Techniques for data-driven curriculum analysis

Conference Paper · March 2014

DOI: 10.1145/2567574.2567591

CITATIONS
15

READS
239

3 authors:

Gonzalo Méndez
University of St Andrews
13 PUBLICATIONS **99** CITATIONS
SEE PROFILE

Xavier Ochoa
Escuela Superior Politécnica del Litoral (ESPOL)
90 PUBLICATIONS **1,171** CITATIONS
SEE PROFILE

Katherine Chiluiza
Escuela Superior Politécnica del Litoral (ESPOL)
40 PUBLICATIONS **128** CITATIONS
SEE PROFILE

Techniques for Data-Driven Curriculum Analysis

Gonzalo Méndez
Escuela Superior Politécnica del Litoral
Vía Perimetral, Km. 30.5
Guayaquil, Ecuador
gmendez@cti.espol.edu.ec

Xavier Ochoa
Escuela Superior Politécnica del Litoral
Vía Perimetral, Km. 30.5
Guayaquil, Ecuador
xavier@cti.espol.edu.ec

Katherine Chiluiza
Escuela Superior Politécnica del Litoral
Vía Perimetral, Km. 30.5
Guayaquil, Ecuador
kchilui@espol.edu.ec

ABSTRACT

One of the key promises of Learning Analytics research is to create tools that could help educational institutions to gain a better insight of the inner workings of their programs, in order to tune or correct them. This work presents a set of simple techniques that applied to readily available historical academic data could provide such insights. The techniques described are real course difficulty estimation, dependance estimation, curriculum coherence, dropout paths and load/performance graph. The description of these techniques is accompanied by its application to real academic data from a Computer Science program. The results of the analysis are used to obtain recommendations for curriculum re-design.

Categories and Subject Descriptors

K.3.1 [**Computing Milieux**]: Computers and Education-Computer Uses in Education

Keywords

Learning Analytics, Curriculum Design

1. INTRODUCTION

Analytics could be applied at very different levels in an educational institution. Siemens and Long [18] seminally classified the scope of analysis into five levels: Course, Departmental, Institutional, Regional and National/International. The first two levels are the main focus of Learning Analytics, while the remaining three are usually called Academic Analytics [4]. Due to this distinction, most of previous research in Learning Analytics [9] deal only with the first level: the analysis of the behavior and interaction of students and faculty inside a course. Apart from studies of dropout [24], there is very little research on how to analyze the learning process at Departmental or Program level in order to guide the design or re-design of a program curriculum.

LAK '14, March 24 - 28 2014, Indianapolis, IN, USA

ACM 978-1-4503-2664-3/14/03 $15.00.
http://dx.doi.org/10.1145/2567574.2567591.

To bootstrap the discussion of this also important level, this work proposes a simple set of Learning Analytics techniques that could help program coordinators or faculty groups to gain a better insight on the nature and current performance of their programs and provide insights on possible problems or difficulties that the students may have with it. Such techniques could be useful for what we call Data-Driven Curriculum Analysis. These techniques are designed to work exclusively with readily available academic performance date (final grades) to facilitate its immediate adoption in a widely variety of institutions. To validate the usefulness of the techniques, they are showcased in a case study involving at least 12 years of academic data of a particular Computer Science program that is being re-designed. The techniques, however, could be applied to any existing curriculum.

The structure of the paper is as follows: Section 2 briefly reviews the related work on curriculum design techniques. Section 3 provides details on the context and data used as case study. Section 4 describes a technique to establish the actual difficulty of course based on the value and distribution of grades. Section 5 provides a correlation analysis to determine the dependence between courses. Section 6 proposes the use of Exploratory Factor Analysis to find the underlying structure and coherence of the curriculum. Section 7 identifies problematic courses that could lead to student dropout. Section 8 analyzes the performance of the students under different academic loads, in order to recommend the number of courses per semester. Finally, Section 9 draws conclusions from the analysis performed and the recommendations for the re-design of the program in the case study.

2. RELATED WORK

Curriculum analysis and design is a process that follows generally the following systematic approach: precise identification of the curricular design specifications and constraints (student outcomes, competences, learning goals); identification of a curricular conceptual model (research based learning, inquiry learning, problem based learning, case based learning, etc.); developing and testing of the curricular design; and refining the design with the feedback of students and stakeholders [15]. Due to the nature of the design process, some of its components can be revisited after several iterations. A key component in this process is the establishment of the conceptual model, which is related to: instructional contexts that are needed to be emphasized, teaching/learning methods, resources, etc. Many studies are mainly devoted to study the effects of some components of

a curricular conceptual model or the innovation of teaching strategies in the curriculum [15, 7, 23], but few of them examine their curriculum in order to base their redesign decisions on data of the corresponding academic program.

One example of an attempt to a data-driven curriculum redesign is what Sundermand and Price conduct in the curriculum incubator [20]. In their work, they experimented with teaching strategies, learned from them and introduced curricular modifications but mainly at the course level. Albert [1] states, from the medical education point of view, that most of the decisions related to curriculum and teaching are based on "opinions, intuitions, and personal preferences"; thus, she presents a curriculum for new rheumatology trainees that is based on trends that appeared after analyzing clinical problems that had the greatest educational relevance in the rheumatology field. In this case, the decision-making process about the curriculum was mostly based on teaching experiences of relevant educational pieces. Nonetheless, and despite these efforts, no studies present the analytical approach that is presented in this research. It was not found any literature that offers a varied set of techniques to identify difficult courses and their impact on GPA or desertion in a program, as the techniques presented in this article.

3. CASE STUDY

The data analyzed in this article consists on the academic performance information of Computer Science (CS) undergraduate students from the Electrical and Computing Engineering College of ESPOL University in Guayaquil, Ecuador. The dataset spans the academic history from 1978 up to 2012 and is composed by the grades of 2543 individuals that eventually enrolled in the CS program of ESPOL. The name of the ESPOL' CS program has changed over time: 1) Electrical Engineering with a minor in Computing, 2) Computer Engineering, and 3) Computer Science. The two later degrees compose the most recent history of the investigated time interval and have been offered in three different concentrations or majors: a) Multimedia Systems, b) Technological Systems, and c) Information Systems.

According to the performance scoring system of ESPOL, a class grade can take a real value from the interval [0.00, 10.00] and a student passes a given class by obtaining a grade equal or greater than 6.00, otherwise he/she fails. The general CS curriculum categorizes its courses according to the topics and contents they cover in: 1) Basic Sciences, 2) Humanities, and 3) Professional Training courses . Moreover, each CS profile has its own concentration courses and contains several both selective and elective credits.

At ESPOL, students are allowed to adjust the number of courses they want to take within the selective elective and free-selective categories, according to the amount of credits they accumulate throughout their degree. Therefore, some students may complete the specified number of credits for these categories with more (or fewer) courses than others. Because of this, the entire academic history generated by the students considered in this analysis defined a huge set of unique courses. The heterogeneity of this set is not only explained by the diversification introduced by each program's majors and their selective elective and free-selective courses, but also by the curricular modifications that were performed on the CS programs during the time interval considered.

In order to prepare such varied data for a proper analysis, a few processing steps were performed on the original dataset to *clean* it properly. First, we dealt with any missing information introduced by curricular modifications. Some of these changes include 1) Courses that were removed from some versions of the curricula, 2) Courses whose type was changed from compulsory to elective, 3) Courses that were split into more than one courses, 4) Courses whose names were modified.To cope with these events, a course-course matching process was conducted to determine the agreements among two or more distinct courses that can be considered equivalent across different versions of the curriculum. At the end of this course-matching process, the set of unique courses was slightly reduced but it still contained a considerable amount of missing grading information.

To reduce the amount of missing data, we discarded in our analysis courses not highly frequently chosen by students and English courses that escape from the conceptual design of CS. With this step, we selected a common core of 27 classes for which grading information was readily available. The set of courses included in the last version of the dataset is illustrated in Figure 1. This set is composed by 8 basic sciences courses, 16 on professional instruction and 3 from the humanities category.

For some analysis, due to discontinuities in the data due to program changes, only the last 12 years, where the curriculum has stayed fairly static, were used to avoid the introduction of noise.

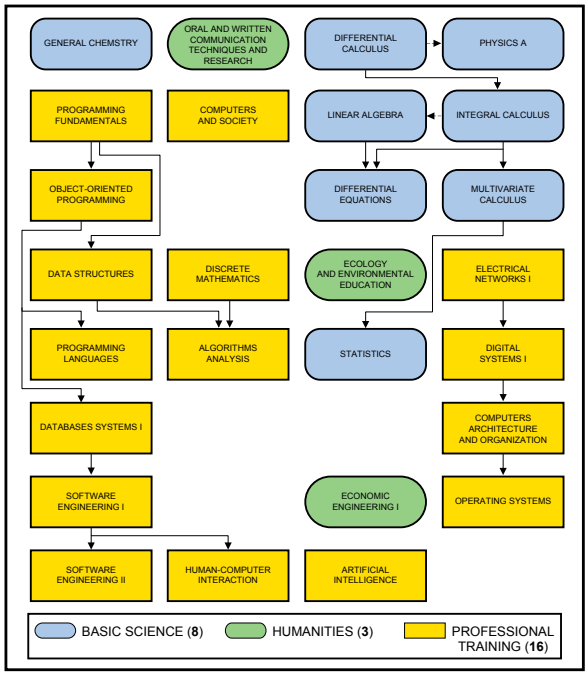


Figure 1: Courses included on the core curriculum of the ESPOL Computer Science program

4. DIFFICULTY ESTIMATION

When looking at the difficulty levels of the courses they are composed of, curricula constitute very diverse sets. In particular for CS, given the varied nature of the knowledge areas involved in the professional training process of computer scientists, courses may exhibit distinct difficulty levels.

The objective characterization of courses in terms of their difficulty is not a trivial task. Factors such as instructors' particular characteristics, students' affinity with the course' concepts, or grading stringency standards applied are the causes why the computation of a single value to represent the difficulty level of a course is not an easy problem. In this regard, the number of credits contributed by a course within a curriculum is normally considered as a difficulty indicator because it is intended to suggest the work and time loads that students are supposed to experience in a certain period of time (a week, for example) to successfully pass a given class. However, the credit system overlooks other factors that are not necessarily related to time and effort. Special considerations such as extracurricular activities, other courses taken at the same time, and aspects not related to the academic life, can make a course easier or harder independently of its number of credits.

The historical average grade of a course j, denoted here as HAG_j and defined according to Equation 1, where r_{ij} is the grade obtained by the student i in course j and N_s^j is the total number of students that have taken the class j, might be the most basic data-inspired indicator that comes to mind when searching for objective estimators of difficulty. However, HAG not only can be also biased by the factors mentioned above, but it also ignores the overall academic performance of students that generated the grades involved in its computation. Thus, this value is not suitable to suggest, for example, whether a course rated as easy was easy because it was always taken by high-performing students or because its contents were not challenging enough in such a way that even low-performing students got high grades.

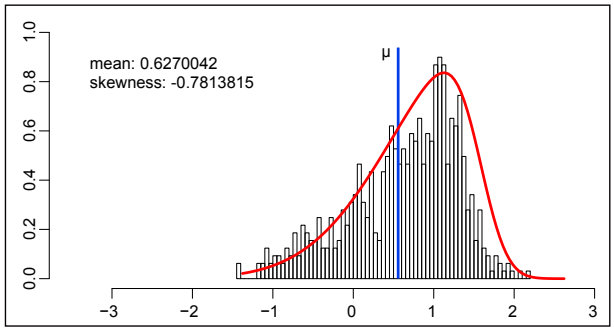
$$HAG_j = \frac{\sum_i r_{ij}}{N_s^j} \tag{1}$$

In this section, we propose a technique to compute empirical, data-based, courses' difficulty estimators for courses of an actual CS curriculum. We also apply two of the approaches suggested in [5] to estimate courses difficulty levels and their corresponding grading stringency indices. Then, we compare these indicators with the results of a perception study that measured the opinion of students about the difficulty of courses, impact of courses' grades on GPA, and importance of some courses to continue in the program.

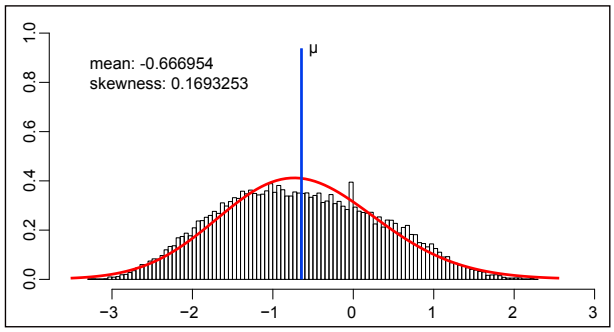
4.1 Description

The *influence* of a course on a student's academic history is related to which extent it contributes positively or negatively to the student's overall performance. If the grade point average (GPA) is considered as a general performance indicator, a course contribution can be seen as positive or negative according to whether the grade obtained on it is greater or lower, respectively, than the student's GPA. Thus, the distance between the GPA and a particular course grade can be seen as a measure of how much that class shifted, up or down, the student's GPA. That is, how much the given course move students away from their GPA or bring them closer to it.

In order to characterize the difficulty of a course we propose to construct a statistical distribution of the distances between the GPAs of all the students that took the given course and the grades they got on it. Examples of these distributions are shown in Figure 2 for two courses of the ESPOL CS curriculum: Algorithms Analysis (Figure 2a) and the Oral and Written Communication Techniques and Research course (Figure 2b).



(a) Algorithms Analysis



(b) Oral and Written Communication Techniques

Figure 2: GPA - course grade distances distribution for two courses from the ESPOL CS curriculum

As it can be observed the shapes of these two distributions (shown in red) differ significantly in the longitude and leaning of their longer tail. In the case of Algorithm Analysis, the mass of the distribution is concentrated to the right of the figure, which means a negative skewness. On the contrary, values for the Oral and Written Communication Techniques and Research course are consolidated to the left side of the distribution mean, which leads to a positive skewness distribution. The skewness values for the distribution computed from the differences between the students' GPA and their course grades, are always related to whether the course grades have been greater or lower than the GPAs of the students involved. In that sense, the distribution skewness can be considered as a difficulty indicator of a course. Thus, The more negative the skewness, the more difficult the course is.

Caulkins et. al. [5] propose that a course j has a grading stringency index β_j and multiplicative magnitude α_j that represents its difficulty level. These two values are computed according to Equations 2 and 3, respectively. In these equations, GPA_i represents the overall academic performance of student i and r_{ij} and N_s^j are defined as explained above.

$$\alpha_j = \frac{\sum_i {GPA_i}^2}{\sum_i (r_{ij} * GPA_i)} \tag{2}$$

$$\beta_j = \frac{\sum_i (GPA_i - r_{ij})}{N_s^j} \tag{3}$$

Note that the equation for the β values include the distance between the GPA and the course grade of students ($GPA_i - r_{ij}$) on which we based the skewness distribution computation. Thus, the grading stringency standard is also an indicator of how much a course moves the student's GPA. The use of the skewness however, also takes into account the shape of the distribution of grades.

4.2 Application

The three approaches detailed in section 4.1 were applied over the courses that compose the curriculum show in Figure 1. Table 1 details these three magnitudes for the 27 courses, which have been ordered by their distribution skewness value.

Parallel to these computations, a perception study was conducted with 80 students of the CS program. The perceptions about the following courses' characteristics were gathered using a survey: difficulty level, negative impact on GPA, and importance in the program. The survey included questions that asked the student to identify a maximum of six courses that were considered: difficult, with a high failure rate, and important to continue the CS program. Descriptive results of this survey are shown in Figure 3. As can be seen, courses like Operating Systems, Programming Fundamentals, Object-Oriented Programming, Data Structures, among others, are perceived either difficult and/or negatively related on the GPA of a student, coinciding with what appears in Table 1. In addition, the importance of a course to continue in the CS program was identified by students. The following are the courses that appeared between 10% to 50% of the times as important courses to continue in the CS program: Physics A (10%), Electrical Networks Systems I (10%), Research Methods Applied to Computing (11%), Physics C (14%), Differential Calculus (15%), Statistics (18%), Programming Languages (23%), Differential Equations (24%), Operating Systems (30%), Linear Algebra (46%), Programming Fundamentals (50%). These courses' importance indicators will be used to contrast the findings in section 7.

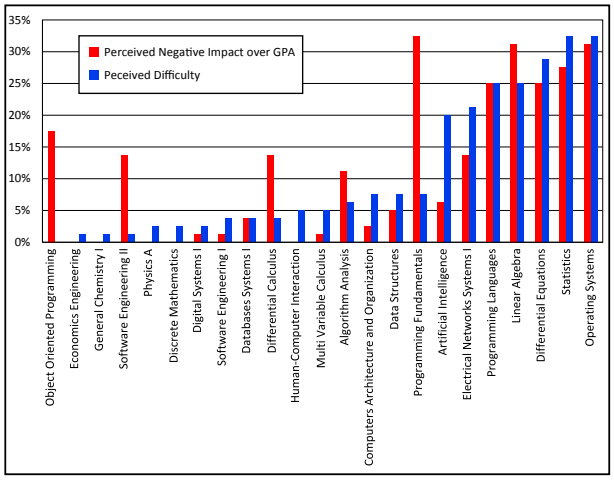


Figure 3: Students' perceptions about difficulty and negative impact of courses on their GPA

The results obtained through the three approaches presented above were contrasted with students' perceptions about the level of difficulty and negative impact of courses in their GPAs. It was found that the majority of the courses that had a negative skewness, higher *Alpha* and *Beta* values are also perceived as courses with high level of difficulty and a negative impact on the overall GPA of students, once they are approved. These findings might be useful for faculty members in planning, for example, the split of some courses into others or the inclusion of teaching/learning strategies that help students to reach better scores from (difficult) courses. Curriculum design is a process that goes from setting student outcomes, that must be achieved at the end of a program, to guaranteeing that these outcomes are actually developed and met [10]. As such, when designing curriculum, data-evidences are key inputs of the implicit decision making process that is conducted. Finally, it is important to note that these measurements of difficulty are averaged among different instructors with different methodologies and grading stringency. Obtaining these values for pairs Course-Instructor should provide a more reliable estimate of the real difficulty.

5. DEPENDANCE ESTIMATION

Curricula for undergraduate programs are normally structured in such a way that their courses are not totally independent from each other. That is, concepts reviewed in advanced courses commonly depend on the contents covered previously, in other basic courses. This pre-requisite based system causes that students follow a logical sequences that guarantee that the contents associated to specific concepts or Knowledge Areas are covered incrementally. Based on this fact, this section presents a simple correlational analysis as a measurement of the inter-course dependance within a curriculum.

5.1 Description

As a simple measurement of the dependance degree, a Pearson's correlation coefficient could be applied to the final grades to estimate the linear relationship between the students' academic performance on each pair of courses in a curriculum. We propose this technique under the assumption that the contents of certain courses are of higher importance for other subsequent subjects than others. Consider, for instance, that the contents covered on the Differential Calculus course are the base over which the Integral Calculus topics are developed. Under this hypothesis, we are supposed to find a strong correlation on the performance of a student taking the Integral Calculus course and the performance he/she obtained on the corresponding pre-requisite.

5.2 Application

The correlation analysis was applied to the core courses of the ESPOL CS curriculum. The results of these computations are shown in the color-map of Figure 4. The correlation degree between two courses are represented by the color of the corresponding cell: high correlation values are shown in dark colors and lower values are shown in white tones.

As it can be observed, the correlation values found in this part of the alaysis are not specially high among most of the courses. This fact suggests that the inter-course dependances within the curriculum analyzed are not strong. The three highest Pearson's correlation coefficients of the depicted map occurs for the Computers and Society course when is observed with: 1) Human-Computer Interaction (0.6226), 2) Discrete Mathematics (0.614), and 3) the Op-

Table 1: Skewness values for some GPA-course grades distances distributions

#	Course	*skewness*	α	β
1	Algorithms Analysis	-0.7814	1.1435	1.0791
2	Operating Systems	-0.7751	1.0891	0.7334
3	Programming Fundamentals	-0.7620	1.3458	2.0529
4	Object-Oriented Programming	-0.7282	1.2001	1.3907
5	Data Structures	-0.7118	1.1452	1.1113
6	Digital Systems I	-0.7042	1.1912	1.3570
7	Electrical Networks I	-0.6992	1.2300	1.4962
8	Programming Languages	-0.6841	1.2032	1.4374
9	Discrete Mathematics	-0.6799	1.1503	1.1343
10	Economic Engineering I	-0.6705	1.1426	1.0519
11	Multivariate Calculus	-0.6107	1.2126	1.4332
12	Integral Calculus	-0.6005	1.2026	1.3283
13	Differential Calculus	-0.5978	1.2842	1.7069
14	Statistics	-0.5912	1.2519	1.7823
15	Artificial Intelligence	-0.5793	1.1214	0.9268
16	Linear Algebra	-0.5494	1.3042	1.8891
17	Software Engineering I	-0.5385	0.9891	-0.0499
18	Physics A	-0.5310	1.2302	2.4057
19	Differential Equations	-0.5244	1.3066	1.8509
20	General Chemistry	-0.4212	1.1108	1.2535
21	Computers and Society	-0.3889	1.0895	0.6651
22	Human-Computer Interaction	-0.3859	1.0396	0.3970
23	Databases Systems I	-0.3680	1.1201	0.8373
24	Software Engineering II	-0.2021	0.9742	-0.1283
25	Ecology and Environmental Education	-0.1473	1.0061	0.2150
26	Computers Architecture and Organization	0.0009	0.9750	-0.1167
27	Oral and Written Comm. Techniques	0.1693	0.9432	-0.3152

erating Systems (0.582) courses. It is interesting to see that the Computers and Society course is not an immediate prerequisite of its most correlated courses. However, the contents reviewed on this class constitute the most basic CS topics that students are exposed to at the very beginning of their academic path. Also important to note is that the performance of the students in Programming Fundamentals, one of the most difficult initial courses, has very little correlation with their performance in any other course. It seems that the level of stringency in this course is much higher than needed, as obtaining a high or low grade does not heavily influence the further achievements in the program.

De Koning et al. [6] carried out a study for a three-year bachelor program and concluded that predictors of first-year courses were not the same predictors for subsequent years, even though they also affirm that prior grades and learning activities were strong and stable predictors. Despite cognitive skills and previous knowledge of students are important predictors of achievement, recently, affective variables like: motivation, self-drive, dedication time, learning preferences, among the most important ones, have been identified as important as those linked to scores [19, 3]. We have found no significant correlations between the scores reached in prerequisite courses and their subsequent courses; thus, in our case other factors have to be explored and studied to predict students' achievement.

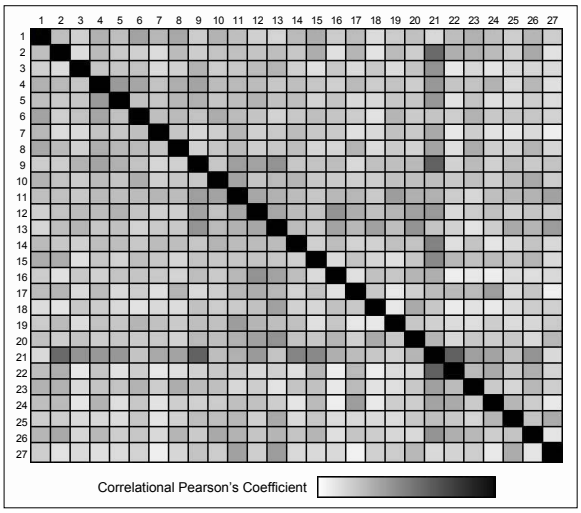


Figure 4: Colormap of courses performances correlations. Courses appear codified according to Table 1

6. CURRICULUM COHERENCE

In this paper, we refer to a curriculum as coherent when it constitutes a set of courses that allows students to achieve, by the completion of their degrees, several logically interrelated outcomes or competences. Several curricular guidelines for undergraduate programs in computing, such as the proposed by the ACM and the IEEE Computer Society, normally define a Body of Knowledge (BoK) to cover the CS professional domain according to the desired competences of their future graduated students. In such guidelines, a BoK

is organized in several Knowledge Areas (KAs) which are further composed by various Knowledge Units (KUs) that ultimately constitute the building blocks of the courses included in a curriculum.

In this work, we hypothesize that when a curriculum is coherent, it should mask, somehow, an underlying structure of courses grouping together in sets of courses that cover concepts associated to several interrelated professional competences. Moreover, we believe that such grouping should reveal the level at which a set of courses are coherently aligned with the learning outcomes aimed by a given curricular design.

In this section, we propose the use of Exploratory Factor Analysis (EFA) as a technique to reveal the grouping course structures hidden underneath a curricular design.

6.1 Description

The goal of EFA is to uncover the underlying structure of a relatively large set of variables by identifying interrelationships among them. This technique is of special usefulness to discover which variables in the set form coherent subsets that are relatively independent of one another [21]. It outputs several unobserved latent variables, known as factors, that summarize patterns of correlation among the observed variables. These factors are thought to reflect the underlying processes that have created the correlations among the observed input variables.

The hypothesis behind the use of EFA in the context of the academic performance achieved by students who enroll in the courses that compose a specific curriculum can be stated as follows: Several individual variables (in this case, the grades obtained by the students in each of the analyzed courses) combine with the grades of some other courses to form factors that represent the big areas of the students' professional training process.

6.2 Application

In this part, we based our analysis on the academic performance of the last six generations of CS undergraduate students who successfully completed their degree at ESPOL between 2000 and 2012. A maximum-likelihood factor analysis was performed on the correlation matrix of the dataset composed by the performance achieved by 333 students on each of the 27 courses shown in Figure 1. The analysis was executed by using the functionalities provided by the *psych* statistical package[1] available for the R statistical software [16]. A *varimax* rotation was applied on the first version of the loadings matrix resulting from the initial extraction of factors.

From the 27 factors, The first six had eigenvalues greater than 1. We investigated the cumulative variance that can be captured from the dataset by executing the EFA with 5, 6 and 7 factors. The values found for this parameter in the three tests were 33, 36 and 37 percent, respectively. Moreover, in each analysis, a hypothesis test was conducted to measure whether each amount of factors was sufficient. The p value for the three tests were 0.0827, 0.284 and 0.541, respectively. Since the idea of Factor Analysis is to summarize the patterns of correlations with as few factors as possible, five factors were selected.

The groups of courses included in the five factors found in our analysis are shown in Figure 5. These set of courses result after the application of a cutoff value of 0.3 on the loadings matrix of the orthogonally-rotated factors. Any course with a loading value greater than the predefined threshold has been retained in the analysis.

[1]http://cran.r-project.org/web/packages/psych/index.html

Table 2: Factors summed squared loadings and variances

	Factors				
	1	**2**	**3**	**4**	**5**
SS loadings	2.82	1.99	1.55	1.33	0.98
Proportion Var	0.11	0.08	0.06	0.05	0.04
Cumulative Var	0.11	0.18	0.24	0.3	0.33

Table 2 details the sum of squared loadings (SSL) associated to each factor. As mentioned before, the five factors capture a cumulative variance of 0.33. That is, together, they account for 33% of the variance present in the 27 performance scores. The proportion of variance that is individually captured by each factor is also shown along with the corresponding cumulative value.

6.3 Interpretation of Factors

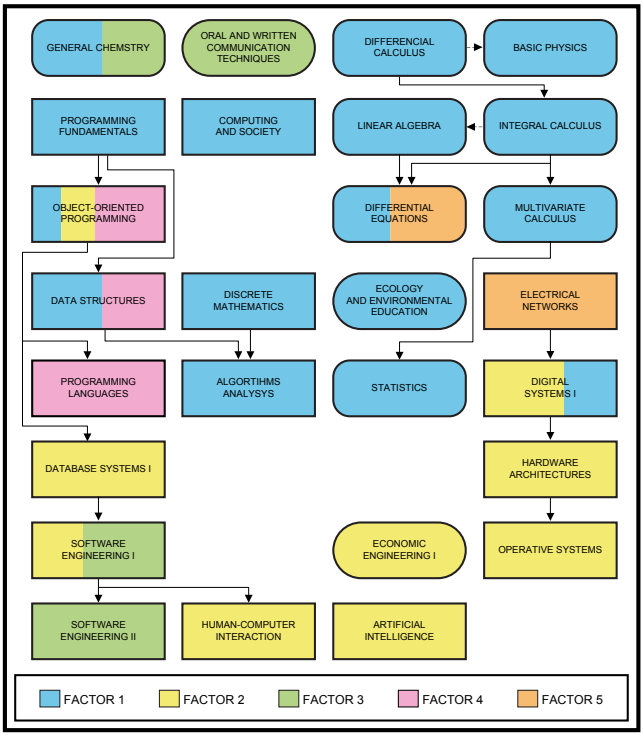
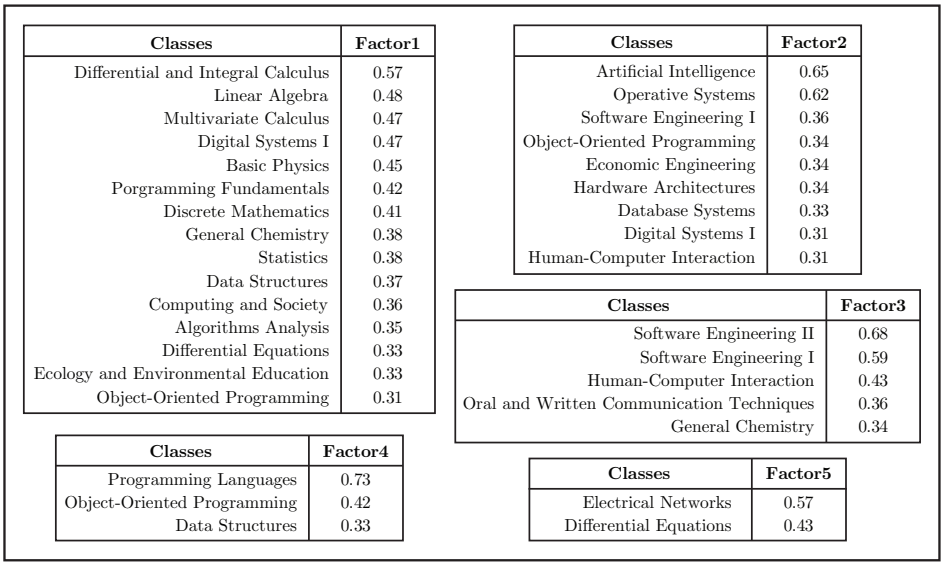


Figure 6: Courses categorized by the factors to which they belong

The groups of courses defined by the generated factors reflect to a really large extent the logic clusters defined in the curricular design of the analyzed CS program. Following, we provide the interpretations of each factor found according to the known characteristics of the analyzed curricular design.

Factor 1 (The basic training factor): Elements in this group can be classified in two categories: Basic science courses, and fundamental CS courses. The first category cover topics aimed the understanding of abstract ideas and



Classes	Factor1
Differential and Integral Calculus	0.57
Linear Algebra	0.48
Multivariate Calculus	0.47
Digital Systems I	0.47
Basic Physics	0.45
Porgramming Fundamentals	0.42
Discrete Mathematics	0.41
General Chemistry	0.38
Statistics	0.38
Data Structures	0.37
Computing and Society	0.36
Algorithms Analysis	0.35
Differential Equations	0.33
Ecology and Environmental Education	0.33
Object-Oriented Programming	0.31

Classes	Factor2
Artificial Intelligence	0.65
Operative Systems	0.62
Software Engineering I	0.36
Object-Oriented Programming	0.34
Economic Engineering	0.34
Hardware Architectures	0.34
Database Systems	0.33
Digital Systems I	0.31
Human-Computer Interaction	0.31

Classes	Factor3
Software Engineering II	0.68
Software Engineering I	0.59
Human-Computer Interaction	0.43
Oral and Written Communication Techniques	0.36
General Chemistry	0.34

Classes	Factor4
Programming Languages	0.73
Object-Oriented Programming	0.42
Data Structures	0.33

Classes	Factor5
Electrical Networks	0.57
Differential Equations	0.43

Figure 5: Factors and variable-factors correlations grouped after applying a cutoff of 0.3

concepts. These abstractions lie at several levels: mathematical (Calculus and Linear Algebra), logical (Digital Systems), natural phenomena (Chemistry, Physics, Ecology and Environmental Education) and data interpretation (Statistics).

On the other hand, the second category groups fundamental CS courses. Computers and Society, Programming Fundamentals, Data Structures and Algorithm Analysis are the first set of courses of the CS program offered to students. In particular, the Discrete Mathematics course is a good example of a subject in which mathematical abstractions are thought with a focus strongly oriented to computing applications.

Four of the courses included in Factor 1 are also correlated at a considerable level with some other factors. The analysis of the following factors will provide the corresponding interpretations of these shared correlations.

Factor 2 (The advanced CS topics factor): With the exception of the Economic Engineering I course, all the subjects in this group cover advanced, high-level CS topics. This group includes courses related to the design of Database Systems, the study of Computers' Architecture and Organization, Object-Oriented Programming, Software Engineering, Human Computer Interaction, Digital Systems, Artificial Intelligence, and Operating Systems. In general, these courses refer to the identification of a system's components and their interrelationships. The only course that seems to be misplaced in this subset is the Economic Engineering class.

Factor 3 (The client interaction factor): This factor seems to cover aspects related to communication skills needed to deal with final users and stakeholders involved in the software developing process. In that sense, it includes the Oral and Written Communication Techniques and Research course and other courses that are also correlated with Factor 2. More specifically, these courses are mainly related to the software building process and interaction with clients and final users. In this point, it is important to mention that most of these courses (with the exception of General Chemistry) are organized in ESPOL in such a way that students do have actual clients for which they have to develop functional software projects during the class. Thus, it is comprehensible that, besides being part of the advanced CS topics factor, they share correlation with Factor 3.

Factor 4 (The programming factor): The courses grouped under this factor constitute those exclusively dedicated to develop programming skills and abstraction conceptualizations. These courses are, in the curricular design of the ESPOL CS program, in what is called the programing sequence of the curriculum. The concepts covered by all of these courses are related to programming skills.

Factor 5: This group seems to include courses that cannot be grouped with any other of the CS curriculum. It is interesting to see that, these courses are more linked to electrical engineering courses, which were included in the curriculum as a requisite of the college (Electrical and Computer Engineering). These courses are commonly seen by students as subjects that do not fit in their degree because are not directly applied to solving problems related to CS.

The main conclusion of the curriculum coherence analysis conducted in this section is that it allows us to identify which courses of a curricular design form logical structures aimed to develop interrelated professional competences. To confirm the soundness of the found groups, additional EFAs could be performed on other type of data related to the courses that compose a curriculum. For instance, instead of considering the students' academic performance information, we might consider the distances between each class grade obtained by the students and their GPA.

It has to be said that a very important component of this analysis is the interpretation of the factors output by the EFA. Coherence requires subjective judgment and that can be done only when the context on which the curriculum takes place is very well known. This implies that all the identified structures have to be further analyzed by human evaluators with a deep understanding of the nature of the program and its particularities in order to see the logic behind the discovered grouping structures. In any case, having

an objective statistical method over which a judgment can be done is a positive starting point to take decisions on the curricular design within academic institutions.

Regarding the case study, the results suggest that some courses do not align with the main CS competences factors (Differential Equations and Electric Networks). This result, together with the knowledge that these courses were initially considered only because they were part of the Electrical Engineering curriculum, should spark a discussion about their real importance in the CS curriculum. Also, it was surprising to find that Programming Fundamentals is not considered in the programming factor (factor 4). This result, linked with the previous result of the lack of correlation between grades of Programming Fundamentals with the rest of the programming courses, warranty a deeper qualitative analysis of the content and methodologies of this course.

7. DROPOUT AND ENROLLING PATHS

In this study, academic dropout is defined as a student who enrolled in an academic program that eventually, at some point before completing the corresponding degree, permanently abandoned it. This type of academic desertion has been widely investigated for several decades in works mainly oriented to explain its causes. Factors such as race and gender [11], socio-economic and family backgrounds [17], student's engagement with school activities [2], extracurricular load [12], and the like, have been repeatedly explored not only to explain dropout but also to construct predictive models for the early detection of potential desertion scenarios.

However, no works have been conducted to explore the existence of relationships, if any, between dropouts and curricular designs. This section proposes the use of frequent sequence analysis as a technique to answer the following question: Are there any frequent enrolling paths followed by students within a curriculum that lead to dropout settings? If so, to which extent a given enrolling path is frequently observed as a previous necessary condition for desertions? Our analysis refers to an enrolling path as a specific sequence of courses taken (or failed) by a student along his/her academic history. Following section explains the theoretical motivations for the application of sequence mining and details how it can be used to track students' enrolling patterns that lead them to quit their academic programs.

7.1 Description

The goal of sequence mining is to discover sequential patterns of frequently repeated events in a given database of temporal transactions [13]. Among many others, scenarios where this type of mining task has been used include: sales databases of items bought by customers [8], analysis of users' activity in web searching and browsing [14], and disease identification based on sequences of symptoms [22].

In sequence mining, frequent patterns are considered to emerge over time as the events that compose the sequences occur repeatedly. A sequence α is considered frequent if it is contained by a minimum number of input-sequences that form the database D. This minimum value is called the *support* or $frequency$ of α. To illustrate the concept of *support*, consider the scenario where a bookstore keeps historical record on the books bought by each of its customers. This information could be mined to find behavioural patterns in the users' buying habits to predict, for example, how likely is that a given customer that has bought book A eventually come back to buy book B, given that 80% of previous buyers of book A actually did so.

Considering that students' academic history takes places over a period of time with very well known courses' timestamps, sequence mining techniques can be applied to find frequent patterns in the way students enroll in the courses that compose their programs' curricula. More specifically, frequent sequence analysis can be conducted over the failure segments of the students' academic history to investigate whether particular failing course patterns conduct students to more dropout scenarios than others.

7.2 Application

The failing academic events of 610 dropouts were investigated to find the most frequent courses and sequences of courses that were failed by students before abandoning their undergraduate studies. The analyzed subset was part of a greater cohort of 1591 students that enrolled at the CS program of ESPOL along the last twelve years (2000-2012). Thus, the original CS population evidences a dropout rate of 38.34%.

The SPADE algorithm [25], an approach for fast discovery of sequential patterns, was applied to the described subset of enrolling information using a minimum support of 0.3. The results of this algorithm are detailed in Table 3, which clearly shows that before deciding to quit, a large proportion of the investigated students failed courses on the basic training levels of their curriculum (that is, courses from Factor 1 of section 6.3). These result implies that most of these students decided to dropout immediately after dealing with fundamental concepts related to general Science and Engineering. Among all these courses, the most frequent event associated with dropout is the failure of the Physics A course, which was failed by about 61% of the students that eventually left the CS program.

Since the investigated dropouts did not take place at advanced levels or high phases of their academic history, students who abandoned the CS program did it without having the opportunity to interact with actual CS topics. The Programming Fundamentals course is the first CS course that is part of the more frequent steps previous to dropouts. It appears third in the list with a *support* of 0.53.

Results of the sequence mining performed are consistent with those obtained from the perception study described in section 4. Interestingly, the Physics A course is perceived as an important course that enables a student to continue the CS Program (10%) but not as important as the Programming Fundamentals course (50%). Therefore, failing such courses refrain students to continue in their CS education. Again, 6 out of 11 perceived important courses are from the basic sciences strand, these courses are mostly the same found in the set of courses of the failing enrolling path that lead to dropping out the CS Program. This apparent relationship can be interpreted as a good indicator of the importance given to them by students that eventually graduate.

This section proposed sequence mining for the identification of enrolling paths defined by students within a curriculum that frequently lead to dropouts scenarios. The identification of these critical trajectories in curricular designs is of special importance when decisions have to be taken to reduce or minimize the desertion rates. Finding critical paths

Table 3: Frequent failing enrolling paths that lead to dropout scenarios ($Support = 0.3$)

Sequence	*Support*
<{Physics A},{Dropout}>	0.608196721
<{Differential Calculus},{Dropout}>	0.570491803
<{Programming Fundamentals},{Dropout}>	0.532786885
<{Integral Calculus},{Dropout}>	0.496721311
<{Physics A, Differential Calculus},{Dropout}>	0.43442623
<{Linear Algebra},{Dropout}>	0.432786885
<{Differential Calculus,Integral Calculus},{Dropout}>	0.385245902
<{Physics C},{Dropout}>	0.347540984
<{Physics A,Integral Calculus},{Dropout}>	0.327868852
<{General Chemistry},{Dropout}>	0.319672131
<{Differential Equations},{Dropout}>	0.31147541

to avoid dropout and to reduce the negative impact of certain courses in desertion rates is a key input for curricular design. Administrators, faculty members and curricular designers can take data-driven decisions and plan accordingly. Moreover, this technique can be also applied to academic events of other types, such as successful scenarios to detect enrolling paths that are more likely to conduct students to graduation.

8. LOAD/PERFORMANCE GRAPH

The balance between academic load and academic performance is of vital importance for students' success. The achievement of an optimal trade-off between these two factors is highly related to the curricular design in the sense that it defines (or, at least, suggests) the way in which students should enroll in the courses of their degrees. How courses are assigned to different levels, the established prerequisites, and the list of possible optimal courses are some of the factors that influence the academic history of students of all levels. This implies that a good curricular design should seek the maximization of students' academic performance at the same time that it properly manages their working load. Given this context, mechanisms to characterize the load versus performance profile of a curriculum are highly desired. This section explains a technique to build such profiles by proposing some measurements for the load and performance magnitudes.

8.1 Description

Academic load can be associated to several factors: 1) number of courses taken in a particular academic period, 2) total difficulty pursued by the student, and 3) total grading stringency to which the student is subjected. On the other hand, performance can be seen in function of other several factors such as: 1) number of courses a student pass in a given academic period, 2) total courses' difficulty achieved, and 3) total grading stringency of the courses the student managed to pass.

Constructing visualizations of the trajectories followed by students in the load/performance graph could provide curriculum designers with information about how well different students manage the recommended class load and what is the actual preferred load of different kind of students.

8.2 Application

Figure 7 plots the density plot of the semestral load and performance information for all the students of the investigated dataset. Load is represented here by the difficulty of the courses taken by the student. On the other hand, performance is expressed in terms of the total difficulty of the passed courses. This graph represents the relation between the total difficulty achieved on a semester and the total difficulty pursued on the same period of time. We calculated the total difficulty pursued by a student as the sum of the individual courses taking at a given academic period. Similarly, the total difficulty achieved is defined as the sum of the difficulties of the passed courses. Being a density plot, this graph shows how many times students had an specific value of in the load/performance plane.

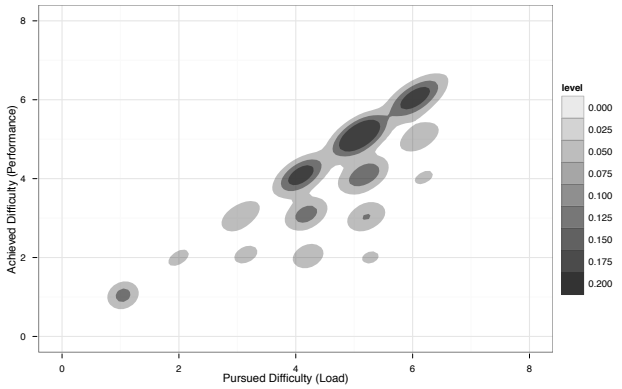


Figure 7: Semestral load versus performance density plot

As it can be easily seen in the Figure 7, most students have a load around 4, 5 or 6 (darker tones). There is a fairly large population that usually achieve the pursued difficulty, that is, they approve all the courses that they take. There is also a noticeable population of students that usually fail 1 or 2 courses (dark spots under the diagonal), meaning that these students were taking more load than they could effectively manage. In the curriculum the recommended load is 6 courses. However, the largest population of students are not taking that load, but only 5 courses or even 4 courses. Even with that load, a fair amount of them is failing at least 1 course. It is clear that a more realistic recommended load of 5 courses (or 4 if a difficult course is taken) is more advisable for the students of this program.

9. CONCLUSIONS

Historical academic performance data, while limited, provide an easy gateway to start conducting Learning Analytics studies on the characteristics and relation of diverse courses inside a program curriculum. The information that could be obtained, even with very simple techniques, could help faculty and institutional administrator to understand their programs beyond a merely theoretical viewpoint. The techniques presented in this work have the potential to start informed discussions inside faculty members that could lead to an improvement of the learning process of the student, not by changing particular aspects of a course, but deciding which courses are taught and in which sequence.

The application of these techniques to the case study has provided valuable information that will be used in an expected re-design of the CS program at ESPOL. The difficulty estimation analysis suggests that some courses, in particular Programming Fundamentals, should be re-designed or even divided to reduce their complexity. The methodology and evaluation of others courses, such as Oral and Written Expression and Research, should be reviewed to provide level of stringency closer to the mean. The dependence analysis highlighted the fact that current pre-requisites are not as important as previously thought and that the performance of student in perceived difficult courses is not correlated with their performance in other courses, suggesting that a relaxation of the difficulty of initial courses will not have a major impact further in the program. The curriculum coherence analysis discovered that there are four major classes of courses in the program: Basic Concepts, Advanced CS concepts, Programming and Project/Client related. A fifth class, grouping the Electrical Engineering courses, seems to be disconnected from the overall flow of the program and should be considered for elimination. The dropout and enrollment path analysis suggests that failing basic courses, not really related with CS, is one of the main indicators of future desertion. Maybe the curriculum re-design should provide more core CS courses at the beginning of the program. Finally, the load/performance analysis indicates that the current workload is too heavy for the majority of students. A re-design should consider lowering the recommended load to 5 courses, while allowing high-performing students to advance at a faster pace. All these recommendations should be evaluated by the program stakeholders in order to to provide the qualitative analysis needed to validate them.

Far from trying to be a comprehensive set of analytics tools for curriculum (re-)design, this work attempts to discuss how very simple statistical and computational techniques could be applied to historic academic performance data to gain insight on possible problems or improvement opportunities in whole program curricula. Moreover, the real objective behind this research is to encourage other researchers to critique and improve over these simple techniques in order to extend the impact of Learning Analytics beyond the scope of individual courses.

10. REFERENCES

[1] L. Albert. Curriculum design: Finding a balance. *Journal of rheumatology*, 34(3):458–459, 2007.

[2] I. Archambault, M. Janosz, J.-S. Fallu, and L. S. Pagani. Student engagement and its relationship with early high school dropout. *Journal of adolescence*, 32(3):651–670, 2009.

[3] V. V. Busato, F. J. Prins, J. J. Elshout, and C. Hamaker. Intellectual ability, learning style, personality, achievement motivation and academic success of psychology students in higher education. *Personality and Individual Differences*, 29(6):1057 – 1068, 2000.

[4] J. P. Campbell, P. B. DeBlois, and D. G. Oblinger. Academic analytics: A new tool for a new era. *Educause Review*, 42(4):40, 2007.

[5] J. P. Caulkins, P. D. Larkey, and J. Wei. Adjusting gpa to reflect course difficulty. 1996.

[6] B. B. de Koning, S. M. Loyens, R. M. Rikers, G. Smeets, and H. T. van der Molen. Generation psy: Student characteristics and academic achievement in a three-year problem-based learning bachelor program. *Learning and Individual Differences*, 22(3):313 – 323, 2012.

[7] J. W. Denton, V. Franke, and K. N. Surendra. Curriculum and course design: a new approach using quality function deployment. *Journal of Education for Business*, 81(2):111–117, 2005.

[8] M. Ester, H.-P. Kriegel, and M. Schubert. Web site mining: a new way to spot competitors, customers and suppliers in the world wide web. In *Proceedings of the eighth ACM SIGKDD international conference*, pages 249–258. ACM, 2002.

[9] R. Ferguson. Learning analytics: drivers, developments and challenges. *International Journal of Technology Enhanced Learning*, 4(5):304–317, 2012.

[10] A. A. for the Advancement of Science. *Designs for Science Literacy*. Oxford University Press, Mar. 2001.

[11] W. J. Jordan, J. Lara, and J. M. McPartland. Exploring the causes of early dropout among race-ethnic and gender groups. *Youth & Society*, 28(1):62–94, 1996.

[12] R. B. McNeal Jr. Extracurricular activities and high school dropouts. *Sociology of education*, pages 62–80, 1995.

[13] S. Parthasarathy, M. J. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 251–258. ACM, 1999.

[14] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining access patterns efficiently from web logs. In *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 396–407. Springer, 2000.

[15] P. Pukkila, J. DeCosmo, D. C. Swick, and M. Arnold. How to engage in collaborative curriculum design to foster undergraduate inquiry and research in all disciplines. *Developing and Sustaining a Research-Supportive Curriculum: A Compendium of Successful Practices*, pages 341–357, 2007.

[16] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[17] R. W. Rumberger. Dropping out of high school: The influence of race, sex, and family background. *American Educational Research Journal*, 20(2):199–220, 1983.

[18] G. Siemens and P. Long. Penetrating the fog: Analytics in learning and education. *Educause Review*, 46(5):30–32, 2011.

[19] K. Singh, M. Granville, and S. Dika. Mathematics and science achievement: Effects of motivation, interest, and academic engagement. *The Journal of Educational Research*, 95(6):323–332, 2002.

[20] J. A. Sunderman. Curriculum Incubation : Data-driven Innovative Instructional Design. In *ASEE Annual Conference*, 2012.

[21] B. G. Tabachnick and L. Fidell. *Using Multivariate Statistics: International Edition*. Pearson, 2012.

[22] F. S. Turner, D. R. Clutterbuck, C. A. Semple, et al. Pocus: mining genomic sequence annotation to predict disease genes. *Genome biology*, 4(11):R75–R75, 2003.

[23] P. Wolf. A model for facilitating curriculum development in higher education: A faculty-driven, data-informed, and educational developer–supported approach. *New Directions for Teaching and Learning*, 2007(112):15–20, 2007.

[24] A. Wolff, Z. Zdrahal, A. Nikolov, and M. Pantucek. Improving retention: predicting at-risk students by analysing clicking behaviour in a virtual learning environment. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, pages 145–149. ACM, 2013.

[25] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60, 2001.