"Exploration and implementation of computerized adaptative testing methods"

Leclercq, Matthieu ; Linsmeau, Clément

ABSTRACT

La vie de tout étudiant est rythmée par des tests. Examens écrits, pratiques ou oraux, tests formatifs ou encore présentation orale sont d'autant de façons d'évaluer le niveau d'un étudiant pour savoir si il a assimilé la matière d'un cours. Dans les tests écrits, un large ensemble, similaire pour tous, de questions est demandé à tous les étudiants sans prendre en compte leur niveau individuel. Dans ces étudiants, il y en a de tous niveaux qui sont pourtant tous testés de la même façon. Avec un éventail restreint de questions particulier à chaque étudiant, il serait cependant possible d'ajuster la difficulté du test à chacun d'entre eux. Cela permettrait de déterminer leur niveau réel plus rapidement et avec autant de précision que si toutes les questions leur avait été posées et permet également de mieux évaluer les niveaux extrêmes. Cette méthode de test existe déjà sous le nom de "Test Adaptatif". Ce type de test propose une question s'ajustant aux réponses que l'étudiant aura précédemment données. L'Université Catholique de Louvain (UCLouvain) n'utilise pas ce genre de test actuellement. Nous avons jugé utile l'implémentation d'un système permettant aux professeurs de l'UCLouvain de réaliser leurs propres tests adaptatifs sur la plateforme INGInious. Chaque étudiant pourra, dès lors que le système sera implémenté, se tester en quelques questions. Bien qu'encore peu répandus, les tests adaptatifs offrent une alternative comportant certains avantages par rapport aux tests classiques. Nous espérons donner plus de visibil...

CITE THIS VERSION

UCLouvain

epl

École polytechnique de Louvain

Mémoire : "Exploration and implementation of computerized adaptative testing methods"

Auteurs: **Matthieu Leclercq, Clément Linsmeau**
Promoteur: **Kim Mens**
Lecteurs: **Anthony Gego, Olivier Goletti, Olivier Bonaventure**
Année académique 2021–2022
Master [120] en sciences informatiques

Remerciements

Nous aimerions tout d'abord remercier notre promoteur, le professeur Kim Mens ainsi qu'Olivier Goletti l'un de ses assistants. Le temps qu'ils ont pris pour suivre notre Mémoire, pour nous prodiguer des conseils d'orientation et de redirection lorsque nous étions indécis sur les chemins à emprunter ainsi que leurs relectures ont guidés notre route jusqu'à l'aboutissement de notre travail.

Nous tenons aussi à remercier Anthony Gego qui nous a aidé pour intégrer notre système sur la plateforme INGInious en nous assistant en cas de problèmes ou en nous fournissant des références vers la documentation appropriée.

Nous remercions les professeurs Olivier Bonaventure et Yves Deville qui, afin de valider notre système, nous ont donné accès aux statistiques de leurs cours.

Finalement, nous remercions nos familles et nos amis pour le soutien qu'ils nous ont apporté tout au long de cette année de travail. Plus particulièrement, Julien Lienard pour son avis objectif et ses retours sur les versions de notre système.

Résumé

La vie de tout étudiant est rythmée par des tests. Examens écrits, pratiques ou oraux, tests formatifs ou encore présentation orale sont d'autant de façons d'évaluer le niveau d'un étudiant pour savoir si il a assimilé la matière d'un cours. Dans les tests écrits, un large ensemble, similaire pour tous, de questions est demandé à tous les étudiants sans prendre en compte leur niveau individuel. Dans ces étudiants, il y en a de tous niveaux qui sont pourtant tous testés de la même façon. Avec un éventail restreint de questions particulier à chaque étudiant, il serait cependant possible d'ajuster la difficulté du test à chacun d'entre eux. Cela permettrait de déterminer leur niveau réel plus rapidement et avec autant de précision que si toutes les questions leur avait été posées et permet également de mieux évaluer les niveaux extrêmes. Cette méthode de test existe déjà sous le nom de "Test Adaptatif". Ce type de test propose une question s'ajustant aux réponses que l'étudiant aura précédemment données. L'Université Catholique de Louvain (UCLouvain) n'utilise pas ce genre de test actuellement. Nous avons jugé utile l'implémentation d'un système permettant aux professeurs de l'UCLouvain de réaliser leurs propres tests adaptatifs sur la plateforme INGInious. Chaque étudiant pourra, dès lors que le système sera implémenté, se tester en quelques questions. Bien qu'encore peu répandus, les tests adaptatifs offrent une alternative comportant certains avantages par rapport aux tests classiques. Nous espérons donner plus de visibilité à ce type de test en implémentant ce système.

Table des matières

Introduction

Déterminer si une personne a acquis suffisamment de connaissances concernant une matière est fondamental pour s'assurer que cette personne sera apte à faire face à des problèmes requérant des compétences liées à cette matière. Le cadre venant en premier à l'esprit quand on parle d'évaluer des personnes est celui des études. Qu'elles soient primaires, secondaires, ou supérieures, les études sont un lieu d'apprentissage (pour l'étudiant) et d'appréciation de l'apprentissage (pour le professeur) pour une certaine matière (le cours).

Il existe différentes façons pour mesurer les compétences d'un étudiant : examens écrits, examens oraux, examens pratiques, devoirs cotés, tests certificatifs, sont quelques exemples parmi d'autres. À la base d'un bon test, il y a son contenu, le type de questions. Comment les choisir pour évaluer au mieux les étudiants ? Doit-on poser les mêmes questions à tous ? Doit-on utiliser différents types de questions pour différents cours ? Devrions-nous utiliser seulement des questions à choix multiples, seulement des questions ouvertes, un mélange des deux, ou un tout autre type de question ?

Dans ce mémoire, nous allons explorer une manière d'évaluation encore peu répandue : les tests adaptatifs. Il s'agit de tests dans lesquels, si l'étudiant répond correctement à une question, la prochaine question sera plus difficile. En revanche, s'il répond mal, la prochaine question sera plus facile. Il a été prouvé que s'ils sont utilisés dans un cadre précis et bien défini, ils permettent d'évaluer des personnes avec un même niveau de précision que les tests classiques (dits linéaires), mais en prenant moins de temps. Ils sont également plus à même d'évaluer des étudiants avec des niveaux de connaissance plus extrêmes. Dans un test linéaire plus facile, les bons étudiants auront simplement la note maximale ou, inversement, dans un test plus difficile, les moins bons étudiants risquent de se retrouver avec la note minimale. Un test adaptatif n'aura pas ce genre de limitation et est donc plus pertinent pour ce type d'élèves.

Par ailleurs, l'École Polytechnique de Louvain (abrégée EPL par la suite) dis-

pose d'une plateforme web d'aide à l'apprentissage qui permet aux professeurs de proposer des exercices aux étudiants. Cette plateforme d'exerciseur, INGInious, est le cadre idéal pour implémenter ces tests. En effet, INGInious se rapproche de ce cadre précis et bien défini dont les tests adaptatifs ont besoin pour être plus efficaces que des tests linéaires.

Ce mémoire tente de répondre à deux questions de recherche présentées dans la figure 1.

QR1 Est-il possible d'implémenter un système permettant l'utilisation des tests adaptatifs ?

QR2 Est-il possible de rendre la conception de tels tests réalisable par des professeurs ?

FIGURE 1 – Questions de recherche

Suite à ces questions de recherches, nous essaierons de savoir s'il est possible de valider notre modèle en faisant passer un test adaptatif formatif à des étudiants pour quelques-uns des cours de l'EPL.

Pour atteindre ces objectifs, nous expliquerons en détail ce qu'est réellement un test adaptatif ainsi que la théorie dont nous aurons besoin pour faire fonctionner ces tests au cours du chapitre 1.

Le chapitre 2 présentera en détail l'approche que nous avons décidé de suivre pour rendre effectives la conception et l'utilisation de ces tests adaptatifs.

Le chapitre 3 présentera l'implémentation concrète de ces tests adaptatifs au sein de la plateforme INGInious travaillant de pair, au moyen d'une API REST, avec un programme `R` s'occupant de la partie statistique.

Le chapitre 4 exposera une validation de notre prototype sur INGInious en transformant des tests initialement assemblés sur le site Wooclap en tests adaptatifs sur INGInious.

Le chapitre 5 présentera un mode d'emploi à destination des professeurs, pour fabriquer eux-mêmes leurs tests adaptatifs ainsi qu'à destination des étudiants, pour qu'ils puissent facilement utiliser l'interface des tests adaptatifs sur INGInious.

Le chapitre 6 parlera des limites et des contraintes liées à notre système en proposant des améliorations futures qui pourraient solutionner certains de ces problèmes.

Finalement, nous exposerons notre point de vue personnel sur le Mémoire ainsi qu'une réponse aux questions de recherche précédemment introduites en présentant un résumé du travail effectué.

Chapitre 1

Théorie des tests adaptatifs

Dans ce chapitre, nous expliquons en détail ce que sont les tests adaptatifs, leurs avantages et inconvénients par rapport aux tests linéaires, leurs étapes de déroulement ainsi que la théorie statistique derrière ces tests.

1.1 Qu'est-ce que sont les tests adaptatifs ?

Il est primordial d'évaluer les connaissances des personnes. Pour cela, dans nos études par exemple, la manière la plus répandue d'évaluer est de faire passer un test dit linéaire pour estimer à quel point la personne connaît la matière. Il semble logique que la plupart des points de celle-ci soient couverts par le test. Il semble logique aussi que ce test soit similaire pour chaque étudiant pour ne pas avoir d'injustice entre deux tests d'étudiants. Une alternative à ces tests linéaires connue sous le nom de "Tests adaptatifs" (ou CAT pour Computerized Adaptive Test) existe. Cette alternative fait partie des "tests par ordinateurs" (abrégé CBT plus tard pour Computer-Based Tests) qui utilise des systèmes informatiques pour faire passer des tests contrairement aux tests papier-crayon traditionnels.

Dans un test adaptatif, chaque question posée s'adapte au niveau actuel de la personne, de l'étudiant testé. Si l'étudiant passant ce test réussit une question qui lui est posée, la prochaine question sera plus difficile que la précédente. Si par contre il échoue à cette question, la suivante sera plus simple. Pour cela, il faut évaluer le niveau de chaque question pour savoir laquelle est la plus pertinente à poser pendant le test à l'étudiant. En parallèle, il faut pouvoir déterminer le niveau de l'étudiant automatiquement au début du test et après chaque question. Il y a donc un niveau différent pour chaque question et pour le niveau de l'étudiant. La longueur d'un tel test peut varier d'un étudiant à un autre. Cette longueur dépendra du nombre de questions qu'il a à réaliser pour que son niveau soit déterminé avec

une précision suffisante. Il faut aussi introduire différentes conditions d'arrêt pour stopper un test adaptatif.

Deux raisons principales nous poussent à nous intéresser à ce genre de tests : premièrement, utilisés à bon escient et dans le bon contexte, ces tests se montrent plus efficaces que les tests linéaires [13]. Nous expliquerons dans quels contextes utiliser ces tests après avoir listé leurs avantages et inconvénients. Deuxièmement, ces tests commencent seulement à se propager en Belgique maintenant. Par exemple le site Wallangues permet depuis 10 ans d'évaluer son niveau dans les trois langues nationales ainsi qu'en anglais. Celui-ci est construit grâce à 4 tests adaptatifs portant sur la grammaire, le vocabulaire, la compréhension orale et la compréhension écrite. Des missions sont ensuite mises à disposition en fonction du niveau de langue obtenu par ce test. Nous voulions apporter notre pierre à l'édifice et parler de cette alternative encore méconnue jusqu'ici. Dans le monde, ils sont déjà utilisés. En voici quelques exemples :

- Aux Pays-Bas, les prérequis en arithmétique de tous les étudiants de première primaire sont désormais évalués grâce à un package CAT (WISCAT-pabo testing package) [8].
- PISA (Programme for International Student Assessment) créé par l'OCDE (Organisation de Coopération et de Développement Économiques) permet à plus de 90 pays dans le monde de tester les connaissances et les compétences d'étudiants de 15 ans en lecture, mathématiques et en sciences. Toute la partie lecture du test est réalisée avec des tests adaptatifs.
- En France, le site Pix rend disponible, à tout citoyen français, un service publique pour évaluer, développer et certifier ses compétences numériques. Celui-ci adapte ses tutoriels après le test en fonction des réponses de la personne au test adaptatif [20].
- La plateforme Concerto permet de créer son test adaptatif en ligne et de le rendre disponible directement sur le site [15].

Nous n'utilisons pas cette dernière plateforme pour 3 raisons : Premièrement, l'ensemble des exercices dont nous nous servons pour créer des tests adaptatifs existent déjà sur la plateforme d'exercices de l'EPL, INGInious. Deuxièmement, comme une utilité potentielle de notre système était d'implémenter des examens, en cas de controverse il faut que nous soyons en mesure d'accéder au code source pour être capable de justifier pourquoi telle question a été posée à tel moment et surtout pouvoir démontrer les raisons pour lesquelles un score X, représentation concrète du niveau de l'étudiant, a été assigné à un étudiant Y. Finalement, un changement de plateforme est à éviter : les professeurs et les étudiants sont déjà habitués à INGInious alors autant ne pas les désorienter.

1.2 Comparaison entre les tests linéaires et les tests adaptatifs

Dans cette section, nous comparons les avantages et inconvénients des tests adaptatifs par rapport aux tests linéaires pour savoir quand privilégier plutôt l'un que l'autre. Les tests linéaires sont les tests les plus répandus au sein de nos études. C'est pour cette raison que la comparaison se fait avec ces tests-là.

1.2.1 Avantages des tests adaptatifs

Le gros point fort des tests adaptatifs par rapport aux tests linéaires est leur différence de longueur. En effet, comme le test est personnalisé, chaque étudiant devra répondre à moins de questions pour un niveau de précision identique voire supérieur de son niveau [13].

De plus, il a été montré qu'un étudiant avec un niveau de connaissance moyen par rapport à la difficulté d'un test linéaire sera évalué avec une précision équivalente par les 2 types de tests, mais ce ne sera pas le cas pour des étudiants avec des niveaux de connaissances extrêmes [9]. Les étudiants avec un niveau supérieur/inférieur au niveau moyen prévu par le test seront mieux évalués avec un test adaptatif.

Un dernier avantage que nous pouvons soulever est celui de la difficulté de tricher à ce genre de tests : les étudiants ne sont pas forcément confrontés aux mêmes questions que leurs camarades ce qui rend la triche collaborative complexe.

1.2.2 Inconvénients des tests adaptatifs

Pour créer un test linéaire, il suffit de créer les questions et de les poser aux étudiants de façon séquentielle. Pour un test adaptatif, cela est bien plus laborieux. En effet, il faut partir de questions auxquelles certains étudiants ont déjà répondu, car cela est nécessaire pour donner un niveau de difficulté à chaque question. Il est difficile pour un professeur d'estimer parfaitement la difficulté relative de chacune des questions d'un tel test. Il ne pourra pas être sûr qu'une question sera plus facile qu'une autre sans avoir demandé leur avis à des étudiants (en les testant). Voilà pourquoi il vaut mieux partir d'un ensemble de questions déjà existantes avec des statistiques de réponses.

De plus, implémenter un système permettant de créer ces tests est une tâche complexe. Il faut déterminer quelle question poser à quel moment, ne pas poser plusieurs fois la même question, fixer une fin au test, etc. La création d'un tel système est l'un des problèmes auquel notre Mémoire s'attaque.

L'un des autres problèmes majeurs qui s'oppose aux tests adaptatifs est le nombre de questions en réserve. En effet, si on veut pouvoir trouver des questions

pertinentes pour chaque étudiant, il faut un nombre conséquent de questions et c'est l'un des éléments que nous allons explorer. On peut aussi mentionner qu'il est difficile de gérer un nombre de questions pour chaque thème à l'intérieur d'un même domaine. Divers problèmes d'ordre éthique peuvent aussi survenir : comme les étudiants ne sont pas évalués de la même façon, est-ce que ce genre de test est équitable ? D'autres contraintes seront expliquées dans les prochaines sections.

Comme nous le voyons ici, les tests adaptatifs ne sont pas la panacée pour évaluer des étudiants. Il existe en effet un certain nombre de contraintes et limitations qui empêchent ces tests d'être utilisés pour toute évaluation. Il faut donc les utiliser dans le bon contexte.

Mais alors, qu'est-ce qui fait un bon contexte pour des tests adaptatifs ? Il faut que les deux principaux inconvénients soient déjà surmontés. Tout d'abord, il faut détenir un large ensemble de questions auxquelles des étudiants ont déjà répondu. Ces questions devraient couvrir au mieux le spectre des difficultés différentes pour pouvoir évaluer des étudiants de tous niveaux. Ensuite, il faut un système implémenté qui permet de créer ces tests. Ainsi, la réponse donnée par l'étudiant doit être facilement analysable par un ordinateur (comme toute la partie vérification de la réponse est automatique). Nous parlerons plus en détail des contraintes dans le Chapitre 6.

1.3 Déroulement de tests adaptatifs

Dans cette section, nous expliquons en détails chaque étape des tests adaptatifs et nous introduisons aussi certaines des contraintes que ces tests nous imposent. Cette section est fortement inspirée de l'excellent livre de David Magis : *Computerized Adaptive and Multistage Testing with R* [13].

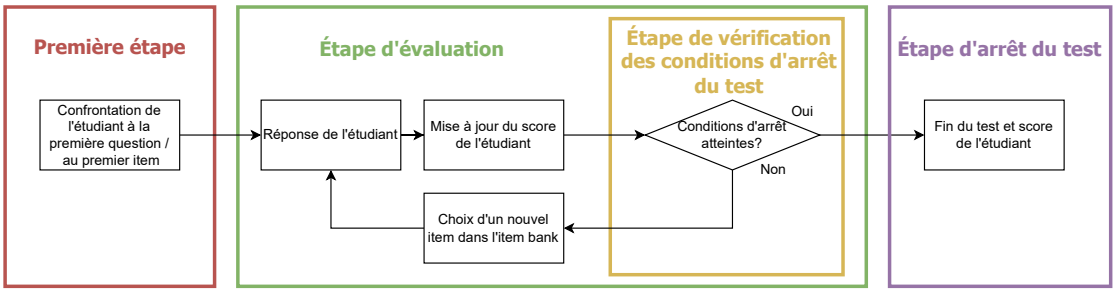

FIGURE 1.1 – Étapes d'un test adaptatif

1.3.1 Création de la banque d'items

Avant de commencer le test, il faut préalablement créer un ensemble de questions auxquelles les étudiants devront répondre. Cet ensemble de questions se nomme la *banque d'items*. Nous gardons l'appellation item car c'est la plus représentative de ce concept. Il faudra rassembler un nombre conséquent de questions pour créer cette banque d'items. Chaque item dans cette banque d'items a déjà reçu un niveau avant le début du test. On dit que la banque d'items a été calibrée. Nous verrons comment calibrer cet ensemble dans la section 1.4.

1.3.2 Première étape

La première étape du déroulement d'un test adaptatif (en rouge sur la figure 1.1) consiste à sélectionner un premier item dans la banque d'items pour confronter l'étudiant à celui-ci. Ce premier item est choisi comme étant le plus proche par rapport au niveau moyen des étudiants ayant déjà répondu à cette banque d'items. Tous les étudiants précédents n'ont pas besoin d'avoir passé toutes les questions pour que le test fonctionne.

Il est possible de calibrer la difficulté du test si des informations à propos du niveau des étudiants passant actuellement le test sont disponibles. Par exemple, si le niveau moyen des étudiants actuels est plus élevé que celui des étudiants ayant déjà passé le test, il serait possible de poser une première question plus difficile que celle prévue. Il est aussi possible de ne pas partir d'une seule question mais d'un ensemble de questions pour ne pas que chaque étudiant débute avec la même première question. Après cette première étape, nous passons à l'étape d'évaluation (en vert sur la figure 1.1)

1.3.3 Étape d'évaluation

Cette étape est composée de 4 parties : la réponse de l'étudiant, la mise à jour du score de l'étudiant, le choix d'un nouvel item dans la banque d'items et l'étape de vérification des conditions d'arrêt du test. Nous considérons cette dernière partie comme une sous-étape (en jaune sur la figure 1.1) et sera expliquée lors de la prochaine section.

Premièrement, la réponse de l'étudiant dépend du style de question qui est posé. Il est très complexe de poser des questions ouvertes dans ce genre de test car l'évaluation de la réponse doit être réalisée de manière automatique par un ordinateur. Le plus souvent, les questions posées seront soit des questions à choix multiples (QCM) ou une variante comme les questions à réponses multiples (QRM), soit des questions avec une réponse unique (un simple mot ou une réponse numérique).

Dans notre cas il sera aussi possible de poser des questions de programmation. Nous expliquerons dans la section 3.2 la raison pour laquelle la plateforme INGInious aide beaucoup à répondre à cette contrainte.

Vient ensuite la mise à jour du score de l'étudiant. Après chaque question, le score de l'étudiant est recalculé en tenant compte de sa dernière réponse ainsi que de toutes les réponses qu'il a déjà données. Plus il répondra à des questions, plus la précision de son score augmentera. Il faut poser un certain nombre de questions pour que ce score soit suffisamment précis. Pour estimer le niveau de l'étudiant, de nombreuses techniques existent (fonction de vraisemblance, modèle de Bayes,etc.). Nous expliquons ces techniques dans la section 1.4, et le choix de la technique pour notre système dans le Chapitre 2.

La troisième partie vérifie les conditions d'arrêt du test (en jaune sur la figure 1.1). La prochaine section y est dédiée. Si celles-ci ne sont pas respectées, le test continue avec la prochaine et dernière partie de l'étape, le choix d'un nouvel item dans la banque d'items.

Comme l'étudiant, chaque question possède un score. Lors de cette dernière partie, le système calcule quelle est la question la plus pertinente à poser à l'étudiant en fonction de ses précédentes réponses, ainsi que du score de chaque question à laquelle il n'a pas encore répondu. Bien entendu, les questions auxquelles il a déjà répondu sont écartées de la banque d'items d'où sont tirées les prochaines questions. Cela empêche de poser plusieurs fois la même question. Il existe aussi une multitude de méthodes de sélection d'item avec chacune leurs avantages et leurs inconvénients. La section 1.4 passera en revue celles-ci. Le chapitre 2 expliquera celle que nous avons choisie ainsi que la raison de ce choix.

L'étudiant répond donc à cette question nouvellement choisie dans la banque d'items, son score est mis à jour, etc. Cette étape de test boucle jusqu'à ce que l'une des conditions d'arrêt du test devienne vraie et que le système passe à l'étape d'arrêt du test (en violet sur le schéma 1.1).

1.3.4 Étape de vérification des conditions d'arrêt du test

Lors de l'étape d'évaluation, après que le score de l'étudiant ait été mis à jour, on ne passe pas directement à la partie de choix d'un nouvel item. En effet, si nous faisions cela, le test ne s'arrêterait jamais ou planterait lorsque la banque d'items devient vide. Il faut tout d'abord vérifier les conditions d'arrêt du test. Il existe 4 règles principales [13].

— La règle de **précision** arrête le test lorsque l'erreur type associée au niveau de l'étudiant est devenue inférieure ou égale à la valeur du seuil pré-défini. Cette condition permet d'obtenir une précision du score de l'étudiant minimale plus affinée, mais risque d'augmenter le nombre de questions posées. Chaque étudiant risque de répondre à un nombre différent de questions. Cela peut causer des problèmes si le test a une limite de temps. La section 3.3 explique pratiquement le calcule de ce seuil.
— La règle de **longueur** quant à elle, impose un nombre maximal de questions au test. A l'inverse de la précédente règle, cette règle peut réduire le niveau de précision du score de l'étudiant mais permet de poser un même nombre maximal de questions à tous. En pratique, cette règle ne devrait être utilisée que très peu.
— La règle de **classification** ne peut être utilisée que dans le cas où le niveau de connaissance peut être fixé à un certain seuil. Ce seuil est défini arbitrairement par le créateur du test. Le but du test est alors de déterminer si le candidat a un niveau de connaissance supérieur ou inférieur à ce seuil. Si l'intervalle de confiance du niveau actuel du candidat chevauche le seuil, il faut continuer le test. Dans l'autre cas (l'intervalle de confiance ne couvre pas le seuil) le candidat peut être classé comme maîtrisant la compétence (si l'intervalle de confiance est supérieur au seuil) ou non (si l'intervalle est inférieur au seuil).
— En dernier lieu, la règle d'**information** se concentre sur la quantité d'information que les items restants dans la banque d'items pourraient nous apporter. Si cette quantité n'est pas suffisante par rapport à un seuil prédéfini, le test s'arrête. La quantité d'information apportée peut être, par exemple, la différence entre l'erreur type actuelle et l'erreur type après que l'étudiant ait répondu à la question choisie.

Ces règles peuvent être soit utilisées seules ou à plusieurs. Dès lors qu'au moins l'une de ces règles est respectée, le test s'arrête et il ne propose plus de nouvelle question à l'étudiant. Dans le cas contraire, l'étape suivante sera de proposer une nouvelle question et de continuer le test (retour à l'étape verte du schéma 1.1).

1.3.5 Étape d'arrêt du test

Cette étape est la dernière étape et survient quand l'une des conditions d'arrêt du test est vraie. Elle retourne alors l'estimation finale du score de l'étudiant. D'autres informations, telle que l'erreur type finale avec son intervalle de confiance autour du niveau de l'étudiant, peuvent aussi être communiquées.

1.4 Principes théoriques

Passons maintenant à la théorie pour réaliser toutes ces actions décrites plus tôt : Comment donner un niveau à l'étudiant ? À une question ? Comment déterminer la question la plus adaptée à être posée à un certain moment ? Dans cette section, nous expliquerons plusieurs méthodes pour répondre à ces questions. La figure 1.3 reprendra chaque méthode choisie et comment nous l'utilisons à la fin du chapitre.

Tout ce que nous avons expliqué jusqu'à présent se base sur l'"Item Response Theory" (IRT). C'est grâce à cette théorie que nous pouvons répondre à toutes ces questions. C'est aussi pour cette raison que nous parlons d'item et de banque d'items. L'IRT est une famille de modèles mathématiques qui permet l'évaluation de traits latents. Un trait latent est toute chose inobservable, mais qui peut se manifester. C'est exactement ce dont nous avons besoin : le trait latent que nous voulons mesurer est le niveau d'habilité / le niveau de compétence / la capacité / l'intelligence de l'étudiant. Il est possible d'évaluer plusieurs traits latents mais ici, bien qu'il possède plusieurs noms, il n'y en a qu'un que nous tentons de déterminer. Comme introduit auparavant, c'est au travers de questions, d'items, que l'évaluation va être réalisée. Dans la plupart des cas, il est possible de traiter les réponses de deux manières différentes : dichotomique (la réponse donnée est soit vraie soit fausse) ou polytomique (la réponse donnée est bonne à 20%, 40%, etc.). On peut voir un modèle de la famille IRT comme une fonction mathématique qui décrit la probabilité d'observer chaque réponse possible à l'item. Celle-ci dépend du niveau de l'étudiant et des paramètres de la question. Bien entendu plus le niveau de l'étudiant est élevé, plus la probabilité de bien répondre à une question augmente.

Le modèle IRT le plus simple et le plus utilisé est appelé le modèle de Rasch ou modèle logistique à un paramètre (1PL) et est un modèle dichotomique ([11], [16]). Sa fonction de réponse d'item (abrégée IRF pour Item Response Function) prend en compte le score actuel de l'étudiant et le score de la réponse. L'IRF permet de donner la probabilité, pour un étudiant d'un certain niveau, de correctement répondre à une question. Plus le niveau est élevé, plus l'étudiant est bon ou plus la question est considérée comme difficile. L'équation peut s'écrire comme ceci [13] :

$$IRF(x, y) = Pr(X_{xy} = 1|x, y) = \frac{exp(D[x - y])}{1 + exp(D[x - y])} \tag{1.1}$$

où x représente le niveau de l'étudiant et y la difficulté de la question. A l'intérieur de l'exponentielle, nous voyons donc que le niveau de la question est soustrait au niveau de l'étudiant, le tout multiplié par D ensuite. Si l'étudiant et la question ont le même niveau, l'intérieur de l'exponentielle sera nul. La probabilité que l'étudiant réussisse la question est donc de 50%. Si la question est plus dure/facile, la soustraction

donnera un résultat négatif/positif et la probabilité sera inférieure/supérieure à 50%. Pour les modèles améliorés, il faut rajouter des paramètres. D est une valeur que l'on peut fixer à 1 pour simplifier les équations. Dans le second modèle (2PL) cette valeur D devient le paramètre a qui permet à l'équation d'avoir différentes pentes. Plus ce paramètre est élevé plus le modèle distinguera entre les étudiants à faible et à haute capacité. On peut rajouter un 3ème paramètre, c, qui est traduit par une asymptote basse qui représente la possibilité de deviner la réponse ou de la trouver par chance (3PL). C'est le paramètre de pseudo-supposition. Le 4ème et dernier paramètre, d, est traduit par une asymptote haute qui représente un paramètre d'inattention (4PL). Ce paramètre décrit la situation où l'étudiant rate la question alors qu'il connaît la réponse.

L'équation avec tous ses paramètres peut s'écrire comme cela :

$$IRF(x, y, a, c, d) = Pr(X_{xy} = 1|x, y, a, c, d) = c + (d - c)\frac{exp(a[x - y])}{1 + exp(a[x - y])} \quad (1.2)$$

Pour revenir sur le modèle 3PL, il faut simplement fixer d à 1 et pour revenir au modèle 2PL il faut simplement fixer c à 0. Pour observer l'impact que chaque paramètre a sur la fonction, la figure 1.2 montre 4 graphiques faisant varier ces paramètres.

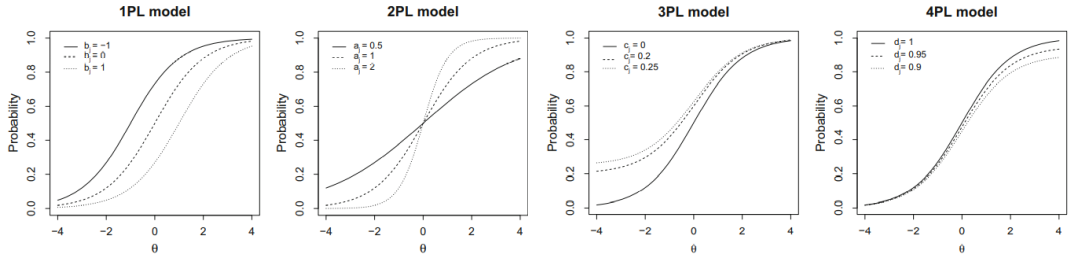

FIGURE 1.2 – Schémas mettant en évidence les différents impacts des paramètres sur la courbe de probabilité de répondre correctement pour le modèle dichotomique (Source : *Computerized Adaptive and Multistage Testing with R* [13])

Les axes x et y du schéma 1.2 représentent respectivement le niveau d'habilité de l'étudiant et la probabilité de réussir la question.

1.4.1 Détermination de la prochaine question à poser à l'étudiant

Grâce à ces quatre paramètres pour chaque question, il est maintenant possible de choisir la question la plus pertinente dans la banque d'items pour un étudiant. Comme expliqué dans la section 1.3.3 il existe bon nombre de techniques pour cela. Nous allons ici exposer 2 de ces techniques pour sélectionner la question. Notons ici qu'il n'y a pas de meilleures techniques. Chacune à ses avantages et ses inconvénients.

— **Critère bOpt** ou **règle d'Urry** est la règle la plus intuitive. Elle sélectionne la question à laquelle l'étudiant n'a pas encore répondu dont le niveau est le plus proche du niveau de l'étudiant. Cette façon de faire est la plus simple à mettre en oeuvre : il s'agit d'une simple soustraction entre des valeurs que nous connaissons déjà sans devoir passer par d'autres calculs. Seul problème, cette difficulté de question n'est disponible que dans un modèle dichotomique. Dans un modèle polytomique, aucun paramètre ne représente la difficulté de la question directement.

— **Maximum Fisher Information** (MFI) va sélectionner l'item dans la banque d'item qui maximise l'information à partir de l'estimation actuelle du niveau de l'étudiant. Cette information représente la plus grande différence entre le seuil actuel et le seuil après avoir répondu à la question. Cette estimation est bien sûr déterminée par l'ensemble des réponses aux items qu'il a déjà réalisé. Nous présenterons dans la section 1.4.3 la façon de calculer cette estimation. Il s'agit de la méthode utilisée le plus souvent, car elle est la plus simple et la plus facile à calculer pour tous types d'items.

Les autres techniques que nous n'expliquons pas sont soit des techniques améliorant la précision de la MFI (**Maximum Likelihood Weighted Information**, **Maximum Posterior Weighted Information**, etc.) mais en augmentant la complexité de calcul soit l'inverse (sélection d'item aléatoire, ou techniques ne fonctionnant que pour des items dichotomiques avec une plus grande complexité calculatoire que la règle d'Urry comme la procédure **thOpt**). Nous avons choisi la méthode MFI et nous discutons de ce choix dans le prochain chapitre. Cette méthode est représentée en brun dans la figure 1.3.

1.4.2 Détermination des paramètres

Mais alors comment estimer ces paramètres qui nous permettent d'évaluer les questions ? Les techniques utilisées pour calibrer le modèle se déroulent comme cela : avant la construction du test adaptatif, il y a déjà à disposition des réponses d'anciens étudiants aux questions. Ensuite, sont utilisés des algorithmes d'Espérance-Maximisation. Ces algorithmes contiennent deux phases. La première,

l'espérance, calcule les paramètres en fonction des niveaux estimés à l'étape de maximisation. L'étape de maximisation recalcule les niveaux des étudiants en maximisant la vraisemblance du niveau de ceux-ci par rapport à ces paramètres. Comme l'un dépend de l'autre, il faut un ensemble de données de départ pour démarrer l'algorithme. C'est à cela que servent les données des étudiants précédemment interrogés. Lorsque l'algorithme ne permet plus d'affiner les paramètres et que la maximisation ne s'améliore plus, l'algorithme aura déterminé les paramètres optimaux. Concrètement, nous utilisons un algorithme de maximum de vraisemblance marginal (MML pour Marginal Maximum Likelihood) qui nous renvoie les paramètres pour le modèle 2PL (y et a). Cette méthode est représentée en bleu dans la figure 1.3.

1.4.3 Détermination et mise à jour du niveau de l'étudiant

Pour le niveau de l'étudiant nous utilisons le modèle de Bayes qui est un modèle fonctionnant avec des items dichotomiques ou polytomiques. Cette méthode est représentée en gris dans la figure 1.3. Le modèle de Bayes va calculer le niveau de l'étudiant en fonction des réponses que celui-ci a donné aux précédentes questions. Un niveau moyen est donné au début du test à l'étudiant car l'algorithme n'a pas de réponses sur lesquelles se baser à ce moment-là. En pratique, il va déterminer quel niveau d'habilité maximise la distribution postérieure. La distribution postérieure est composée de la fonction de vraisemblance et de la distribution antérieure. La fonction de vraisemblance est la multiplication des probabilités d'opter pour la réponse que l'étudiant a effectivement choisi, et cela pour chaque question à laquelle il a déjà répondu :

$$Vrai(x) = \prod_{q=1}^{Q} \prod_{k=0}^{K_q} P_{qk}(x, \mathbf{p}_q)^{Y_{qk}} \tag{1.3}$$

Le premier terme de l'équation 1.3 représente la multiplication pour chaque question à laquelle l'étudiant a déjà répondu. Le second terme représente chaque possibilité de réponse pour une question q. Y_{qk} est égal à 1 si l'étudiant à répondu avec une réponse appartenant au groupe k à la question q et 0 sinon. Ce terme permet de ne multiplier que les probabilités des réponses qu'il a effectivement donné. Finalement $P_{qk}(x, \mathbf{p}_q)$ représente la probabilité pour une question q avec une réponse k et ses paramètres $\mathbf{p}_q$ d'être choisie pour un niveau d'étudiant x. Pour un modèle dichotomique, cette probabilité peut ressembler à l'équation 1.1 ou 1.2. Il est possible de s'arrêter là et de maximiser cette équation. Il s'agit alors de l'estimateur de vraisemblance maximum (MLE). Pour obtenir le modèle de Bayes, il faudra multiplier celui-ci par la distribution antérieure qui permet d'adapter le modèle à la population à laquelle est confronté le test :

$$Bayes(x) = f(x)Vrai(x) \tag{1.4}$$

Nous expliquons dans la section 3.3 pourquoi nous avons fait le choix de deux techniques différentes pour déterminer le niveau des questions et le niveau de l'étudiant.

Nous ne nous attarderons pas ici sur les modèles polytomiques car nous ne les utilisons que pour déterminer le niveau actuel estimé de l'étudiant et dans le choix de la prochaine question mais pas pour déterminer le niveau de la question (lors de la calibration).

Nous ne discuterons pas non plus des modèles pluri-dimensionnels ici car comme dit auparavant un seul trait latent nous intéresse, le niveau d'habilité de l'étudiant.

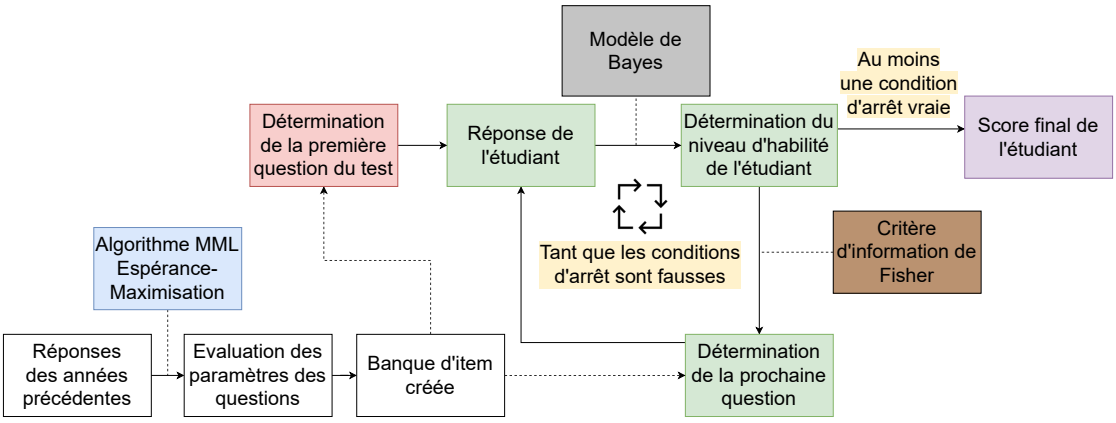

FIGURE 1.3 – Schéma récapitulatif des tests adaptatifs

Pour résumer toute la partie théorique, nous exposons dans le schéma 1.3 la suite des opérations qui nous permettent en partant de questions auxquelles des étudiants ont déjà répondu d'arriver au score final d'un étudiant. Le code couleur rappelle les différentes étapes d'un test adaptatif de la figure 1.1.

1.5 Travaux apparentés

Dans ce mémoire nous nous sommes principalement reposés sur l'excellent livre de David Magis : *Computerized Adaptive and Multistage Testing with R* [13]. Celui-ci explique les principes théoriques de l'IRT, du CAT mais aussi des MST (le Multistage Tests une approche hybride entre un test linéaire et un CAT). Tout cela en plus de présenter le package `R` que nous utilisons pour notre implémentation. Un autre mémoire sur le même sujet a aussi été réalisé il y a quelques années par Marie Visschers et Audrey Wenders *Utilisation de tests adaptatifs pour les cours*

d'informatique [22] et nous a aidé à comprendre et appréhender le package `R` vis à vis d'INGInious.

Il existe maints articles, livres, etc. débattant des techniques pouvant être utilisées pour mettre à jour le score de l'étudiant après qu'il ait répondu à une question, ou en d'autres termes des estimateurs de capacité (Ability Estimation). Lord explique plus en détail le maximum de vraisemblance dans *Applications of item response theory to practical testing problems* [10]. L'estimateur de Bayes est plus développé par Birnbaum dans *Statistical theory for logistic mental test models with a prior distribution of ability* [4]. La vraisemblance pondérée [21], l'estimateur a posteriori [5], et l'estimateur robuste [14] [18] sont aussi d'autres estimateurs avec chacun leurs articles dédiés.

Pour la sélection de la question la plus appropriée pour un étudiant, il existe une multitude d'articles et de livres en discutant plus en détail : nous pouvons citer une fois de plus *Computerized Adaptive and Multistage Testing with R* [13] qui réalise un aperçu sur 14 méthodes pour la sélection d'item. La plupart des explications sont elles-mêmes accompagnées de références vers d'autres articles. Pour ce qui est des articles : Choi compare 6 méthodes pour de la sélection d'items polytomiques [7] , Chang et Ying discutent de l'efficacité du critère FMI par rapport à d'autres critères [6] et Segall présente un modèle qui se base sur la variance minimum attendue (MEV) [19].

Finalement, nous pouvons citer *A Heuristic Method for Large-Scale Cognitive-Diagnostic Computerized Adaptive Testing* par Jill-Jênn Vie et co. qui utilise des Q-matrices et des graphiques préalablement créés pour construire des tests adaptatifs pouvant supporter plusieurs composants de connaissances (un équivalent aux traits latents) pour les modèles pluri-dimensionnels [20].

Chapitre 2

Approche

Dans ce chapitre, nous présentons nos choix par rapport à la théorie expliquée et introduisons les différents éléments dont nous aurons besoin pour mettre en place concrètement des tests adaptatifs.

2.1 Choix dans la théorie

Pour chacune des étapes de la figure 1.1 de la page 6 nous avons besoin de faire des choix dans les modèles statistiques possibles que nous avons présenté dans le chapitre 1. Nous présentons chacun de ces choix ici en fonction de l'étape du test.

2.1.1 Création de la banque d'items

Pour la création de la banque d'items, grâce aux statistiques sur les étudiants et les questions que nous détenions déjà, un algorithme d'espérance-maximisation était la voie à suivre pour évaluer au mieux les paramètres des questions. L'algorithme MML brièvement introduit dans la section 1.4.2 est considéré dans la littérature comme le modèle le plus général (il peut manipuler des patterns de réponse parfaits et est applicable à différents types de modèles IRT). Il est complexe à calculer mais des algorithmes pour ce faire sont communément implémentés [13].

2.1.2 Première étape

Dans notre modèle, nous faisons l'hypothèse que le score moyen des items du test est le score demandé pour la réussite du test (correspondant à 10/20 dans la majorité des tests linéaires). C'est-à-dire qu'une question avec un score moyen (calculé auparavant dans la banque d'items) sera une question de difficulté moyenne dans le test. Nous faisons aussi le choix de ne pas prendre en compte le niveau des

étudiants passant actuellement le test (par rapport à un niveau moyen pour le test) pour désigner la première question.

Pour le choix de la première question à poser à un étudiant, nous avons décidé de choisir la même première question pour tous les étudiants débutant le test. Cette solution allie simplicité et efficacité tout en permettant une certaine équité pour les étudiants. Deux étudiants répondant de la même façon aux questions devraient au final obtenir le même score. Si nous démarrions avec deux questions différentes, la suite de questions serait aussi différente comme leur score n'évoluerait pas de la même façon.

2.1.3 Étape d'évaluation

Pour cette étape, nous voulions choisir une alternative permettant de poser différents types de questions (QCM, QRM, question de programmation, question à réponse unique) tout en sachant qu'il serait très compliqué d'utiliser des questions ouvertes comme l'évaluation de la réponse de l'étudiant doit être automatique.

Pour mettre à jour le niveau de l'étudiant, il nous semblait logique d'utiliser une méthode fonctionnant avec des items dichotomiques et polytomiques. Le modèle de Bayes s'est imposé comme celui avec la plus grande précision.

Pour choisir la nouvelle question, nous avons décidé d'utiliser la méthode MFI. Celle-ci nous semblait être un bon compromis entre complexité calculatoire et précision du choix de la question en plus d'être disponible pour les questions polytomiques. Cette méthode est complexe mais nous savions qu'elle était déjà implémentée dans certains packages prévus pour les tests adaptatifs, car c'est la plus utilisée [13]. Choisir une question déterminée par le MFI plutôt qu'une autre technique minimise au final le nombre de questions posées dans le test.

2.1.4 Étape de vérification des conditions d'arrêt du test

Puisque l'un des objectifs principaux est de minimiser le nombre de questions posées tout en garantissant une évaluation au plus proche du niveau réel de l'étudiant, nous avons décidé d'utiliser deux règles en coordination : La règle de précision et la règle d'information (voir section 1.3.4).

Comme nous choisissons à chaque étape la question qui réduit le plus l'intervalle de confiance (et donc la question qui précise le plus le niveau de l'étudiant) grâce à la méthode MFI, nous minimisons le nombre de questions pour que la règle de précision devienne le plus rapidement vraie pour chaque étudiant.

Bien que nous choisissons la question qui réduit le plus l'intervalle de confiance, il se peut que la quantité d'information de cette question ne soit quand même pas suffisante pour avoir un impact significatif. La règle d'information arrête alors le test.

Nous avons également fait le choix de ne pas utiliser la règle de longueur car il nous était compliqué de trouver une valeur fixe pertinente sans données expérimentales.

2.1.5 Étape d'arrêt du test

Pour choisir le score final de l'étudiant, nous faisons le choix de garder la valeur qui se trouve au milieu de l'intervalle de confiance. Choisir la borne supérieure permettrait de favoriser l'étudiant en tout point (meilleur score final). Choisir la borne inférieure rendrait une note plus sévère (score final moins élevé). Nous avons décidé de rester neutre et de choisir l'entre deux. De plus, il s'agit du point qui en moyenne sera le plus proche du réel niveau de l'étudiant.

2.2 Outils nécessaires

Nous allons également avoir besoin de plusieurs outils pour pouvoir concrétiser ces tests. Nous présentons ici ces différents besoins pour obtenir un test adaptatif utilisable.

Interface de réponse Premièrement, il faut mettre à disposition des étudiants une interface avec laquelle ils peuvent interagir et qui leur fournit des questions ainsi que la possibilité d'y répondre. Cette interface doit également être capable de corriger automatiquement la réponse de l'étudiant puisque cette correction doit servir à juger le niveau de l'étudiant pour lui fournir une question suivante plus pertinente.

Implémenter les statistiques Deuxièmement, nous avons présenté les différents modèles statistiques que nous avons choisis pour les différentes étapes du test adaptatif, mais il faut également les intégrer dans notre solution sous forme de fonctions appelables par notre interface de réponse.

Un lien entre les statistiques et l'interface de réponse Finalement, nous avons besoin d'un moyen pour permettre une discussion entre l'interface de réponse et l'implémentation des statistiques. Ce canal de discussion doit permettre aux deux parties d'envoyer des informations, il faut que les communications puissent se faire dans les deux sens.

Chapitre 3

Implémentation

Dans ce chapitre, nous détaillons nos implémentations. Nous allons tout d'abord commencer par expliquer le fonctionnement de la plateforme sur laquelle fonctionne notre test adaptatif, INGInious. Ensuite, nous allons expliquer la partie statistique développée via le langage `R`. Finalement nous allons discuter de la façon dont nous avons fait interagir les deux via une API REST.

3.1 Structure globale

Pour comprendre la structure que nous avons mis en place, la figure 3.1 en montre une représentation. Chaque élément sera détaillé dans ce chapitre. Pour résumer, nous pouvons dire qu'INGInious (une plateforme d'exerciseur), travaille de pair avec `R` et la librairie `catR` pour fournir à l'étudiant des questions pertinentes (d'après la théorie des tests adaptatifs) tout en fonctionnant de manière transparente vis-à-vis de l'étudiant. Grâce à une connexion via l'`API REST`, nos différents agents communiquent entre eux pour se fournir l'un l'autre les informations manquantes. Dans le schéma 3.1 nous vous présentons comment INGInious [1] et `R` communiquent et dans quelles situations ces échanges d'informations sont nécessaires.

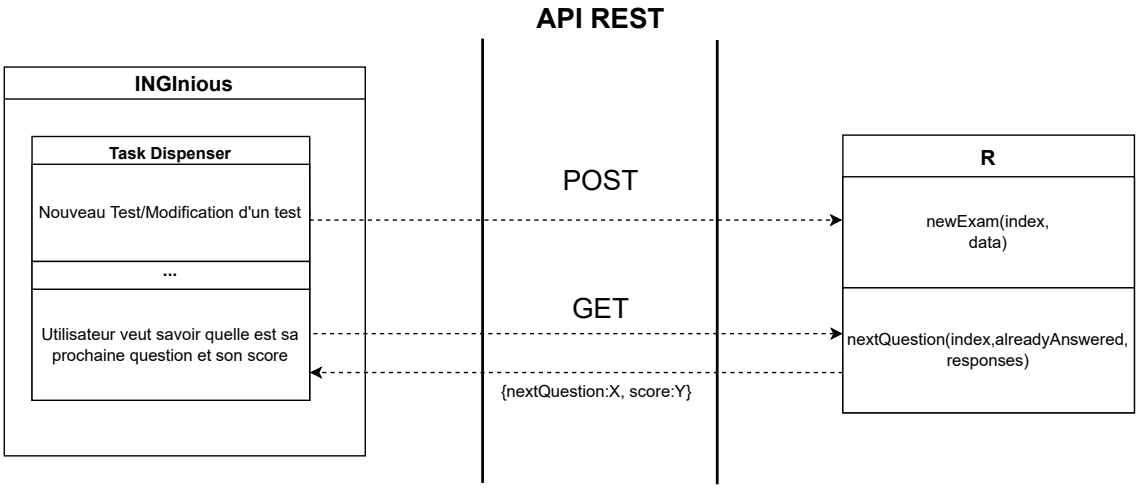

FIGURE 3.1 – Schéma du comportement interne

3.2 INGInious

Tout d'abord, il est important pour la compréhension de la suite d'en savoir un peu plus sur INGInious que nous utilisons. Cet outil répond à notre premier besoin dans les outils nécessaires, détaillés dans le chapitre 2, à savoir une interface de réponse qui permet aux étudiants de répondre à des questions qui leurs sont proposées et de les corriger automatiquement. Il s'agit d'une application web de type "exerciseur", qui propose à des professeurs de créer des tâches pour les étudiants. Ces tâches peuvent être de différents types tels que des exercices de programmation, des questions à choix multiples, des questions ouvertes, etc. L'application donne aussi la possibilité aux professeurs d'implémenter les corrections pour que celles-ci soient automatiquement exécutées. Chacun d'entre eux dispose d'un ou plusieurs `cours` dans lesquels sont présentes une ou plusieurs `tâche(s)` qui sont les questions posées à l'étudiant. INGInious est développé en Python et utilise entre autres une base de données mongoDB ainsi que la technologie des Dockers. Les enseignants ont donc la possibilité de programmer ou de définir une correction d'exercice, que celui-ci soit un simple QCM (Questionnaire à Choix Multiples), un exercice de programmation ou encore une question algorithmique complexe. Ce qui nous intéresse principalement ici, c'est cette capacité à donner un score automatiquement à l'étudiant, car c'est un élément indispensable aux tests adaptatifs. Nous allons intégrer notre travail dans cette application Open Source et nous allons détailler le fonctionnement de notre implémentation.

3.2.1 Structure d'INGInious

Pour comprendre où se situe nos ajouts, nous allons d'abord introduire la structure actuelle d'INGInious. Pour des informations plus complètes, veuillez vous référez à la documentation d'INGInious [2].

La plateforme a deux facettes, la partie administrateur et la partie utilisateur. Dans un contexte académique, on peut supposer que les administrateurs seront les professeurs et les utilisateurs seront les étudiants. Les étudiants se connectent à la plateforme et ont une liste de `cours` administrés par différents professeurs. Chacun de ces cours est composé d'une ou plusieurs `tâche(s)` assignée(s) à l'étudiant.

Les administrateurs de leur côté doivent créer ces questions, implémenter une façon d'automatiser les corrections et enfin les assigner aux étudiants. Habituellement, les professeurs auront une liste de questions qu'ils mettront à disposition des étudiants avec éventuellement des délais pour certaines ou un nombre de soumissions maximum pour d'autres, etc. Chaque question est modulaire et permet aux professeurs de les paramétrer comme ils le souhaitent mais, par défaut, les étudiants se voient attribuer toutes les questions du cours (moyennant certaines dates d'accessibilité définies par les professeurs). Les administrateurs ont aussi la possibilité de basculer la distribution de questions en mode `aléatoire` mais il n'y a, pour l'instant, aucun autre mode de distribution que ces deux-là (complet ou aléatoire). C'est dans cet endroit d'INGInious que va se greffer notre implémentation. Nous allons ajouter un nouveau moyen de distribuer les questions aux étudiants.

3.2.2 Le plugin

INGInious a un système de gestion de plugin qui permet à des contributeurs de rajouter des éléments de distribution de questions et de pouvoir changer les interfaces de ces rajouts. Nous avons décidé de réaliser notre ajout sous forme de plugin. La figure 3.2 montre les différents fichiers que nous allons utiliser. Les fichiers HTML de `admin` ont notamment été changés pour rajouter des interfaces aux administrateurs ou pour importer des questions. Les fichiers de `student` ont été modifiés pour ajouter une visualisation du score pour l'étudiant. Le fichier `__init__.py` est la base de notre implémentation car il articule tout le mécanisme des tests adaptatifs (figure 1.3 de la page 14) .

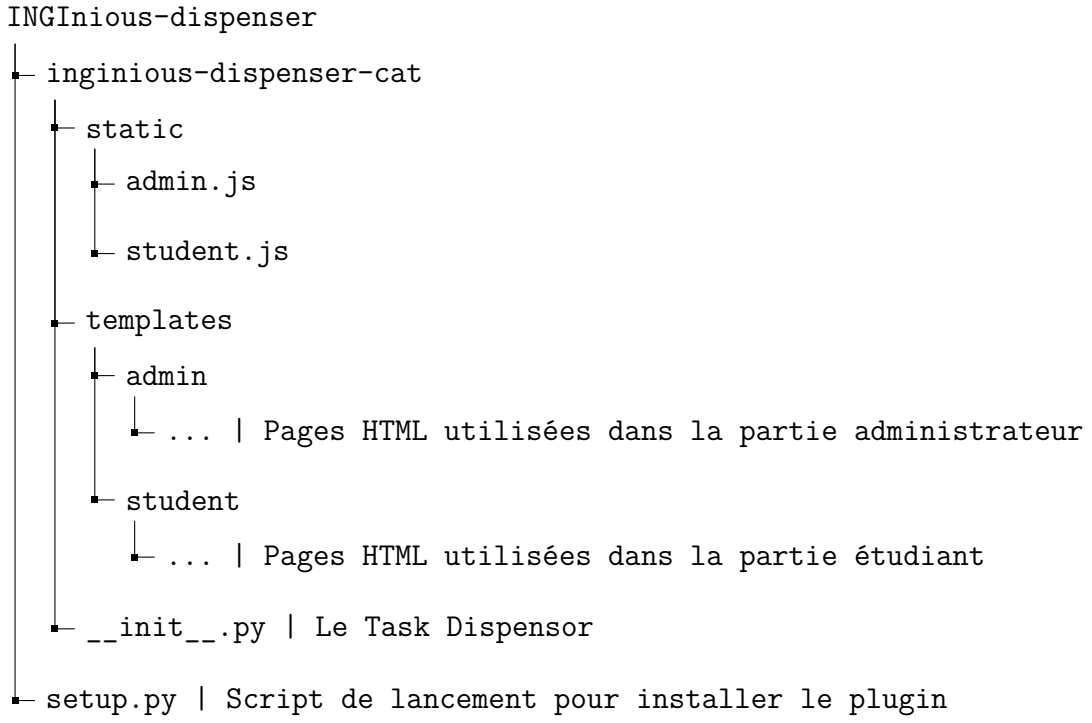

FIGURE 3.2 – Structure des fichiers du plugin INGInious

3.2.3 Task Dispenser

L'un des modules d'INGInious est appelé le Task Dispenser. L'exerciseur étant codé en Python, il s'agit en fait ici d'un objet qui a pour rôle de fournir les tâches, ou plus précisément les questions créées par les professeurs, attribuées à chaque étudiant ainsi que la façon dont INGInious doit afficher ces tâches. C'est donc le Task Dispenser qui va informer le site que l'étudiant X doit répondre aux questions Y et Z par exemple. Nous avons donc créé un plugin qui va implémenter un nouvel objet de type `TaskDispenser` qui aura comme rôle de fournir aux étudiants les questions qui leur sont associées en fonction de ce à quoi ils ont déjà répondu et de comment ils y ont répondu, en suivant le principe du test adaptatif. Le Task Dispenser a aussi un second rôle qui est d'afficher correctement les questions à l'étudiant en lui empêchant de revenir aux questions précédentes (celles auxquelles il a déjà répondu) et également de lui afficher son score actuel calculé par le test adaptatif.

Dans un premier temps, le Task Dispenser doit connaître les questions qu'il a à sa disposition et pour cela une page administrateur est configurée sur INGInious

pour permettre aux professeurs d'importer des questions depuis un autre cours, sur lequel des réponses (bonnes ou mauvaises) ont déjà été fournies par les étudiants. Cette importation permettra de garder les statistiques des questions dont nous auront besoin pour l'évaluation des paramètres des questions (comme montré sur la figure 1.3 de la page 14). Une fois ces questions importées et sélectionnées par l'administrateur, celui-ci n'a plus qu'à cliquer sur un bouton "Appliquer les changements" pour que toutes les questions soient envoyées au Task Dispenser. Celui-ci va alors construire un grand tableau comprenant les questions et les réponses des étudiants à celles-ci, pour pouvoir appliquer l'algorithme d'Espérance-Maximisation (étape bleue de la figure 1.3).

Une fois les questions importées par l'administrateur, les étudiants peuvent désormais se connecter sur la page d'affichage des questions. Notre Task Dispenser va alors avoir comme rôle d'évaluer, pour chaque étudiant, où il en est dans son test et quelles sont les prochaines étapes que l'étudiant va devoir réaliser (étape d'évaluation dans la figure 1.1 de la page 6). Pour cela, les statistiques entrent en jeu puisqu'il faut calculer plusieurs éléments tels que le score actuel de l'étudiant, la précision de son score, la prochaine question à lui poser, etc. Pour cela, nous avons décidé d'utiliser le langage `R` pour bénéficier de ses librairies. Notre Task Dispenser doit communiquer avec `R` pour avoir accès aux informations dont il a besoin. Nous en parlerons dans la section 3.5 après avoir discuté de notre implémentation en `R`.

Ajouts dans la base de données Comme dit précédemment, INGInious utilise une base de données mongoDB et dispose d'un certain nombre de collections. Nous avons eu besoin de rajouter deux collections pour que le Task Dispensor puisse sauvegarder et réutiliser plusieurs informations dont il a besoin.

1. `cat_info` {"courseidfrom","courseid"} : Retient d'où ont été importées les questions
2. `cat_score` {"courseid","username","finalscore","nombrequestions"} : Retient le parcours de l'étudiant

3.3 `R` et `catR`

Pour implémenter tout ce qui a été introduit précédemment au niveau des tests adaptatifs, notamment dans les sections 1.3 et 1.4 nous avons décidé d'utiliser un package existant, dans le langage de programmation `R`, nommé `catR` [12],[13]. Ce package nous permet de répondre à notre seconde nécessité dans le chapitre 2 qui demandait de pouvoir implémenter et gérer les modèles statistiques choisis. Cet outil nous permet, à condition de lui fournir les informations adéquates, de gérer les

différentes étapes d'un test adaptatif présentées dans la section 1.3. `catR` n'a pas de mémoire (plus précisément, il est dit *stateless*) donc INGInious doit se charger de retenir l'état de chaque étudiant et de chaque examen pour lui. Ces paramètres devront néanmoins lui être fournis sous une structure particulière pour qu'il puisse les interpréter correctement. Ces paramètres sont les suivants :

1. La banque d'items, qui contient chaque question avec ses statistiques associées
2. La liste des questions auxquelles l'étudiant à déjà répondu
3. Le score `theta` qui correspond au score estimé de l'étudiant
4. La précision `semTheta` qui correspond à la précision du score `theta`

Ces paramètres seront utilisés dans les appels aux fonctions `catR`. Nous allons lister les fonctions que nous utilisons dans ce package, il est à noter que `itemBank` correspond à la banque d'items, `theta` correspond au score, `alreadyAnswered` correspond aux questions auxquelles l'étudiant à déjà répondu, `itemBankPartial` correspond à une partie de la banque d'items où seules les lignes des questions déjà posées se trouvent et `responses` correspond à un vecteur de réponses de l'étudiant (des valeurs entre 0 et 1 correspondant au score qu'il a obtenu pour ces questions).

1. `startItems(itemBank)` : Fonction qui nous renvoie la première question à poser à un étudiant.
2. `nextItem(itemBank,theta,alreadyAnswered)` : Fonction qui nous renvoie la prochaine question pertinente à poser à l'étudiant en considérant son niveau actuel et les questions auxquelles il a déjà répondu.
3. `thetaEst(itemBankPatial,responses)` : Fonction qui évalue le niveau de l'étudiant en fonction de ses réponses.
4. `semTheta(theta,itemBankPartial,responses)` : Fonction qui évalue la précision de `theta`.

La banque d'items La banque d'items est, comme expliqué dans la section 1.3, une base de données reprenant les statistiques de chaque question. Pour `catR`, la banque d'items doit se présenter sous la forme d'une matrice $n \times 4$ où n est le nombre de questions. La ligne i de la matrice, montre les statistiques de la question i. Comme la matrice est composée de 4 colonnes, nous avons donc 4 valeurs à remplir qui correspondent aux 4 paramètres des modèles IRT (Item Response Theory) présentés dans la section 1.4 à savoir la discrimination, le niveau de la question, la pseudo-supposition et le paramètre d'inattention. Nous avons décidé de partir sur un modèle logistique à 2 paramètres (2PL) comme indiqué dans la

section 1.4.2 car c'est ce que calcule notre modèle tout en ayant un bon compromis entre efficacité et accessibilité. Nous aurons alors uniquement besoin des deux premières colonnes qui seront dans l'ordre la discrimination de la question et sa difficulté (son score). Les deux colonnes restantes seront mises à leurs valeurs par défaut, à savoir respectivement 0 et 1. La figure 3.3 donne un exemple de banque d'items que nous utiliserons. Dans cette figure, chaque ligne correspond aux statistiques d'une question et chaque colonne fait référence à l'un des paramètres de l'IRT.

Discrimination	Difficulté	Pseudo-supposition	Paramètre d'inattention
...	...	...	...
0.154	0.581	0	1
−1.245	2.47	0	1
1.4269	−1.25	0	1
−0.1447	0.9874	0	1
...	...	...	...

FIGURE 3.3 – Exemple de banque d'items

Cette matrice restera sauvegardée en RAM par `R` pour ne pas devoir la recalculer à chaque question de chaque étudiant. Nous y reviendrons plus en détail dans la section 3.5.

Listes des questions auxquelles l'étudiant à déjà répondu Pour savoir quelles questions poser à l'étudiant, il faut garder en mémoire celles auxquelles il a déjà répondu. En effet, lui reposer une même question ne donnera aucune information supplémentaire et lui fera juste perdre son temps, puisqu'on suppose ici qu'aucun feedback ne lui sera attribué et donc que sa réponse sera identique. `catR` nous demande donc de les retenir pour les lui fournir sous forme de vecteur d'indices correspondant aux indices des questions auxquelles l'étudiant a déjà répondu dans la banque d'items. Cette liste nous permet de créer la matrice `itemBankPartial` qui correspond à une partie de la banque d'items dans laquelle ne se trouvent que les questions auxquelles l'étudiant a déjà répondu (exemple d'utilisation d'`itemBankPartial` dans les fonctions décrites plus haut).

Le score `theta` À chaque itération du test, `catR` calcule pour nous un score nommé `theta` via son estimateur par défaut, à savoir le modèle de Bayes. `Theta` est l'approximation statistique actualisée de l'étudiant à chaque étape d'évaluation du test (voir partie verte de la figure 1.3 et de la figure 1.1). Il permet d'en déduire un score en suivant une loi normale centrée réduite. Ce score est à la fois un paramètre

et un résultat puisqu'il est utile pour calculer `semTheta` (voir fonctions décrites plus haut) et donne également le score de l'étudiant. Concrètement, en `R`, pour avoir le score sur 100 de l'étudiant, il faut faire `pnorm(theta)*100`.

La précision `semTheta` Pour chaque itération du test, `catR` va également calculer une précision nommée `semTheta`. Cela correspond à l'erreur standard estimée de l'habilité estimée. Avec cela, nous pouvons calculer un intervalle de confiance (de 95% par exemple) où se situe le niveau réel de l'étudiant. Pour donner un exemple d'utilisation, si nous voulons un intervalle de confiance de 95%, nous allons multiplier le `semTheta` par 1.96 (valeur correspondant à 95% dans la loi normale réduite) et l'intervalle de confiance sera alors : $[\texttt{theta} - \texttt{semTheta} * 1.96; \texttt{theta} + \texttt{semTheta} * 1.96]$. Pour trouver ce 1.96, il faut utiliser le quantile de la loi normale valant 0.975 ($= \frac{0.95+1}{2}$). Nous devons trouver x où $\phi(x) = 0.975$, la fonction de répartition de la loi normale est donnée par l'équation 3.1 et nous avons utilisé sa réciproque pour trouver notre valeur.

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\inf}^{x} e^{\frac{-1}{2}t^2} \tag{3.1}$$

Nous savons que le niveau de l'étudiant ne va pas sortir d'entre ces deux bornes avec une confiance à 95%. C'est ce `semTheta` qui va nous aider à savoir si nous devons arrêter un test ou le continuer. En effet, si nous nous rendons compte que la précision est suffisamment élevée (`semTheta` petit) nous pouvons en conclure que nous avons assez d'informations pour calculer un score final de l'étudiant. Au contraire, si la précision est trop faible, le score de l'étudiant pourrait être biaisé et nous allons poursuivre le test. A noter que cela ne doit pas être la seule condition d'arrêt mais elle est néanmoins la plus importante comme explicité dans la section 1.3.4 et dans le chapitre 2. Cette précision est donc quelque chose que nous voulons maximiser tout en gardant un nombre raisonnable de questions.

Choix de la prochaine question L'un des éléments clé du test adaptatif est de choisir la prochaine question à laquelle l'étudiant doit répondre. Nous avons fait le choix d'utiliser les méthodes pré-intégrées au package `catR` et de garder celle définie par défaut, à savoir la méthode du maximum d'information de Fisher (MFI). Nous avons effectués plusieurs essais tels que décrits plus tard pour le graphique 3.5 et nous nous sommes rendu compte que la MFI était la plus intéressante pour nous comme expliqué précédemment dans la section 1.4.1.

Les conditions d'arrêt Une fois que nous avons toutes ces fonctions, nous devons savoir à quel moment arrêter le test (partie jaune sur la figure 1.3 de la page 14). Comme expliqué dans la section 1.3.4, plusieurs facteurs peuvent jouer dans le choix de stopper le test ou non. L'un des plus importants est celui de la précision du test, représenté par notre `semTheta` ici. Il faut donc choisir une valeur qui maximisera la précision et minimisera le nombre de questions, c'est donc un compromis entre les deux. Une autre condition d'arrêt que nous avons implémentée est l'arrêt du test si l'application n'a plus de question pertinente à proposer à l'étudiant. Nous avons décidé de définir cette non-pertinence par le fait qu'une question ne soit pas dans l'intervalle de confiance du niveau de l'étudiant (plus ou moins une marge de confiance). Si la question proposée par le MFI n'est pas déclarée comme pertinente, nous arrêtons le test.

Exemple concret d'utilisation

Avec tous ces paramètres, nous pouvons dès lors illustrer un comportement de test adaptatif. Sur la figure 3.4 nous pouvons voir le déroulement avec les paramètres affichés. Les valeurs sur la surface bleue correspondent au `semTheta` tandis que l'amplitude de la zone bleue correspond à l'intervalle de confiance calculé à 95% (pour rappel, il s'agit donc de `theta` $\pm$ `semTheta` * 1.96). La ligne rouge montre le niveau de difficulté de chaque question qui a été posée et pour savoir si l'étudiant y a bien répondu il suffit de regarder si la question suivante a un niveau de difficulté plus élevé (du moins pour cet exemple). Pour ce test, nous avons décidé de fixer le seuil d'arrêt lorsque l'intervalle autour de `theta` atteint 0.3 ou moins. On peut aussi préciser que l'étudiant passant le test avait un niveau moyen (aux alentours de 50% sur un test linéaire) pour cet examen.

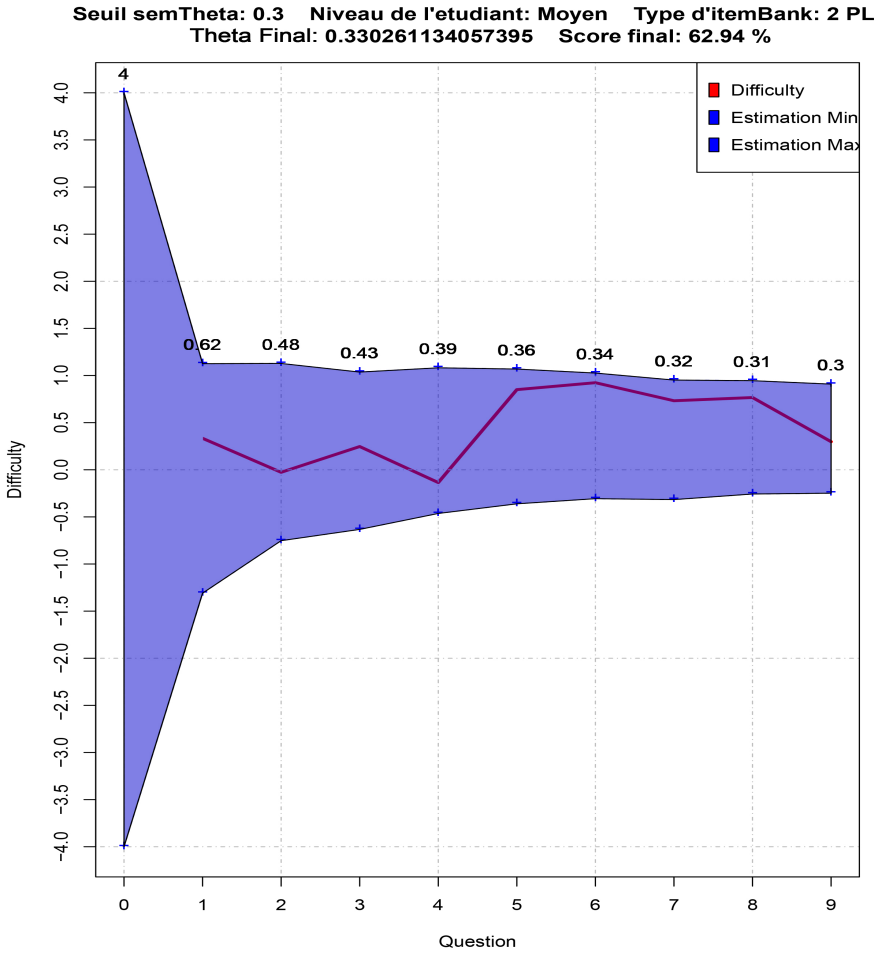

FIGURE 3.4 – Comportement d'un test adaptatif

3.4 Le modèle 2PL

Comme expliqué dans la section 1.4, il existe plusieurs paramètres qui peuvent influer sur la précision (`semTheta`) ainsi que sur le choix de la prochaine question. Nous avions expliqué que le score d'une question ne dépend pas seulement du taux de réussite mais de plusieurs autres facteurs, dont nous allons discuter. Concrètement, il s'agit d'utiliser (ou pas) les différentes colonnes de la banque d'items. Nous avons décidé de choisir un modèle 2PL pour avoir un compromis entre simplicité et efficacité. Nous utilisons donc deux paramètres : le taux de réussite et la discrimination. Le modèle de Rasch [16] (1PL) se base sur le fait que toutes les questions ont la même discrimination (première colonne de la banque d'items 3.3)

ce qui, en pratique, est rarement le cas. Si nous reprenons notre cas pratique (INGInious), les étudiants ont généralement le choix de réaliser ou non les exercices comme ceux-ci sont souvent utilisés comme complément de cours et ne sont pas toujours obligatoires. On peut dès lors rapidement détecter un biais, et donc une discrimination, en supposant que les bons étudiants vont avoir tendance à essayer tous les exercices, même les plus difficiles, alors que les moins bons étudiants risquent de ne même pas essayer les exercices compliqués. On se retrouve alors avec des exercices compliqués qui ont un score de réussite élevé du fait de cette tendance. Nous avons décidé d'introduire ce principe dans notre modèle pour obtenir plus d'informations via la réponse d'un étudiant ce qui explique notre choix du modèle 2PL. Cette discrimination est calculée via un package `R` nommé `ltm` [17]. Pour calculer cette discrimination, nous avons besoin d'une matrice qui reprend les réponses de chaque étudiant pour chaque question. Par exemple, si nous avons en tout 150 étudiants qui ont répondu à 10 questions, nous aurons une matrice $n * m$ ($n = 150$ et $m = 10$) où la ligne correspond à un étudiant et la colonne correspond à une question. La valeur à cet indice est le score de l'étudiant à cette question. Un détail qu'il est cependant important de préciser est que cette valeur dans la matrice doit être binaire (soit 0 soit 1) pour que la méthode de `ltm` fonctionne. On considère que l'étudiant a donc soit réussi soit raté une question ce qui implique que, pour le calcul du discriminant, il n'y a ici pas de notion polytomique.

Une fois implémenté, nous pouvons comparer le gain d'information obtenu en générant un graphique. Ce graphique montre l'évolution du nombre de questions nécessaire pour atteindre une certaine précision de `semTheta`, par une série d'étudiants (fictifs) en fonction du seuil de précision que nous imposons et du modèle utilisé (1PL et 2PL). Ce que nous cherchons à montrer avec ce graphique est l'impact qu'aura ce gain d'information sur le nombre de questions posées en moyenne. Pour pouvoir générer ces graphiques, il nous faut générer un vecteur de difficulté de questions ainsi qu'un vecteur de niveau d'étudiants (qui ont une probabilité p de bien répondre à une question de difficulté y). Une fois cette matrice créée, nous pouvons calculer une moyenne de difficulté ainsi qu'une discrimination nécessaire aux modèles 1PL et 2PL. Il est maintenant possible de créer le graphique de comparaison des deux modèles 3.5.

En comparant les résultats entre 1PL et 2PL dans la figure 3.5, on observe nettement que les valeurs pour le modèle 2PL sont, dans tous les cas, plus intéressantes car le nombre de questions posées (en fonction de la précision) est toujours sensiblement moins élevé. Cela va nous permettre de descendre le seuil de `semTheta` tout en gardant un nombre de questions raisonnable. Baisser ce seuil permet d'avoir une meilleur précision quant au score final de l'étudiant et nous pouvons en conclure

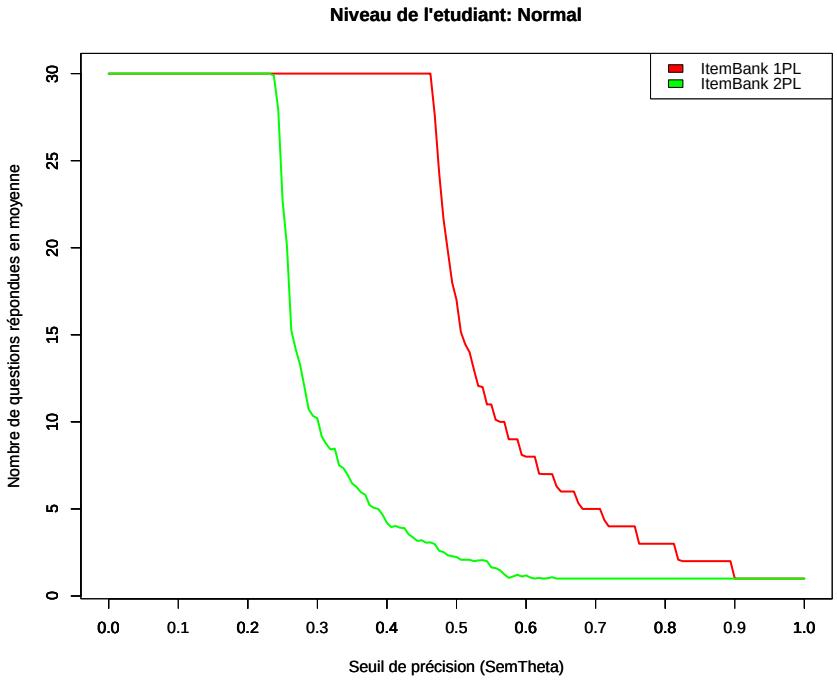

FIGURE 3.5 – Comparaison 1PL et 2PL

qu'il est préférable (quand cela est possible) d'utiliser un modèle 2PL. Nous voyons également assez facilement le problème qui se pose quant à la sélection du seuil le plus pertinent sur ce schéma : nous voulons minimiser le nombre de questions tout en minimisant le seuil de précision, pour le modèle 1PL cette minimisation pourrait se faire aux alentours de 0.5 alors qu'elle serait plutôt égale à 0.3 dans le cas du 2PL.

3.5 API REST

Pour établir une connexion entre INGInious (le Task Dispenser) et `R`, nous avons créé une API REST. Une API REST est une méthode pour faciliter la communication entre plusieurs interfaces grâce à des protocoles web. Cette API REST répond à notre troisième et dernière nécessité présentée dans le chapitre 2. En effet, cette API REST va nous permettre de faire le lien entre l'interface de réponse (à savoir INGInious) et l'implémentation des modèles statistiques (à savoir `catR`) pour que ceux-ci puissent communiquer entre eux. Le principe est assez simple, quand le Task Dispenser a besoin d'une information il envoie toutes

les données nécessaires sous format `JSON` à `R` qui lui répond (sous format `JSON` également). Pour ce faire, nous avons deux cas où ces deux parties discutent via des adresses URL, nous pouvons voir comment cela s'articule sur la figure 3.1 du début de chapitre.

1. `newExam` : Lorsqu'un administrateur crée le test ou rajoute des questions car `R` doit calculer les scores de chaque question et former sa banque d'items.
2. `nextQuestion` : Lorsqu'un utilisateur répond à une question et qu'INGInious doit lui fournir la suivante.

Lorsque le test est créé, le Task Dispenser doit envoyer un appel à `newExam` avec deux arguments dans son `JSON`. Le premier (`data`) doit être une matrice de réponses pour chaque question telle que présentée dans la section 3.4 pour générer les discriminations. Le second (`index`) correspond à l'index auquel `R` doit stocker ses informations pour pouvoir, par la suite, communiquer à quel étudiant correspond quelle banque d'items.

Ensuite lorsque chaque étudiant répond à une question (ou commence le test pour la première fois) le Task Dispenser à besoin de savoir quelle question poser. Il va demander à `R` via `nextQuestion` en lui donnant du contexte via ses paramètres. Il y a ici 3 paramètres à donner : l'index auquel est stockée la banque d'items, une liste d'identifiants des questions auxquelles l'étudiant à déjà répondu et qu'il ne faut plus sélectionner et finalement une liste de scores (entre 0 et 1) correspondant à chaque question à laquelle l'étudiant à déjà répondu. Par exemple, si l'étudiant passe le test `"foo"` et qu'il en est à sa troisième question, un appel type du Task Dispenser ressemblerait à : `{"itemBankID":"foo","alreadyAnswered":{5,7},"responseList":{0.57,0.42}}`. Il ne s'agit donc pas tout à fait d'un fonctionnement *stateless* puisque `R` va calculer ce dont il a besoin puis le stocker dans une matrice pour l'utiliser plus tard (qu'il gardera donc dans sa RAM). Si on décide d'appeler `nextQuestion` avec l'index "XY" nous aurons donc besoin d'avoir une première fois appelé `newExam` avec l'index "XY". Cette fonction a, quant à elle, une valeur de retour. Dans cette valeur, deux arguments sont présents : `index` et `score`. `Index` correspond à l'index de la question, dans la banque d'items, qu'il faudra poser à l'étudiant. `Score` montre le niveau actuel estimé de l'étudiant (c.f. 3.3). Ces deux valeurs peuvent être égales à -1 quand celles-ci n'ont pas lieu d'être. Respectivement, quand il n'y a plus de questions à poser (fin du test) et quand l'étudiant n'a pas encore commencé le test. `Index` sera alors égal à -1 lorsqu'il n'aura plus de question pertinente à poser à l'étudiant (ou que la précision est assez élevée) alors que `score` sera égal à -1 tant que l'étudiant n'aura pas au moins répondu à une première question. Il est à noter que le score ici se situe donc au milieu de notre intervalle de confiance

(comme discuté dans la section 2.1.5) mais cela pourrait être ajusté en imaginant par exemple donner la borne inférieure (de moins bonnes notes en général) ou la borne supérieure (de meilleures notes en général).

En conclusion, cette API REST interagit via deux adresses URL, `newExam` et `nextQuestion`. La première sera une requête POST alors que la seconde sera une requête GET qui récupérera la prochaine question à poser et le score de l'étudiant. La figure 3.6 montre l'utilisation de notre API REST.

POST : `/newExam`

index : Une référence au test (peut être un `String` ou un `Integer`)

data : Matrice (2 dimensions) contenant les réponses des étudiants

GET : `/nextQuestion`

index : Une référence au test (peut être un `String` ou un `Integer`)

`alreadyAnswered` : Un vecteur contenant les questions auxquelles l'étudiant a déjà répondu. Vide ou -1 si début du test.

`responses` : le score de l'étudiant à chacune des questions de `alreadyAnswered`. (`responses`[5] est donc son score à la question `alreadyAnswered`[5]. Vide ou -1 si début du test.)

RETURN :

`nextQuestion` : L'index de la prochaine question à poser.

`score` : Le score actuel de l'étudiant. Peut être -1 si l'étudiant n'a pas encore de score

FIGURE 3.6 – API REST

Conclusion

Cette implémentation répond à la première question de recherche (figure 1 de la page 2) en montrant qu'il est possible de créer un système articulant les différents pans des tests adaptatifs.

Chapitre 4

Validation du prototype

Nous avons testé notre implémentation dans un contexte réel avec de vrais étudiants et un vrai cours donné sur la plateforme INGInious. Cela nous a permis de tester aussi bien notre partie graphique que notre implémentation des tests adaptatifs. Pour cela, nous avons eu la chance de pouvoir compter sur la collaboration de deux professeurs de l'UCLouvain : le professeur Yves Deville et le professeur Olivier Bonaventure. Ces deux professeurs nous ont donné accès respectivement aux questions du cours de `Calculabilité, logique et complexité` et du cours de `Réseaux informatiques`. Nous avons eu au total plus d'une centaine de participation et nous allons discuter de cette expérience dans ce chapitre.

4.1 Wooclap

Tout d'abord, il est important de discuter de la provenance des questions. Il se fait que nous n'avons pas trouvé de cours INGInious offrant un ensemble de questions déjà implémentées qui ne soient pas trop longues. En discutant avec le professeur Olivier Bonaventure, il nous a cependant proposé d'utiliser ses questions `Wooclap` [3].

Wooclap est un site internet permettant de réaliser des sondages et des Questions à Choix Multiples (QCM) et de les diffuser via un lien URL. Ces sondages restent disponibles pendant un intervalle de temps limité et sont fréquemment utilisés par les enseignants pour questionner les étudiants pendant les cours magistraux. Ce site permet notamment l'anonymat des réponses ce qui permet une réponse sans peur du jugement. Le professeur Olivier Bonaventure et le professeur Yves Deville utilisent justement ces questions lors de chacun de leurs cours magistraux. C'est une opportunité pour nous d'avoir alors accès à une liste de questions parcourant l'ensemble du cours tout en ayant des statistiques de réponses dessus.

L'inconvénient de cette méthode est que nous n'avons pas d'inter-dépendance entre les questions. Plus précisément, à cause de l'anonymat, nous ne savons pas quels étudiants ont bien répondu à une question X mais mal répondu à une question Y. Cela nous empêche dès lors d'utiliser notre modèle 2PL puisque celui-ci nécessite d'avoir cette inter-dépendance entre les questions pour pouvoir calculer la discrimination entre les questions (comme expliqué dans la section 1.4 et plus en détail dans la section 3.4). Nous avons donc dû basculer sur un modèle 1PL pour cette expérience ce qui implique une perte notable de précision (comme montré par le graphique 3.5 de la page 30). Il faudra dès lors garder en tête que les résultats peuvent être différents d'une autre expérience si celle-ci est réalisée avec des questions provenant directement d'INGInious qui nous permettrait d'utiliser un modèle 2PL.

4.2 Le sondage

Nous avons demandé aux étudiants de répondre à un sondage une fois le test fini. Ce sondage proposait de donner un avis sur le test adaptatif et demandait également aux étudiants de répondre s'ils estimaient avoir été correctement notés par le système, ou si ceux-ci pensaient mériter une meilleure note ou une moins bonne note. Grâce aux 99 étudiants qui ont répondu au sondage, nous pouvons nous faire une idée de la pertinence de notre modèle.

Dans ce sondage, 55 étudiants ont répondu avoir été bien notés, ce qui veut dire que plus de la moitié des étudiants estime que notre modèle a été pertinent dans leur cas. Dans le groupe des 44 étudiants restants, 29 ont estimé avoir été sur-évalués (donc mériter un score moins élevé) et 15 ont estimé avoir été sous-évalués (donc mériter un score plus élevé). Nous avons tracé le graphique 4.1 pour visualiser cette tendance tout en tenant compte de leur score final ainsi que du nombre de questions qui leur a été posé.

Le graphique 4.1 nous permet de nous rendre compte de deux points assez visibles. Le premier est que la majorité des étudiants estimant avoir été sous-évalués ont reçu une notre inférieure à 50% alors que la majorité des étudiants estimant avoir été sur-évalués ont reçu une note supérieure à 50%. La seconde tendance de ce graphique montre que les très bons étudiants, à qui nous avons attribué une note aux alentours de 100% ont dû répondre à un bien plus grand nombre de questions. Nous tentons dans les prochaines sections d'analyser ces deux points.

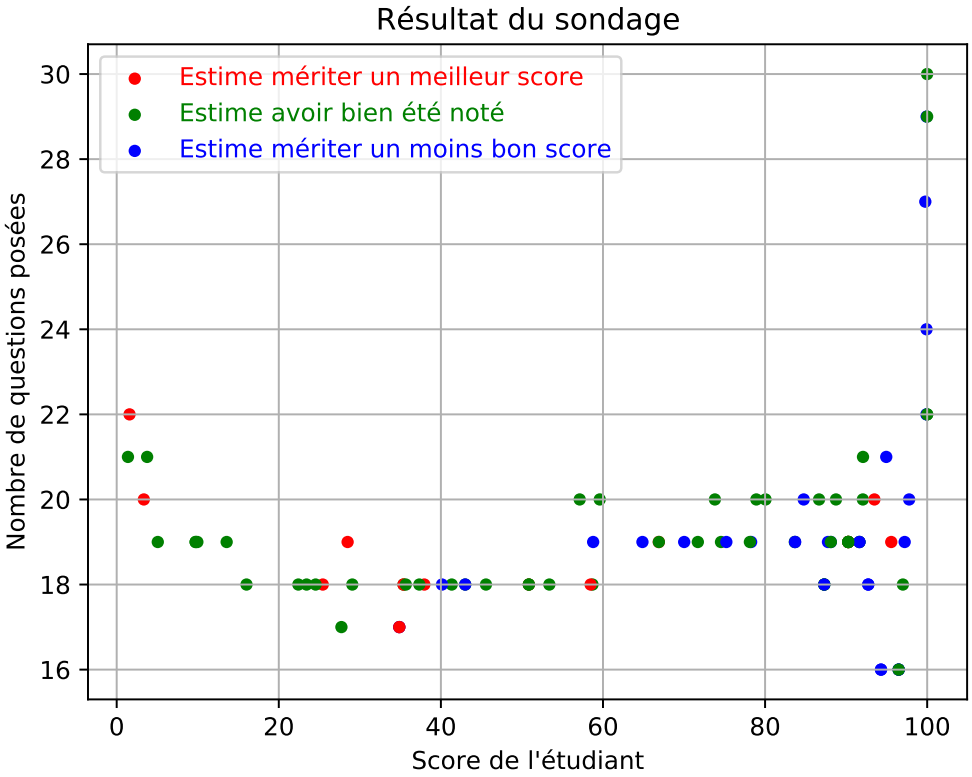

FIGURE 4.1 – Résultat du sondage aux étudiants

4.2.1 Les mauvaises évaluations

Pour essayer de comprendre pourquoi de nombreux étudiants ont estimé avoir été incorrectement noté, nous avons posé plusieurs hypothèses. Certaines sont d'ordre sociologique et nous ne les confirmerons pas ici mais nous pouvons quand même les énoncer.

La première hypothèse qui nous semble la plus intuitive est que les étudiants s'auto-évaluent mal. Soit ils souhaitent plus car ils pensent connaître la matière alors que ce n'est pas le cas, soit ils sont modestes et pensent alors mériter moins. Ce biais entre forcément en compte mais nous ne sommes pas en mesure de savoir avec quelle intensité. Une autre hypothèse est que les étudiants savent que les professeurs ont accès aux soumissions et au sondage et ont donc peur de se sentir confiant en disant qu'ils estiment effectivement mériter une note haute mais de

nouveau, il nous est impossible de vérifier cette hypothèse.

La dernière de nos hypothèses est que les étudiants peuvent avoir eu le sentiment d'avoir bien réussi le test du fait qu'ils avaient donné un plus grand nombre de bonnes réponses que de mauvaises, alors qu'ils n'ont en réalité réussi que des questions évaluées comme *faciles*. Inversement, certains étudiants peuvent avoir le sentiment d'avoir donné plus de mauvaises réponses que de bonnes réponses et pensent dès lors mériter une moins bonne note.

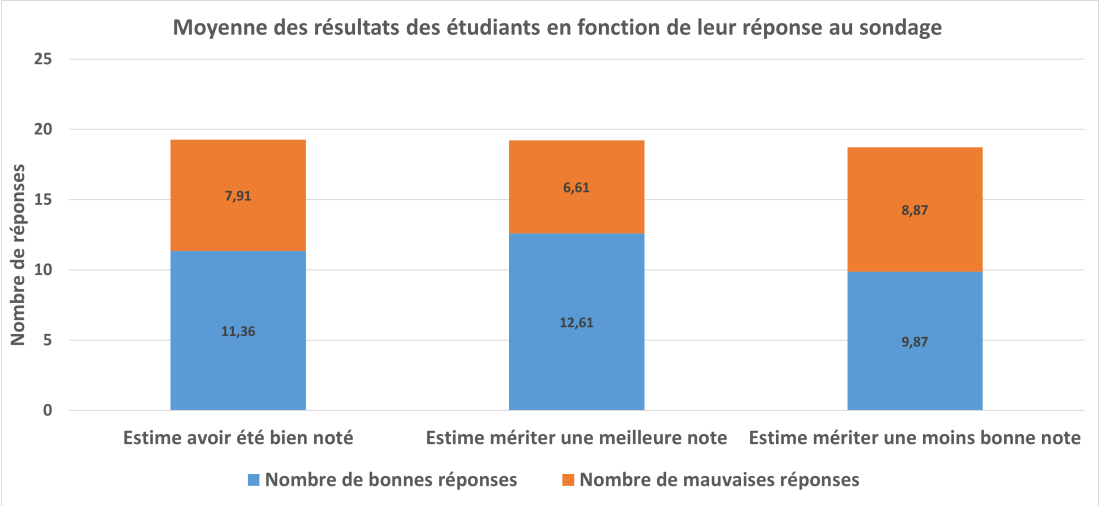

FIGURE 4.2 – Résultats des sondages

Le graphique 4.2 affiche la tendance du nombre de bonnes et mauvaises réponses des étudiants en fonction de leur avis et met en évidence la potentielle véracité de notre dernière hypothèse. En effet, nous pouvons constater que les étudiants estimant mériter une meilleure note ont plus de bonnes réponses que les étudiants qui estiment avoir été bien noté alors que les étudiants estimant mériter une moins bonne note ont donné plus de mauvaises réponses que les étudiants estimant avoir été bien noté. Le graphique 4.2 tend donc à démontrer que notre hypothèse est correcte et que ce biais doit être tenu en compte.

4.2.2 Un nombre plus élevé de questions

La seconde constatation du graphique 4.1 est que les très bons étudiants devaient répondre à un bien plus grand nombre de questions. On observe aussi une légère augmentation du nombre de questions pour les étudiants ayant un score très bas. Nous nous concentrerons sur les étudiants avec un meilleur score comme

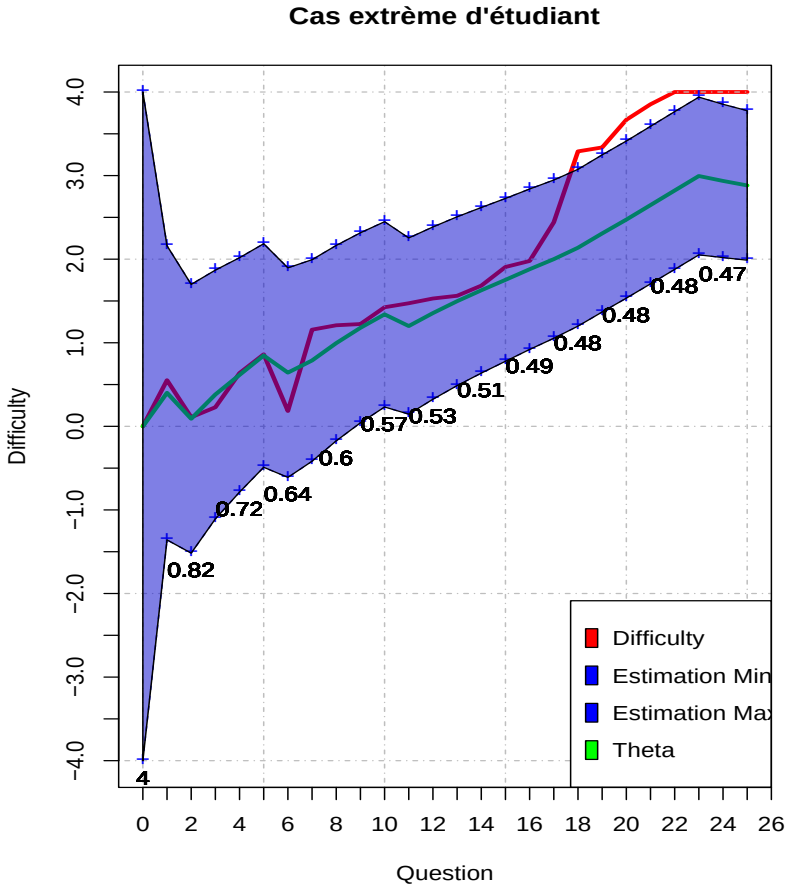

FIGURE 4.3 – Parcours extrême étudiant

l'augmentation est plus flagrante mais nous pouvons supposer une ressemblance entre les deux extrêmes à ce niveau. Pour comprendre ce phénomène, nous avons créé un graphique retraçant les variations des paramètres du test adaptatif de l'un de ces étudiants au score élevé pour en comprendre le comportement. Le graphique 4.3 nous montre ce comportement en affichant la difficulté de chaque question posée ainsi que le `theta` et l'intervalle de confiance `semTheta` à chaque étape du test. On peut voir une évolution linéaire du `theta` mais une évolution logarithmique de l'intervalle de confiance (le `semTheta`). Cette décroissance logarithmique peut être aperçue également dans notre précédent graphique 3.5 de la page 30.

Rappelons que nous utilisons un modèle 1PL ici. Il est tout à fait normal que certains étudiants arrivent à la limite de précision puisque le seuil de précision est défini arbitrairement. Celui-ci peut ne pas être parfait pour certains types

d'étudiants comme expliqué lors de la section 3.3 à la page 27. Sur ce graphique, nous pouvons néanmoins nous rendre compte que la difficulté des questions peut effectivement sortir de l'intervalle de confiance (à partir de la question 17 du graphique 4.3). Comme la difficulté n'est sortie que de très peu de l'intervalle de confiance, elle n'a dès lors pas rempli notre condition d'arrêt car la marge entre ces deux points (ici entre `Difficulty` et `Estimation Max`) était trop faible. Il aurait été probablement possible de rajouter d'autres conditions d'arrêt pour éviter à cet étudiant d'avoir à répondre à trop de questions. Le chapitre 6 : Limites, Contraintes et Solutions en discutera plus en détails. Notons également que cet événement est très probablement lié à notre choix (discuté dans la section 2.1.4 du chapitre 2 de la page 18) de ne pas utiliser la règle de longueur.

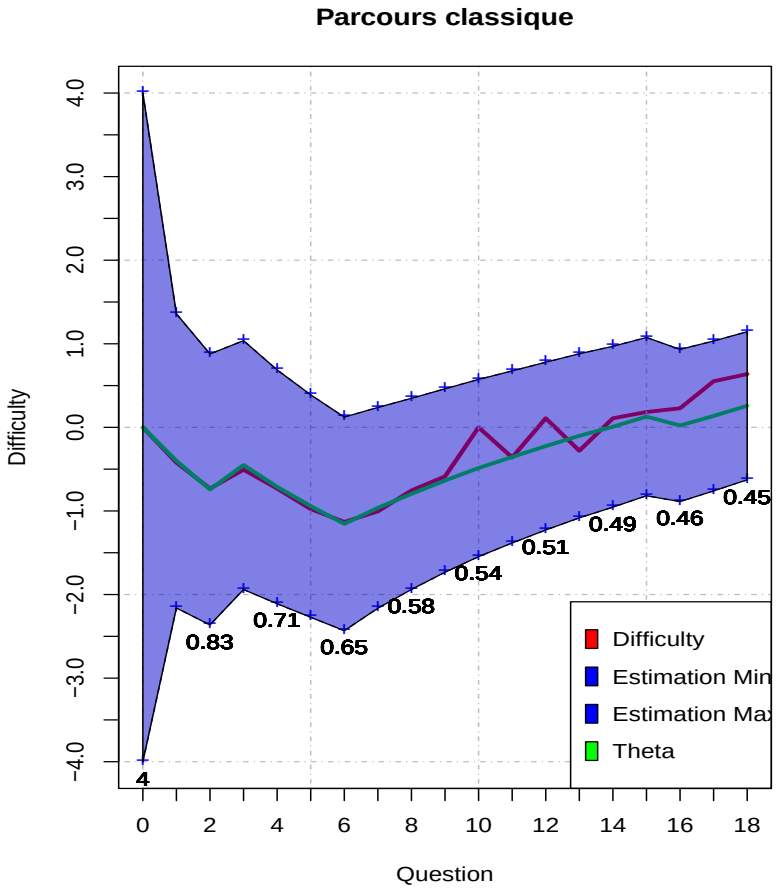

FIGURE 4.4 – Parcours d'étudiant classique

Finalement, nous avons comparé notre graphique 4.3 qui montre un parcours atypique, à un étudiant ayant un niveau moyen et ayant reçu un nombre de questions raisonnable. Le graphique 4.4 nous montre un tel parcours dans notre test adaptatif. On peut voir que l'étudiant a commencé à rater plusieurs questions (le `theta` diminue dans le premier tiers du test) avant de se rattraper car les questions étaient jugées comme plus faciles et il a finalement été évalué avec un `theta` positif ce qui signifie une note supérieure à 50%. Nous pouvons d'ailleurs aussi voir que la difficulté des questions est bien restée dans l'intervalle de confiance (souvent très proche du niveau `theta`) et cela nous montre que le test adaptatif s'est comporté de la manière attendue. Il est à noter que cet étudiant a d'ailleurs répondu avoir été correctement noté lors de notre sondage (point vert de la figure 4.1 de la page 35).

Conclusion

Bien que quelques modifications aient dû être effectuées du fait de ne pas avoir accès aux questions d'un cours répondant aux critères nécessaire pour l'implémentation d'un test adaptatif, nous avons pu créer un test adaptatif uniquement à partir d'éléments fournis par les professeurs. D'après le sondage soumis aux étudiants, il semble que nous puissions conclure que ce test est concluant. Dès lors, notre seconde question de recherche (figure 1 de la page 2) est également validée.

Chapitre 5

Mode D'emploi

Ce mode d'emploi est un tutoriel complet pour guider les utilisateurs, qu'ils soient professeurs ou étudiants, à travers notre implémentation sur l'application INGInious. Ce mode d'emploi sera détaillé en deux phases. La première sera une explication de la partie étudiante/utilisateur et la seconde sera concentrée sur la partie professeur/administrateur. Ce mode d'emploi suppose une connaissance d'INGInious au préalable et ne parcourra donc que les ajouts de notre implémentation.

5.1 Partie pour les étudiants/utilisateurs

Les étudiants ne voient la partie adaptative qu'à travers une black box. Pour eux, seules compte les questions auxquelles ils doivent répondre et le score qu'ils auront suite à leurs réponses. Ils devront accéder à une question, y répondre puis passer à la suivante jusqu'à ce qu'il n'y en ait plus (voir étape d'évaluation 1.1 de la page 6). C'est le seul événement, dans le déroulement des tests adaptatifs, qui nécessite une intervention de l'étudiant.

FIGURE 5.1 – Affichage initial

Plus concrètement, la figure 5.1 nous montre comment un cours s'affiche lorsqu'un étudiant accède à ce cours qui utilise notre implémentation. Il n'y voit qu'une

seule question pour l'instant qui sera la même pour chacun. Une fois qu'il aura complété cette question, en fonction de ce que le professeur a réglé en créant la question, il aura un retour sur sa réponse. Une fois cela fait, il devra retourner sur l'écran de sélection de questions. Pour cela, il doit cliquer sur le nom du cours en haut de la page comme le montre la figure 5.2.

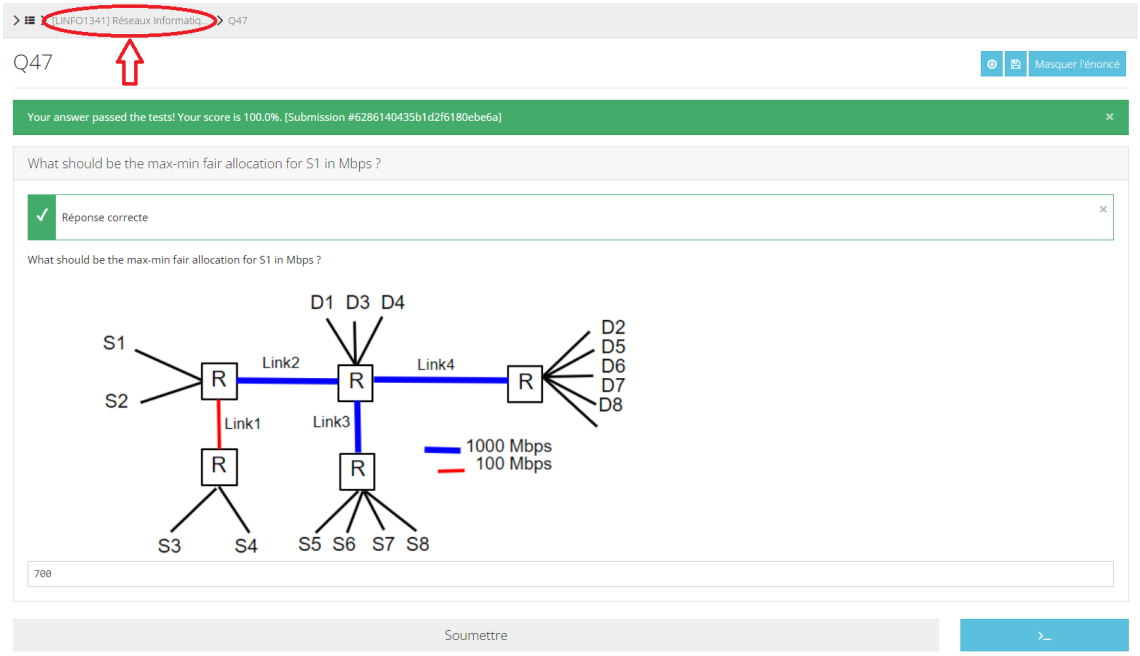

FIGURE 5.2 – Affichage d'une bonne réponse

Une fois de retour sur la page de sélection de questions après avoir répondu à une question, la page de sélection de questions lui indique maintenant son score actuel ainsi qu'une nouvelle question comme nous pouvons le voir sur la figure 5.3. Il est à noter que la question à laquelle l'étudiant a répondu est toujours visible mais est maintenant grisée et qu'il ne peut désormais plus que regarder sa réponse sans possibilité de soumettre à nouveau.

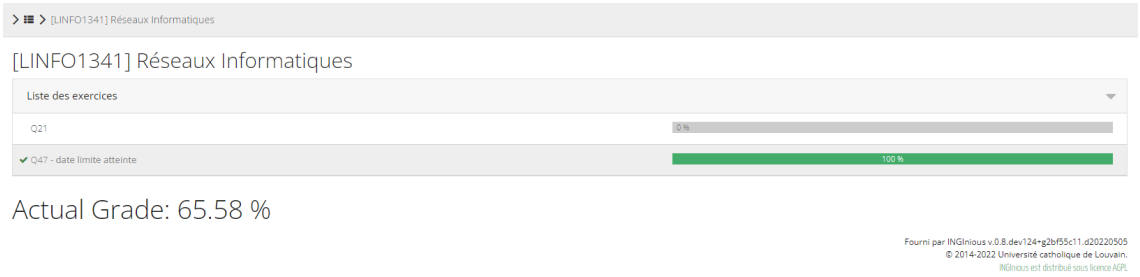

FIGURE 5.3 – Affichage de la liste de questions

Il devra continuer à effectuer ces étapes jusqu'à ce qu'il n'y ait plus aucune nouvelle question qui s'affiche et que son score soit affiché comme étant "Final Grade" comme on peut le voir sur la figure 5.4.

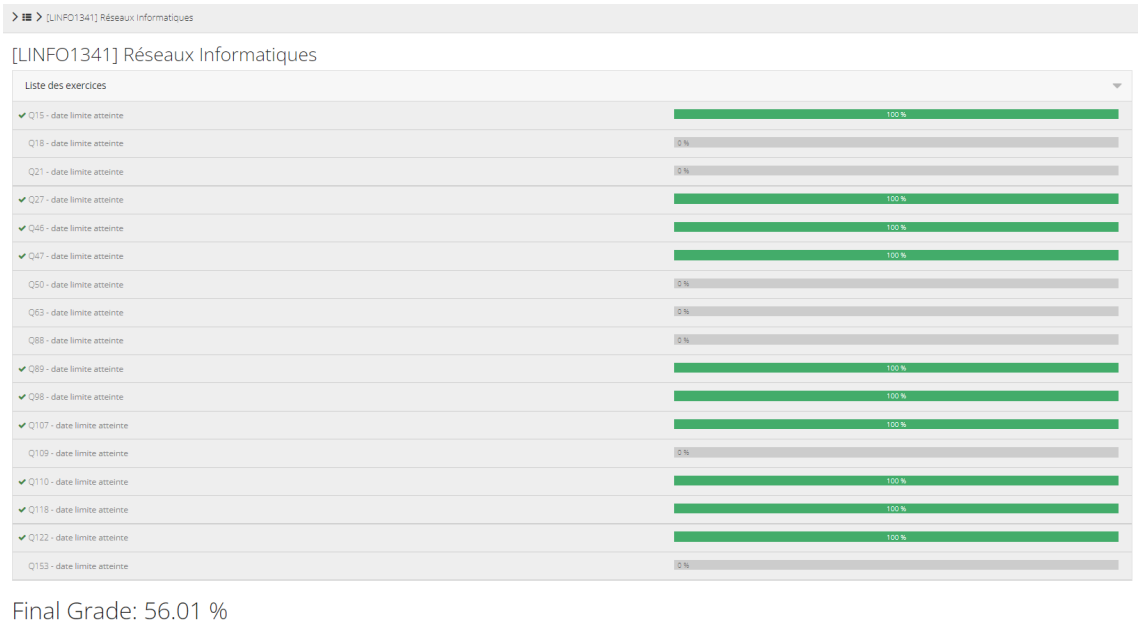

FIGURE 5.4 – Affichage final

5.2 Partie pour les professeurs/administrateurs

Les étapes à suivre pour initialiser un test adaptatif sur INGInious sont un peu plus complexes mais ne nécessitent aucune connaissance dans la théorie des tests adaptatifs. Le seul pré-requis à cela est de disposer d'un cours sur lequel des soumissions ont déjà été effectuées pour chaque question que l'on voudrait mettre dans le test adaptatif. Nous allons donc supposer dans cette partie que le professeur qui veut créer un test adaptatif dispose d'un tel cours avec un nombre raisonnable de réponses pour chaque question qu'il souhaite intégrer au test. Il faut également que celui-ci ait la possibilité d'avoir accès au niveau administrateur d'un cours.

Pour créer un test adaptatif, il y a deux moyens avec notre implémentation. Le premier est de créer un nouveau cours vierge qui récupérera les exercices et leurs réponses associées d'un cours source, ou alors de transformer ce cours déjà existant en un test adaptatif. Nous allons expliquer le fonctionnement dans le cas où le professeur voudrait garder son cours d'origine et en créer un nouveau de type adaptatif mais il est tout à fait possible de suivre ce tutoriel s'il souhaite transformer un cours déjà existant. Les démarches à suivre sont les mêmes à l'exception de l'importation de questions (5.2.2) qu'il faudra simplement passer.

5.2.1 Passer en test adaptatif

Pour passer un cours en test adaptatif, qu'il soit vierge ou déjà existant, l'administrateur doit se rendre dans le cours puis aller dans la partie administration du cours (figure 5.5). Ensuite, il devra se rendre dans la section `Exercices`. Une fois sur cette page, il faudra changer le distributeur (figure 5.6) et choisir le distributeur de tests adaptatifs (figure 5.7). Il est à noter que c'est ici que notre implémentation entre en jeu. Le distributeur que l'administrateur choisi correspond au `Task Dispenser` discuté lors de la section 3.2.3.

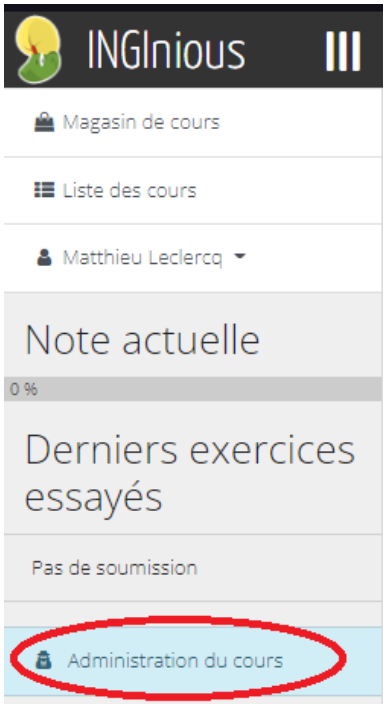

FIGURE 5.5 – Accéder à l'administration

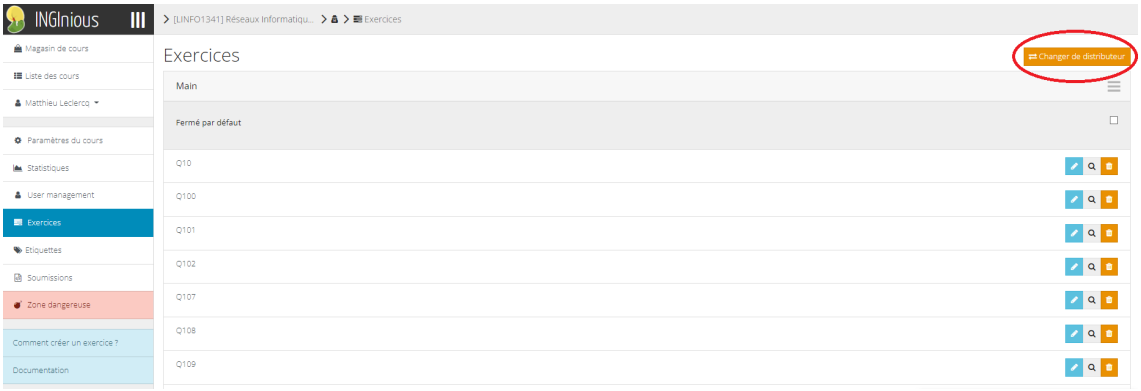

FIGURE 5.6 – Changer le distributeur

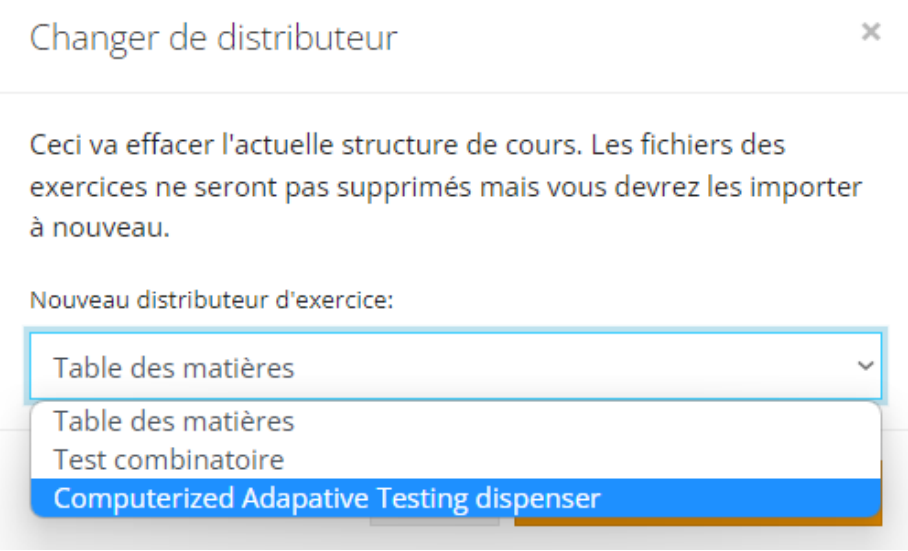

FIGURE 5.7 – Choisir le distributeur de tests adaptatifs

5.2.2 Importer des questions

Maintenant que le cours est créé et que le distributeur de questions est passé en mode `Computerized Adaptive Testing dispenser`, il faut dire à INGInious quelles questions utiliser. Pour cela, il faudra donc, comme dit précédemment, soit les importer d'un cours déjà existant soit en avoir qui sont déjà créées et auxquelles des étudiants ont déjà répondu dans le cours même.

Nous allons donc supposer ici que de telles questions doivent être importées depuis un autre cours et si cela n'est pas nécessaire (qu'il s'agit donc du cas où le professeur souhaite changer son cours déjà existant) alors il peut simplement passer à l'étape suivante.

Pour importer des questions, la seule chose nécessaire est de connaître l'identifiant du cours source. Pour le trouver, le plus simple est de regarder l'adresse `url` du cours lors de la sélection de questions comme le montre la figure 5.8. Dans cet exemple, l'identifiant est donc `cnp3`.

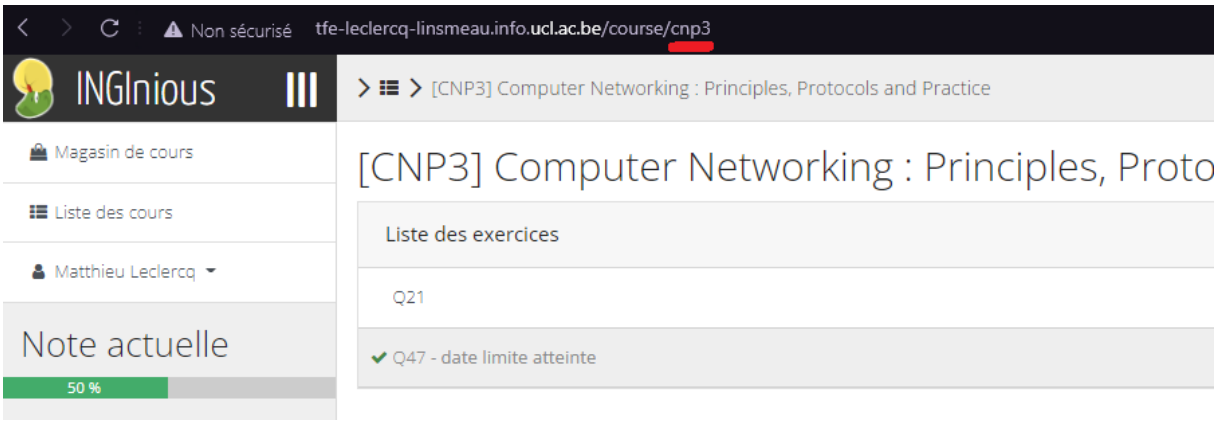

FIGURE 5.8 – Trouver l'identifiant du cours

Une fois cet identifiant connu, dans la section `Exercices` du cours (dans `l'administration du cours`) que l'on veut transformer en test adaptatif, il faudra cliquer sur le bouton `import`, comme le montre la figure 5.9. Ensuite, dans la fenêtre qui s'ouvre, il suffit d'indiquer l'identifiant du cours source (dans l'exemple ici, il faudra donc écrire `cnp3`) comme indiqué par la figure 5.10 et cliquer sur le bouton vert `Import`.

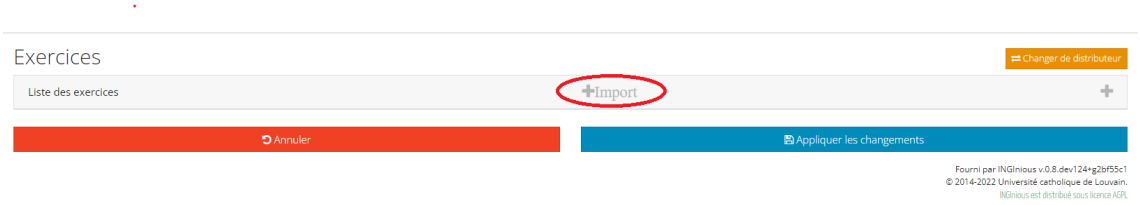

FIGURE 5.9 – Importer les questions

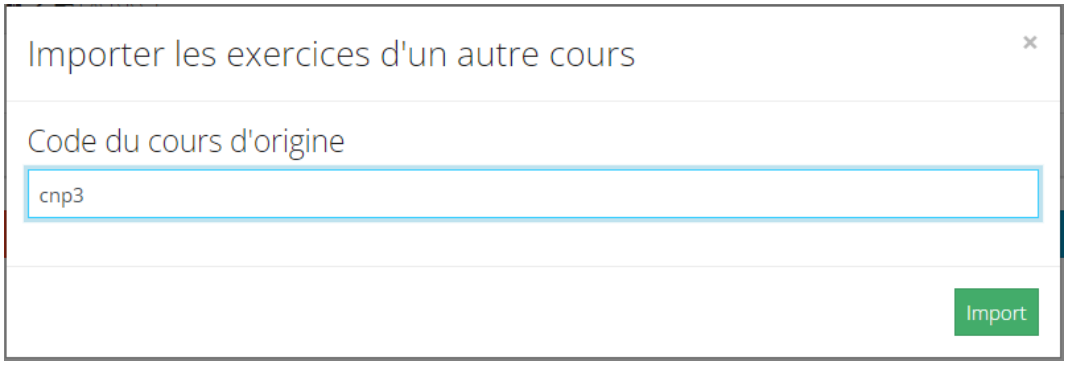

FIGURE 5.10 – Fenêtre d'importation des questions

5.2.3 Choix des questions

Une fois les questions ajoutées dans le cours, il ne reste plus qu'à choisir lesquelles nous voulons ajouter au test adaptatif. Pour ce faire, il faudra cliquer sur le bouton "+" dans la page `Exercices` dans l'administration du cours comme le montre la figure 5.11. Par la suite, il ne reste qu'à sélectionner toutes les questions qu'on veut rajouter puis cliquer sur le bouton `Ajouter`. Il est à noter qu'il est également possible de sélectionner tous les exercices d'un seul coup en cliquant sur le bouton `Tout sélectionner` comme montré sur la figure 5.12.

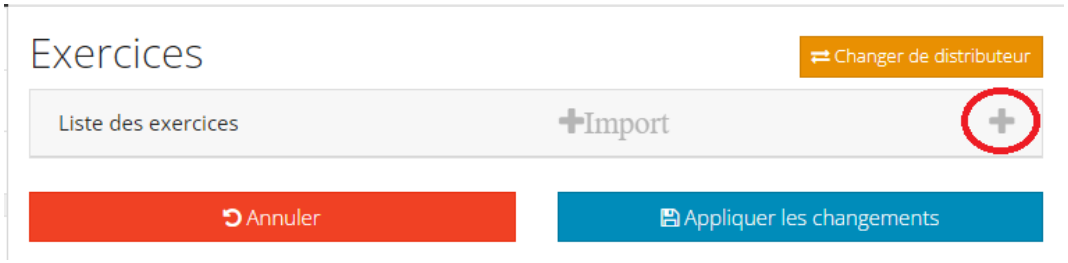

FIGURE 5.11 – Où sélectionner les questions

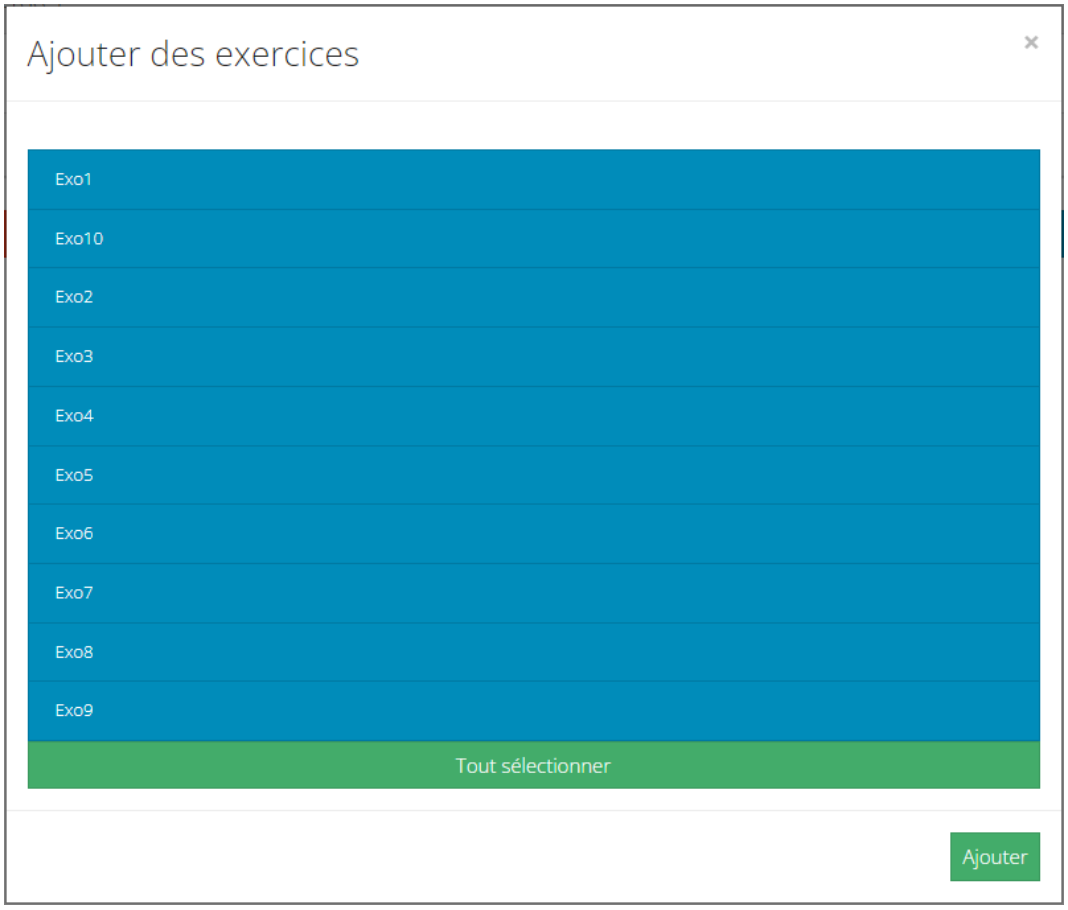

FIGURE 5.12 – Sélectionner et ajouter les questions

5.2.4 Finalisation

Maintenant que toutes les questions ont bien été ajoutées comme souhaité, il ne reste plus qu'à cliquer sur `Appliquer les changements` (5.13). Si tout s'est bien passé, sur la fenêtre initiale du cours où les questions sont affichées pour les étudiants, une seule question devrait être affichée comme discuté lors de la section 5.1 pour les étudiants.

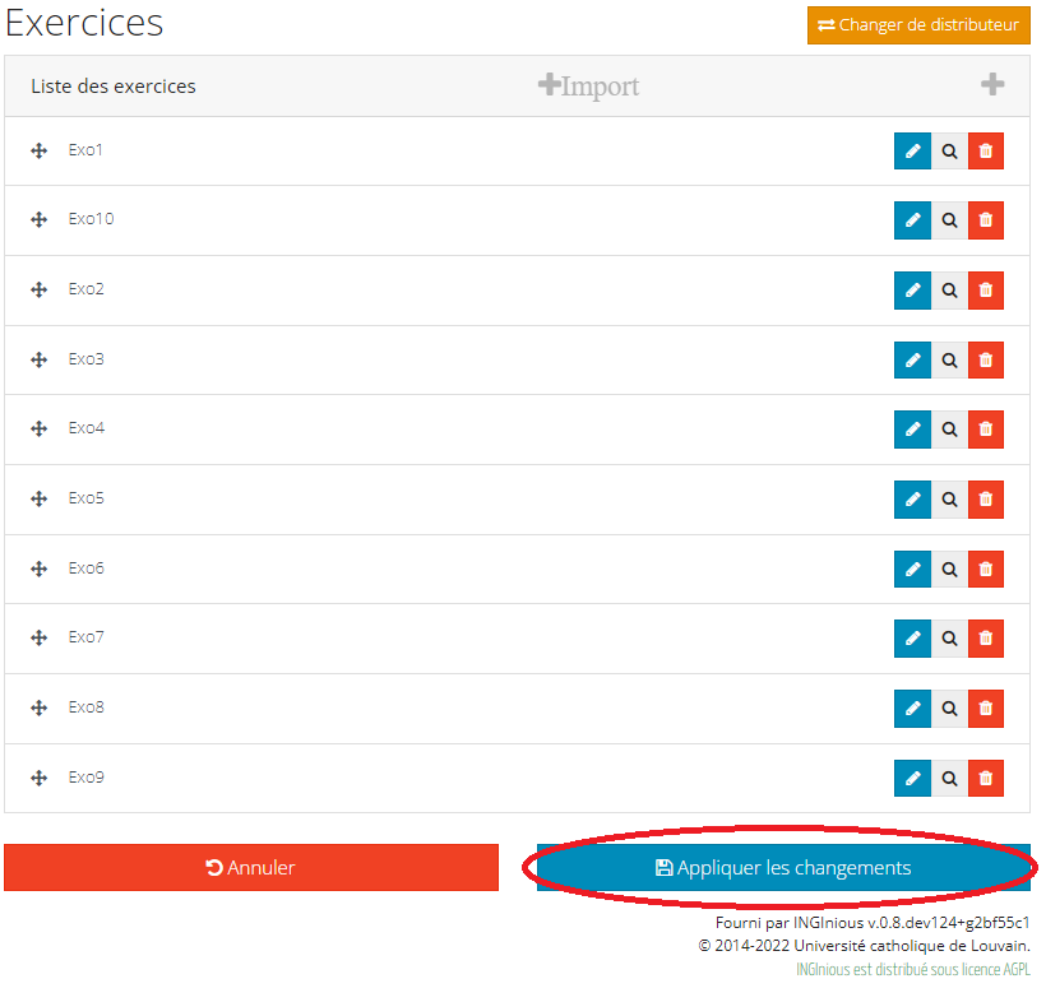

FIGURE 5.13 – Appliquer les changements

Chapitre 6

Limites, Contraintes et Solutions

Nous avons exposé dans la section 1.2 les avantages et les inconvénients des tests adaptatifs par rapport aux tests linéaires. Dans ce chapitre nous développerons plus en détails ces inconvénients. Nous aborderons aussi d'autres limites et contraintes que ces tests nous imposent afin qu'ils soient le plus utiles et efficaces. Nous terminerons chaque section en décrivant quelles solutions nous pourrions apporter dans le futur à ces différents problèmes.

6.1 Problèmes liés au concept de tests adaptatifs

Les conditions préalables à la création de tests adaptatifs utiles sont nombreuses et proviennent du concept même de tests adaptatifs.

6.1.1 Nombre de questions

Pour qu'une question apporte le plus d'informations pour améliorer la précision du score de l'étudiant, il faut un panel de questions suffisant. Plus il y aura de questions, plus il est probable que le système choisisse une question améliorant grandement la précision du score de l'étudiant. Comme le test adaptatif doit évaluer des étudiants de tous niveaux, il faut que ces questions soient distribuées de manière uniforme sur toute la surface des niveaux de difficultés. En effet, si les questions sont toutes difficiles, les étudiants moins performants seront moins bien évalués et inversement. Le problème peut aussi se poser pour des étudiants moyens avec un échantillon de questions qui seraient trop difficiles ou trop faciles. Dans ce cas, la question la plus pertinente à poser à l'étudiant n'augmentera alors pas beaucoup la précision de son score. Comme la précision n'augmente pas, soit un plus grand nombre de questions sera posé, soit le test s'arrêtera à cause de la condition d'arrêt d'information (section 1.3.4). On revient alors au problème du nombre de questions

dans l'échantillon qui doit être suffisant.

Dans un test il faut aussi poser des questions sur différents thèmes. Ainsi si chaque question choisie appartient au même chapitre, le test n'évaluera pas correctement l'étudiant sur l'intégralité de la matière. S'il s'agit d'un test de langue par exemple, il faudrait poser des questions de grammaire, de vocabulaire, de lecture, etc. Le choix de la question ne devrait dès lors pas être basé uniquement sur le score de la question mais aussi sur la partie de la matière que la question couvre. Une autre solution pourrait être de diviser le test en de plus petits tests adaptatifs séparés par chapitre (comme le fait la plateforme Wallangues) mais nous revenons alors au problème du nombre de questions potentiellement insuffisant pour chacun de ces différents tests. Pix s'attaque à ce problème en utilisant désormais une méthode permettant de prendre en compte un grand nombre de compétences dans la création de ses tests adaptatifs [20].

De plus certaines parties de la matière peuvent être plus importantes que d'autres. Il faudrait alors poser un nombre de questions proportionnel à l'importance du chapitre auquel elles se rapportent.

Pour s'attaquer au premier problème qui est le nombre suffisant de questions, une solution pourrait être d'augmenter le nombre de questions artificiellement. En créant préalablement des questions avec un certain nombre de paramètres, il est possible de poser des questions semblables en faisant varier ces paramètres. Deux possibilités peuvent être dès lors être décrites : soit les difficultés des questions semblables sont les mêmes soit elles sont différentes. Par exemple le calcul d'intégrale d'une fonction polynomiale peut être vu de difficulté équivalente à une autre (même si les paramètres de la question sont différents). Si par contre on change la fonction pour que celle-ci contienne en plus des exponentielles, la difficulté de la question pourrait augmenter. Le but est de créer plus rapidement un ensemble d'items grâce à un modèle de question prédéfini.

Au départ d'une question il est donc possible de créer un large panel de questions avec des difficultés identiques ou différentes.

— L'utilité d'avoir plusieurs items avec des **difficultés identiques** et venant d'un même modèle serait de poser des questions semblables pour chaque étudiant sans que ceux-ci ne puissent tricher. Pour estimer les paramètres des items préalablement au test, il ne faudrait poser à chaque étudiant qu'une de ces questions comme leurs difficultés sont à peu de choses près identiques. Bien entendu, il faudrait empêcher, dans la mesure du possible,

de poser plusieurs questions de ce même groupe de questions lors du test adaptatif.

— Pour des items semblables aux **difficultés différentes**, le modèle sera plus général. Ici, chaque question doit être évaluée séparément car les difficultés ne sont pas les mêmes. L'avantage lors du test adaptatif est que ces questions peuvent être posées à un même étudiant car bien que les questions soit semblables, leurs difficultés ne le sont pas.

6.1.2 Temps total d'un test

Les examens de l'université sont soumis à une contrainte de temps. Au bout d'un certain laps de temps, les étudiants doivent rendre leur copie s'il s'agit d'un test papier ou le système clôture le test s'il s'agit d'un CBT. Il faut trouver une solution pour que cette contrainte soit prise en compte dans les tests adaptatifs. Nous pouvons imaginer que chaque question dans la banque d'items doit être créée avec un temps de résolution moyen posé préalablement. Par exemple, si le test doit durer 2 heures et qu'il faudrait poser 8 questions pour avoir une précision du score de l'étudiant suffisante, il faudrait que chaque question dure approximativement 15 minutes. Cela veut dire qu'il faut connaître préalablement le temps du test car si un test dure moins longtemps, les questions de 15 minutes deviennent trop longues et il faut, soit changer toute la banque d'items, soit accepter un niveau de précision du score de l'étudiant plus bas.

Une autre solution pourrait être de prendre en compte la durée moyenne d'une question lors du choix d'une nouvelle question dans le cas où l'on manquerait de temps. Dans cette situation, on risque de tomber sur une question donnant moins d'informations pour améliorer la précision du score de l'étudiant et il faudrait alors poser plus de questions alors qu'on manque déjà de temps.

6.1.3 Un processus itératif

Comme nous l'avons expliqué dans la section 1.2, laisser quelqu'un évaluer la difficulté des questions d'un test n'est pas rigoureux. Cela pourrait aussi avoir des conséquences plus importantes comme l'échec d'un étudiant qui a les compétences requises pour réussir car il a mal répondu à une question difficile qui a été évaluée comme moyenne ou facile. Un professeur aura certainement tendance à trouver ses questions plus faciles que ses étudiants. Pour connaître une difficulté approchant la réalité pour chaque question, il faut que des étudiants aient déjà répondu à ces questions. Dans leur Mémoire, Audrey Wenders et Marie Visschers [22] exposent une formule qui estime le nombre d'étudiants qui devraient avoir répondu à une question afin que l'estimation de celle-ci soit la plus juste possible. 385 est le nombre d'étudiants qu'elles obtiennent pour un niveau de confiance de 95% avec une marge

d'erreur de 5% autour de la valeur de difficulté de la question pour une question dichotomique. Plus des étudiants auront répondu à la question, plus le niveau de difficulté sera précis. On ne peut donc pas décider de créer un test adaptatif avec une banque d'items aux niveaux de difficulté précis du jour au lendemain. Ce qui peut être fait par contre est de poser ces questions à des étudiants d'une année pour ensuite transformer ce panel de questions en test adaptatif pour l'année prochaine. Chaque année, plus d'étudiants répondront à ces questions, et le niveau des questions sera plus précis. Il s'agit donc d'un processus qui s'améliore au fil du temps, en répétant les mêmes questions auprès d'étudiants sur plusieurs années.

Cela n'est pas encore implémenté sur INGInious. Il serait cependant possible de le faire au moment de l'archivage d'un cours INGInious en fin d'année (bouton rouge "Zone de danger" sur la figure 5.6 de la page 44). Les données que l'administrateur d'un test adaptatif veut archiver seraient alors utilisées pour recalibrer les items de la banque d'items afin d'avoir une estimation de la difficulté des questions plus précise pour les années suivantes.

6.1.4 INGInious, Wooclap et création automatique de tests adaptatifs

Un nombre important de questions auxquelles des étudiants ont déjà répondu, il y en a sur INGInious et Wooclap. Le problème c'est que les questions actuellement sur la plateforme INGInious n'ont pas été créées en pensant qu'elles seront utilisées dans des tests adaptatifs (il y a peut-être des questions plus courtes que d'autres, plus de questions sur un thème qui n'est pas très important, etc.). Le même problème se pose avec les questions Wooclap transférées sur INGInious. Elles n'ont été créées que pour être posées lors des cours magistraux (besoin de contexte du cours magistral pour pouvoir y répondre, certaines questions sont non pertinentes pour déterminer le trait latent, l'anonymat des étudiants rend le modèle 2PL inutilisable, etc.). L'adaptation manuelle de Wooclap vers INGInious ou même d'un cours pré-existant vers un test adaptatif nouvellement créé sur INGInious est à réduire au minimum. En effet, sur le long terme nous voudrions arriver à un système presque automatique de création de tests adaptatifs. Si on veut créer des questions INGInious qui seront à terme utilisées dans un test adaptatif, il faudra donc penser à tous les inconvénients précédemment cités pour que le test soit utilisable.

6.2 Problèmes liés à l'implémentation

Dans cette section nous parlerons des problèmes qui nous sont imposés par les implémentations préexistantes. Nous tenterons d'y apporter une solution envisageable pour un travail futur.

6.2.1 INGInious

Dans INGInious, deux problèmes peuvent survenir. Premièrement, un professeur administrateur pourrait inclure par mégarde une question qui n'a pas de statistiques de réponse. Le système ne pourrait dès lors pas trouver de difficulté pour cette question et cela pourrait compromettre les statistiques. Il serait possible de détecter cela lors de la création de la banque d'items et d'expliquer que le test n'a pas pu être créer à cause de certaines questions sans statistiques en pointant quelles questions sont à retirer.

Deuxièmement, il est impossible de prendre des questions de différents cours pour créer un test adaptatif. En effet les étudiants n'étant pas forcément les mêmes pour les deux cours et les questions n'étant pas les mêmes non plus, il n'y aura pas de convergence du niveau de questions lorsque l'algorithme d'Espérance-Maximization va créer la banque d'items (section 1.4.2). Le problème ne se posera pas lorsqu'on voudra utiliser les mêmes questions d'une année à l'autre. Dans ce cas, il n'y aura qu'une dissociation entre les étudiants et pas entre les questions et les étudiants.

Etudiant\Question	Q1_1	Q1_2	Q1_3	...	Q2_1	Q2_2	Q2_3	...
E1_1	1	0	1	...	N/A	N/A	N/A	...
E1_2	1	0	0	...	N/A	N/A	N/A	...
E1_3	0	1	1	...	N/A	N/A	N/A	...
...	...	...	...	...	...	...	...	...
E2_2	N/A	N/A	N/A	...	1	1	0	...
E2_2	N/A	N/A	N/A	...	0	1	1	...
E2_3	N/A	N/A	N/A	...	1	0	0	...
...	...	...	...	...	...	...	...	...

TABLE 6.1 – Tableau de réponses des étudiants aux questions de 2 cours distincts

Dans le tableau 6.1, similaire à la matrice décrite dans la section 3.4, on peut observer que les étudiants du premier cours ($E1\_N$) ne répondent pas aux questions du second cours ($Q2\_N$) et inversement. Il y a une dissociation totale entre les deux parties du tableau et cela empêcherait la convergence de l'algorithme.

Dans le cas où on rajouterait des étudiants dans un même cours (par exemple lorsqu'une nouvelle année débute), ce problème ne se poserait pas. Il n'y aurait pas de dissociation comme décrit précédemment car ces nouveaux étudiants répondent aux mêmes questions que les anciens.

Un point que nous pouvons soulever, qui n'est pas une limitation stricte car il pourrait être changé avec du temps, est qu'INGInious a été conçu pour accueillir des tests linéaires classiques. Pour améliorer certains aspects d'interface ou certaines facilités cela prendrait beaucoup de temps car il faudrait changer des parties qui forment certaines bases d'INGInious.

Nous pouvons citer par exemple le score de l'étudiant donné pour le test adaptatif qui est différent du pourcentage de complétion des questions du cours (Schéma 6.1).

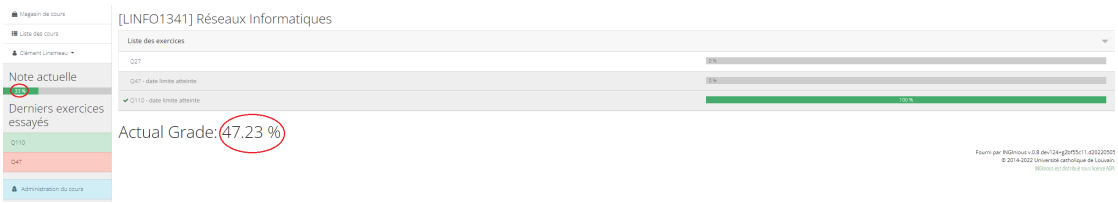

FIGURE 6.1 – Le pourcentage de complétion du cours (à gauche) et le pourcentage de score de l'étudiant (à droite) sont inégaux.

Sur l'interface de l'étudiant, les deux pourcentages sont présents mais seulement le pourcentage sous les questions est déterminant. Nous avons expliqué précédemment que notre système est un plugin à INGInious et nous n'avons pas d'accès à ce pourcentage de complétion du cours. Si le plugin est définitivement inclus dans INGInious, ces deux résultats pourront être fusionnés en un seul. Un autre problème peut être résolu par cette méthode : l'administrateur du cours n'a pour l'instant pas accès sur INGInious au score de l'étudiant mais seulement au score de complétion du cours (notons que le score réel n'est accessible que dans la base de données mongoDB pour l'instant). Quand la fusion aura eu lieu, l'administrateur aura accès au score réel de l'étudiant comme à tout autre score.

Nous pouvons aussi mentionner le bouton permettant d'aller à la question suivante. Actuellement, l'étudiant doit revenir sur la page du cours pour accéder à la prochaine question, ce qui est incommode. Pour un test adaptatif, ce bouton ne devrait apparaître qu'après que l'étudiant ait répondu à la question actuelle car il faut que le système détermine la question suivante avant de permettre un accès

vers elle. Pour cela, il faudrait dévier le bouton de la question suivante vers une nouvelle question dynamiquement décidée par notre système.

6.2.2 Paramètres du package `R`

Dans la figure 4.3 de la page 37 nous pouvions observer qu'un trop grand nombre de questions étaient posées. Comme nous le supposons dans le chapitre 4, cela est dû au fait qu'il y avait toujours des questions pertinentes à poser bien que celles-ci n'améliorent pas tellement la précision du score de l'étudiant. L'intervalle `semTheta` ne devenait pas assez rapidement petit et la condition d'arrêt n'était pas respectée. On peut dès lors supposer que ce seuil était mal ajusté. Une des solutions possibles serait de rajouter une condition sur le gain de la note finale de l'étudiant dans l'intervalle de confiance. Si la variation du `theta` dans l'intervalle ne fait pratiquement plus changer la note finale, il ne devient plus pertinent de continuer le test, peu importe la valeur du `semTheta`. Si par exemple nous avons un intervalle de confiance de 1 (donc plus grand que le seuil maximum) mais que le `theta` est de 3, alors nous savons que le `theta` ne va varier (à 95%) qu'entre 2 et 4. Dès lors, sachant qu'un score `theta` de 2 reste très élevé (pratiquement un score de 100%), la variation entre 2 et 4 n'a presque plus d'impact sur le score final et nous pourrions arrêter le test. C'est le cas qui s'est présenté dans la figure 4.3 à la page 37. Cette nouvelle condition aurait pu limiter le nombre de questions.

6.3 Problèmes d'ordre éthique

Certaines raisons éthiques liées aux tests adaptatifs peuvent être mentionnées. Premièrement, on pourrait se demander s'il est éthique, voire équitable de poser des questions différentes à des étudiants différents. Ce raisonnement peut être en partie défait par plusieurs arguments : les examens oraux réalisent la même chose et sont intégrés depuis bien longtemps dans les cursus universitaires. Certains examens papiers changent l'ordre des questions et les paramètres des questions. Deuxièmement, un nombre différent de questions pour chaque étudiant peut paraître injuste aux yeux des étudiants bien que cela améliore la précision de leur score. Finalement, chaque question a un impact plus grand sur sa note finale comme moins de questions sont posées. L'étudiant pourrait alors se dire qu'il a été jugé sur un panel non suffisant de questions et qu'il n'a pas eu de chance sur la sélection. Ces raisonnements pourraient largement être la base de recours. Nous controns cette problématique pour l'instant en ne réalisant que des tests formatifs comme exposé lors de notre validation (Chapitre 4).

6.4 Situation idéale

Nous concluons ce chapitre en présentant une situation idéale pour que soit construit un test adaptatif utile, utilisable et efficace grâce à notre système amélioré des points que nous avons cités précédemment. Pour commencer, il faut un grand nombre de questions uniformément réparties sur toute la surface de difficultés, reprenant chaque point de la matière et auxquelles un nombre important d'étudiants ont déjà répondu. Soit ces questions doivent durer plus ou moins le même temps, soit le système doit prendre en compte le temps dans sa sélection de questions s'il y a une limite de temps. Finalement, il faut que ce test soit formatif faute de quoi des recours pourraient émerger. Pourtant, si toutes ces conditions sont remplies, les tests adaptatifs offrent une évaluation plus rapide et plus sûre du niveau de l'étudiant qu'un test linéaire classique.

Conclusion

Dans ce Mémoire, nous avons étudié la nature des tests adaptatifs. Nous avons d'abord commencé par expliquer que, pour ces tests, les réponses de l'étudiant ont un effet direct sur le choix des questions suivantes. Nous avons ensuite exploré les étapes de déroulement de ces tests et étudié la théorie s'y rapportant. En décrivant les outils dont nous avons eu besoin, nous avons pu adapter cette théorie à un contexte réel. Pour répondre à notre première question de recherche nous avons implémenté notre système supportant les tests adaptatifs sur INGInious, la plateforme d'exerciseur de l'Université Catholique de Louvain. En créant un test adaptatif formatif pour les étudiants au moyen de l'interface d'INGInious, nous avons répondu à notre seconde question de recherche en plus de valider notre modèle auprès des étudiants. La validation nous a permis de récolter et d'exposer aux moyens de statistiques le point de vue général des étudiants plutôt positif.

Après avoir réalisé cette implémentation, nous nous sommes rendu compte que de nombreux obstacles peuvent alourdir, voire rendre impossible, la création d'un test adaptatif. Nous pensons qu'à terme, ces tests pourront être une solution envisageable parmi d'autres. Nous avons essayé de réduire le plus possible ces obstacles pour que ceux-ci soient transparents aux yeux des utilisateurs (enseignants et étudiants). Certains de ces obstacles subsistent néanmoins et empêchent que les tests adaptatifs ne soient un réel substitut aux tests linéaires mais plutôt une alternative à envisager. Nous avons expliqué ces obstacles mais le plus contraignant dans notre implémentation est l'impossibilité de gérer le temps de chaque question.

Nous sommes néanmoins heureux de pouvoir proposer une nouvelle alternative, qui semble assez bien correspondre aux attentes des étudiants. Les évaluations sont un des éléments inhérents aux études et il nous semble important d'avoir un éventail de méthodes d'évaluation le plus large possible pour que les enseignants aient la possibilité d'examiner les étudiants le plus équitablement possible.

Bibliographie

[1] Inginious open source. `https://github.com/UCL-INGI/INGInious`.

[2] Inginious' documentation. `https://docs.inginious.org/en/latest/`.

[3] Wooclap. `https://www.wooclap.com`. Accessed : 01-05-2022.

[4] Allan Birnbaum. Statistical theory for logistic mental test models with a prior distribution of ability. *Journal of Mathematical Psychology*, 6(2) :258–276, 1969.

[5] R Darrell Bock and Robert J Mislevy. Adaptive eap estimation of ability in a microcomputer environment. *Applied psychological measurement*, 6(4) :431–444, 1982.

[6] Hua-Hua Chang and Zhiliang Ying. A global information approach to computerized adaptive testing. *Applied Psychological Measurement*, 20(3) :213–229, 1996.

[7] Seung W Choi and Richard J Swartz. Comparison of cat item selection criteria for polytomous items. *Applied psychological measurement*, 33(6) :419–440, 2009.

[8] Theo JHM Eggen and Gerard JJM Straetmans. Computerized adaptive testing of arithmetic at the entrance of primary school teacher training college (wiscat-pabo). *The Transition to Computer-Based Assessment*, page 134, 2009.

[9] Amy Hendrickson. An ncme instructional module on multistage testing. *Educational Measurement : Issues and Practice*, 26(2) :44–52, 2007.

[10] Frederic M Lord. *Applications of item response theory to practical testing problems*. Routledge, 2012.

[11] Frederic M Lord and Melvin R Novick. *Statistical theories of mental test scores*. IAP, 2008.

[12] David Magis and Juan Ramon Barrada. Computerized adaptive testing with R : Recent updates of the package catR. *Journal of Statistical Software, Code Snippets*, 76(1) :1–19, 2017.

[13] David Magis, Duanli Yan, and Alina A Von Davier. *Computerized adaptive and multistage testing with R : Using packages catR and mstR*. Springer, 2017.

[14] Robert J Mislevy and R Darrell Bock. Biweight estimates of latent ability. *Educational and psychological measurement*, 42(3) :725–737, 1982.

[15] Stefan Oppl, Florian Reisinger, Alexander Eckmaier, and Christoph Helm. A flexible online platform for computerized adaptive testing. *International Journal of Educational Technology in Higher Education*, 14(1) :1–21, 2017.

[16] Georg Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

[17] Dimitris Rizopoulos. ltm : An r package for latent variable modelling and item response theory analyses. *Journal of Statistical Software*, 17(5) :1–25, 2006.

[18] Christof Schuster and Ke-Hai Yuan. Robust estimation of latent ability in item response models. *Journal of Educational and Behavioral Statistics*, 36(6) :720–735, 2011.

[19] Daniel O Segall. A sharing item response theory model for computerized adaptive testing. *Journal of Educational and Behavioral Statistics*, 29(4) :439–460, 2004.

[20] Jill-Jênn Vie, Fabrice Popineau, Françoise Tort, Benjamin Marteau, and Nathalie Denos. A heuristic method for large-scale cognitive-diagnostic computerized adaptive testing. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S '17, page 323–326, New York, NY, USA, 2017. Association for Computing Machinery.

[21] Thomas A Warm. Weighted likelihood estimation of ability in item response theory. *Psychometrika*, 54(3) :427–450, 1989.

[22] Audrey Wenders, Marie Visschers, and Olivier Bonaventure. Utilisation de tests adaptatifs pour les cours d'informatique. 2016.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**