and spring groups. Table 1 shows the relevant statistics.

	Fall (n= 96)		Spring (n =111)		df = 205	
	M	*SD*	*M*	*SD*	*t*	*p*
Exam Avg.	.81	.18	.83	.11	.69	.49
Final Exam	.78	.24	.78	.21	.14	.89
Final Grade	.81	.19	.82	.13	.22	.83

Table 1: Results of t-tests

In addition to examining student performance, the survey allowed the authors further insight into student perceptions of testing frequency. See Appendix A for the survey questions.

Frequency and distribution statistics were calculated on the survey questions with 5 representing strongly agree and 1 representing strongly disagree. Seventy percent of the students (n = 106) agreed or strongly agreed they preferred to have content broken into smaller and more frequent exams with a mean score of 3.89 (*SD* = 1.17). Fifty-five percent of the students (n = 106) agreed or strongly agreed they experienced increased anxiety with fewer exams in a course (*M* = 3.36, *SD* = 1.39). Eighty-one percent of the students (n = 106) preferred having more tests to provide frequent feedback so they could adjust their study skills (M = 4.01, *SD* = 1.01). Students were more confident in courses with multiple exams (n = 106, *M* = 3.85, *SD* = 1.12), and 69 percent indicated they were motivated to study more when there were frequent exams (n = 106, *M* = 3.71, *SD* = 1.17).

Seventy-seven percent of the students in the spring class (n = 96) thought their final grade in Computer Programming I would be higher due to having more frequent exams. Students overwhelmingly (n = 106) preferred taking an exam during the regularly scheduled class period instead of a scheduled evening exam with 88 percent selecting the class period.

6. DISCUSSION OF RESULTS

The results of this study were parallel to Mines (2014) and Murphy and Stanga (1994) as they also did not find a relationship between number of exams and student performance. One factor that might explain the absence of significant differences is the impact of more frequent testing decreases with each additional exam (Kuo & Simon, 2009; Bangert-Drowns, et al., 1991). For instance, adding one exam for a total of two exams is found to have a greater impact than adding a fourth exam for a total of five exams. Adding two more exams to the spring classes did not make an impact, perhaps because the fall group had three exams already.

The student survey results showed that the majority of students preferred more exams to fewer exams and thought fewer exams added anxiety. These findings about student exam frequency preference mirror other studies where students have indicated they preferred more exams (Kling, et al., 2005; Gholami & Moghaddam, 2013).

The analysis of written comments had one major theme: the difficulty of attending evening exams due to other commitments, primarily a job. A common evening exam has been used in this course for many years. It was established so all students would take the same exam at the same time and the time allotment could be longer than a regular class period. Many college students have evening commitments including college social or academic activities, part-time jobs, and athletic practices and events making evening exams difficult to attend. Given both the survey statistics and the written comments, more frequent exams will likely be given during the regular class period.

7. LIMITATIONS AND SUGGESTIONS FOR FUTURE RESEARCH

The final exam was worth 200 points which was approximately 25 percent of the course grade. Students know exactly how many points they need to earn on the final exam to get a specific grade in the class so they may only complete enough of the exam to earn those points. The final exam may not be the best indicator of overall student performance if they do not answer all questions they know. Student evaluations of the instructors could not be compared as the fall evaluations were not available due to an unsuccessful pilot of electronic evaluations. Exam scores and final grades in the data set were not associated with a student grade so analysis on different majors, gender, nationality, or GPA could not be performed. Other limitations include day versus night testing, limited sample size, minor differences in exams, and differences in instructor teaching styles and experience. Future research could study the number of exams given in other levels of programming or information systems courses, the impact of daily or weekly quizzes on performance, or test to find a better measure of overall student learning.

8. CONCLUSION

The number of exams to give in a course will continue to be explored since there is no conclusive evidence to determine whether frequent or infrequent exams have a greater impact on student learning. This study supports other research that shows that students prefer more frequent exams (Bangert-Drowns, et al., 1991; Leeming, 2002). The results showed students like more frequent exams due to decreased test anxiety, higher confidence in knowing the material, and increased motivation to study. College faculty relying on only a mid-term and final exam should reflect on these factors and consider adjusting the number of exams given.

9. REFERENCES

Bangert-Drowns, R. L., Kulik, J. A., & Kulik, C. C. (1991). Effects of frequent classroom testing. *Journal Of Educational Research, 85(2),* 89.

Butler, A. C., & Roediger, H. I. (2007). Testing improves long-term retention in a simulated classroom setting. *European Journal of Cognitive Psychology, 19*4-5), 514-527. doi:10.1080/09541440701326097

Gholami, V., & Moghaddam, M. M. (2013). The effect of weekly quizzes on students' final achievement score. *I.J.Modern Education and Computer Science, 1*, 36-41.

Kling, N., McCorkle, D., Miller, C., & Reardon, J. (2005). The impact of testing frequency on student performance in a marketing course. *Journal Of Education For Business, 81(2),* 67-72.

Kuo, T., & Simon, A. (2009). How many tests do we really need?. *College Teaching*, *57*(3), 156-160.

Leeming, F. C. (2002). The exam-a-day procedure improves performance in psychology Classes. *Teaching Of Psychology*, *29*(3), 210-212.

Michael, J. (1991). A behavioral perspective on college teaching. *The Behavior Analyst*, *14*(2), 229–239.

Mines Jr, R. O. (2014). The impact of testing frequency and final exams on student performance. *American Society for Engineering Education Southeast Section Conference.*

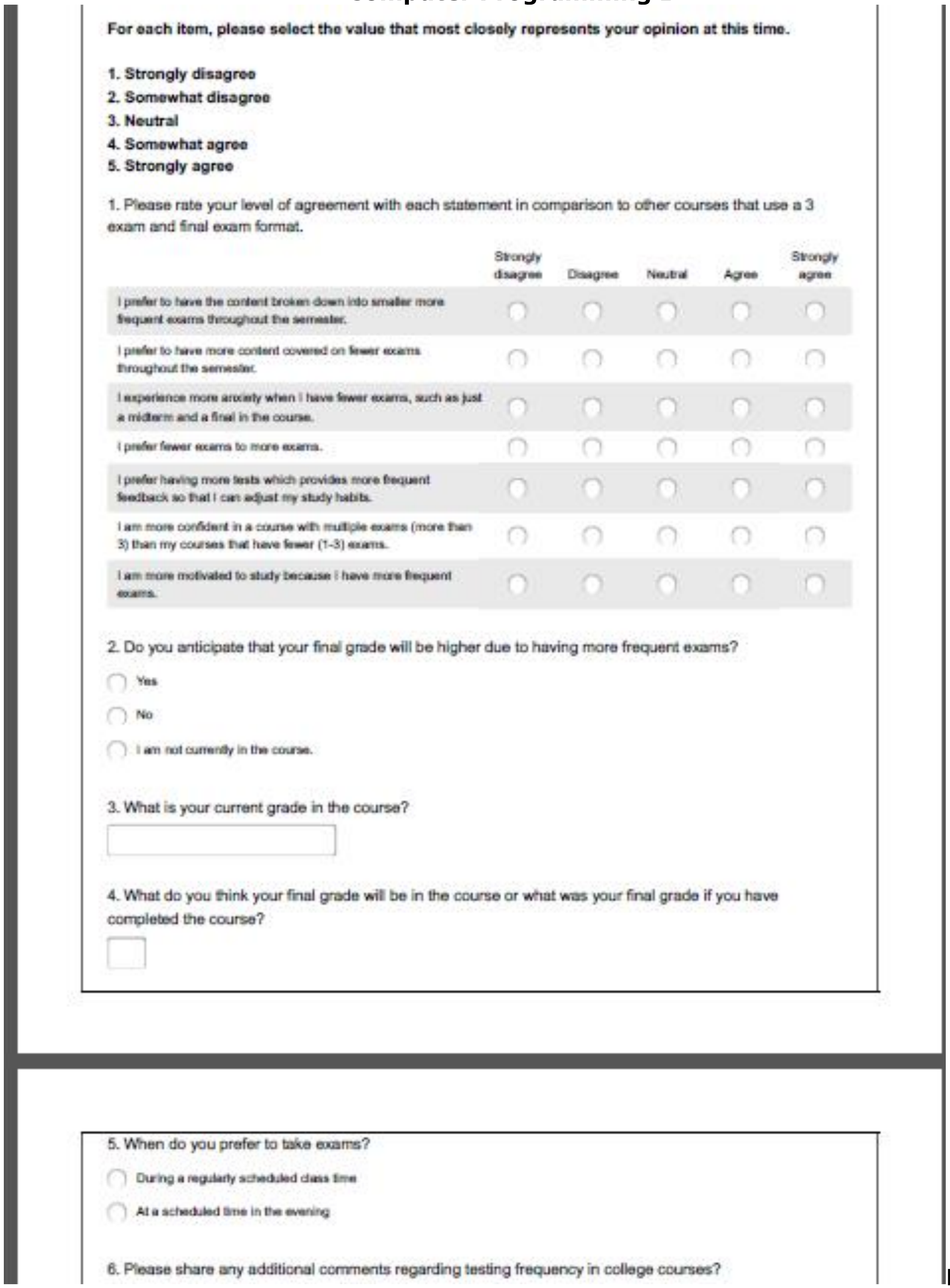
Murphy, D. P., & Stanga, K. G. (1994). The effects of frequent testing in an income tax course: An experiment. *Journal of Accounting Education, 12*(1), 27-41.

Ramshe, M. H. (2014). A review of the studies on the frequent administrations of englich tests. *Journal of Language Teaching and Research, 5*(6), 1412-1416.

Roediger III, H. L., & Karpicke, J. D. (2006). The power of testing memory basic research and implications for educational practice. *Perspectives On Psychological Science*, *1*(3), 181-210. doi:10.1111/j.1745-6916.2006.00012.x

Appendix A

Computer Programming I



For each item, please select the value that most closely represents your opinion at this time.

1. Strongly disagree
2. Somewhat disagree
3. Neutral
4. Somewhat agree
5. Strongly agree

1. Please rate your level of agreement with each statement in comparison to other courses that use a 3 exam and final exam format.

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I prefer to have the content broken down into smaller more frequent exams throughout the semester.	○	○	○	○	○
I prefer to have more content covered on fewer exams throughout the semester.	○	○	○	○	○
I experience more anxiety when I have fewer exams, such as just a midterm and a final in the course.	○	○	○	○	○
I prefer fewer exams to more exams.	○	○	○	○	○
I prefer having more tests which provides more frequent feedback so that I can adjust my study habits.	○	○	○	○	○
I am more confident in a course with multiple exams (more than 3) than my courses that have fewer (1-3) exams.	○	○	○	○	○
I am more motivated to study because I have more frequent exams.	○	○	○	○	○

2. Do you anticipate that your final grade will be higher due to having more frequent exams?

○ Yes

○ No

○ I am not currently in the course.

3. What is your current grade in the course?

4. What do you think your final grade will be in the course or what was your final grade if you have completed the course?

5. When do you prefer to take exams?

○ During a regularly scheduled class time

○ At a scheduled time in the evening

6. Please share any additional comments regarding testing frequency in college courses?

Exam Frequency Survey

Pursuing a Vendor-Endorsed ERP Award for Better Job Prospect: Students' Perceptions

LeeAnn Kung
kung@rowan.edu
Department of Marketing & Business Information Systems
Rowan University
Glassboro, NJ 08028, USA

Hsiang-Jui Kung
hjkung@georgiasouthern.edu
Department of Information Systems
Georgia Southern University
Statesboro, GA 30460, USA

Abstract

This paper identifies factors that motivate students to pursue a vendor-endorsed ERP award by integrating concepts from motivation theory and constructs from technology acceptance literature. We developed a web-based survey with closed- and open-ended questions to collect both quantitative and qualitative data, respectively. Students in information systems courses were solicited to participate in the survey. We collected data from 2010 to 2014. Our analysis shows that Perceived Value and Social Influence are significant predictors of students' intentions to pursue a vendor-endorsed ERP award.

Keywords: IT Certification, Perceived Value, IS Curriculum, Enterprise Resource Planning, UTAUT, Motivation.

1. INTRODUCTION

Information system (IS)/information technology (IT) majors often need something “extra” in addition to their college degrees to distinguish themselves from other IT job seekers especially in an economic recession. IT certification is one such means because possession of it indicates adequate knowledge and skills. Cantor (2002) categories two types of IT certification: vendor-specific and vendor-neutral. Vendor-specific is product-related such as Microsoft Office Specialist, CISCO career certifications, Oracle certifications, SAP TERP10 certificate and others. Vendor-neutral type of certificate focuses more on foundational concepts relative to underlying technology (Randall & Zirkle, 2005). The Computer Science and Telecommunications Board (2001) suggested that educational institutions partner with IT vendors and professional associations to offer IT certification training. Davis, Siau and Dhenuvakonda (2003) recommended that universities provide more real-world tool training such as Enterprise Resource Planning (ERP) and other systems. In recent years, a hybrid of certification has emerged, that is, vendors collaborate with higher education institutions to offer vendor-supported curricula and/or vendor-endorsed awards. The basic model is to use the vendor’s product as a tool (not as the focus) to teach students business concepts, methodology, and application of the technology. Especially in an economic recession and recovery, students think IS curriculum should be changed to be more “competitive” with more technical skills and business sense built in (Pratt, Hauser & Ross, 2010). The hybrid model fulfills this need. Scholars have noted that, if universities integrate industry certifications and academic degrees, it creates a win-win situation for both industry and academia and better help students get jobs (David, David, & David, 2011; Hitchcock, 2007; Simmonds, 2002). Due to its relative novelty, however, only a few studies have been conducted to evaluate the value of such programs for students and industry. This study fills the gap by investigating students' perceptions of whether a vendor-endorsed ERP award, specifically an SAP-endorsed program, provides better job outlooks. We describe the program next.

Systems, Applications & Products in Data Processing (SAP SE) is a German multinational software corporation that develops enterprise software to manage business operations and customer relations. SAP offers a University Alliances Program (UAP) to higher education institutions worldwide. Through UAP, universities gain access to SAP technologies and materials. They also enable University Alliances members to offer their students full access to the SAP Student Academies. SAP also offers SAP Student Recognition Award endorsement and an SAP specific certificate, the SAP TERP-10.

This study examined the SAP Student Recognition Award. In order to receive this award, students at the UAP universities must 1) have taken at least three courses that have a minimum of 30% hands-on SAP content in each course, and 2) have earned a grade of "C" or better in each course. In the United States, many UAP universities (e.g., Central Michigan University, Georgia Southern University, Rider University, etc.) offer the SAP Student Recognition Award. Students who earn the award demonstrate the breadth and depth of their knowledge using state-of-the-art software and valuable skills relevant to their careers and chosen fields. To earn the SAP TERP-10 certificate, students or people who are interested have to pay a hefty fee for a boot camp training and the certification exam.

To answer our research question of "what are the factors related to the students' intentions in pursuing the vendor-endorsed award", we draw concepts and constructs from motivation theory and technology acceptance which we describe in the next section.

2. THEORETICAL BACKGROUND

We wanted to study students' intentions to pursue a vendor-endorsed ERP award. In other words, we were curious about the "why". Psychologists and human behaviorists have been searching for the reasons behind human behavior for a long time. Motivation can also be defined as one's direction to behavior or what causes a person to repeat a behavior and vice versa (Covington, & Müeller, 2001). A motive is what prompts the person to act in a certain way or at least develop an inclination for specific behavior (Pardee, 1990). Over time, researchers have developed a number of different theories to explain motivation, for example, the incentive theory, the psychoanalytic theory, and the humanistic theory (Ajzen, 1991). Researchers have broadly accepted and explained motivation from two dimensions: intrinsic and extrinsic.

Intrinsic motivation refers to a motivation that is driven by an interest or enjoyment in the task itself and exists within the individual (Ryan, & Deci, 2000a) rather than relying on any external pressure. Students who are intrinsically motivated are more likely to willingly engage in the task as well as working to improve their skills, which will increase their capabilities (Wigfield, Guthrie, Tonks, & Perencevich, 2004). An example of *intrinsic motivation* in the workplace occurs when an employee becomes an IT professional because he or she wants to learn about how computer users interact with computer networks. The employee has the *intrinsic motivation* to gain more knowledge (Root, 2015).

Extrinsic motivation, on the other end of the spectrum, is the performance of an activity in order to attain an outcome. The sources of intrinsic and extrinsic motivation are different but not mutually exclusive (Ryan & Deci, 2000b). The source of *extrinsic motivation* comes from outside of the individual. The harder question to answer is where externally do people get the motivation to carry out and continue to persist. Usually *extrinsic motivation* is used to attain outcomes that a person would not get from *intrinsic motivation*. Two common types of *extrinsic motivations* are rewards such as money and grades, or coercion, and threat of punishment.

Many information systems researchers have published various theories that could be used to explain the adoption of information technology innovations (Teo, Wei, & Benbasat, 2003; Thompson, Higgins, & Howell, 1991; Venkatesh & Davis, 2000. Venkatesh and his colleagues reviewed and compared user acceptance models with the goal to develop a unified theory of technology acceptance (Venkatesh, Morris, Davis, & Davis, 2003). They integrated every major parallel aspect of user acceptance determinants from eight, earlier well respected models and named their proposed model the "Unified Theory of Acceptance and Use of Technology" (UTAUT). Venkatesh and his colleagues (2003) conducted longitudinal field studies across heterogeneous contexts and found three constructs to be significant predictors of intention: *performance expectancy*, *effort expectancy*, and *social influence*. These constructs are composite, which means they encapsulated the eight models of the similar concepts. UTAUT has been demonstrated to be up to 70 percent accurate at predicting user acceptance of information technology innovations while previous models had an average of 40 percent accuracy (Venkatesh et al., 2003). We combined the motivation concepts and major constructs from UTAUT pertaining to this study and established the research model. We present our research model and hypotheses in the next section.

3. RESEARCH MODEL

The research model is shown in Figure 1, followed by our hypotheses.

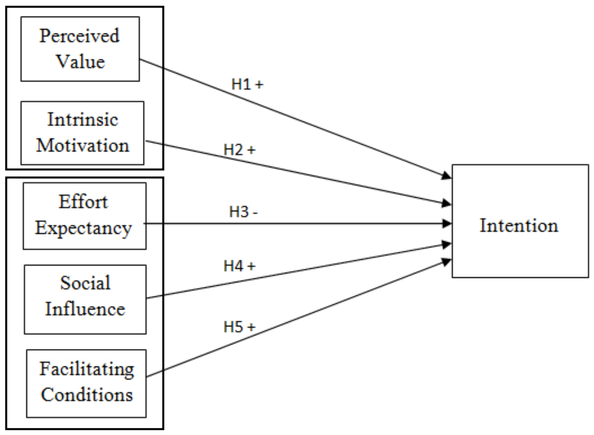


Figure 1: Research Model

Perceived Value

Motivation is the biological, social, emotional, or cognitive force that initiates, guides, and maintains goal-oriented behaviors. It is what causes an individual to take action, whether to enroll in college to earn a degree, or, in this study, to pursue an award. "Extrinsic motivation refers to the performance of an activity because it is perceived to be instrumental in achieving valued outcomes that are distinct from the activity itself, such as improved job performance, pay, or promotions…" (Davis, Bagozzi & Warshaw, 1992, p.1112). The primary focus of extrinsic motivation is the outcome. The incentive theory suggests that people are motivated to do things because of external rewards. Performing a certain action, for example, attending class to get a good grade, going to work to get paid, etc., has a purpose. Applying this concept, we developed a construct, *perceived value*, to capture the essence of extrinsic motivation in this study. We argue that, if students perceive value in the SAP ERP Award (believe that it will increase their marketability), they are more likely to be motivated to earn the award. Therefore, we hypothesize a positive relationship between the *perceived value* and intention to pursue the award.

Hypothesis 1: Students' *perceived value* of the SAP ERP Award will positively relate to their intention to pursue the award.

Intrinsic Motivation

In contrast to *extrinsic motivation*, *intrinsic motivation* is based upon taking pleasure in the activity rather working toward an external reward. Researchers in IS/IT use "*Affect*" to represent *intrinsic motivation*. For example, Compeau and her colleagues defined "*Affect*" as an individual's liking of the behavior (Compeau & Higgins, 1995; Compeau, Higgins & Huff, 1999). Venkatesh et al. (2003) formed "Attitude" in UTAUT combining both positive and negative sides of *Affect*, *intrinsic motivation* from motivational theory, and "attitude toward behavior," from the theory of reasoned action, theory of planned behavior, and the Technology Acceptance Model (Chau & Tam, 1997). Venkatesh et al. (2003) presented contradictory findings about the relationship between attitude and intention and proposed an indirect influence of attitude on intention. To answer our research question, we posit that it is imperative to examine whether *intrinsic motivation* has a direct positive effect on students' intention. Therefore, we adopted the validated *Attitude* construct from UTAUT but use only the positive evaluation items (such as "fun", "interesting" and "like") as the *intrinsic motivation* construct in our research model. Focus on the positive motivation internally, we propose that:

Hypothesis 2: Higher *intrinsic motivation* will positively relate to higher intention to pursue the SAP ERP Award.

Effort Expectancy

Effort expectancy is rooted as a part of *Perceived Behavioral Control* (PBC) which refers to the perceived degree of ease or difficulty of performing a particular behavior. Ajzen (1991) theorized that perceived behavioral control contributes to one's intention and behavior. It is assumed that perceived behavioral control is determined by the total set of accessible control beliefs. It reflects individual's confidence that they are capable of performing the behavior by assessing self-efficacy and controllability of behavior. By nature, the easier a task is perceived the higher the willingness to perform it.

Hypothesis 3: Higher levels of *effort expectancy* will correlate to lower levels of the SAP ERP Award pursuit intention.

Social Influence
Venkatesh and his colleagues (2003) merged three similar constructs: subjective norm, social factors, and image from other models and formed a construct which they named *social influence*. They defined *social influence* as "the degree to which an individual perceives that important others believe he/she should use the new systems" (Venkatesh et al., 2003, p.451). They also proposed that people's behaviors are influenced by the way in which they believe others will view them as a result of using an object; in this case, pursuing and then earning the SAP ERP Award.

Social influence has been treated as a direct determinant of behavior intention in many models and validated in many studies. It is reasonable to hypothesize that the higher degree to which students believe that their important others, such as parents or hiring managers, value the SAP ERP Award, the higher probability that they will pursue the award.

Hypothesis 4: Students' *social influence* will positively relate to their intentions to pursue the SAP ERP Award.

Facilitating Conditions
According to the theory of planned behavior, perceived behavioral control is determined by the total set of accessible control beliefs. Control beliefs is defined as an individual's beliefs about the presence of factors that may facilitate or impede performance of the behavior (Ajzen, 2001). Venkatesh and his colleagues (2003) captured three different constructs, namely perceived behavioral control, *facilitating conditions*, and compatibility from other models and built a construct called "facilitating conditions". *Facilitating conditions* are defined as "the degree to which an individual believes that an organizational and technical infrastructure exists to support use of the technology" (Venkatesh et al., 2003, p.453). In this study, *facilitating conditions* refer to the SAP ERP environment, faculty, and technical support the university provides. Although UTAUT concluded that this variable was not significant as a determinant of intention, we want to re-investigate the relationship between *facilitating conditions* and intention in this study for two reasons. First, other researchers such as Taylor and Todd (1995) have found that *facilitating conditions* is a significant predictor of behavior. Secondly, we want to test this construct in different context. In an academic setting, the accessibility of faculty and facility (lab, ERP system) is important to students' learning, and the justification of providing such resources is important to university administration. We included it in our model to see whether the relationship between *facilitating conditions* and intention is different in an academic setting.

Hypothesis 5: *Facilitating conditions* will positively relate to students' intentions to pursue the SAP ERP Award.

4. RESEARCH METHOD

Data Collection Procedure
We surveyed students from ERP and non-ERP related IS courses about their perception toward Georgia Southern University's SAP ERP Award. Georgia Southern University is a public university in the southeastern United States and has been participating in the SAP university alliance program for more than ten years. Most students who participated in the survey were from the College of Business Administration. We collected data from 2010 to 2014. The non-ERP related courses use ERP simulation games to demonstrate business processes in those courses. Students are exposed to SAP ERP even in the non-ERP awarded courses. To study students' intentions to pursue the SAP ERP Award, we developed an online survey using Blackboard, an online course management system. The survey was available for ten days each time it was administrated. Students were told about an extra credit opportunity for participating. Students are required to report if they are taking more than one course concurrently to prevent multiple responses from the same individual. They were given different extra credit opportunities for any second course. After the survey was closed, Blackboard generated a report in a spreadsheet. Each student's account was flagged for dichotomous outcome, indicating whether the survey had been filled out or not. No personal information was captured. This flagged field was used solely for the purpose of assigning extra credits. Only aggregated data were kept and analyzed.

Measures
The survey contained 24 questions including three background questions (major, minor, and academic status: freshman/sophomore/junior/ senior/graduate), one dichotomous question

about current behavior (currently enroll in SAP ERP Award program or not), nineteen Likert scale items (see Appendix A), and an open-ended question (the reason(s) of the student's intention toward SAP ERP Award). All of the Likert scale items were on a 5-point scale with "1" being "strongly disagree" and "5" being "strongly agree." From our review of the literature we selected four control variables that might potentially affect a student's intention to pursue the SAP ERP Award: year (when the survey was conducted), major (IS vs non-IS), course level (graduate vs undergraduate), and whether the course is ERP related. We describe the main constructs and items in our model next.

Perceived value (PV) that we developed for this study to represent extrinsic motivation was operationalized by two items to measure students' perception of value of the ERP award related to their marketability. The two items were "If I have an Georgia Southern University SAP ERP Award, it will increase my chance of getting a better job" and "I would find Georgia Southern University SAP ERP Award useful in my job hunting."

Intrinsic motivation refers to performing an action or behavior for the sake of enjoyment without external incentive. We modified the attitude toward using technology construct from UTAUT for intrinsic motivation in this study. Three items measure intrinsic motivation: SAP ERP makes work more interesting; working with SAP ERP is fun; I like working with SAP ERP.

Other constructs were adopted from UTAUT and modified for this study. We used 4 items to measure *effort expectancy*, 4 items to measure *social influence*, 3 items for *facilitating condition*, and 3 items for *intention* (see Appendix A for more detail).

Data Analysis

We obtained 333 valid responses, of which 146 were from IS majors and 187 from non-IS majors; 94 were from graduate students and 239 were from undergraduate students. Before analyzing the data, we verified the assumptions of normality, homoscedasticity, linearity, and independence with IBM SPSS Version 21 (2012). We tested for normality with normal probability plots and the results showed all the variables did not depart from normality severely. Homoscedasticity was checked using plots of residuals versus predicted values. We used a plot of the observed versus predicted values to test for linearity. All correlations turned out to be significant, with $p < .001$ (see Table 1, Appendix B). No multicollinearity problem was found. No outliers among the cases were found.

Table 1 presents descriptive statistics and correlation coefficients for all model variables. The correlation coefficients between the scale variables are Pearson's product moment correlation coefficients. Cronbach's alpha values, measures of internal consistency reliability, are reported on the diagonal in Table 1. All Cronbach's alphas exceed the cutoff of .7, which indicate high internal consistency reliability. Finally, we conducted Harman's one factor test (Harman, 1960) to assess common method bias. The un-rotated factor solution indicated that no single factor accounts for a significant portion of the variance in our data, which suggests that common method bias is not a significant threat to the validity of this study's results.

5. RESULTS

Hierarchical Regression

We decided to use hierarchical regression analysis to test effects of independent variables in different stages. We entered all control variables (Year, IS Major/Not, Graduate/Under, and SAP ERP Related Course/Not) into the analysis in the first step, then added the two motivation variables (PV and Intrinsic Motivation) in the second step, and then included all the other predictors in the last step. Table 2 (see Appendix C) displays the results of the entire hierarchical regression analysis. Portion of the variance explained (R^2) increased in each step. All three models are significant with all $p < .001$. Our final model which includes all control variables, independent variables and moderators explained about 48% of the variance in the intention to pursue the SAP ERP Award.

In the first model with only control variables, the course level (Graduate/Under) and whether it is an SAP ERP related course are significant predictors. In model 2, these two factors remain significant. Another control variable, student's major, becomes significant in model 2. Worthy of noting is that both *intrinsic motivation* and *perceived value* are significant at the .001 level. In model 3 we added the three constructs that we adapted from UTAUT. Among them, only *social influence* is significant at the .01 level. The only other significant variable excluding the control variables in model 3 is *perceived value*.

Hypotheses Test Results

Hypothesis 1 stated that *perceived value* will positively relate to intention to pursue the SAP

ERP Award. In model 2 and model 3, *perceived value* is a significant predictor with t = 9.907, p < .001 and t = 7.404, p < .001 respectively. Therefore, Hypothesis 1 is supported. Hypothesis 2 predicted that intrinsic motivation will positively relate to intention to pursue the SAP ERP Award. Interestingly, in model 2, intrinsic motivation is significant at the .001 level (t = 5.065, p < .001); while in model 3, it becomes insignificant with other variables present (t = 1.537, p = .125). Therefore, Hypothesis 2 is partially supported. Hypothesis 3 predicted that *effort expectancy* will negatively relate to intention to pursue the SAP ERP Award and this statement was not supported (t = 1.867, p = .063) with a positive coefficient and non-significant p-value. Hypothesis 4 predicted that *social influence* will positively relate to intention to pursue the SAP ERP Award and the statement is supported (t = 3.338, p = .001). Hypothesis 5 predicted that *facilitating conditions* will positively relate to intention to pursue the SAP ERP Award. Hypothesis 5 is not supported (t = -.254, p = .8) with a negative coefficient of -.021.

6. DISCUSSION

The data analysis results showed that three of the five hypotheses are supported. *Perceived value*, intrinsic motivation (conditionally) and *social influence* are significant predictors that positively correlated to students' intentions to pursue the SAP ERP Award. Interesting notes are the different results between this and other prior UTAUT studies. *Effort expectancy* and *facilitating conditions* are not significant predictors from our data and the direction of *facilitating conditions*' negative effect is opposite from previous studies.

Perceived Value

We tested both motivation types with the control variables in model 2. The *perceived value*, that is, extrinsic motivation, turned out to be a significant predictor of intention at the .001 level. Both our qualitative and the quantitative data confirm that *perceived value* is very important in students' decision forming of whether to pursue the SAP ERP Award regardless of degree of ease or *facilitating conditions*.

If students think that earning the SAP ERP Award increases their job-obtaining possibilities, it is more likely that they will want to take and complete the awarded courses. Students most appreciate these courses are electives included in the curriculum with no additional tuition or fees associated with them as demonstrated by the following selected comments (see Appendix D).

Intrinsic Motivation

Intrinsic motivation is significant at the .001 level in model 2 when we only tested the direct effects of the two motivation types with the control variables. However, in model 3, after we added other independent variables, *intrinsic motivation* is not a significant predictor anymore. Students did not seem to take these courses for enjoyment. Venkatesh and Davis (2000) suggested that affective reactions (e.g. *intrinsic motivation*) may operate through *effort expectancy*.

Regardless the degree of ease perceived, the program was evaluated as “worth” pursuing. Students pursue the SAP ERP Award because they recognize such award will increase their marketability (extrinsic) rather than because they would enjoy using ERP systems (intrinsic) no matter how much efforts would require. This phenomenon was confirmed that intrinsic enjoyment is not the best motivator while extrinsic motivation is. Students would take "not-so-easy" courses if they perceive high return of their efforts.

Social Influence

Social influence was formed by capturing essences of normative beliefs, subjective norms, social factors and image. From the *social influence* perspective, people who are important to “us” personally and professionally influence “our” behavior and choices. To measure this effect, we asked students to indicate whether any influences, either personal or professional, affected their intentions/decisions to pursue the ERP award. From both the quantitative and qualitative data, students value the opinions of people who influence their behavior such as their advisors and employers (see Appendix D).

Social influence is significant at the .001 level in model 3 and positively correlates to intention. Our data and results confirmed findings from UTAUT and other studies in this perspective. Although only a few comments were specifically about *social influence*, some of the students did mention that they felt the award was worth pursuing simply because they "have heard a lot about it", “I heard only good things about the program and I believe it will give me higher chance to get a better job”, and that their friends thought the SAP ERP skills are important when they look for jobs. One non-IS major stated that "I heard it boosts your base salary by having any SAP certificate".

Business schools should play up this factor as one student suggested “I think SAP America and Georgia Southern University need to do a better job of displaying Georgia Southern University's SAP assets to companies and corporations that use SAP ERP." By doing so universities can achieve many-fold benefits for all parties: students, schools and industry. Students gain marketable skills and knowledge, schools gain reputations, and industry gain competent workers.

Facilitating Conditions
Venkatesh et al. (2003) hypothesized a non-significant correlation between *facilitating conditions* to intention to use technology. We, however, proposed the direct positive relationship because of the education context. We believed that the accessibility and availability of facilitating resource are crucial in learning any IT skill; however, such a belief was not supported in this study. Contradictory to prior quantitative studies, *facilitating conditions* was not a statistically significant factor to predict students' intentions to pursue the SAP ERP Award while its slope is negative. From the qualitative data, however, students did articulate the need for better *facilitating conditions* in the open-ended question comment (see Appendix D).

This contradiction signals a need for further investigation into the relationship between *facilitating conditions* and intention. It would be interesting to study whether the academic setting has any effect on the relationship.

Other Findings
We found from our qualitative data that other than those afore-discussed factors, Georgia Southern University students value their ERP award program not only for the job potential but also for its unique design and requirements. Students most appreciate the fact that it is embedded in courses and unlike any other certification, they do not have to pay extra to pursue the award (see Appendix D).

On the other hand, students who decided not to pursue this ERP award most expressed the "lack of time" factor, that is, they did not have enough time to take three additional courses before scheduled graduation. Students reasoned that they were not informed until it was "too late" to pursue it (see Appendix D).

Contributions to Theory
This study contributes to incorporate motivation theory and UTAUT's theoretical validity to the management of information technology-based initiatives in education. Venkatesh and his colleagues (2003, p.470) directed "…future work should attempt to identify and test additional boundary conditions of the model in an attempt to provide an even richer understanding of technology adoption and usage behavior". Unlike numerous previous studies that tested and validated UTAUT by using a specific technology as an artifact, this study tested the boundary and applicability of UTAUT in a new but technology-related area - IT certification.

Most prior IT certification value research focused on the perspectives of employers, managers or professionals who already had certifications (Venkatesh, Thong, & Xu, 2012). A few studies have examined students' perspectives. This current study fills this gap by studying students' perspectives, intentions, and their motivating factors. Higher education institutions can design better programs if we understand what motivates students. Attention should be paid to determine students' underlying beliefs. Students in this study understood the benefits of receiving the award; one benefit is that it demonstrates skills and knowledge, which in turn increases their marketability in the competitive business environment. Another important benefit is the differentiation effect. Certification separates the owners from other job seekers, and the award program also brings distinction to the university.

Practical Implications
Scholars and practitioners have advocated closing the gap between business school curriculum and business community needs for practically trained personnel (David et al., 2011). Gartner (2010) conducted its IT Market Compensation Survey sampling 358 U.S.-based IT organizations from March 2009 to February 2010. At Time 1 (2010) of data collection of this study, the Gartner survey (2010) exposed a slow IT job market overall, but certain IT jobs/skills, for instance, system architecture design, database administration, ERP and networking management, are still high in-demand. Those desired skills could be demonstrated by awards and certifications.

Most current Computerworld's 2016 IT Salary Survey conducted in the fall of 2015 shows people who work in enterprise resource planning reported a bigger year-over-year compensation gain — 5% —than survey respondents in any other area of IT. The same survey also shows a solid interest in certifications among the 3,301 respondents: More than half (54%) said they have IT-related certifications, and 44% said they plan to pursue an IT certification within the next 24 months. Scholars have been pointing out that if universities integrate industry certification with an academic degree, it would create a win-win situation for both the industry and academia, and better help students get jobs (David et al., 2011; Hitchcock, 2007; Simmonds, 2002). From our qualitative data, this study also helps business schools understand what students consider most – cost and time.

7. CONCLUSIONS

We observed one interesting fact that non-business majors have begun to enroll in the award program since 2011. Although typically students majoring in IS partake and receive the SAP ERP Award, the award can also be earned by students who minor in IS. The IS minor and the SAP ERP Award enhance career options for students who are interested in working for businesses that use SAP ERP. Students majoring in accounting, finance, human resource management, operations management, or logistics are especially likely to benefit from an IS minor and SAP ERP Award (MacKinnon et al., 2006). A follow-up study could be done on the effect of marketing of the award program in order to increase student enrollment.

We plan to further develop and validate the new construct established for this study: *perceived value*. We plan to increase the reliability and validity by adding more items and then testing them via expert evaluation and pilot-test with students. Another construct, *facilitating conditions*, should be investigated further because of mixed results from this and other studies. From the qualitative data, students did mention that they are "grateful" to have the opportunity to pursue the award by taking courses. It "saves money" because they "pay tuition instead of thousands of dollars" for an industry certificate. It would be beneficial for future studies to evaluate the financial impact of pursuing an industry certification. Other schools can adapt this built-in-curriculum model to create a win-win-win situation for students-schools-employers.

8. REFERENCES

Ajzen, I. (1991). The Theory of Planned Behavior. *Organizational Behavior and Human Decision Processes*, 50(2), 179-211.

Cantor, J. (2002). Skills certifications and workforce development: Partnering with industry and ourselves [Electronic version]. *Leadership Abstracts,* 15(1), Retrieved March 22, 2013, from http://www.league.org/publication/abstracts/leadership/labs0102.html.

Chau, P., & Tam, K. (1997). Factors affecting the adoption of open systems: an exploratory study. *MIS Quarterly*, 21(1), 1-24.

Chen, K., Razi, M., & Rienzo, T. (2011). Intrinsic factors for continued ERP learning: a precursor to interdisciplinary ERP curriculum design. *Decision Sciences Journal of Innovative Education*, 9 (2), 149-176.

Compeau, D. R., & Higgins, C. A. (1995). Computer self-efficacy: development of a measure and initial test. *MIS Quarterly*, 19(2), 189-211.

Compeau, D., Higgins, C. A., & Huff, S. (1999). Social cognitive theory and individual reactions to computing technology: A longitudinal study. *MIS Quarterly*, 23(2), 145-158.

Computer Science and Telecommunications Board (2001). Report of a Workshop on Science, Technology, Engineering, and Mathematics (STEM) Workforce Needs for the U.S. Department of Defense and the U.S. Defense Industrial Base (NAE) Released 2011-12-20

Covington, M., & Müeller, K. (2001). Intrinsic versus extrinsic motivation: An approach/avoidance reformulation. *Educational Psychology Review*, 13(2), 157-176.

David, F., David, M, & David, F. R. (2011). What are business schools doing for business today? *Business Horizons,* 54(1), 51-62.

Davis, F., Bagozzi, R., & Warshaw, P. (1992). Extrinsic and intrinsic motivation to use computers in the workplace1. *Journal of Applied Social Psychology*, 22(14), 1111-1132.

Davis, S, Siau, K, & Dhenuvakonda, K. (2003). A fit-gap analysis of e-business curricula vs. industry needs. *Communication of the ACM*, 46(12), 167-177.

Eom, S., Wen, J., & Ashill, N. (2006). The determinants of students' perceived learning outcomes and satisfaction in university online education: an empirical investigation. *Journal of Innovative Education*, 4(2), 215-235.

Harman, H. (1960). Modern Factor Analysis. Chicago: University of Chicago Press.

Hitchcock, L. (2007). Industry certification and academic degrees: Complementary, or poles apart? Paper presented at the *Proceedings of the ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce*, St. Louis, MS.

IBM Corp. Released 2012. IBM SPSS Statistics for Windows, Version 21.0. Armonk, NY: IBM Corp

MacKinnon, R., Rogers, C., Kung, H.-J., Gardiner, A., Whitworth, J. E., & Williams, S. (2006). Creating an ERP emphasis in the IS curriculum. *Issues in Information Systems*, 7(1), 284-288.

Mohamed, S, & McLaren, T. (2009). Probing the gaps between ERP education and ERP implementation success factors. *AIS Transactions on Enterprise Systems*, 1(1), 8-14.

Pardee, R. (1990). Motivation theories of Maslow, Herzberg, McGregor & McClelland. A literature review of selected theories dealing with job satisfaction and motivation.

Pratt, J., Hauser, K., & Ross, S. (2010). IS staffing during a recession: comparing student and IS recruiter perceptions. *Journal of Information Systems Education*, 21(1), 69-84.

Randall, M., & Zirkle, C. (2005). Information technology student-based certification in formal education settings: who benefits and what is needed. *Journal of Information Technology Education,* 4, 287-305.

Root, G. N. III. Examples of intrinsic workplace motivation, http://smallbusiness.chron.com/examples-intrinsic-workplace-motivation-11382.html, Retrieved 30 August 2015.

Ryan, R., & Deci, E. (2000a). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55 (1), 68–78.

Ryan, R., & Deci, E. (2000b). Intrinsic and extrinsic motivations: classic definitions and new directions. *Contemporary Educational Psychology*, 25(1), 54-67.

Simmering, M., Posey, C., & Piccoli, G. (2009). Computer self-efficacy and motivation to learn in a self-directed online course. *Journal of Innovative Education*, 7(1), 99-121.

Simmonds, A. (2002). Learning experience with an industry certification course at university. Paper presented at the Proceedings of Australasian Computing Education Conference in Research and Practice in Information Technology, Adelaide, Australia.

Syler, R., Cegielski, C., Oswald, S., & Rainer, K. (2006). Examining drivers of course performance: an exploratory examination of an introductory CIS applications course. *Journal of Innovative Education*, 4(1), 51-65.

Taylor, S., & Todd, P. (1995). Understanding information technology usage: a test of competing models. *Information Systems Research*, 6(2), 144-176.

Teo, H., Wei, K., & Benbasat, I. (2003). Predicting intention to adopt interorganizational linkages: an institutional perspective. *MIS Quarterly*, 27(1), 19-49.

Thompson, R., Higgins, C., & Howell, J. (1991). Personal computing: toward a conceptual model of utilization, *MIS Quarterly*, 15(1), 124-143.

Thong, J. (1999). An integrated model of information systems adoption in small businesses. *Journal of Management Information Systems*, 15(4), 187-214.

van der Heijden, H. (2004). User acceptance of hedonic information systems. *MIS Quarterly*, 28(4), 695-704.

Venkatesh, V., & Davis, F. (2000). A theoretical extension of the technology acceptance model: four longitudinal field studies. *Management Science*, 46(2), 186-204.

Venkatesh, V., Morris, M., Davis, G., & Davis, F. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly*, 27(3), 425-478.

Venkatesh, V., Thong, J., & Xu, X. (2012). Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. *MIS Quarterly*, 36(1), 157-178.

Wigfield, A., Guthrie, J., Tonks, S., & Perencevich, K. (2004). Children's motivation for reading: domain specificity and instructional influences. *Journal of Educational Research*, 97, 299–309.

Appendix A

Survey Items

Construct	Source	# of	Survey Items
Perceived Value	New, developed for this study Based on motivation theroy	2	• If I have Georgia Southern University SAP ERP Award, it will increase my chance of getting a better job • I would find Georgia Southern University SAP ERP Award useful in my job hunting
Intrinsic Motivation	Based on motivation theroy Modified UTAUT construct: Attitude toward using technology	3	• SAP ERP makes work more interesting • Working with SAP ERP is fun • I like working with SAP ERP
Effort Expectancy	UTAUT	4	• My interaction with SAP ERP would be clear and understandable • It would be easy for me to become skillful at using SAP ERP • I would find SAP ERP easy to use • Learning to use SAP ERP is easy for me
Social Influence	UTAUT	4	• People who influence my behavior think that I should use SAP ERP • People who are important to me think that I should use SAP ERP • The senior management of this business has been helpful in the use of SAP ERP • In general, the organization has supported the use of SAP ERP
Faciliataing Condition	UTAUT	3	• I have the resources necessary to use the SAP ERP • I have the knowledge necessary to use SAP ERP • A specific person (or group) is available for assistance with SAP ERP difficulties
Intention	UTAUT	3	• I intend to pursue Georgia Southern University SAP ERP Award • I predict I would be in Georgia Southern University SAP ERP Award program • I plan to pursue Georgia Southern University SAP ERP Award in the future

Appendix B

Table 1: Descriptive Statistics and Correlations

	Mean	SD	PV	IM	EE	SI	FC	INT
PV	4.134	0.766	**0.770**	.384**	.430**	.524**	.464**	.586**
IM	3.57	0.892		**0.894**	.684**	.538**	.484**	.401**
EE	3.739	0.785			**0.863**	.572**	.606**	.458**
SI	3.609	0.701				**0.786**	.598**	.527**
FC	3.907	0.690					**0.730**	.425**
INT	3.943	0.990						**0.921**

**p < .001

PV: Perceived Value
IM: Intrinsic Motivation
EE: Effort Expectancy
SI: Social Influence
FC: Faciliataing Condition
INT: Intention

Appendix C

Table 2: Hierarchical Regression Results

	Step 1			Step 2			Step 3	
	b	β		b	β		b	β
Constant	3.563			0.451			-0.009	
Year 2	0.150			0.039	0.017		0.039	0.017
Year 3	0.071			-0.055	-0.021		-0.011	-0.004
Year 4	-0.260			-0.208	-0.085		-0.142	-0.058
Year 5	-0.262			-0.350*	-0.142		-0.285*	-0.115
Graduate level	-0.248*			-0.264**	-0.121		-0.25**	-0.114
SAP related	0.534**			0.319**	0.132		0.324**	0.134
IS Major	0.187			0.187*	0.094		0.172*	0.087
Perceived Value				0.586***	0.455		0.474***	0.368
Intrinsic Motivation				0.258***	0.234		0.1	0.09
Effort Expectance							0.148	0.117
Social Influence							0.273**	0.194
Facilitating Conditions							-0.021	-0.015
R^2	0.116			0.446			0.479	
R^2 Change	0.116			0.330			0.033	
F change	6.095***			96.327***			6.71***	
*** $p < .05$; ** $p < .01$; ***$p <.001$**								

Appendix D

Table 3: Student Responses to the Open-ended Question

Factors	**Student Comments**
Perceived Value	• "With several companies beginning to use SAP, it is obvious that it is a worthwhile goal to pursue." • "I have obtained an internship allowing me to enhance my knowledge of SAP. Without pursuing the SAP ERP Award, I probably would not have received the internship." • "With the SAP ERP Award, I definitely increase my chances of landing a good job position, and I also differentiate myself from other people trying to get that same position." • the award "…will help [me] to get further in job hunting." • "It's helping me find a job right now. Got a few really good leads."
Social Influence	• "it is recommended by my advisor. Yes, it's highly recommended in the IS department as something that will help me more marketable in my search to pursue jobs. " • "Current employer is implementing SAP" • "I work for GulfStream and I use SAP every day, and it will make me look more marketable to my employer."
Facilitating Conditions	• "Much easier to attain when teachers are available to help." • "The classes taught need a T.A." • "Some students need SAP tutor for help, but great program."
Others	• "I am very happy that Georgia Southern University provides SAP ERP Award classes that I can take on my time." • "I think it is great that Georgia Southern University allows you to obtain this award through the course work." • "I think that for us as student it is a great opportunity for us to have at a great price. Considering how expensive the SAP classes are."
Not Pursuing the award	• "I would go for it but I simply don't have the time" • "No. Would postpone my graduating time." • "No, not able to take all the necessary classes."

Developing an Approach to Harvesting, Cleaning, and Analyzing Data from Twitter Using R

Stephen Hill
hills@uncw.edu
Information Systems and Operations Management
University of North Carolina Wilmington
Wilmington, NC 28403 USA

Rebecca Scott
rebecca.a.scott@ttu.edu
Marketing and Supply Chain
Texas Tech University
Lubbock, TX 79409

Abstract

Using data from social media can be of great value to businesses and other interested parties. However, harvesting data from social media networks such as Twitter, cleaning the data, and analyzing the data can be difficult. In this article, a step-by-step approach to obtaining data via the Twitter application program interface (API) is described. Cleaning of the data and basic sentiment analysis are also described.

Keywords: Analytics, Social Media, Text Mining, Data Cleaning, Classification

1. INTRODUCTION

Social media has become nearly ubiquitous in modern society with users interacting with friends and celebrities, posting photos of their children, and sharing their opinions on a variety of topics. The ubiquity of social media has driven the rapid growth of social media networks. For example, two of the predominant social media networks in the United States, Facebook and Twitter, have grown steadily since their founding. Facebook reported over one billion daily active users in early 2016 (Company Info, n.d.) and Twitter reported over 300 million monthly active users (Company, n.d.). The sheer size of these networks and the information that their users are willing to share present rich opportunities for businesses to market to existing and potential customers and to accrue valuable customer information. In this article, the authors focus on harvesting tweets and related data from the Twitter social media network. The process for cleaning and preparing this data and approaches to basic analysis of the data are described.

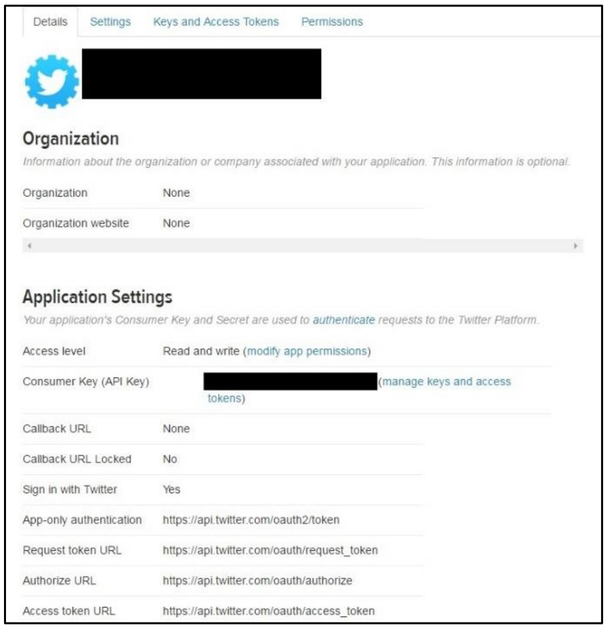
Twitter provides developers access to their network via its application program interface (API). Accessing, harvesting, and analyzing social media data via Twitter's API is not a new phenomenon. Doing these tasks in R, a popular open-source statistical programming software package, is also not new. However, providing a single reference via which an interested data analyst can be guided, in a step-by-step manner, through the process of Twitter data harvesting, cleaning, and analysis is valuable. This article attempts to present such a reference. The closest such reference is from Dannemann and Heimann (2014), but this reference has proven to be outdated, particularly in its description of access to the Twitter API. Other relevant examples include Breen (2011), Bryl (2014), and de Vries (2016). However, these examples either rely on out-of-date approaches to access the Twitter API or are incomplete in that they provide very limited examples of analysis.

2. COLLECTING TWITTER DATA

To retrieve data from Twitter a user must first gain access to the Twitter API. Access to the API will allow the user to collect archived tweets (limited to approximately seven to nine days of historical tweets, subject to API rate limits) or real-time tweets (subject to API rate limits). Twitter may change the process by which users are granted access to the API. The authors have verified that the process described in this article is valid as of late May 2016.

Accessing the Twitter API

In order to gain access to the Twitter API to collect data a user must first sign up for Twitter (Free, https://twitter.com/signup). A Twitter user can then register as a Twitter developer (Free, https://dev.twitter.com/). Twitter developer registration is a one-time task and does not need to be repeated. Once registered as a developer, the user should then create a Twitter application at https://apps.twitter.com/. Select "Create New App". The user will then be prompted to "Create an application".



[redacted]

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	[redacted] (manage keys and access tokens)
Callback URL	None
Callback URL Locked	No
Sign in with Twitter	Yes
App-only authentication	https://api.twitter.com/oauth2/token
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token

Figure 1: Twitter Application Information Screenshot

Enter the requested information. Note that it is acceptable to use a placeholder URL for the requested Website. Agree to the Developer Agreement, and a Twitter application has been developed. The user can then access their application by clicking on the application name. Figure 1 shows the application information for a sample application developed by this article's authors. Personally identifying and other confident information has been redacted from Figure 1.

To use the newly created application to access the Twitter API and collect tweets data, the user should now open R. For this article, the authors used R version 3.3 (R Development Core Team, 2016) and RStudio version 0.99.902 (RStudio Team, 2016). If the user does not have R and/or RStudio, they may be obtained from https://www.r-project.org/ and https://www.rstudio.com/products/rstudio/download/, respectively. Note that both software packages are available at no cost and in versions for Windows, Mac OS X, and Linux. When installing these software packages, be sure to install R first before installing RStudio. RStudio is optional but provides a useful development environment for R.

Several R packages are needed for this project. See Table 1 below for a listing of the required packages. Each package should be installed via the R install.packages command if not done previously. The packages should then be loaded via the R library command.

Package	
caret	RJSONIO
caTools	ROAuth
dplyr	rpart
e1071	rpart.plot
ggplot2	SnowballC
httr	streamR
plyr	stringr
qdap	tm
rattle	twitteR
RColorBrewer	wordcloud
RCurl	

Table 1: Necessary R Packages

Appendix 1 contains the R script that is used to access the Twitter API. Each line in the script is numbered to enable easier referencing in the article. Line numbers should be removed when the script is used in R. Also, R's commenting sign, "#", is used to denote comments in the script.

To connect to the API, the user must first obtain the appropriate certificates for interaction with Twitter's servers. This is accomplished by line (1). Lines (2) to (4) are then used to prepare for Twitter API access authorization.

Next, the user should assign their Consumer Key and Consumer Secret to appropriately named R objects (Lines (5) and (6)). The user can obtain their Key and Secret from the "manage keys and access tokens" link on the Twitter Application information screen (see Figure 1). Note that the article authors' Key and Secret have been replaced by X's in the sample lines of code. The user should replace the X's with their own Key and Secret. Lines (7) and (8) are then used to establish the connection to the Twitter API.

The Twitter application access token and secret are then assigned to appropriately named R objects. As before, the author's token and secret have been replaced by X's. The user should replace the X's with their own token and secret values.

The Twitter API access authorization can be saved to a .Rdata file by line (11). The user can then, in the future, load the authorization via line (12) and gain reauthorization to access the API via line (13). If this is done, there is no need to run lines (1) to (11) for future instances of data harvesting. At this point, the user has gained access to the Twitter API and is ready to harvest data.

Collecting Archived Twitter Data
The searchTwitter function as shown in Line (14) was run on May 26th, 2016 to collect up to 25,000 tweets featuring the hashtag "#Warriors" and no older than May 19th, 2016. This code returned 25,000 tweets with the oldest tweet from May 23rd, 2016. The searchTwitter function can be used to gather tweets for a set of search terms (with terms separated by "+") and/or by geocoded location. A geocoded location is provided as a point specified by latitude and longitude and a radius (in miles or kilometers) from that point. For example, line (15) shows the same search as in line (14), but modified to only collect tweets generated within 50 miles of San Francisco, California.

Collecting Real-Time Twitter Data
The filterStream command that is part of the streamR package is used to collect real-time Twitter data (Barbera, 2016). The authors used the filterStream command to collect 30 minutes of tweets generated during the March 10th, 2016 Republican Presidential debate. A total of 79,456 tweets were collected. Line (16) shows the R code used to collect the tweets. The "timeout = 1800" portion of the code specifies the duration (in seconds) during which tweets should be collected.

3. PREPARING TWITTER DATA FOR ANALYSIS

Once tweet data has been harvested from Twitter, it must then be prepared for analysis. Before preparing the data it is prudent to save an archive of the raw data from Twitter. This is done via lines (17) and (18). Then, to enable easier re-use of the preparation and analysis R code, the tweets data is given a generic name of "some_tweets" in line (19). The tweet text is then isolated from the remainder of the tweet data and stored as "some_txt" via line (20).

Lines (21) through (27) prepare the text data by removing any "RT" and "@" portions for retweeted text, the "@" for tweets directed at a user , punctuation marks, numbers, web links, extra spaces, and non-graphical characters. These changes are done via R's gsub function and regular expressions (regex). Lines (28) through (30) convert all text to lower case. A function in line (29) from Sanchez (2012) is used to prevent R's tolower() function from producing errors that prevent the upper to lower case conversion from taking place. Lines (31) and (32) then remove the originally searched for term "gopdebate" from the text and also removes a related Twitter hashtag. Other frequently appearing, but uninformative text can be removed in a similar manner if desired. The table in Appendix 3 shows how a particular tweet is modified by the procedures in lines (21) to (32).

The text is then converted into a corpus (body of text) by line (33). The tm package is then used in line (34) to remove common words, known as stop words, which often have little analytical value. Examples of English stop words include "a", "is", and "the". With text data that is now cleaned, the user can proceed to analysis of the data.

4. BASIC ANALYSIS OF TWITTER DATA

There are a variety of possible avenues for the analysis of Twitter data. In this section, a few basic approaches to analysis are presented. The first approach is a simple word cloud via the wordcloud R package. Line (36) shows the R code used to develop the word cloud shown in Figure 2. Note that the wordcloud function requires that a corpus be passed to it. Word clouds are often used to provide an appealing visualization of the frequency and importance of words that appear in a body of text (Heimerl, Lohmann, Lange, & Ertl, 2014). Lines (37) then converts the corpus to a data frame object. Line (38) assigns the cleaned (but not yet stemmed) text to a variable in the some_tweets data frame. This is done to facilitate analysis.

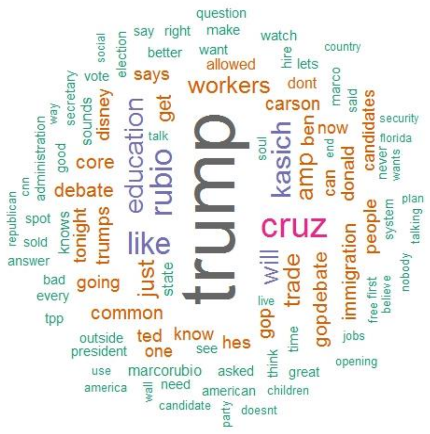


Figure 2: Republican Presidential Debate Twitter Wordcloud

A common analysis technique for text data is sentiment analysis. A significant portion of sentiment analysis work in the academic focuses on capturing the polarity (e.g., positive, neutral, or negative) of a text author's feelings, emotions, or thoughts (Pang and Lee, 2008). A simple way to conduct sentiment analysis is to consider each text document (each tweet in the case of this article) as a "bag of words" (Salton and McGill, 1986). Each word in the document is then matched against a dictionary of words that have been classified as either positive or negative. Positive words are assigned a sentiment of +1, negative words are assigned a sentiment of -1, and unmatched words are assigned a sentiment of 0. The sentiment of the document is then the sum of the sentiment of each word in the document. In this article, dictionaries of positive and negative sentiment words from Hu and Liu (2004) are used. Lines (39) and (40) read in these dictionaries.

A custom function from Bryl (2014) is used to score the sentiment of each tweet. Before using this function, lines (41) and (42) should be run to appropriately format the data for use in the function. The custom function is shown in line (43). The user may wish to save this function as a separate .R file and then source this file as needed (see line (44)). The sentiment scores are generated by line (45) and are then copied to the some_tweets data frame by line (46). The data frame is saved (as a CSV file) for backup purposes by line (47). The user can then analyze the sentiment scores. For example, line (48) shows the R code used to develop the histogram of tweet sentiment scores shown in Figure 3. Sentiment scores can be categorized as Negative, Positive, or Neutral by lines (49) to (51).

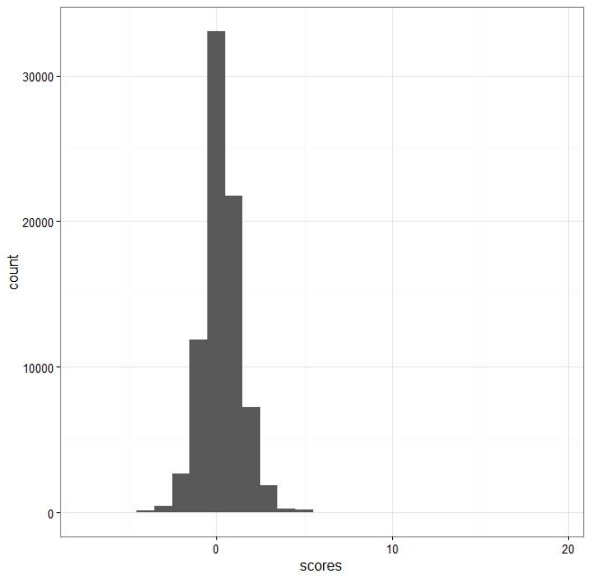


Figure 3: Republican Presidential Debate Twitter Sentiment Histogram

The final data manipulation that is described in this article is stemming. Lines (52) and (53) are used to stem the tweets. Stemming is used to reduce a word down to its root. Doing so reduces the number of unique terms (i.e., words) that are considered during analysis (Meyer, Hornik, & Feinerer, 2008). For example, the word "applied", when stemmed, becomes "appli". After stemming the tweet that was shown in the table in Appendix 3, becomes "anoth tonight".

Once stemming is complete, the frequencies of terms (words) in the text corpus is examined by line (54). If the user wishes to see terms that appear frequently, line (55) can be used. In this example, terms that appear 20 or more times across all of the tweets in the corpus are shown. Line (56) is used to eliminate terms from the corpus that appear infrequently. Setting the numerical value in this line to a number closer to 1 will result in more terms being retained. Smaller values will result in fewer terms being retained. Line (57) will display the number of terms remaining. In this example, 296 unique terms were retained. Lines (58) and (59) convert the remaining terms into a data frame and give each column in the data frame a name that corresponds to the terms. Line (60) transfers the sentiment score categories to the data frame created in lines (58) and (59).

The user can now consider a variety of analytics techniques with an objective to determine what terms that appear in a tweet are most predictive of whether or not the tweet is classified as "Negative" or "Not Negative". In this article, a classification tree is used. Classification trees rely on recursive portioning to predict the classification of entities in a dataset (Breiman, Friedman, Stone, & Olshen, 1984). In this article, a classification tree will be constructed that predicts whether a tweet is "Negative" or "Not Negative".

Before constructing the tree, the dataset is split. Lines (61) through (63) are used to split the data into a training set comprised of 70% of the data and a testing set with the remaining 30% of the data. The training set is then used for predictive model building while the testing set is set aside for evaluation of model quality. The splitting is performed via the sample.split function from the caret package.

The rpart package is then used to develop the classification tree. Line (64) generates the tree and line (65) displays the tree. The resulting classification tree is shown in Appendix 5. The tree classifies a tweet as either "Negative" (total sentiment less than zero) or "Not Negative" (total sentiment of zero or greater). To classify a tweet as negative or not negative, the user of the tree examines the tweet and begins at the top of the tree with the statement "disast >= 0.5". This statement is True (follow the Yes branch of the tree) if the tweet contains at least one word with the root "disast". For example, if the tweet contains the word "disaster", the user would follow the Yes branch of the tree and classify the tweet as "Negative". If the tweet does not contain a word with the root "disast" then the No branch is followed. This process is repeated until the tweet is classified.

Line (66) uses the classification tree to predict the classification of tweets in the testing dataset. Of the 23,836 tweets in the testing portion of the dataset, 20,311 were correctly classified by the tree and 3,525 were incorrectly classified. This gives the tree an out-of-sample accuracy of 85.2%. This accuracy compares favorably to a naïve (no information) accuracy of 81.0%. The accuracy calculations are performed in line (67) by the confusionMatrix function from the caret package.

5. CONCLUSIONS AND OPPORTUNITIES FOR FUTURE WORK

This article provides a convenient and easy to follow step-by-step approach to harvesting, cleaning, and analyzing data from Twitter. The approach accesses the Twitter API via the R statistical programming software. Archived tweets or streaming, real-time tweets are then collected. The tweets data is then cleaned and prepared for analysis. The article closes with a brief description of basic sentiment analysis of the data.

The step-by-step approach provided in this article is valuable to businesses and other interested parties. Analyzing social media, such as Twitter, allows businesses to evaluate customer impressions of their goods and services. This in turn can allow organizations to use social media as an effective customer service tool.

There are a variety of options available for this work to be expanded. For example, tweets could be collected that relate to a particular event (e.g., the Super Bowl, elections, etc.) and sentiment regarding these events could then be analyzed. Any number of text mining techniques and other analytical techniques for classification could also be applied.

6. REFERENCES

Barbera, P. (2016). *Introducing the streamR package*. *Pablobarbera.com*. Retrieved 30 May 2016, from http://pablobarbera.com/blog/archives/1.html.

Breen, J. (2011). *Mining Twitter for Airline Consumer Sentiment*. *Inside-r.org*. Retrieved 30 May 2016, from http://www.inside-r.org/howto/mining-twitter-airline-consumer-sentiment.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC Press.

Bryl, S. (2014). *Twitter sentiment analysis with R*. *AnalyzeCore.com*. Retrieved 30 May 2016, from http://analyzecore.com/2014/04/28/twitter-sentiment-analysis/.

Company Info. (n.d.). Retrieved May 31, 2016, from http://newsroom.fb.com/company-info/.

Company. (n.d.). Retrieved May 31, 2016, from https://about.twitter.com/company.

Danneman, N., & Heimann, R. (2014). Social media mining with R. Packt Publishing Ltd.

de Vries, A. (2016). *Text Analysis 101: Sentiment Analysis in Tableau & R*. *The Information Lab*. Retrieved 30 May 2016, from http://www.theinformationlab.co.uk/2016/03/02/text-analysis-101-sentiment-analysis-in-tableau-r/.

Heimerl, F., Lohmann, S., Lange, S., & Ertl, T. (2014). Word cloud explorer: Text analytics based on word clouds. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 1833-1842). IEEE.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 168-177). ACM.

Meyer, D., Hornik, K., & Feinerer, I. (2008). Text mining infrastructure in R. *Journal of statistical software*, 25(5), 1-54.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2), 1-135.

R Development Core Team. (2014). R: A Language and Environment for Statistical Computing (Version 3.3) [Software]. Vienna, Austria: R Foundation for Statistical Computing. Available from http://www.R-project.org.

RStudio Team (2016). RStudio: Integrated Development for R (Version 0.99.902) [Software]. Boston: RStudio Inc.. Available from http://www.rstudio.com.

Salton, G. & McGill, M., editors (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Sanchez, G. (2012). *Catching errors when using tolower*. *Gastonsanchez.com*. Retrieved 30 May 2016, from http://gastonsanchez.com/how-to/2012/05/29/Catching-errors-when-using-tolower/.

Editor's Note:

This paper was selected for inclusion in the journal as an EDSIGCon 2016 Meritorious Paper. The acceptance rate is typically 15% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a paper in 2016.

Appendix 1 (Twitter API Access R Script)

The R script below is used to gain access to the Twitter API. The user must register as a Twitter develop and create a Twitter "application" before executing this code. Comments are embedded in the script and are indicated by R's commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(1)     download.file(url='http://curl.haxx.se/ca/cacert.pem', destfile='cacert.pem')
(2)     reqURL <- 'https://api.twitter.com/oauth/request_token'
(3)     accessURL <- 'https://api.twitter.com/oauth/access_token'
(4)     authURL <- 'https://api.twitter.com/oauth/authorize'
(5)     consumerKey <- ' XXXX ' #Replace X's with your Consumer Key
(6)     consumerSecret <- 'XXXX' #Replace X's with your Consumer Secret
(7)     Cred <- OAuthFactory$new(consumerKey=consumerKey,
                consumerSecret=consumerSecret,
                requestURL=reqURL,
                accessURL=accessURL,
                authURL=authURL)
(8)     Cred$handshake(cainfo = system.file('CurlSSL', 'cacert.pem', package = 'RCurl'))
(9)     access_token = 'XXXX' #Replace the X's with your Access Token
(10)    access_secret= 'XXXX' #Replace the X's with your Access Token
(11)    save(Cred, file='twitter authentication.Rdata')
(12)    load('twitter authentication.Rdata')
(13)    setup_twitter_oauth(consumerKey,consumerSecret,access_token,access_secret)
```

Appendix 2 (Twitter Archived and Streaming Tweets Collection Examples)

The R script below is used harvest tweets. Line (14) is used for archived tweets and line (15) is used for real-time, streaming tweets. As in the other appendices, comments are embedded in the script and are indicated by R's commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(14)    tweets = searchTwitter("#Warriors",n=25000, retryOnRateLimit=120, lang="en",
        since="2016-05-15", resultType="recent")
(15)    tweets = searchTwitter("#Warriors",n=25000, retryOnRateLimit=120, lang="en",
        geocode="37.7749,-122.4194,50 mi", since="2016-05-19", resultType="recent")
(16)    filterStream(file.name = "tweetsGOP.json",
        track = c("GOPDebate", "gopdebate", "GOPdebate"), language = "en",
        timeout = 1800, oauth = Cred)
```

Appendix 3 (Twitter Data Cleaning and Preparation for Analysis)

The R script below is used to perform cleaning of the tweets data. As in the other appendices, comments are embedded in the script and are indicated by R's commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(17)    tweet_archive = do.call("rbind", lapply(tweets, as.data.frame))
        #OR for streaming tweets
        tweets_archive <- parseTweets("tweetsGOP.json", simplify = FALSE)
(18)    write.csv(tweet_archive,file="tweets.csv")
(19)    some_tweets = tweets_archive
(20)    some_txt = sapply(some_tweets, function(x) x$getText())
(21)    some_txt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", some_txt)
(22)    some_txt = gsub("@\\w+", "", some_txt)
(23)    some_txt = gsub("[[:punct:]]", "", some_txt)
(24)    some_txt = gsub("[[:digit:]]", "", some_txt)
(25)    some_txt = gsub("http\\w+", "", some_txt)
(26)    some_txt = gsub("^\\s+|\\s+$", "", some_txt)
(27)    some_txt = gsub("[^[:graph:]]", " ", some_txt)
(28)    try.error = function(x)
(29)    {
                y = NA
                try_error = tryCatch(tolower(x), error=function(e) e)
                if (!inherits(try_error, "error"))
                y = tolower(x)
                return(y)
        }
(30)    some_txt = sapply(some_txt, try.error)
(31)    some_txt = gsub("gopdebate", "", some_txt)
(32)    some_txt = gsub("cnndebate", "", some_txt)
(33)    corpus = Corpus(VectorSource(some_txt))
(34)    corpus = tm_map(corpus, removeWords, stopwords("english"))
(35)    corpus = tm_map(corpus,PlainTextDocument)
```

Action	Tweet Text
Original Tweet	Another #GOPDebate tonight. https://t.co/TIEy5DwSzo
Punctuation Removed	Another GOPDebate tonight httpstcoTIEy5DwSzo
Numbers Removed	Another GOPDebate tonight httpstcoTIEyDwSzo"
Removed Web Links	Another GOPDebate tonight
Converted Text to Lower Case	another gopdebate tonight
Removed Specific Hashtags	another tonight

Appendix 4 (Basic Twitter Data Analysis)

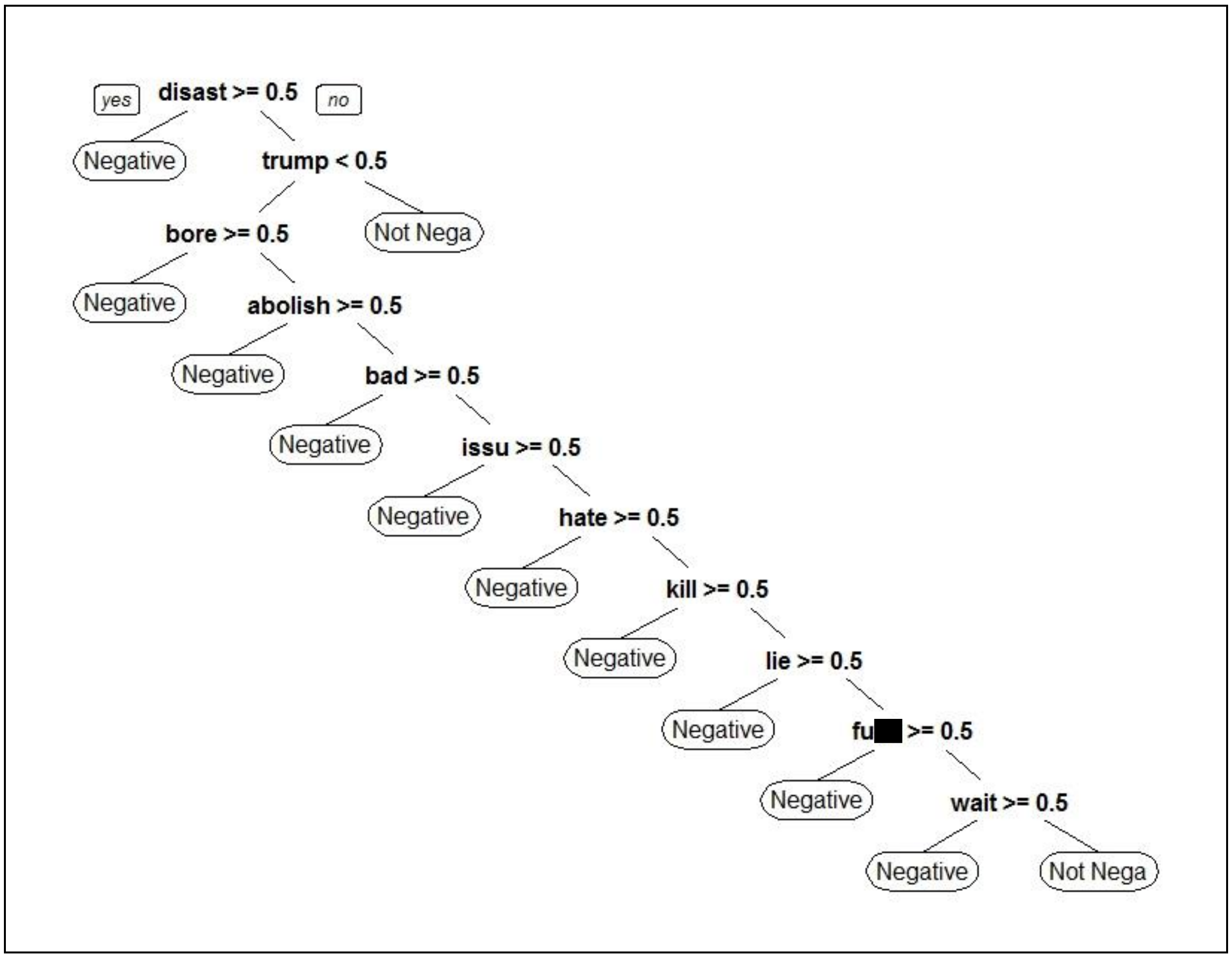
The R script below is used to perform basic analysis of the tweets. As in the other appendices, comments are embedded in the script and are indicated by R's commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(36)    wordcloud(corpus, scale=c(5,0.5), max.words=100,
      random.order=FALSE, rot.per=0.35,
      use.r.layout=FALSE, colors=brewer.pal(8, "Dark2"))
(37)    corpus_df = as.data.frame(corpus)
(38)    some_tweets$unstem = corpus_df$text
(39)    pos <- scan('C:/XXXX/positive-words.txt', what='character', comment.char=';') #Replace the
        X's with the correct path to this file
(40)    neg <- scan('C:/XXXX/negative-words.txt', what='character', comment.char=';') #Replace the
        X's with the correct path to this file
(41)    Dataset <- some_tweets$unstem
(42)    Dataset <- as.factor(Dataset)
(43)    score.sentiment <- function(sentences, pos.words, neg.words, .progress='none')
        {
         require(plyr)
        require(stringr)
        scores <- laply(sentences, function(sentence, pos.words, neg.words){
        sentence <- gsub('[[:punct:]]', "", sentence)
        sentence <- gsub('[[:cntrl:]]', "", sentence)
         sentence <- gsub('\\d+', "", sentence)
         sentence <- tolower(sentence)
        word.list <- str_split(sentence, '\\s+')
        words <- unlist(word.list)
        pos.matches <- match(words, pos.words)
        neg.matches <- match(words, neg.words)
        pos.matches <- !is.na(pos.matches)
        neg.matches <- !is.na(neg.matches)
        score <- sum(pos.matches) - sum(neg.matches)
        return(score)
        }, pos.words, neg.words, .progress=.progress)
        scores.df <- data.frame(score=scores, text=sentences)
        return(scores.df)
        }
(44)    source("scoresent.R")
(45)    scores <- score.sentiment(Dataset, pos.words, neg.words, .progress='text')

(46)    some_tweets$scores = scores$score
(47)    write.csv(some_tweets, file="tweetsandscores.csv", row.names=TRUE)
(48)    ggplot(some_tweets,aes(x=scores)) + geom_histogram(bins=27) + theme_bw()
(49)    some_tweets$scorescat[some_tweets$scores < 0] <- "Negative"
(50)    some_tweets$scorescat[some_tweets$scores >= 0] <- "Not Negative"
(51)    some_tweets$scorescat = as.factor(some_tweets$scorescat)
(52)    corpus = tm_map(corpus, stemDocument, language = "english")
(53)    corpus = tm_map(corpus,PlainTextDocument)
(54)    frequencies = DocumentTermMatrix(corpus)
(55)    findFreqTerms(frequencies, lowfreq=20)
(56)    sparse = removeSparseTerms(frequencies, 0.995)
(57)    sparse
(58)    tweetsSparse = as.data.frame(as.matrix(sparse))
(59)    colnames(tweetsSparse) = make.names(colnames(tweetsSparse))
(60)    tweetsSparse$scorescat = some_tweets$scorescat
(61)    split = sample.split(tweetsSparse$scorescat, SplitRatio = 0.7)
(62)    train = subset(tweetsSparse, split==TRUE)
```

```
test = subset(tweetsSparse, split==FALSE)
tweetTREE = rpart(scorescat ~ ., data=train, method="class")
prp(tweetTREE)
predictTREE = predict(tweetTREE, newdata=test, type="class")
confusionMatrix(predictTREE,test$scorescat)
```

Appendix 5 (Classification Tree)



Microsoft Excel®: Is It An Important Job Skill for College Graduates?

Sam K. Formby
formbysk@appstate.edu

B. Dawn Medlin
medlinbd@appstate.edu

Virginia Ellington
ellingtonve@appstate.edu

Computer Information Systems & Supply Chain Management Department
Appalachian State University
Boone, NC 28608, USA

Abstract

Several studies have found that a large percentage of middle-skilled jobs require at least a basic understanding of spreadsheets, and some even advanced level skills. A study was conducted at a four-year university to identify Excel skill sets that were determined as necessary by employers of the university's current students, advisory boards, recruiters, and other relevant parties. The findings suggested that the needs and opportunities for Excel® based analytical skills is pervasive in businesses of all sizes and ubiquitous in business.

Keywords: Microsoft Excel®, job opportunities, analytical skills, business curriculum

1. INTRODUCTION

Excel was launched by Microsoft in 1985, and has grown in use by businesses in their need for charts, graphs, statistical data computations, and formula creation. With that growth came the need for individuals with spreadsheet skills.

Previous studies have found several skills such as communication and relationship building skills to be necessary in the workplace, while 80% of middle-skilled jobs have been found by online recruiters to require at least a basic understanding of Excel skills. Several reports have also found that advanced analytical skills, Excel® in particular, results in increased marketability and increased compensation for graduates.

For business students, spreadsheet knowledge is imperative in order for the likelihood of success in the job market. Therefore, a business school should pose the question "What Microsoft Excel® skills are necessary to be taught to students for them to be successful in acquiring a job?" A second question can direct a business program by questioning, "What is the purpose of a college education?" Gerstein and Friedman (2016) point out that the answer to this question has varied over the years and still varies greatly today in different institutions. Should the Bachelor of Science in Business Administration (BSBA) curriculum focus predominantly on theory, concepts, critical thinking and knowledge? How much effort should be spent on skills development? With greater intensity employers are demanding skills and competencies that ensure students are productive and resilient in life in spite of the high degree of change being experienced in many professional fields. Barrett (2015) quotes Ronald Reagan in 1967 who said "we can no longer afford intellectual luxuries in universities." His point was that that education should create productive and economically sustainable members of society, implying that knowledge alone without skills is incomplete education. Over 80% of freshmen entering college say that the purpose of education is to get a good paying job and businesses are demanding basic skills beyond traditional topical knowledge in the employment process (Barrett, 2015). Freidman and Friedman (2015) make a compelling case that institutions must stress and teach skills that help students be successful and survive and thrive in the new knowledge economy.

The need for graduates to start jobs with sound functional analytical skills, e.g. proficiency with Microsoft Excel®, is therefore becoming more of a prerequisite for employment. According to Dana Manciagli (2013), author of *Cut the Crap, Get a Job,* found that students must have not only a proficiency in Excel, but have advanced skills. "A white paper study commissioned by Microsoft and released by IDC, October 2013, reported that the top two skills cited in over 14 million job postings for the top 60 job growth occupations of the economy were oral and written communications and Microsoft Office skills. Microsoft Excel® was cited as the most ubiquitous analytics tool in business. Geiger (2015) reported on a study finding that 78% of middle-skilled jobs require digital skills like Excel®. Middle-skilled jobs are fast growing job categories that place more emphasis on skills than on the having a bachelor degree, and in many cases pay more than traditional jobs requiring the bachelor degree alone. These jobs require significant business understanding, but with the added emphasis on skills to apply quantitative business intelligence to decision making. Some could argue that business graduates should aspire to more senior management positions, but it is more appropriate to consider these middle-skilled positions as part of an accelerated path for business graduates into management. The business education, coupled with the analytical skills with tools like Microsoft Excel®, place the business graduates in a highly favored position for future leadership. Business school graduates are therefore often in competition for these higher paying jobs and the skills proficiency makes the difference in the hiring selection (Geiger, 2015). Soergel (2015) reports additional details on the middle-skill job study which concluded that salaries are 13-38% higher based on the analytical tools skills a candidate has when interviewed. Soergel concludes with the following quote from the study, "Jobs requiring advanced analytical tools skills offer the strongest opportunity for middle-skill job seekers in terms of salary and growth as well as career advancement. Effectively, entire segments of the U.S. economy are off-limits to people who don't have basic analytical skills." Thus new BSBA graduates need to have these analytical skills to compete for these higher paying middle-skilled jobs.

There also appears to be a shift in hiring criteria such that skills are gaining in importance, and degree, school name, and GPA are dropping in importance, (The Role of Higher Education, 2012). This message was also reinforced at the Association to Advance Collegiate Schools of Business (AACSB) workshop "Co-Lab Connecting Business Schools with Practice" in June 2016. In the session "Recruitment, Retention, and Engagement" panel speakers commented that talent acquisition was the #1 issue with many companies today and candidates with cyber and analytics skills and competencies were drawing significantly higher salary offers. Many companies are also establishing baseline quantitative assessment tests at part of the screening and interview process to ensure essential skills and competencies are present before hiring.

In the broader context of business analytics and big data trends, Microsoft Excel® is still the "ubiquitous and popular choice" for data analysts, (Dumbill, 2012). Madhu Reddy, Senior Product Planner for Microsoft Big Data, stated that the interoperability Excel® with other BI and big data systems and applications is an obvious high priority in Microsoft, (Dumbill, 2012). In many cases, analytical skills are becoming more important in the employment decision than whether or not a candidate has a bachelor degree. A search for what specific Microsoft Excel® skills are important to employers will find a number of very general suggested lists such as Sravani (2016) but very few of these sources specifically address the needs of business school graduates.

This paper summarizes exploratory research collected over a multiyear period aimed at identifying specific Microsoft Excel® skills important for BSBA graduates to have at graduation. The authors acknowledge the differences of opinion, some strong, on the subject, but take the position that importance of Microsoft Excel® skills is significant to employers and affects opportunities for BSBA graduates whether or not one agrees that it is the role of the business school curriculum to address it. Another question that may be asked is whether it should be the student's responsibility to learn these skills independently thus demonstrating the initiative to bridge the gap between traditional education and the needs of the workplace. Regardless of where educators feel the responsibility lies, employers are placing increasing importance on these skills at graduation.

2. METHODOLOGY AND RESULTS

By, definition, exploratory research often relies on collecting and analyzing data and information in a variety of ways from disparate primary and secondary sources, (Shields & Rangarajan, 2013). Data was gathered through focused discussions in advisory board meetings and discussions with employers, recruiters, and students returning from internships. More quantitative data was also collected from Monster.com, employer surveys and student course feedback. Specific Microsoft Excel® skills that enhance business student success in internships and full-time employment opportunities were identified. Though exploratory research is generally only useful in gaining understanding of a phenomenon of interest and not direct problem solving, the feedback from these activities was consistent enough to allow creation of a recommended set of analytical and Excel® skills that would enhance graduates' success.

Advisory Board Meetings – Evaluating the Need

In 2012-2013, discussion items were included on the agendas in advisory board meetings at the dean's and several department levels. The question asked was open ended. Describe the types of analytical tasks assigned and Microsoft Excel® skills expected of new BSBA graduates. Members of these board were all managers, and though the feedback was consistent, it was not as detailed or specific as desired. However, the conclusion was clear: more advanced Excel® skills are desired of both business student interns and graduates. The descriptions of analytical tasks were a little more specific with descriptions like inventory management, scheduling, financial and account analysis, performance analysis and metrics creation. These discussions clearly validated the need identified earlier through the spontaneous feedback from advisors and recruiters.

Job Posting Study

Following these discussions, a study of "entry level, bachelor degree, business jobs" in Monster.com was made using the available search tools. A comparison was made in six states of all jobs meeting the above criteria and all jobs meeting the same criteria but also calling for Microsoft Excel® skills. The results are shown in Table 1. There was surprising consistency of the results across several regions of the U.S. In the body of the job descriptions, the most frequent terminology used by the employers was advanced Excel® skills or proficiency in Excel®. These results are based on a generic BSBA search.

State	% Requiring Excel®
North Carolina	43.48%
Virginia	51.43%
Georgia	44.53%
South Carolina	46.15%
California	51.44%
New York	51.03%

Table 1. Monster.com Results

This element of the exploratory research confirmed that the feedback we received from our advisors and employers was not specific to our BSBA programs, but a more universal issue. Though the majority of our graduates fill positions in the Southeast U.S., this element of the study suggests that the desires of employers are relatively consistent across the U.S.

Employer Survey

As part of a larger 2013-2014 survey of 107 advisory board members and employer managers, two seven point Likert scale statements were presented. The first was "Data analytics (quantitative tools to analyze business data to support decision making) is a very important skill for students." Overall 94% agreed or strongly agreed with this statement; 67% strongly agreed; 27% agreed and none disagreed with the statement. The second statement was "Being able to create spreadsheets, charts and graphs and analyze data with Excel® are very important skills that students need when they graduate." Overall 96% strongly agreed or agreed; 77% strongly agreed; 19% agreed, and none disagreed.

The conclusion from these initial three exploratory activities was that the need for stronger Excel® skills was very significant and warranted more detailed investigation into the specific skills needed or expected by employers.

Identifying Specific Excel® Skills

In 2013-2014, at the end of each summer, students returning from summer internships and entering into their senior year of study were invited to a "debrief" session to discuss what the school could to better to prepare them for success in their internships. They were asked to describe their specific work assignments and what analytical tools they used. Without exception, all of the interns reported that they spent a significant portion of their time working in Excel® using Pivot Tables and a variety of other functions and spreadsheet operations. There was a wide diversity of business disciplines being performed but Excel® was the common tool used to support all of the reported assignments, and interestingly pivot tables were used more than any other tool within Excel®.

Creating and Piloting a Learning Experience
In 2014-2015, additional web research was conducted in an effort to identify specific Excel® skills that would be of greatest benefit for BSBA students to have after their junior year in support of internships and at graduation. Though many reports of the need for Excel® skills were found on websites, very few details identifying specific skills were found, except in a few topical areas such as accounting and operations research.

Using data gathered from students returning from summer internships as a starting point, two major employers in the area were contacted to gather more specific information about the skills required and the types of business applications new graduates are likely to be challenged with. Based on past hiring experiences with these two employers, it was felt that if the graduate skill levels met their expectations, then they would likely be acceptable for most of the other employers. One of these employers utilizes an Excel® test in the new college graduate interview process. The other employer asks detailed questions about specific Excel® skills during the interview. In addition, several visits (approximately 16 hours total) were made in 2015 to both company locations to talk to work teams and hiring managers about specific day-to-day activities requiring Excel® and the desired level of skills for new college graduates in their businesses. A specific list of skills was developed from all of the input received.

Several of the more vocal advisory board members from the earlier 2012-2013 meetings were asked to comment on the list, and they confirmed that the initial identification of specific skills was a good starting point. The list of specific skills is in Table 2.

Data from real constituent businesses, the U.S. Census Bureau, and a variety of internet sources were used. In addition, the book "Problem-Solving Cases in Microsoft Access and Excel" by Monk, Brady, and Cook (2014) was used as a support text in some case assignments.

1	Absolute, relative and mixed addressing in functions and formula and between worksheets
2	Pivot tables and charts
3	Scenario Manager (alternative analyses)
4	Solver (optimization)
5	Goal Seek
6	Performing Access queries
7	Importing text and other types of data files and queries from Access into Excel®
8	Evaluating and cleaning dirty data
9	Data entry validation tools, using drop down menus, instruction and error messages, comments, and comparison of data entries to expected ranges
10	Creating Dashboards and KPIs with multiple pivot charts synchronized
11	Creating Slicers and Timelines also synchronized
12	Linking Slicers and Timelines to multiple Pivot tables
13	Vlookup and Hlookup, comparing lists
14	Sumproduct, Sumif, Averageif, Countif
16	Nested if's
17	Nested functions
18	Index
19	Time and date calculations
20	Switching between range data and table data and using table functions including
21	Using filters, including custom text filters, in tables for content analysis

Table 2. Specific Excel® Skills

Objectives of the course included understanding common real-world business problems and learning Excel® skills that can be used to support better business decisions in each case. Two industry speakers were also brought in as guest lecturers to discuss real-world day-to-day data challenges and the analytics needed to make better business decisions in their companies.

Student Feedback
Towards the end of the course the students completed a survey asking them a number of questions about the perceived value of the different Excel® topics and case based exercises in the course. Almost all topics received the highest rating of "Very important." In the qualitative section of the survey, the most favorable comments related to the extensive practice with Pivot Tables and Charts including all of the various options for sorting, filtering and grouping. Topics related to time and date functions and "Index" only received "Important" ratings, which were the lowest ratings of the topics covered. All students felt that a course, like the one they completed, based on experiential case-based learning and incorporating Excel® skills development would be extremely helpful in preparing future students for summer internships and fulltime employment.

The Investigative Course Project
In addition to the business cases and skill development sessions, the students were also assigned an investigative project to interview management in a real business they have access to "to better understand the common daily analytical needs of businesses today and the analytical capabilities and desired skills of professionals in their organizations from new college graduates through first level managers." In many cases, the students interviewed employers, working parents or other relatives and friends. At the end of the semester students presented their findings in project reports before the class. The project guidelines are provided in the Appendix.

The range of firm sizes was from very small firms such as family owned restaurants to very large global corporations. In the presentations of the student findings, there was an apparent pattern of maturity in the use of Excel® skills in business processes and decision making. Two of the smallest firms were still tracking all costs and orders manually and had no data entered into computers, but were very interested in the students' suggestions for how to facilitate operations with Excel®. Other small companies tended to use Excel® for very basic operational functions like setting up employee schedules and tracking hours worked. The next level of utilization included inventory tracking and management and basic accounting functions such as cost tracking of receipts and expenses. More mature, slightly larger firms were using the available information and data histories to do forecasting of labor and inventory needs. The next level of maturity seemed to be the integration of data from multiple departments into single spreadsheets for enhanced forecasting and performance tracking. Next, there were activities aimed at analyzing opportunities to improve performance metrics such as throughput, order cycle time, increased capacity, etc. Lastly, analyses were being performed to assess specific customer service metrics and utilizing customer system data to find ways to improve customer satisfaction. The students found that even in largest companies which utilized packaged enterprise systems and cloud data warehouses, employees from senior managers down were continually querying selected subsets of data and using Excel® to analyze scenarios to answer specific business questions.

In the project reports, perhaps the biggest or most significant revelation that the students consistently reported was that almost all businesses have a lot of data and there was great potential to use it to make better business decisions, but due to lack of skills and time, they were not effectively using what they had or doing everything they wanted to do. Essentially all of the businesses interviewed stated that they wish that they had new college hires or skilled employees who could help them make smarter business decisions analyzing the data they have.

Several anecdotal reports were also very interesting. One large Fortune 50 company interviewed said that Excel® was so important and necessary for daily performance of duties that they were giving all business school graduates an Excel® test as part of the interview process as a screening tool. Students with excellent academic credentials who could not pass the Excel® test were eliminated from consideration and much more consideration was given to students who passed the test even with lower academic credentials.

Another large multinational manufacturing company reported that they had a policy of paying premium pay, up to 25% more, for student interns who had high levels of Excel® skills. Additionally, they placed them higher in their rankings as potential full time employees.

At the end of the special topics course, the course outline, learning objectives and some of the results were reviewed with one of the hiring managers who provided input into the design of the course. Their recommendation was that students who graduate with the recommended skills would likely be placed close to the top of their candidate list for consideration.

3. FUTURE RESEARCH

The elements of this exploratory research involved a variety of data sources as well as both quantitative and qualitative investigation. Results supported a successful pilot course design with positive student and employer feedback. Based on the activities to date, two recommendations are provided. First, a more comprehensive survey of a broader cross-section of employers is desired with a possible investigative look at commonalities and differences in specific skills and tasks of greatest importance to each major, e.g. accounting, finance, management, marketing, computer information systems, economics, etc. For example, there is evidence that accounting and finance majors may need a slightly different skill set than other business majors. Ideally, curriculum changes could incorporate common skills in earlier core courses and more advanced major courses could incorporate a more specific skillset.

The second recommendation would be to expand on the task specific investigative project to better understand the market needs and opportunities of employers recruiting BSBA students. As described in the course investigative project section, there appears to be some significant patterns of analytics maturity in companies of varying sizes and demographics. Understanding these maturity differences and potential opportunities in employers' capabilities may significantly inform the types of case-based assignments appropriate for future classes.

4. CONCLUSIONS

This paper summarizes seven exploratory research activities aimed at supporting curriculum decisions that would better prepare BSBA students to meet employer expectations relative to Microsoft Excel® skills. The first three elements of this exploratory research, advisory board meeting discussions, a job posting study and an employer survey validated a very real need for BSBA students to have stronger Excel® skills. The job posting study suggests that this need is widespread and potentially a national need.

The internship interviews, pilot course experience including employer feedback, and end of course student survey feedback suggest validity in identifying the initial set of specific skills needed.

Though this is an exploratory study, and additional research is recommended, the results appear quite significant. It is recommended that each institution evaluate the issues presented here with their respective constituency and the effects these issues have on their graduates' competitive marketability and institute appropriate curriculum enhancements.

6. REFERENCES

Block, B. A. (2016). The Professoriate and the Future of Higher Education Kinesiology. Quest, 1-11.

Dumbill, E. (2012). *Planning for big data*. " O'Reilly Media, Inc.".

Friedman, H. H., & Friedman, L. W. (2016). Six Steps to Transform an Ordinary College into an Exceptional Institution. Available at SSRN.

Geiger, B. (2015, March 6). Your Excel skills could land you your next job. http://fortune.com/2015/03/06/microsoft-excel-jobs/

Gerstein, M., & Friedman, H. H. (2016). Rethinking Higher Education: Focusing on Skills and Competencies. Gerstein, Miriam and Hershey H. Friedman (2016), "Rethinking Higher Education: Focusing on Skills and Competencies," Psychosociological Issues in Human Resource Management, 4(2), 104-121.

Manciagli, D. (2013). *Cut the Crap, Get a Job.* Authority Publishing, Gold River, California.

Monk, E., Brady, J., & Cook, G. S. (2014). Problem-Solving Cases in Microsoft Access and Excel. Cengage Learning.

New study reveals most important skills for students. (2013, October 15). https://news.microsoft.com/2013/10/15/new-study-reveals-most-important-skills-for-students/#sm.00001jtbe2bwa3dc0vnmih8qov2ol

Shields, P. M., & Rangarajan, N. (2013). *A playbook for research methods: Integrating conceptual frameworks and project management*. New Forums Press.

Soergel, A. (2015, March 5). Want a Better Job? Master Microsoft Excel. http://www.usnews.com/news/blogs/data-mine/2015/03/05/want-a-better-job-master-microsoft-word-excel

Sravani, (2016, June 7). Essential Excel Skills Employers are looking for in Candidates. http://content.wisestep.com/essential-excel-skills-employers-looking-candidates/

The Role of Higher Education in Career Development: Employer Perceptions (2012, December) https://chronicle.com/items/biz/pdf/Employers%20Survey.pdf

APPENDIX

Excel® Interview Project Guidelines

Identify a company, or work group within a company, where you can interview one or more employees, preferably a manager. You might consider a company where you have worked part-time, interned, or where friends or family members work or have worked. It would be very helpful for you to have some level of understanding of the daily operations and the measures of business performance deemed important by the organization. If you can't readily identify a company, let me help you, or you can try to "cold call" a company and see if they would be willing to talk to you for a few minutes and give you an interview for a student project.

For the interview, please explain that you are a student in an advanced Excel® business analysis class and that you are seeking to better understand the common daily analytical needs of businesses today and the analytical capabilities and desired skills of professionals in their organizations from new college graduates through first level managers.

Before the interview begins let them know the following background information:

1. You will report your findings in a student project report and presentation to your class.
2. Generic aspects of the findings, may be aggregated with feedback from other companies, and used in university studies and research supporting curriculum enhancement decisions, but with no company or individual names mentioned.
3. Anything shared in the interview that they do not wish to be included in the report, will be omitted.
   If they agree to the interview after they understand how the information will be used, then ask them to share some basic demographic information about their business:

1. Approximate Size- number of employees at this location
2. Industry type- Retail, distribution, manufacturing, logistics services, financial services, etc.
3. Primary product or service
4. Primary customer market served
   If, after the interview, they ask your opinion of what more can and should they be doing, you may offer to take back the interview information and then schedule a second appointment to discuss some possible options based on things you are learning in class. I would be glad to meet with you and help you develop return feedback. Being able to offer specific advice on how they might improve some aspect of their business, could be a big plus on a resume` and give you a great topic for discussion in job interviews.

Interview questions:

Focusing on daily business decisions typically made by professionals in your firm, from new college hires to first level managers:

1. What data analyses do you routinely do to help you make business decisions?
a. Can you describe some examples?
2. What analytical software tools do you use?
3. How do you routinely use MS Excel® in your current work?
a. Can you describe some examples?
4. What data do you, or your company, routinely collect to support analyses? What formats are these data in, e.g. hard copy files of paper forms, spreadsheets like Excel®, database management system like MS Access, etc.?
5. What additional analyses would you like to have or be able to do routinely to better support your business? What new data is needed?
a. Can you describe some examples?
6. What analytical skills do you wish new college graduates were stronger in?
a. Can you describe some examples?

Following the interview(s), prepare a short report, approximately 2,000 words. The report should include five sections. During the class time, last week of classes, each student will be asked to summarize, in 3-5 minutes max, what was learned in the interview(s).

Five Sections

Interview questions 1, 2, and 3 support sections 2 and 3. Question 4 supports section 4. Questions 5 and 6 support section 5.

1. Introduction and demographics: include the name and location of the company, persons interviewed, size of the location/company, and brief description of the mission of their business.
2. Current capabilities and practices using Excel®, Access, and/or similar tools for data management and analysis.
3. Types of business decisions supported by these analyses
4. Current data available and format in collection.
5. Additional analyses needed to better support business decisions, and desired level of analytical skills needed by new college graduates.

Comparing Student Interaction in Asynchronous Online Discussions and in Face-to-Face Settings: A Network Perspective

Elahe Javadi,
ejavadi@ilstu.edu
School of Information Technology
Illinois State University
Normal, IL 61790

Judith Gebauer
gebauerj@uncw.edu
Cameron School of Business, ISOM Department
University of North Carolina Wilmington
Wilmington, NC 28403, USA

Nancy L. Novotny
nlnovot@ilstu.edu
Mennonite College of Nursing
Illinois State University
Normal, IL 61790

Abstract

Online discussions enable peer-learning by allowing students to communicate ideas on what they have learned in and beyond the classroom. Peer-learning through online discussions is fostered when online discussions are interactive. Interactivity occurs when students refer to and use perspectives shared by peers, and elaborate, respond to, or propose alternative views to those shared by others. Open interactions in online discussions require students to choose whom they communicate with in the discussion forums. This study examines the extent to which the patterns of student-to-student interactions in online discussions resemble student interactions with the same peers in face-to-face settings. Online discussion data were collected in six sections of an introductory IS course over three semesters. Each section's dataset contains data from four online discussions among students, as well as the results of two familiarity surveys administered at the beginning and at the end of the semester. The results of the data analysis suggest a relationship between face-to-face interactions and patterns of online group idea sharing and integration. Understanding the structure and dynamics of interactions in online discussions can provide design guidelines to help overcome inherent familiarity fault-lines in classes, and to improve the extent and quality of peer-learning in online discussions.

Keywords: Asynchronous online discussions, interaction, familiarity, peer-learning

1. INTRODUCTION

Learning management systems (LMSs) are used extensively in higher education (Waters & Gasson, 2006). LMSs provide a platform where instructors and students can share resources, creative works, and opinions on course-related topics. LMSs' asynchronous online discussion (AOD) tools support peer-learning because they can help remove obstacles such as production blocking and cognitive interference that often exist in verbal face-to-face and synchronous discussions. To enhance peer-learning, interactivity must be fostered. Interactions in AODs occur when students refer to and use ideas posted by their classmates, and when they provide elaborations, responses, counter-arguments, or alternatives thereto (Gruenfeld & Hollingshead, 1993; De Vreede, Briggs, van Duin, & Enserink, 2010). To elaborate and respond, students must attend to each other's ideas (Gruenfeld, Mannix, Williams, & Neale, 1996). They must also reciprocally value ideas in order to afford the cognitive efforts necessary to elaborate (Gruenfeld et al., 1996). In other words, elaborations require attention to the ideas of others as an enabling factor, and valuing others' ideas as a motivational factor (Javadi, Gebauer, & Mahoney, 2013). Previous research on group brainstorming and decision making provides insights into how the characteristics of a group affect group processes, such as information sharing and processing, elaboration, and consensus making. Homan, Van Knippenberg, Van Kleef, & De Dreu (2007), for instance, studied groups in which members possessed diverse information and discovered that fostering pro-diversity beliefs enhanced information elaboration in those groups. Prior research studies have also examined how familiarity among members and group social ties may influence cognitive processes that underlie information integration (Gruenfeld et al., 1996; Goodman & Leyden, 1991). Gruenfeld et al. (1996), for example, compared groups with different levels of familiarity among their members and found that while familiar groups were more effective in information sharing, unfamiliar groups were more effective in information integration. Prior literature has also shown that people tend to cluster around members who they feel most comfortable with (Cunningham et al. 2012). Clustering around familiar partners in online settings may lead to segmentation within discussions that can again limit the breadth and depth of peer-learning in AODs.

Because familiarity has been found to affect information sharing and information integration—two critical processes for creating effective group discussions—the current research examines the association between face-to-face interactions and interactions in online discussions applying social network analysis (SNA) (Gasson & Waters, 2011; Borgatti, Everett & Freeman, 2002). SNA methods have been used in prior literature to study the make-up of online discussions. Waters and Gasson (2012) used SNA to study the effect of course scaffolding on AODs. They specifically examined impacts of instructions given to students (general vs. structured), number of posts by course instructor, and level of moderation by the instructor (low vs. high) in relation to structure of the interactions network in online course discussions. To measure extent and quality of interactions in online course discussions, Waters and Gasson (2012) quantified number of messages, participants in threads, and maximum depth of threads. They also included behavioral measures to distinguish between peer-to-peer versus broadcast messages, and between student-to-student versus student-to-instructor interactions.

The current research focuses on how face-to-face familiarity among students relates with patterns of interactions in online course discussions. Therefore, our main research question is to what extent the structure and dynamics of student interactions in online discussions resemble the structure and dynamics of student face-to-face interactions. We operationalize our research question with two hypotheses:

Hypothesis 1: *The structure of student interaction in online discussions resembles the structure of student face-to-face interaction.*

Hypothesis 2: *The evolution of student interaction in online discussions resembles the evolution of student face-to-face interaction.*

Hypothesis 1 states that the structural properties of online interaction networks are expected to resemble the structural properties of face-to-face interaction networks. Hypothesis 2 implies that as the four commenting networks and the two face-to-face familiarity networks are examined, the earlier commenting networks will show more similarity to the first face-to-face familiarity network while the later discussions networks are expected to show more similarity to the second face-to-face familiarity network. In other words, as online commenting interactions evolve over the course of the semester, that evolution is expected to resemble the evolution of face-to-face familiarity links among the students.

2. METHOD

Student *interaction in online discussions* was operationalized based on the comments that students posted on each other's ideas during four online discussions over a 16-week semester; *interaction in face-to-face* was measured based on a familiarity survey administered at the beginning and at the end of the semester. The survey asked students to self-report the extent to which they knew/interacted with other students at the time of the survey. Online and face-to-face interaction networks were then compared using SNA methods.

Several measures of network structures were examined to compare the structures of online and face-to-face interactions. To examine the evolution of online interaction networks we compared the commenting interactions during four discussions that occurred at different times during the semester. The evolution of face-to-face interactions was examined by comparing networks of familiarity measured at the beginning and at the end of the semester. The structures of the four commenting and two familiarity networks were then compared, taking into account the time at which they were measured. Our hope is that a better understanding of the associations between online and face-to-face interactions can guide the design and implementation of interventions that could bridge familiarity fault-lines and thus promote a higher level of peer-learning in AODs.

Data Set

Research data were collected from six sections of a 200-level course. The collected data include student interactions during four online discussions that took place on the course's LMS. The four discussions comprised twenty percent of the final course grade (five percent each). Per instructions, students were encouraged to think critically about a specific course-related topic and were asked to engage in an online conversation with their classmates. To start the conversation, the instructor used a prompt related to the topic. The articles and topics for each of the four discussions were identical across the six course sections. For each student, a discussion assignment involved posting one original idea and four comments on the contributions of other students. Each discussion was completed in two phases. During the 1st phase all students were required to post their original analysis; and during the 2nd phase, students were required to post four comments on any of the original analyses that were posted in the 1st phase. The two-phase design was chosen to remove the impact of 'time of post' on the extent to which a certain post received comment from others. The instructor provided students with examples of acceptable posts and comments. For instance, "*I agree with the authors' argument that the world is spiky and not flat, but I find the evidence insufficient, mainly because the authors have focused on the number of patents, which is only one indicator of creative production*," was listed as an acceptable post. In contrast, "*I liked the article*," and "*I also think the world is spiky*," were listed as unacceptable posts. For the comments, "*I agree with you, but if I look closer, I find it difficult to measure other forms of creative production. Number of patents is not a perfect indicator, but it is a precise and reliable indicator*," was listed as an acceptable comment and, "*I agree*," as an unacceptable comment.

The familiarity survey was administered twice: at the beginning of the semester and at the end. The first questionnaire asked students to report the extent to which they knew each of their classmates before attending the class, using a scale from 1 to 5. The questionnaire had a table with one row for each student and five columns that indicated the five levels for measuring familiarity. Students were expected to fill the table, one row at a time (keeping the row for their own names empty), and put an X mark on the column which best explained their face-to-face interactions with the student whose name was written in that row. The level 1 anchor represented "not familiar" (*never heard of or have seen this person before attending this class*) while the level 5 anchor represented "very familiar" (*have known this person and/or worked with them before attending this class*). The second familiarity questionnaire asked students to report the extent to which they knew each of their classmates at the end of the semester. The new descriptions for the 5-point scale read as follows: Not familiar (*I don't know this person and I have not talked with them during the semester*) for level 1, and Very familiar (*I know this person and I have worked with them during the semester*) for level 5. An assumption was that the in-class group-based activities, a four-week group project, and the instructor's use of a grouping mechanism to encourage students to partner with less-familiar classmates, would contribute to a higher level of interpersonal recognition reported at the end of the semester.

To compare the structure and evolution of student interactions in the online discussions and interactions in face-to-face, two-dimensional matrices of the commenting links and familiarity links were constructed. The two-dimensional matrices were organized using the student codes in the first row and in the first column. For a course section with ***n*** students, the matrix therefore resulted in ***n×n*** cells. If student ***i*** commented on student j's post ***m*** times, then the entry in cell ***(i,j)*** was set as ***m***. In a subset of analyses in this study, a binary version of commenting matrices were used. In binary matrices, cell values are set as 1 for cell ***(i,j)*** if student ***i*** ever commented on student ***j***'s post and 0 if s/he did not. Familiarity links were stored in similar two-dimensional ***n×n*** matrices. If a student ***i*** rated their familiarity with student ***j*** as ***r*** (on a scale from 1 to 5), then the entry in cell

(i,j) was set to ***r***. In a subset of analyses in this study, the familiarity information was noted in a binary matrix in which a 1 in cell ***(i,j)*** indicates that student ***i*** provided a rating for his/her familiarity with student ***j***, at a level of 3 or higher, whereas a 0 in cell ***(i,j)*** indicates that the rating was less than 3.

Analyses
Social Network Analysis (SNA) methods and measures were employed to compare the interactions in the online discussions and face-to-face interactions. Social network analyses are generally useful to examine the connections among entities (e.g., individuals, institutions, research papers) who together comprise a network. Connections can be of different types, including trust, friendship, merger, or citations. In the current study, two types of connections are of particular interest: commenting and familiarity.

The network analyses included four steps. In the first step, we assessed the online interactions during the four discussions based on four measures of network structures: (1) density (2) centrality, (3) reciprocity, and (4) clustering coefficient. The four measures and their implications for online discussions are explained in the next section. In the second step, we performed node-level analyses on the commenting and familiarity matrices to assess the correlation between a particular student's statuses in the commenting networks and their statuses in the familiarity networks. In particular, we examined the correlation between the number of comments received and the number of familiarity links received for each student, as well as the correlation between the reciprocity for comments and familiarity links for each student. In the third step, we performed dyadic analyses to assess the similarities between interactions in the commenting networks and in the familiarity networks when compared at the dyadic level. Lastly, mixed dyadic-nodal analyses were performed to assess the associations between commenting connections and familiarity, i.e., students' tendency to comment more frequently on ideas posted by familiar others than on those by non-familiar others. The results of each analysis will be described in the following sections.

3. RESULTS AND DISCUSSION

Network-Level Analyses
We utilize two sets of directed graphs, one set describes the online interactions (Figure 1) and the other set summarizes the face-to-face interactions (Figure 2). Figure 1 describes the structure of the online discussion networks (D1 – D4) based on the measures of density, centrality, reciprocity, and clustering coefficient, in each of the six course sections (S1 – S6).

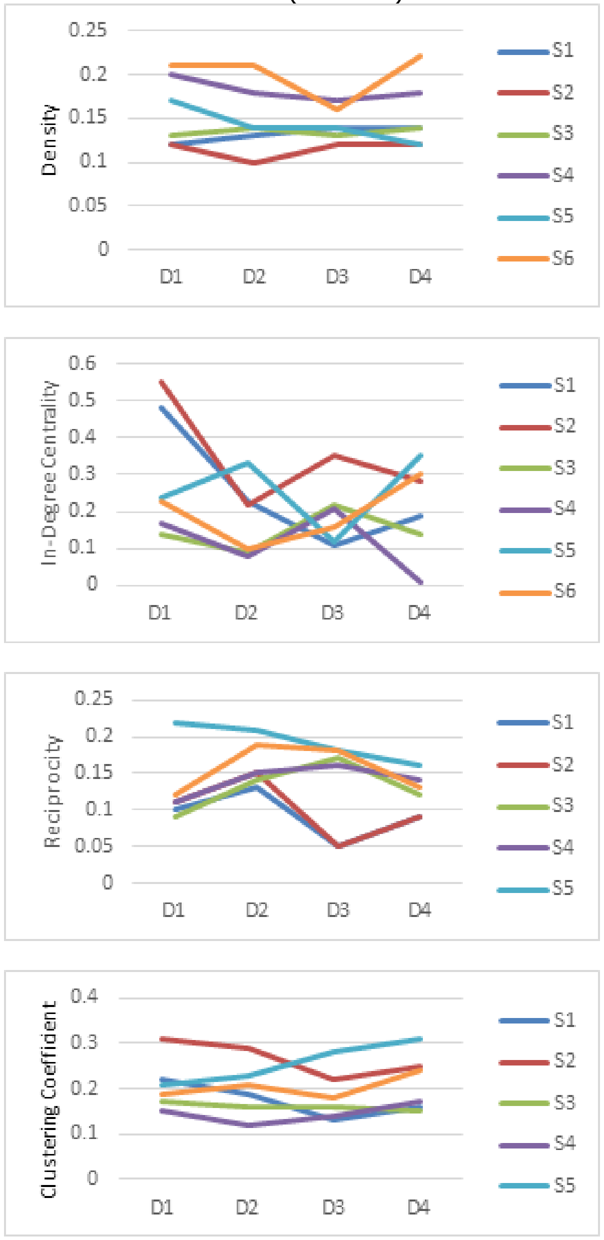


Figure 1: Four Measures of Discussion Network Structures (Discussions D1-D4, Sections S1-S6)

Because students were required to post four comments during each round of discussion, the number of comments posted by each student (i.e., out-degree measures) bears little information. However, it was useful to examine the differences in the number of comments that each student received on his/her posts; i.e., in-degree measures for the nodes in the discussion graphs.

Figure 2 illustrates the changes in density measures of network structures in face-to-face interactions between the beginning and end of the semester within each of the six course sections.

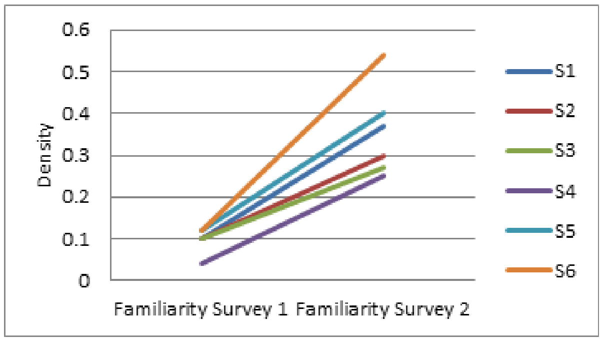


Figure 2: Network Density Measures

Node-Level Analyses

To examine hypothesis 1, node-level analyses were performed on the commenting and familiarity matrices. Node-level analysis examines the extent to which a certain student's statuses—as measured by centrality and reciprocity—in discussion and familiarity networks are correlated. Specifically, we examined the correlation between the number of comments received and existence of a familiarity link with each member in the discussion. In addition, we calculated the correlation between the extent to which a student's comments on other students' comments was reciprocated and the number of the connections (outgoing and incoming) that the student had in the familiarity network. Therefore, the two sets of correlation measures are: (1) the correlation between in-degree centrality measure in the discussion networks and the in-degree centrality measure in the familiarity network (Table 1 in the Appendix); and (2) the correlation between the reciprocity measures in the discussion networks and the degree of centrality in the familiarity networks (Table 2 in the Appendix).

In line with hypothesis 1, positive correlations exist in 84% of the cells in Table 1 (shaded cells), which means that 16% of the observations are not consistent with hypothesis 1. While the results do not fully corroborate hypothesis 1, the positive correlation in the majority of the cells in Table 1 suggests a tendency of face-to-face familiarity to transcend into online interactions, but also calls for additional analysis. Table 2 shows the correlation between reciprocity measures and degree centralizations. Positive correlation is observed in 74% of the cells, which means that 26% of the observations are not consistent with hypothesis 1. Although not fully conclusive, the mere presence of correlation is an intriguing observation that the extent of reciprocity is associated with popularity in class for the majority of the class discussions.

Dyad-Level Analyses

To better understand the association between the dyadic relationships in the discussion and familiarity networks, we used the Jaccard coefficient (Borgatti et al. 2002; Jaccard, 1912), a measure that can show the extent to which a dyad in one network (discussion) co-exists with its corresponding dyad in another network (familiarity). The coefficient is at its maximum of 1, if for every student ***j*** that student ***i*** has expressed familiarity with, student ***i*** has also commented on at least one of student ***j***'s posts. Moreover, for every student ***j*** that student ***i*** has expressed no familiarity with, student ***i*** has refrained from commenting on student ***j***'s posts. In technical terms, to calculate Jaccard's similarity coefficient for two binary vectors, the following are counted: total number of times that an element is 1 in both vectors (J_{11}) and total number of times an element is 0 in one vector and 1 in the other (J_{01},J_{10}). Jaccard's coefficient is then calculated as follows: $\frac{J_{11,}}{J_{01}+J_{10}+J_{11}}$. The Jaccard similarity coefficients for pairwise comparison of discussion and familiarity networks are reported in Table 3 (see appendix); the numbers were calculated using UCINet software (Borgatti et al., 2002).

The Jaccard coefficient numbers are significant in more than fifty percent of the cells (shaded cells) in Table 3. However, S3 is the only experimental group for which the result are consistent with this paper's proposed hypotheses. In S3, Jaccard's coefficients are significant in all eight cells in support of Hypothesis 1. Furthermore, the Jaccard's similarity coefficient has an upward trend for all four discussions. This is consistent with Hypothesis 2, implying that the dyadic dynamic in the discussion networks mirrors the dyadic dynamic in the familiarity networks. The observation in S3 is, thus, fully consistent with our hypotheses, which points to potential control variables that have been omitted in the study design. Additionally, the Jaccard coefficient for D3 & F2 and D4 & F2 are higher and more significant than their counterparts for D1 & F2 and D2 & F2, hence we observe consistency between the evolution of connections in the discussion and familiarity networks. For the Jaccard's coefficients that are not statistically significant, we examined Hamming distance and the derived match coefficient. The match coefficient was significantly higher than Jaccard's coefficient in only one of the not-significant cells. A low Jaccard coefficient implies that there are not many corresponding dyads (cell ***ij***=1) in discussion and familiarity networks. When match coefficient is high despite the low Jaccard's coefficient, it implies that although there are not many corresponding dyads in the two networks, the non-existence of dyads (cell ***ij***=0) in the networks match at a high level.

Mixed Dyad-Node Level Analyses

To further understand the impact of students' face-to-face interactions on students' online interactions, a series of mixed dyad-node level analyses were performed. Mixed dyadic-nodal analyses started with calculating Jaccard similarity measures for students based on students' face-to-face interaction. The Jaccard similarity measures were then used to identify clusters within each course section. For each course section, clustering schemes with 2 to 5 clusters were examined, and the clustering scheme with the highest fit measure was chosen. In all course sections, 2 or 3 clusters yielded better fit measures. The above steps were performed for the two familiarity matrices in all six course sections. The cluster membership information for students in each course section (clusterIDs) were used as basis for three types of mixed nodal-dyadic analyses: structural block model, constant homophily, and variable homophily. These analyses can identify within-cluster commenting tendencies in online discussions for clusters identified in face-to-face interactions. Although mixed dyadic-nodal analyses did not yield any conclusive results in the current study, the analyses are essential for identifying segmentations within a class and for alleviating the impact of within-class segmentations on online discussions. If the analyses are run during the semester and after each discussion is completed, they can guide interventions to counter segmentations effects.

4. DISCUSSION

As explained in the previous section, the observations in a subset of course sections and discussions were consistent with this study's hypotheses. While the presented analyses are not conclusive, they provide some insights into how to examine the relationship between online and face-to-face interactions. To advance our understanding of how online and face-to-face interactions co-evolve and to establish and investigate the direction of causality, the theoretical framework of the study must be strengthened. Future theoretical and empirical studies based on this research project should try to shed light on the nature and dynamic of individuals' information processing habits when online and offline interactions are used in tandem (Walter 1992).

Also, patterns and dynamics of this relationship may be impacted by individual and environmental factors (e.g., average student's age) which should be taken into account using control variables. Such control variables, which would represent natural variances in classroom atmosphere and student characteristics, could explain some of the disparities among the reported comparison measures for the six course sections in the current data set. Due to the institutional limits of classroom research and concerns regarding coercion and privacy, this study's design did not include student characteristics. We also believe that including quality of the posts and comments as an additional variable in the model will help advance the understanding gained from this research study.

5. CONCLUSIONS

In this paper, we discussed the setup, analysis, and preliminary results of an exploratory study to examine the links between online and face-to-face interactions among students. The results suggest an individual's tendency for online interaction with people who are more connected with her/him in face-to-face settings. Online interaction with connected others could therefore limit depth and breadth of peer-learning in courses. Network-level, node-level, and dyadic analyses in this study were mostly consistent with this study's hypotheses that structure and evolution of online interactions mirror those of face-to-face settings.

The implication for educators is that discussion dynamics should be observed and discussion rules and instructions should be evolved throughout the semester to address observed undesirable patterns. Instruction rules should strive to alleviate observed segmentations in face-to-face settings and encourage students' open conversation beyond the acquaintance links with peers in class. For instance, to avoid dominance of a few students in attracting comments, educators can require students to comment on a pre-defined number of peers (as opposed to their chosen subset of peers) throughout the semester. If higher levels of reciprocity are observed (but not desired), instruction rules can limit the number of times a student can engage in debate-type conversations. Conversely, if a debate-type online conversation is desired, instruction rules can guide the depth of discussion threads and encourage involvement of more students in a single discussion thread. It is essential that instructions and rules evolve as students' face-to-face and online interactions evolve during the semester.

Future studies should be conducted to help gain a deeper understanding of how students interact within a specific thread, and to what extent the characteristics of the interactions (e.g., length of thread, timing of responses, and reciprocity within a thread) are associated with the status of the student who initiated the thread. Within-thread patterns of interaction can also provide insights regarding the extent to which a student's familiarity with another student is associated with his/her interactions with a third student (e.g., author of an original post). Future studies could further examine how student involvement in online discussions correlates with student performance in the course, as measured based on exam or assignment grades.

A deeper understanding of discussion dynamics in the virtual classroom can help guide the design of more effective course-related discussions that overcome familiarity fault-lines, and ultimately advance peer-learning. We hope to further contribute to the research stream of knowledge sharing and integration behavior in groups and the role of familiarity.

5. REFERENCES

Borgatti, S. P., Everett M. G., & Freeman L.C. (2002). Ucinet for Windows: Software for Social Network Analysis. Analytic Technologies, Harvard/MA.

De Vreede, G. J., Briggs R. O., van Duin R., & Enserink B. (2000). Athletics in Electronic Brainstorming: Asynchronous Electronic Brainstorming in Very Large Groups, in: J. F. Nunamaker, & R. H. Sprague (Eds.). *Proceedings of the 33rd Hawaiian International Conference on Systems Sciences*, IEEE Computer Society Press.

Gasson, S., & Waters J. (2011). Using A Grounded Theory Approach to Study Online Collaboration Behaviors. *European Journal of Information Systems*, 1-24 doi:10.1057/ejis.2011.24.

Gruenfeld, D. H., & Hollingshead A.B. (1993). Socio-Cognition in Work Groups – The Evolution of Group Integrative Complexity and Its Relations to Task-Performance. *Small Group Research, 24*(3), 383-405.

Gruenfeld, D. H., Mannix E. A., Williams K. Y., & Neale M. A. (1996). Group Composition and Decision Making: How Member Familiarity and Information Distribution Affect Process and Performance. *Organizational Behavior and Human Decision Processes, 67*(1), 1-15.

Goodman, P. S., & Leyden D. P. (1991). Familiarity and group productivity. *Journal of Applied Psychology, 76*, 578–586.

Homan, A. C., Van Knippenberg D., Van Kleef G. A., & De Dreu C. K. W. (2007). Bridging Faultlines by Valuing Diversity: Diversity Beliefs, Information Elaboration, and Performance in Diverse Work Groups. *Journal of Applied Psychology, 92*(5), 1189-1199.

Jaccard, P. (1912). The Distribution of the Flora in the Alpine Zone. *New Phytologist*, (11), 37–50, doi:10.1111/j.1469-8137.1912.tb05611.x.

Javadi, E., Gebauer, J., & Mahoney, J. (2013). The Impact of User Interface Design on Idea Integration in Electronic Brain-storming: An Attention-Based View. *Journal of the Association for Information Systems, 14*(1), 1-22.

Waters, J., & Gasson, S. (2006). Social Engagement in an Online Community of Inquiry. 27th International Conference on Information Systems (ICIS), Milwaukee WI.

Waters, J., & Gasson, S. (2012). Using Asynchronous Discussion Boards to Teach IS: Reflections from Practice. Proceedings of the 32nd International Conference on Information Systems (ICIS), Orlando, Florida, December 2012.

Walther, J. B. (1992). Interpersonal Effects in Computer-Mediated Interaction: A Relational Perspective. *Communication Research*, 19 (1), 52–90

Appendix

Table 1: Correlation among In-Degree Measures for Familiarity and Discussion Networks

Correlations	S1 N=26		S2 N=19		S3 N=29		S4 N=29		S5 N=21		S6 N=17	
	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2
D1	.390	.288	-.134	-.014	.239	.406	.205	.195	.225	.309	-.275	-.310
D2	.354	.132	-.103	.045	.364	.460	.022	-.195	.096	.336	.151	.255
D3	.381	.336	.324	.221	.208	.489	.093	.207	.076	.481	.306	.091
D4	.326	.494	-.024	-.057	.036	.364	.348	.372	.444	.044	.198	.265
Correlation between F1 & F2	.757		.374		.718		.309		.482		.624	

Table 2: Correlation among Reciprocity in Discussion Network and Degree Centrality in Familiarity Network

Correlations	S1 N=26		S2 N=19		S3 N=29		S4 N=29		S5 N=21		S6 N=17	
	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2
D1	.396	.133	-.220	.434	.336	.060	-.360	-.322	.396	.133	-.079	.453
D2	.502	.243	-.119	-.167	.073	.481	-.475	-.202	.502	.243	-.128	.481
D3	.343	-.002	.312	.287	.252	.398	.149	.143	.343	-.002	.005	.083
D4	.550	.329	.070	.268	-.137	.376	.124	-.018	.550	.329	-.190	.238

Table 3: Jaccard Coefficient to Measure Similarity among Discussions (D1-D4) and Familiarity Networks (F1, F2) for Course Sections S1 to S6

Jaccard Coefficient	S1 N=26		S2 N=19		S3 N=29		S4 N=29		S5 N=21		S6 N=17	
	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2
D1	0.14*	0.42	0.1	0.38	0.1*	0.12*	0.05 ***0.83***	0.12*	0.19***	0.21***	0.14	0.58
D2	0.13*	0.39	0.14	0.47	0.16*	0.35*	0.07	0.37*	0.27**	0.64**	0.14	0.58
D3	0.14*	0.5**	0.1	0.37	0.16*	0.37**	0.05	0.31	0.21*	0.62**	0.29***	0.64 (p=0.07)
D4	0.17***	0.48**	0.11	0.44	0.16*	0.43***	0.1	0.4**	0.26**	0.6*	0.18*	0.59
(F1 & F2)	0.21***		0.24***		0.39***		0.19***		0.3***		0.22***	
*:<0.05 **: <0.01 ***: <0.001												