Université Grenoble-Alpes

Etude comparative des pédagogies conventionnelle et Montessori en apprentissage des mathématiques de niveau moyenne section de maternelle

Auteurs
Azat Aleksanyan
Lucas Chabeau
Matthieu François
Etienne Hamard

Tuteurs
Marie-Caroline Croset
Adeline Leclerc-Samson

24 avril 2019

Remerciements

Nous tenons à remercier particulièrement Madame Adeline Leclercq-Samson et Madame Marie-Caroline Croset pour leur aide, leur disponibilité et leur écoute tout au long du projet. Nous remercions aussi le CNRS pour la confiance qu'ils nous ont accordé avec ce projet.

Table des Matières

Introduction

Le programme de notre première année de Master prévoit un projet tutoré faisant appel à nos compétences acquises en statistique et en sciences des données. C'est dans ce contexte que nous avons travaillé en collaboration avec Marie-Caroline Croset et Adeline Leclercq-Samson sur un sujet mêlant sciences cognitives, statistique et fouille de données.

Depuis 4 ans, une équipe de chercheurs de l'institut des sciences cognitives (ISC) de Lyon a testé les capacités en mathématiques des élèves d'une école maternelle située dans une zone REP+ (Réseau d'éducation prioritaire) de l'agglomération Lyonnaise. Une partie de ces élèves a suivi une méthode d'éducation conventionnelle et les autres ont suivi des enseignements suivants la méthode Montessori. Notre objectif est de voir s'il y a ou non une différence entre le niveau en mathématiques des élèves ayant suivi les enseignements conventionnels et ceux ayant suivi les enseignements Montessori.

Nous avons reçu les résultats des tests sous forme de tableurs que nous avons triés pour ne garder que les résultats des élèves de moyenne section. Nous avons ensuite nettoyé nos données et commencé l'étude sur cette population.

Problématique

Nous allons donc chercher s'il existe ou non une différence de niveau en mathématiques entre des élèves d'écoles Montessoriennes et ceux des élèves d'écoles Conventionnelles.

1. Contexte

Ce projet nous a été proposé par l'institut des sciences cognitives - Marc Jeannerod spécialisé en neurosciences. L'UMR (Unité Mixte de Recherche) 5304 créée en 2007 est l'un des deux laboratoires de l'institut des Sciences Cognitives – Marc Jeannerod. L'UMR 5304 est un laboratoire interdisciplinaire qui intègre l'expertise de chercheurs en sciences de la vie (psychologie cognitive, neurosciences) et en médecine (pédopsychiatrie, neuro-pédiatrie) avec celle de chercheurs en sciences humaines et sociales (linguistique computationelle, théorique et philosophie) pour étudier la nature et la spécificité de l'esprit humain.

1.1 Les besoins d'un changement éducatif

Le député et mathématicien Cédric Villani a publié un rapport pour renforcer l'apprentissage des mathématiques à l'école. Les élèves français ont aujourd'hui un niveau inquiétant dans cette discipline. Pourtant, jusqu'en 1985, l'enseignement des mathématiques en France était reconnu comme l'un des meilleurs. L'étude internationale "Trends in International Mathematics and Science Study" (TIMSS) 2015 qui mesure les performances en mathématiques et en sciences des élèves en fin de CM1 classe la France dernière des pays de l'Union européenne. Elle obtient même un score en dessous de la moyenne internationale. Pour mettre un terme à cette tendance inquiétante de la dégradation du niveau des élèves français en mathématiques, le gouvernement est maintenant ouvert à de nouvelles pédagogies pour l'enseignement des mathématiques.

1.2 Les pedagogies étudiées

Pédagogie Montessori : La pédagogie Montessori est une méthode d'éducation créée en 1907 par Maria Montessori.
La pédagogie se base sur quelques principes :
-**La liberté** : les enfants sont libres de choisir l'activité qu'ils souhaitent faire parmi celles qui leur sont proposées.
-**L'autodiscipline** : les enfants sont invités à repérer eux-même leurs erreurs.
-**L'action en périphérique** : les professeurs vont préférer agir sur l'environnement de l'enfant plutôt que directement sur lui. Ils vont chercher à inciter l'enfant à faire une activité plutôt que de directement lui demander de faire cette activité.
-**Le respect du rythme de chacun** : Le rythme de l'enfant est respecté tant qu'il est concentré.
-**L'apprentissage par l'expérience** : Favoriser la pratique pour s'approprier un concept.
-**L'activité individuelle** : La plupart des activités se font en individuel.
-**L'éducation, une aide à la vie** : L'éducation est faite pour préparer l'enfant à une vie dans une société harmonieuse basée sur le respect de l'autre.

Pédagogie conventionnelle : La pédagogie conventionnelle (celle que nous connaissons aujourd'hui dans nos écoles publiques) est celle du modèle transmissif. Selon le triangle pédagogique de Jean Houssaye, cette pédagogie privilégie la relation entre l'enseignant et le savoir. Autrement dit, l'enseignant expose un savoir sous forme de cours magistral, généralement suivi d'exercices ou/et de leçons à apprendre. L'élève doit intégrer et appliquer le savoir exposé par l'enseignant.

1.3 Présentation des données de départ

Notre jeu de données est composé de trois fichiers Excel (.xlsx), avec les résultats de chaque promotion au test cognitif mis en place par l'équipe de recherche de l'institut des sciences cognitives. Un quatrième fichier du même acabit pour l'année 2018/2019 nous est parvenu au milieu de l'étude.

MathsJetons_2015-2016.xlsx : Pour l'année 2015/2016.

MathsJetons_2015-2016.xlsx : Pour l'année 2016/2017.

MathsJetons_2016-2017.xlsx : Pour l'année 2017/2018.

Jetons2019.xlsx : Pour l'année 2018/2019.

Chaque jeu de données représente les résultats question par question (en comptant les sous-questions) des élèves ainsi que leur catégorie pédagogique et des informations telles que l'encadrant, le niveau scolaire, la langue natale, l'âge, le type de classe (mélangé entre plusieurs section ou pas), l'année de passage du test et leur école. Il y a 10 questions divisées en sous-questions, ce qui fait un total de 34 réponses. Chaque question est indépendante et pour répondre à la sous-question suivante il faut une bonne réponse à la sous-question précédente, sauf pour la question 4, toutes ses sous-questions sont indépendantes. Une bonne réponse correspond à un 1 et une mauvaise réponse à un 0, sauf pour la réponse à la question 1 qui est la valeur de comptage maximale de l'enfant. Ici les élèves viennent tous d'une école située en REP+.

1.4 Nettoyage des données

Les données ayant déjà été travaillées lors d'un précédent stage, le travail de nettoyage nécessaire n'a pas été excessif. Il nous a fallu tout de même renommer certaines variables pour les rendre plus lisibles et cohérentes entre elles, supprimer certaines questions car elles n'avaient été posées qu'à certaines classes... Les questions étant posées de manière à ce qu'au sein d'une même tache, il faille réussir les questions dans l'ordre pour passer à celles plus dures nous avions beaucoup de valeurs manquantes dès qu'une question était un peu difficile. Nous avons donc décidé de changer ces valeurs manquantes et les considérer comme une question que l'élève n'aurait pas réussie (donc remplacer NA par 0). Et pour ne pas passer à coté de l'information : "n'a pu aller plus loin", nous avons créé deux jeux de données : un vectoriel (chaque question devient le regroupement des résultats au sous questions. ex : Q2.1 = 1; Q2.2 = 0 devient Q2 = 10), un composé de scores correspondant à la somme des questions au sein d'une même tâche (pour le même exemple que le jeu de données "vectoriel" nous aurions Q2 = 1+0 = 1).

1.5 Recodage des variables

Afin de ne pas influencer notre jugement sur nos résultats, il a été décidé de rendre anonyme les pédagogies enseignées pour chaque classe. Chaque pédagogie fut donc renommée en "P1" et "P2". De ce fait nous n'avons pas pu privilégier une pédagogie plus qu'une autre subjectivement parlant. Aux 3 quarts du projet environ, nous avons reçu les données de nouveaux individus, une cinquantaine, et avons par la même occasion décidé d'enlever cet anonymat. Notre travail étant déjà réalisé, seul l'interprétation sur le jeu de données comportant les nouveaux individus en plus importe. Comme dit précédemment les questions sont divisées en sous questions, ces questions sont regroupables en groupe : un groupe que nous appellerons "variable au-dela", un groupe "variable outils" puis un groupe "variable objet". Chacun de ces groupes fait appelle à une tache pédagogique en particulier.

Variable Au-delà : est calculée en faisant la somme du nombre de bonnes réponses aux questions 2.3, 3.2, 4.2a, 4.2b, 4.2c, 4.2d, 5.2, 6.2, 8.6, 8.7, 8.8 et 8.9. La première question concernant la capacité à compter loin est prise en compte, les individus sachant compter au-delà de 7 ont un point en plus. $audela = 2.3 + 3.2 + 4.2a + 4.2b + 4.2c + 4.2d + 5.2 + 6.2 + 8.6 + 8.7 + 8.8 + 8.9 + \mathbb{1}_{T1>7}$

Variable Outils : est calculée en faisant la somme du nombre de bonnes réponses aux questions des taches 4, 5 et 6 soit les questions des taches sur le surcomptage, la création d'une collection et la comparaison de deux collections. $outils = 4.1a+4.1b+4.1c+4.1d+4.2a+4.2b+4.2c+4.2d+5.1+5.2+6.1+6.2$

Variable Objet : est calculée en faisant la somme du nombre de bonnes réponses aux questions des taches 1, 2, 3 et 8 soit les questions des taches sur la capacité à savoir compter, à dénombrer une collection et à constituer une collection d'objets. La question 1 est gérée par palliés. Nous avons séparé les individus en 5 groupes : ceux qui savent compter jusqu'à 3, puis de 4 à 7, de 8 à 10, de 11 à 16, enfin ceux qui savent compter au delà de 16. Respectivement pour chaque catégorie nous leur avons donné un score de 0, 0.3, 0.6, 0.9 et 1.2. $objet = 1.1 + 1.2 + 2.1 + 2.2 + 3.1 + 3.2 + 8.1 + 8.2 + 8.3 + 8.4 + 8.5 + 8.6 + 8.7 + 8.8 + 8.9 + \mathbb{1}_{T1\in[0;3]} * 0 + \mathbb{1}_{T1\in[4;7]} * 0.3 + \mathbb{1}_{T1\in[8;11]} * 0.6 + \mathbb{1}_{T1\in[12;16]} * 0.9 + \mathbb{1}_{T1>16} * 1.2$

2. Méthodologie

Afin de répondre au mieux à notre problématique nous avons fait le choix d'utiliser plusieurs méthodes statistiques différentes pour analyser nos données. Nous avons dans un premier temps utilisés deux méthodes qui permettent de résumer l'information globale du jeu de données : l'analyse factorielle, et la classification ascendante hiérarchique (pour faire des regroupements de variables). Puis dans un but prédictif nous avons utilisé la régression logistique mixte, la régression linéaire mixte et les forêts aléatoires. Plusieurs tests statistiques ont été réalisés en parallèle.

2.1 Analyse factorielle

Les méthodes d'analyse factorielle que nous avons utilisées ici sont l'analyse des correspondances multiples (ACM) et l'analyse en composantes principales (ACP), qui sont des méthodes de synthétisation du nombre de dimensions pour les données qualitatives et quantitatives. Cela nous permet d'appréhender plus rapidement le jeu de données, et avoir une première idée de ce qui différencie les individus entre eux (ou ce qui les rapproche). L'ACM permet dans un nuage à N dimensions, en cherchant les axes orthogonaux qui maximisent la variance entre les individus, de résumer ces N dimensions en 4 voire 5 dimensions. l'ACM a été réalisée sur les données vectorisées (regroupement des réponses aux sous-questions en un vecteur par question, question 1 exclue), celles ci ont été prises comme variables actives (celles qui définissent le placement des individus sur le graphique) et les variables portant sur la pédagogie, la question 1, et l'âge en illustratives (ajoutée après le placement des individus sur le graphique). Le principe était le même pour l'ACP qui a été faite ensuite.

2.2 Classification Ascendante Hiérarchique

N'ayant aucune information au préalable sur le thème des questions, leur regroupement... etc Mais sachant que certaines questions faisaient appelle aux mêmes compétences. Nous avons utilisé une variante de la classification ascendante hiérarchique (CAH) afin de partitionner nos variables. Nous avons aussi utilisé la CAH classique qui consiste à regrouper les individus selon leur points communs cela en partant d'une inertie interclasse maximale (inertie = somme des variances), pour arriver à une inertie interclasse de 0. Nous avons utilisé la CAH classique afin de partitionner nos individus dans un but descriptif.

2.3 Arbre de régression

L'arbre de régression est une technique d'apprentissage supervisé, qui permet en analysant un grand nombre de données, de prédire une variable à expliquer. Ils sont beaucoup utilisés dans le domaine du marketing, et plus récemment dans le domaine du machine learning (apprentissage automatique). Dans un premier temps il s'agit d'exprimer la variable à expliquer en fonction d'un maximum de variables explicatives, puis d'élaguer l'arbre afin de minimiser l'erreur, soit l'écart entre la valeur prédite et la valeur réelle. Cela revient donc à faire une régression logistique sur les données, puis d'appliquer l'algorithme de construction d'arbre à partir des résultats.

2.4 Modèle linéaire mixte

Les modèles mixtes linéaires sont une extension des modèles linéaires simples permettant des effets fixes et aléatoires. Ils sont particulièrement utilisés lorsqu'il n'y a pas d'indépendance dans les données, telles qu'elles résultent d'une structure hiérarchique.

$y_{kj} = \beta_0 + \beta_1 x_1 + ... + \beta_j x_j + \beta_k + \varepsilon_{ij}$ sachant $k : k_{ieme}$ *groupe de l'individu* ; $j : j_{ieme}$ *variable*

2.5 Régression logistique avec effet mixte

Un GLMM (pour Generalized Linear Mixed Models) est dit “mixte”, car il comporte au moins un effet dit “fixe” (la variable dont on souhaite évaluer l’effet, ici les *Pédagogie*, *Age* et *Année Scolaire*) et au moins un effet dit “aléatoire” (la variable de regroupement, ici *newClasse* ou *Group*). Les effets aléatoires ne sont pas évalués, ils servent seulement à indiquer au modèle que les données ne sont pas indépendantes pour une boite donnée. C’est ce qui permet à la déviance résiduelle d’être bien estimée, et ainsi à l’erreur standard des paramètres de ne pas être biaisée, et aux final d’obtenir des résultats fiables.

$P_k(1|X) = \frac{e^{\beta_0+\beta_1 x_1+\ldots+\beta_j x_j+\beta_k+\varepsilon_{ij}}}{1+e^{\beta_0+\beta_1 x_1+\ldots+\beta_j x_j+\beta_k+\varepsilon_{ij}}}$ sachant $k : k_{iem}$ *groupe de l'individu*

3. Analyse exploratoire

3.1 Analyse Préliminaire

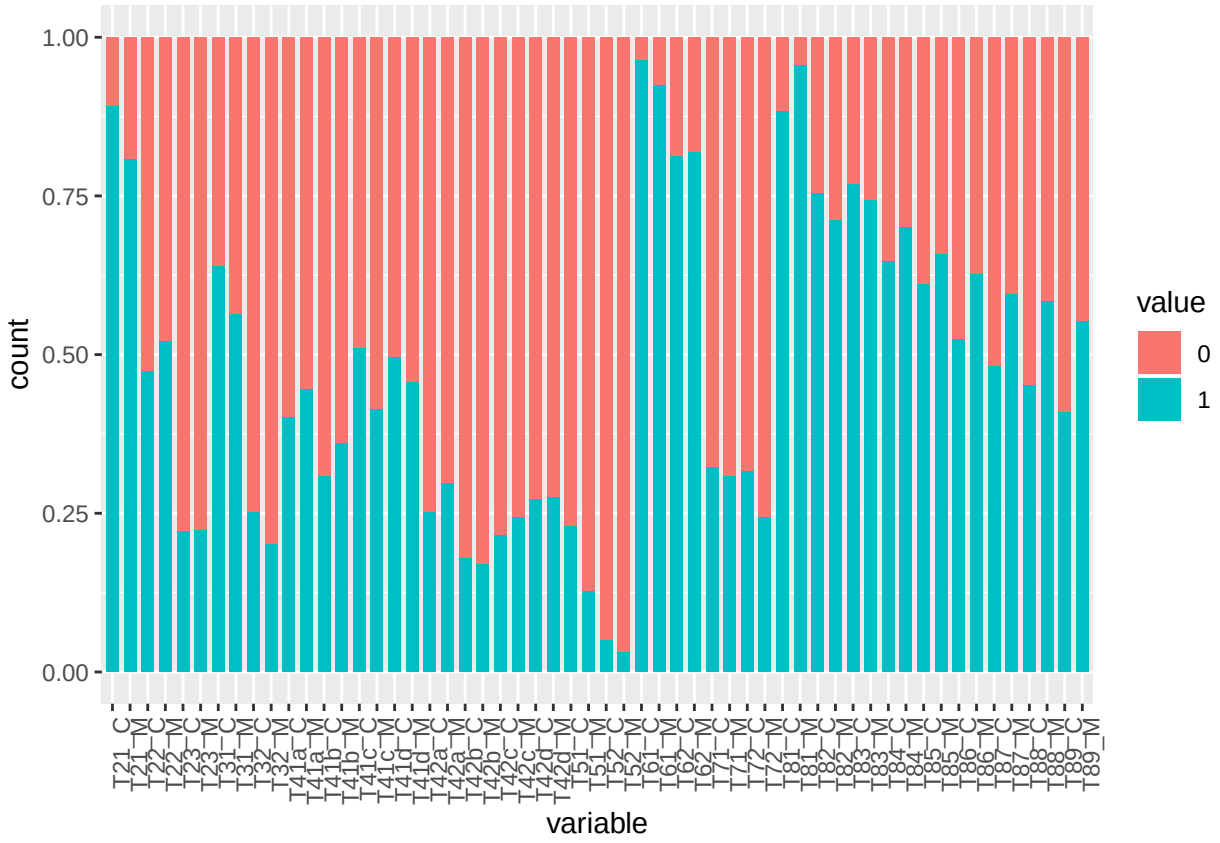


Figure 1: Réussite à chaque question par pédagogie

Le graphique précédent montre la proportion de bonnes et de mauvaises réponses par question et par pédagogie. On peut voir que l'écart entre les deux pédagogies est plus ou moins important selon les questions, mais ce n'étant pas toujours la même pédagogie qui est supérieure à l'autre, cela semble équilibré globalement. Excepté pour les dernières questions, à partir de la question 8.4, la pédagogie Montessorienne est toujours devant la Conventionnelle.

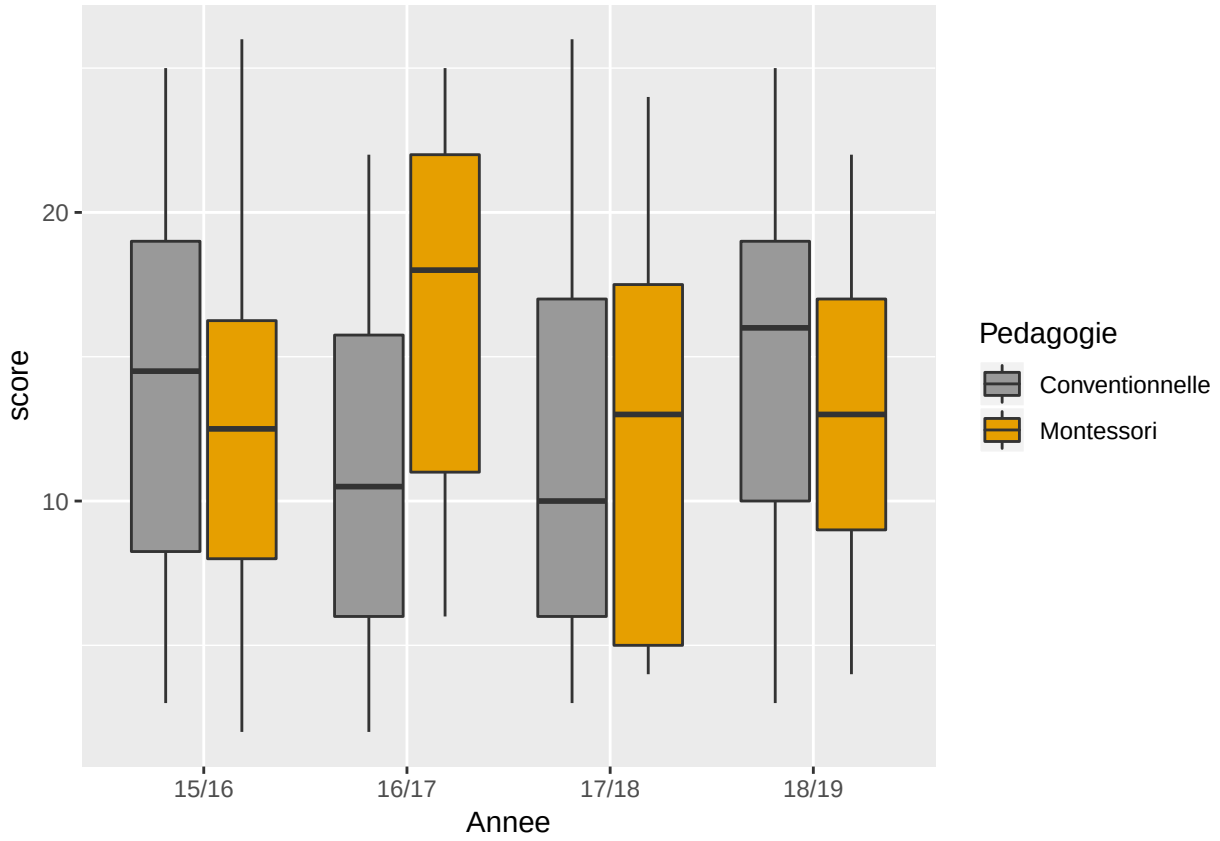


Figure 2: Score total par année par pédagogie

En ne regardant que certaines années on peut voir clairement une différence entre la pédagogie montessorienne et la pédagogie conventionnelle vis à vis du nombre de bonnes réponses. Toutefois lorsqu'on regarde la vue d'ensemble, on peut voir que selon les années, une fois la pédagogie montessorienne est supérieure à la conventionnelle, parfois c'est l'inverse. On peut donc s'attendre à ce qu'on ne puisse pas prédire quelle pédagogie permet d'obtenir un meilleur score global.

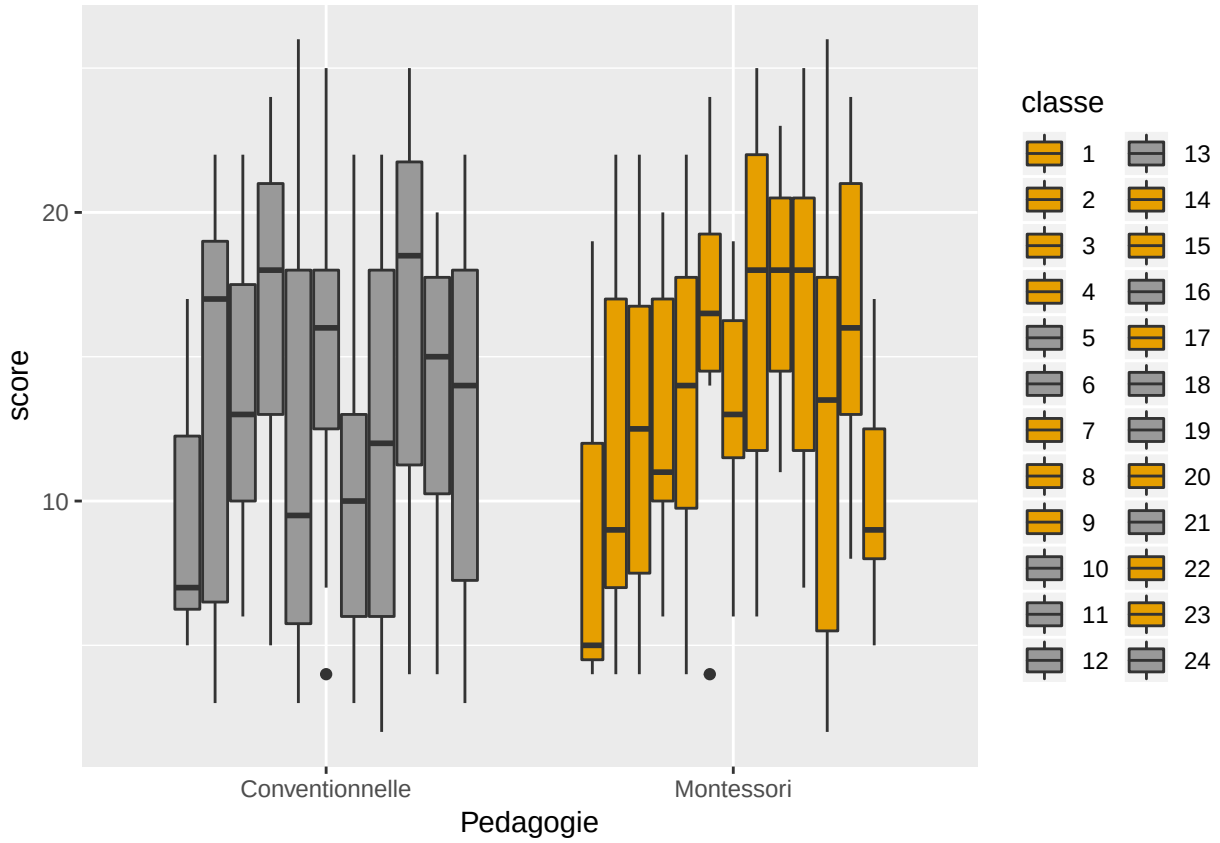
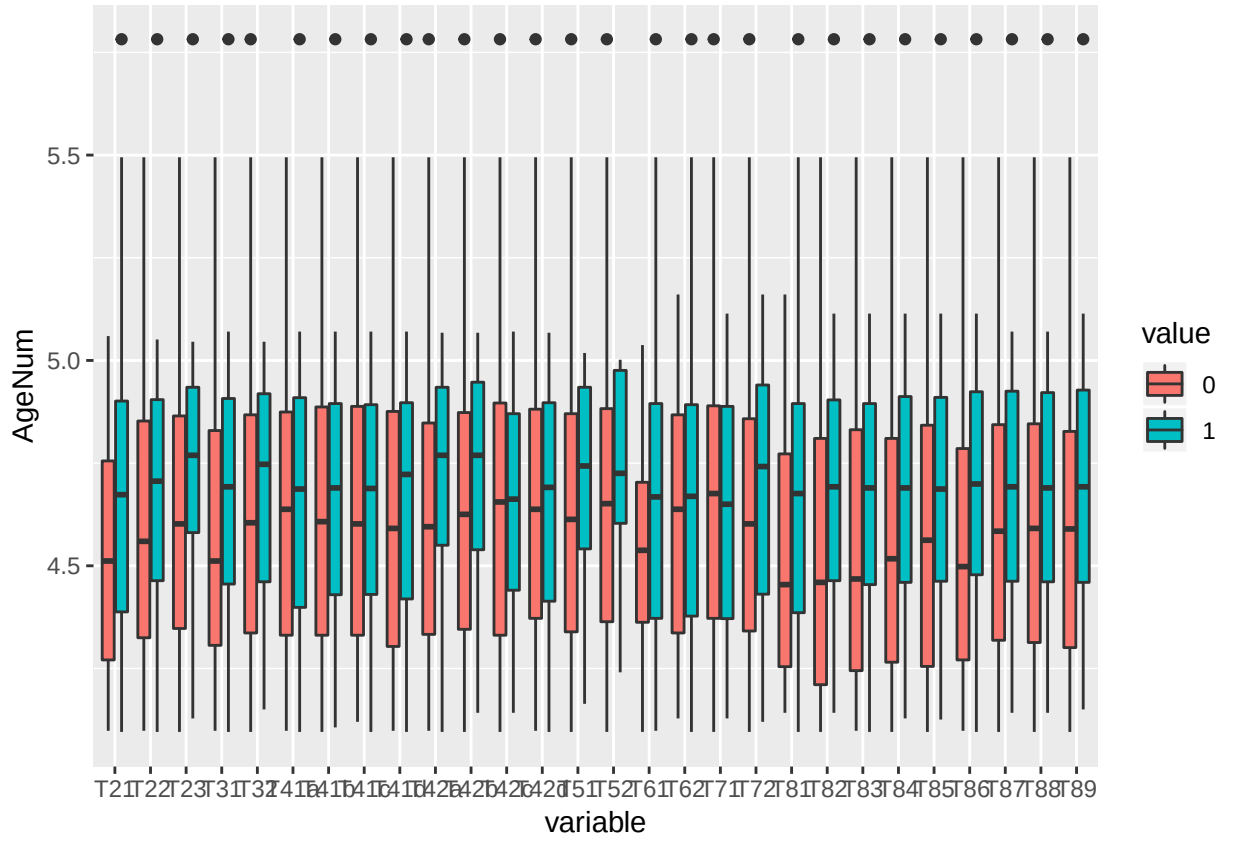


Figure 3: Score total par pédagogie par classe

De plus il semblerait que la classe dans laquelle se trouve l'individu agisse sur le résultat de l'élève. Comme le montre le graphique précédent, la répartition des résultats totaux varie énormément d'une classe à l'autre. Enfin, après réalisation d'un test de Chi2 entre chaque question et la variable concernant le groupe des élèves, on peut observer une dépendance entre leur réussite et le groupe associé.



Enfin le graphique précédent montre que l'âge semble avoir une influence sur la réussite ou non d'une question, et ce, quelque soit la pédagogie enseignée.

Il faudra donc par la suite tenir compte de ces trois facteurs, comme des variables qui peuvent avoir un effet sur la réussite de l'élève en parallèle de la pédagogie enseignée.

3.2 Multivariée

Afin de traiter l'information présente dans le jeu de données de la meilleure façon, nous avons procédé à 2 analyses multivariées : 1 sur le jeu de données qualitatif sous forme de vecteurs, et 1 sur le jeu de données quantitatif sous forme de scores.

3.2.1 Analyse des Correspondances Multiples

La réalisation d'une ACM comme première approche sur le jeu de données a permis de mieux comprendre ce qui différencie les individus dans notre jeu de données et à la fois d'avoir un premier résultat sur la différence entre les deux pédagogies selon cette méthode.

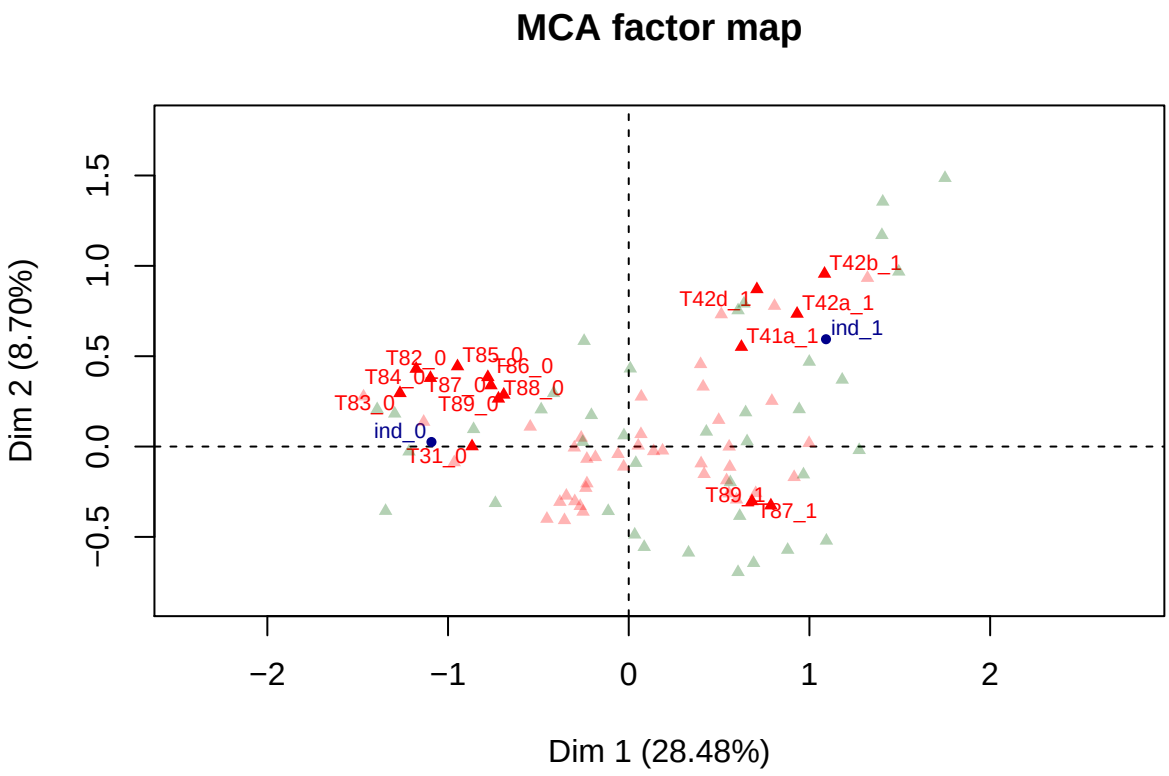


Figure 4: Graphique des modalités sur le plan principal

Le graphique précédent représente les 15 modalités qui contribuent le plus au placement des individus sur le plan principal. On peut donc voir que la dimension 1 oppose des modalités qui concernent la réussite à une question (avec un "_1" à la fin), à droite, à des modalités qui concernent l'échec d'une question (avec un "_0"), à gauche. Plus un élève réussira le questionnaire, plus il se trouvera à droite sur le graphique des individus. Cette interprétation est confirmée par l'ajout de deux individus fictifs : ind_1 et ind_0, qui comportent respectivement des succès à toutes les questions et des échecs à toutes les questions. L'individu ayant réussi en totalité le questionnaire se trouve à droite alors que l'individu ayant raté en totalité le questionnaire se trouve à gauche. De plus nous pouvons voir que les questions qui discriminent le plus la réussite ou non de l'examen sont les questions concernant l'addition, la soustraction et la reconnaissance des chiffres (questions 4 et 8) (de part leur forte contribution). Les caractéristiques qui discriminent le plus les élèves de maternelle seraient donc ces trois capacités. Toutefois cette première analyse n'aura pas permis de différencier les deux pédagogies, la variable projetée en supplémentaire sur le plan principal n'est pas significativement liée à celui ci.

Dans un second temps nous avons refait une ACM mais cette fois ci sur les données vectorielles. Cela afin de prendre en compte la succession de certaines questions qui se regroupent en "compétences".

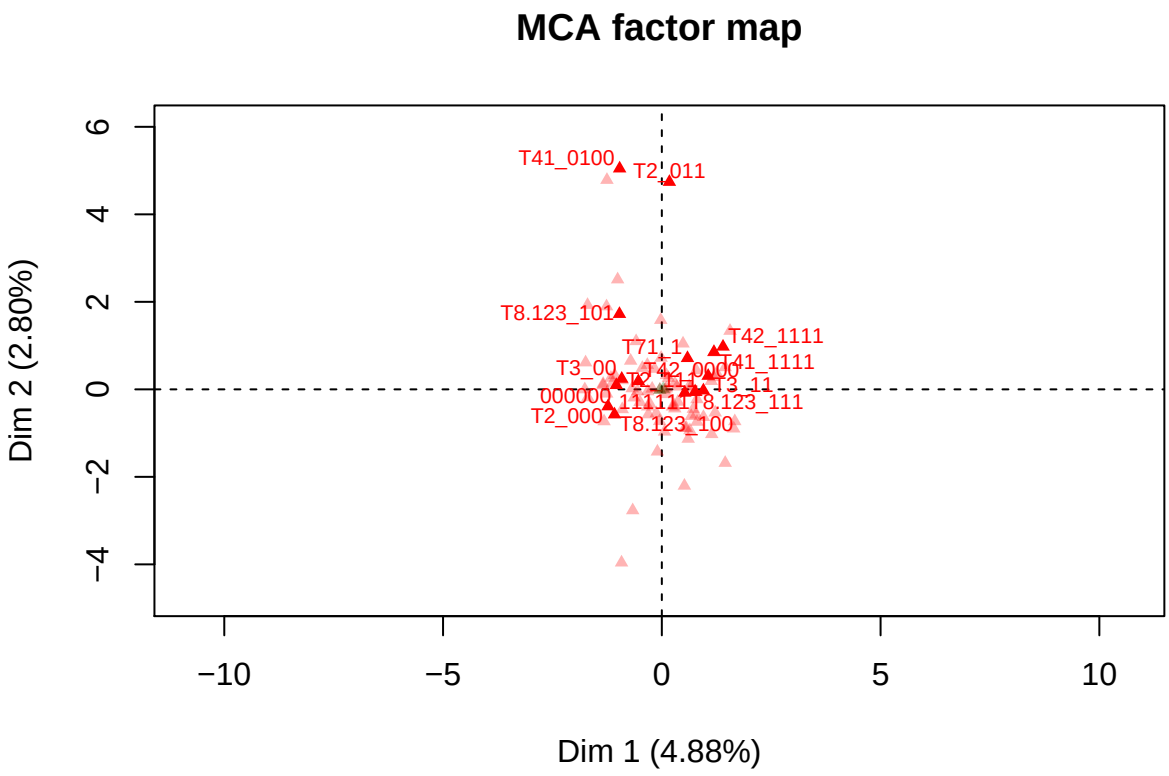


Figure 5: Graphique des modalités sur le plan principal

A nouveau la première dimension oppose les individus ayant réussit la totalité (ou la majorité pour certaines) des questions à droite à ceux qui n'en ont réussi aucune à gauche. Nous ne pouvons pas non plus observer de différence significative entre les 2 pédagogies. On peut voir cette fois ci avec plus de précision les questions qui discriminent la réussite au questionnaire. Ce sont Les question 4, 3, 8 et 5. Soit les taches sur la capacité à soustraire et additionner, la constitution d'une collection, la reconnaissance des chiffre et la création d'une collection équipotente.

Dans ces deux cas nous avons pu aussi observer un lien significatif entre les variables qualitatives, portant sur les réponses à la question 1 et l'âge de l'élève, et le placement des individus sur la première dimension. En conséquent on peut dire qu'il existe un lien entre la réussite à l'examen et le fait qu'un enfant sache compter "loin" et dans une moindre mesure, qu'il soit âgé.

Enfin nous avons voulu voir si en classifiant les individus suite à l'ACM nous obtenions des groupes d'individus propre à une pédagogie ou non. Pour cela nous avons utilisé la classification ascendante hiérarchique (CAH).

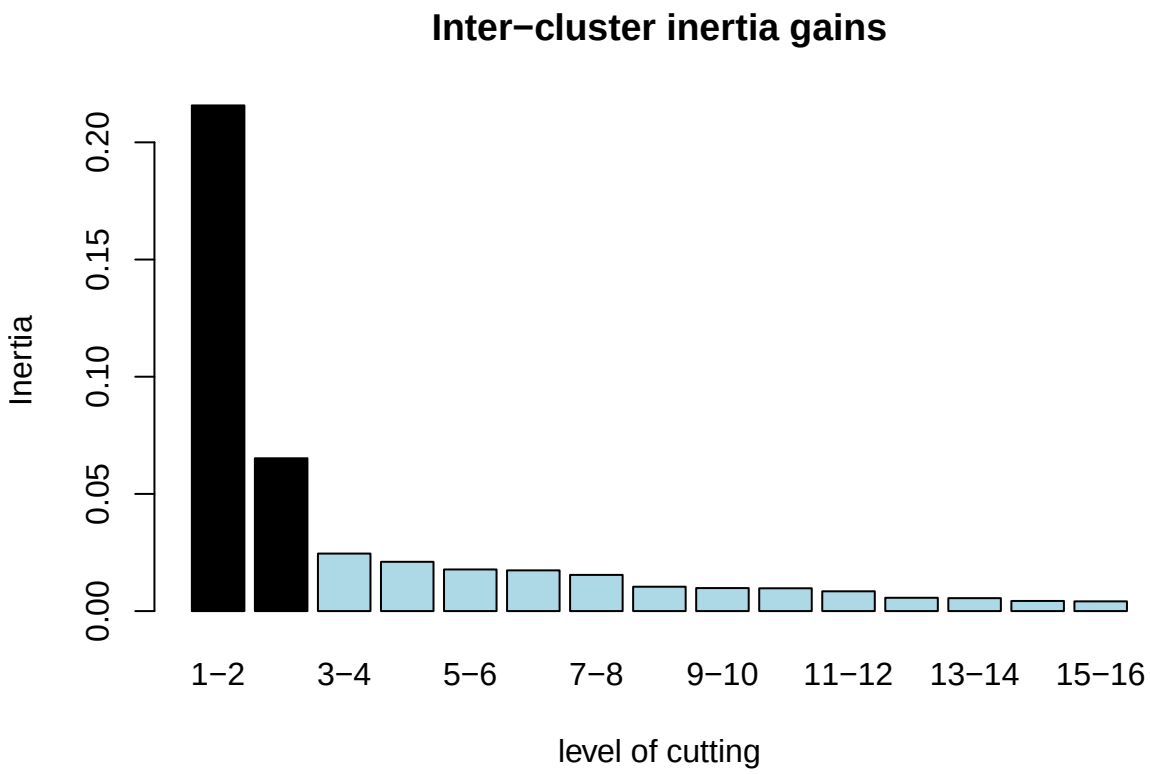


Figure 6: Diagramme des gains d'inertie

Nous pouvons observer un "saut" à la troisième classe, donc nous faisons le choix de retenir trois classes pour la CAH. Mais leur composition ne montre aucune sureprésentation d'une pédagogie plus que l'autre. Le test de chi2 entre la variable concernant la pédagogie et celle concernant la classe n'est pas significatif. Une fois de plus cela ne permet donc pas de montrer une liaison entre la pédagogie et ce qui discrimine nos classe. Au final nous obtenions une classe d'individus qui a une majorité d'échecs, une d'individus qui échouent sur les questions 4.2 (soit une incapacité à additionner ou soustraire au delà de 1), et une qui d'individus qui réussissent globalement.

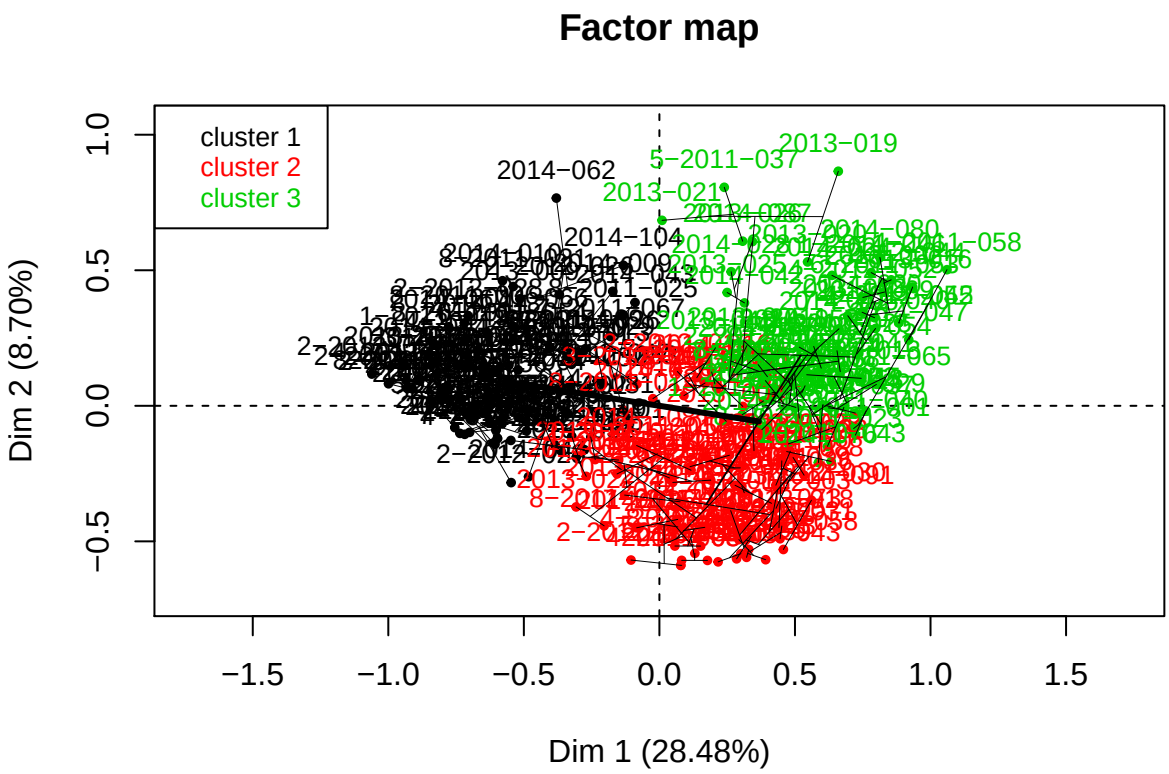


Figure 7: Représentation des classes sur le premier plan de l'ACM

3.2.2 Analyse en Composantes Principales

La réalisation d'une ACP faisant suite à l'ACM a pour but d'étudier le jeu de données différemment. En effet nous avons étudié cette fois ci le jeu de données concernant les scores. Soit un jeu de données quantitatif. Afin de ne pas perdre d'informations nous avons dans un premier temps observé la matrice des corrélations entre les variables de notre jeu de données. Car plus les variables seront corrélées entre elles, plus l'ACP ne montrera que celles ci.

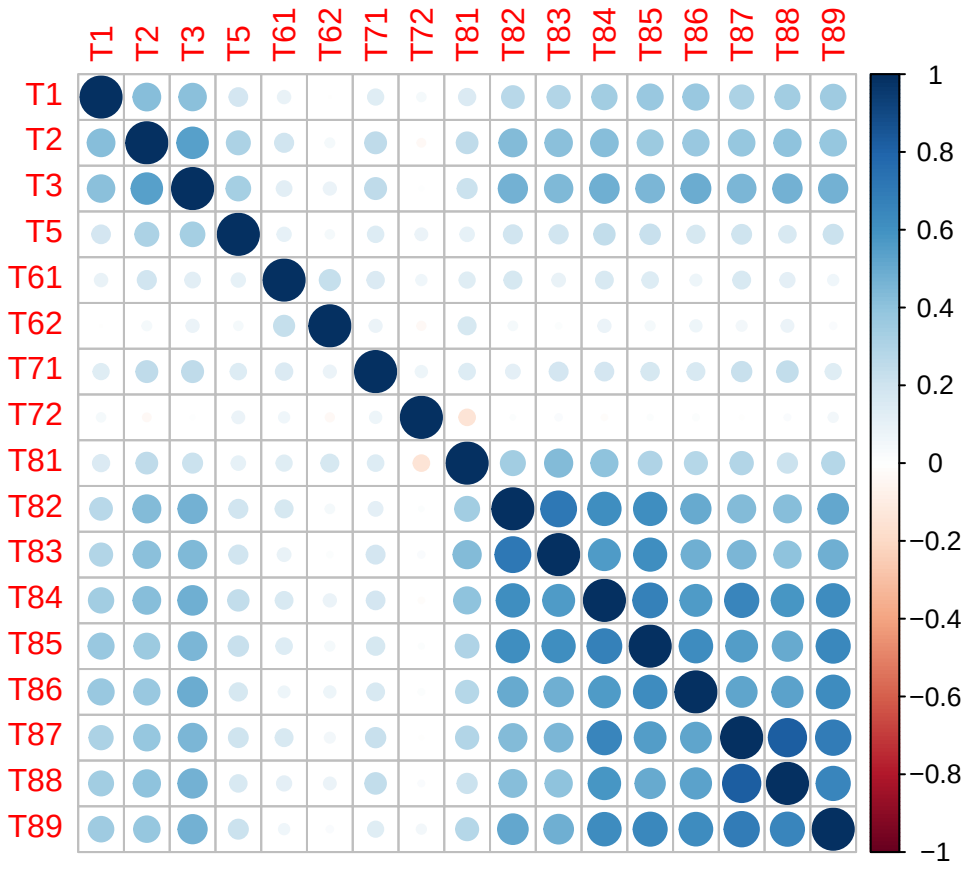


Figure 8: Matrice des corrélations

Nous avons donc fait le choix de regrouper les questions 8 en 2 groupes, aux vues des résultats. Un groupe comprenant les questions 8.1, 8.2 et 8.3, et un comportant les autres questions 8. Reconnaître les chiffres 1, 3 et 2 n'entraînerait donc pas la reconnaissance des autres chiffres jusqu'à 9 forcément.

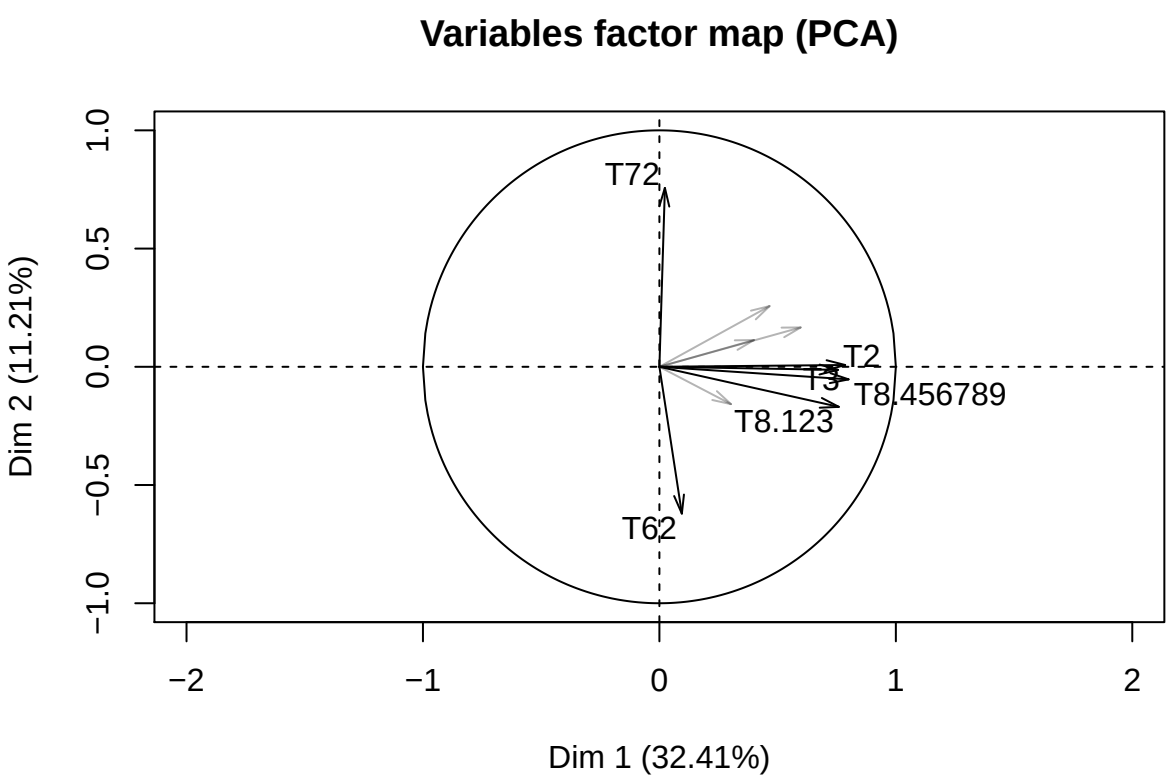


Figure 9: Cercle des corrélations du plan principal

La réalisation de l'ACP nous permet donc de voir que les individus se diffèrent sur la première dimension selon les questions sur le dénombrement d'une collection, la constitution d'une collection et la reconnaissance des chiffres (2,3, et 8). Alors que la dimension 2 les différents selon la corrélation négatives des questions sur la comparaison de deux collections et sur la réunion de deux collections (7.2 et 6.2). A nouveau nous n'observons pas de lien significatif entre la pédagogie enseignée et le placement des individus sur les axes factoriels.

3.3 Classification Ascendante Hiérarchique des variables

N'ayant au début de notre analyse, aucune information sur le thème des questions, leur regroupement... etc Mais sachant que certaines questions faisaient appel aux mêmes compétences. Nous avons utilisé une variante de la classification ascendante hiérarchique (CAH) afin de partitionner nos variables. La CAH est une méthode de classification qui permet de regrouper des individus sein d'une même classe et qu'ils soient le plus semblables possible tandis que les classes soient elles, le plus dissemblables possible.

Nous allons appliquer cette méthode sur les jeux de données en isolant les pédagogies pour comparer les regroupements.

Procédons d'abord à l'isolation de la Pédagogie 1.

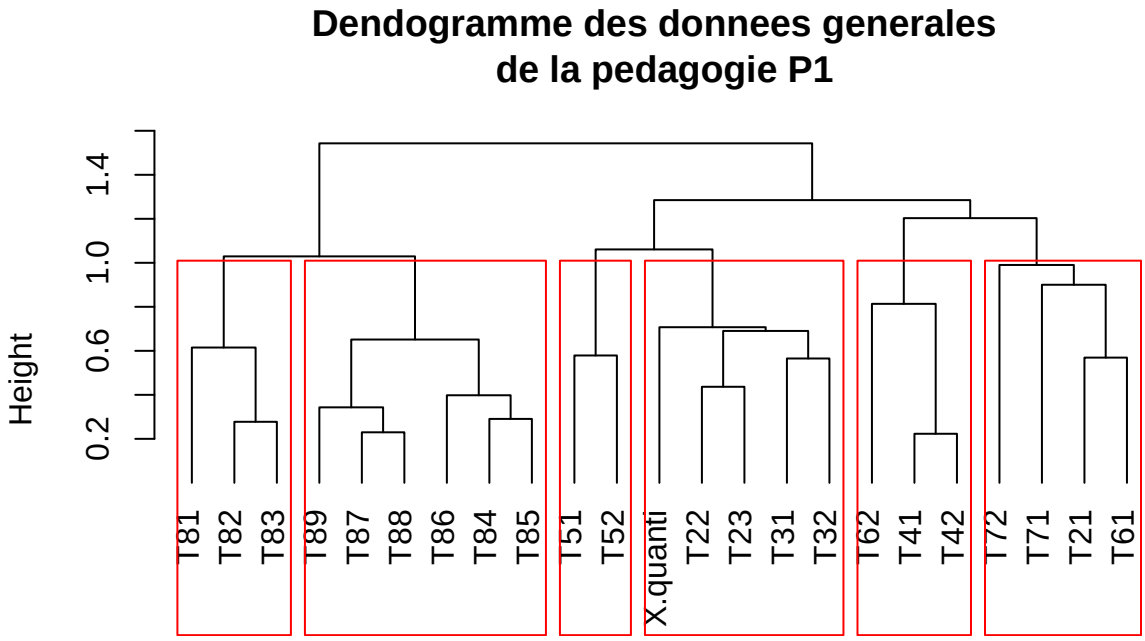


Figure 10: Dendogramme des donnees generales de la pedagogie P2

Ici X.quanti correspond a T1. Nous pouvons observer ici plusieurs regroupement redondant. Le regroupement des questions T81, T82 et T83 et celui des questions T84, T85, T86, T87, T88, T89. Les question T1, T2 et T3 sont aussi fortement attirées, on retrouve en partie la variable “Objet”.

Comparons maintenant avec les regroupements de la Pédagogie 2.

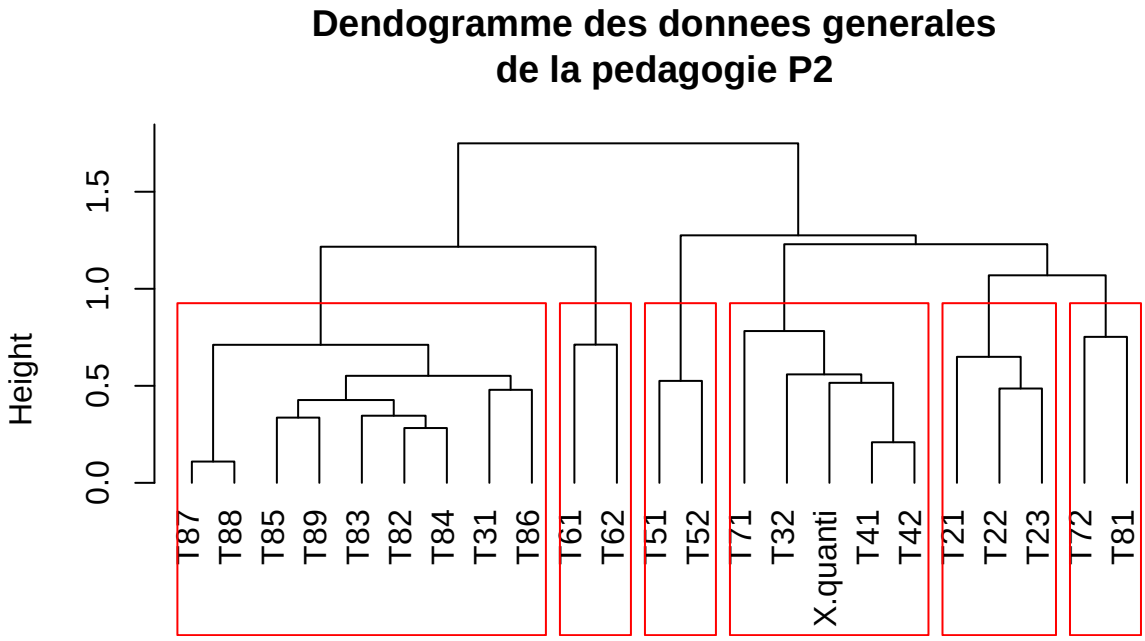


Figure 11: Dendogramme des donnees generales de la pedagogie P2

Nous observons en regroupement des questions T82, T83, T84, T85, T86, T87, T88, T89. La T81 étant séparée du reste. On voit aussi apparaître deux couples de questions, T61 et T62 ainsi que T51 et T52. Ici nous ne voyons aucunes variable prédéfinie ressortir véritablement.

On retrouve plus de similarités entre les classification de la pédagogie 2 qu'entre celles de la Pédagogie 1. Et on remarque que les regroupement qui sont stables entre les changements de jeux de données sont principalement ceux liés aux sous questions de la T8. Pour résumer, les groupes qui ressortent sont pour la Pédagogie 1: (T81, T82 , T83), (T84, T85, T86, T87, T88, T89) et (T1,T2, T3) Et pour la Pédagogie 2: (T82, T83, T84, T85, T86, T87, T88, T89), (T61, T62) et (T51, T52).

Seul la variable 'Objet' est en parti retrouvée et seulement dans le cas de la Pédagogie 1.

4. Modélisation linéaire des résultats à chaque question.

Nous avons voulu modéliser les résultats des élèves à chaque question en fonction de ses caractéristiques individuelles (y compris la pédagogie suivie bien-sûr) pour voir si la pédagogie a un effet sur la réussite au test. Les résultats de certaines questions sont binaires (1 pour réussite, 0 pour échec), la plupart sont concernées (toutes sauf la question 1 qui consiste à faire compter l'enfant le plus loin possible). Les résultats à l'autre question prennent des valeurs continues : la question 1 est dans ce cas mais aussi les résultats que nous avons construit en regroupant des questions (au-delà, objet et outils qui sont expliqués en **1.5**).

Intuitivement, nous sommes d'abord tentés de modéliser les résultats binaires par une régression logistique et les résultats continus par une régression linéaire. Mais, notamment grâce à l'analyse préliminaire faite précédemment, nous soupçonnons un effet aléatoire de la variable classe sur la réponse des élèves, c'est à dire que selon la classe (pas la pédagogie suivie) dans laquelle se trouve l'élève sa réponse sera différente. Cet effet semble être aléatoire (elle n'est pas modélisable linéairement). Il existe justement des méthodes statistiques spécialement faites pour ces cas : il s'agit des **modélisations mixtes**. Ces dernières prennent bien en compte les effets aléatoires tels la différence de résultats selon la classe de nos élèves.

Nous avons donc réalisé nos modélisations avec des **modèles linéaires mixtes** (pour les résultats continus) et des **regression logistique mixte** (pour les résultats binaires).

4.1 Modélisation des questions à résultat continu par modèle linéaire mixte

L'analyse exploratoire nous laisse en effet bien penser que les scores des élèves sont répartis différemment selon la classe dans laquelle il étudie, sa promotion et son âge. Mais seule la classe a été retenue comme effet aléatoire. L'âge a lui un effet fixe, la promotion et la pédagogie de l'élève n'ont finalement pas d'effet significatif (il s'agirait d'un effet fixe s'il était significatif).

Nous avons donc réalisé nos régressions linéaires mixtes sur les regroupements de variable en mettant la variable sur le groupe (la classe) comme effet aléatoire. Et les variables sur la pédagogie, l'âge et la promotion et comme effets fixes.

Pour la modélisation de chaque regroupement de questions (au-delà, objet, outils), nous obtenons bien la présence d'un effet aléatoire, mais après sélection de variables et tests statistique, seul l'âge a un effet fixe significatif sur nos variables à expliquer.

Enfin nous avons rajouté un dernier modèle sur la question concernant la capacité à compter loin. L'effet aléatoire est important, mais est justifié par l'ordre de grandeur de la variable en question.

$$
\begin{aligned}
Au-delà &= -6.19 + 2.493 * age + \beta_k + \varepsilon_{ij} \\
Objet &= -2.88 + 1.90 * age + \beta_k + \varepsilon_{ij} \\
Outils &= -0.594 + 0.741 * age + \beta_k + \varepsilon_{ij} \\
Q1 &= -9.554 + 4.667 * age + \beta_k + \varepsilon_{ij} \\
k : k_{ieme}\ groupe\ dans\ lequel\ est\ l'individu\ ;\ j : j_{ieme}\ variable &
\end{aligned}
$$

avec les p.value suivantes pour le coefficient de l'âge : - pour le modèle d'Au-dela : 0.0006 et les coefficients de l'effet aléatoire ont un écart type de 0.4741162 - pour le modèle d'Objet : 0.0007 et les coefficients de l'effet aléatoire ont un écart type de 2.433568e-07 (il ne semble pas que l'effet aléatoire soit nécessaire pour cette variable) - pour le modèle d'Outils : 0.0257 et les coefficients de l'effet aléatoire ont un écart type de 0.05115118 - pour la question 1 : 0.0436 et les coefficients de l'effet aléatoire ont un écart type de 0.5511588

Pour résumer, nous n'avons pas pu modéliser la réussite à certains regroupements de tâches en fonction de la pédagogie enseignée. Toutefois nous observons qu'il est possible de les modéliser en fonction de l'âge de l'élève en prenant l'effet groupe en compte.

4.2 Modélisation des questions à résultat binaire par régression logistique mixte

N'ayant pas eu de résultats concernant la pédagogie lors de la modélisation linéaire mixte précédente, nous avons affiné notre modélisation en réalisant une régression logistique mixte sur les questions à résultat binaire.

Les principaux résultats coïncident en partie avec les résultats précédents. La pédagogie n'est significativement liée qu'à 2 questions, la question Q5.1 (création d'une collection équipotente), qui a un effet aléatoire **groupe** significatif, et la Q8.9 (reconnaissance d'un chiffre particulier. **Attention !** Les questions 8.1 à 8.8 portaient sur la même thématique mais avec un autre chiffre), en complément d'un effet fixe de l'âge. De plus à nouveau plusieurs variables sont modélisables à partir de l'âge de chaque individu (voir annexe chaque modèle).

$$Q51 : P(1|Montessori) = \frac{e^{-1.1884+-0.8875+\beta_k+\varepsilon_{ij}}}{1+e^{-1.1884+-0.8875+\beta_k+\varepsilon_{ij}}}$$

$$Q89 : P(1|Montessori) = \frac{e^{-6.358+0.6404+1.287*age+\beta_k+\varepsilon_{ij}}}{1+e^{-6.358+0.6404+1.287*age+\beta_k+\varepsilon_{ij}}}$$

$$k : k_{ieme}\ groupe\ dans\ lequel\ est\ l'individu\ ;\ j : j_{ieme}\ variable$$

Avec un écart type pour les coefficients de l'effet aléatoire groupe de la Q51 de 0.1416281 et pour la Q89 de 0.1107895

- Le coefficient de l'âge a une p.value de 0.00617 pour la Q89.
- Le coefficient de la Pédagogie a une p.value de 0.0376 pour la Q51 et de 0.02658 pour la Q89.

Il y a donc une prédisposition à réussir la question T89 pour chez les élèves ayant suivant la pédagogie Montessorienne. Inversement pour la T51 qui est en faveur de la pédagogie Conventionnelle.

Etant limité par les packages disponible de R, nous n'avons pas pu faire les courbe ROC de ces deux régressions logistiques mixtes.

5. Forêt d'arbres décisionnels

Les Forêt d'arbres décisionnels sont une méthode d'apprentissage automatique de régression et de classification basée sur la construction d'une multitude d'arbres de décision. Les forêts d'arbres décisionnels utilisent la méthode du bagging. Le bagging consiste à prendre un jeu de données D de taille n, et créer m nouveaux jeux de données D_i de taille n' en échantillonnant D uniformément et avec remise. Ensuite l'algorithme crée à partir de chaque échantillon D_i un arbre de décision pour ensuite agréger ces arbres et obtenir un modèle stable.

Pour prédire une variable quantitative l'agrégation se fait par la moyenne : $G(x) = \frac{1}{m}\sum_{i=1}^{m} G_i(x)$.

Et pour prédire une variable qualitative on procède à une agrégation par le vote : $G(x) =$ Vote majoritaire $(G_1(x), ..., G_m(x))$ où $G_1(x)$ représente un modèle entraîne sur un ensemble D_i.

Nous avons donc appliqué cet algorithme sur deux jeux de données, le jeu de données contenant les variables regroupées et le jeu de données de base contenant toutes les questions une à une.

Nous avons ici la matrice de confusion. Cette matrice nous donne en ligne les données observées et en colonnes les données prédites. Il y a 64 individus issus de la pédagogie Conventionnelle ayant été bien classés et 32 individus ayant été mal classés et considérés comme des Montessori. Ce qui nous donne un taux de mauvaise prédiction pour les individus de la pédagogie Conventionnelle de 33%.

Pour la pédagogie Montessori, 25 ont été bien classés et 42 ont été classés en Conventionnelle à tort. On obtient 63% de mauvaise classification, il y a donc une minorité de bonne prédiction. Nous ne pouvons pas obtenir une prédiction efficace grâce à ce modèle.

	Conventionnelle	Montessori	class.error
Conventionnelle	64	32	0.3333333
Montessori	42	25	0.6268657

Maintenant voyons pour le jeu de données avec les questions générales. La matrice de confusion nous indique qu'il y a 20% de mauvaise prédiction pour la catégorie Conventionnelle et 79% de mauvaise prédiction pour la catégorie Montessori.

	Conventionnelle	Montessori	class.error
Conventionnelle	80	20	0.2000000
Montessori	50	13	0.7936508

6. Test d'équivalence

Jusqu'à présent nous n'avons pas pu établir de lien entre la pédagogie suivie par les enfants de notre étude et l'avancement de leurs capacités cognitives liées aux mathématiques. Si nous n'avons pu établir de lien, nous allons maintenant voir si les deux pédagogies sont significativement équivalentes sur l'assimilation de notions mathématiques (pour les élèves de moyenne section).

6.1 Calcul d'un score total pour chaque élève.

Nous avons choisi de comparer nos deux échantillons sur le score total de chaque élève au test. Pour calculer ce score, nous avons simplement fait la somme des résultats de l'enfant à chaque question. (Rappelons qu'à part la Q1, les questions ont pour résultat 0 ou 1.)

$$Score = \sum_{i=1}^{29} Q_i$$

Où Q_i est le résultat de l'élève à la question i (Donc bien-sûr dans cette formule, la question 4b par exemple ne correspond pas à Q_{4b} mais Q_5).

Nous devons vous préciser que nous avons utilisé le même traitement pour la question 1 que lors de la création de la variable **Objet** pour éviter qu'elle n'ait trop de poids par rapport aux autres. C'est à dire que la question 1 peut prendre les scores suivants : 0, 0.3, 0.6, 0.9 et 1.2 . (0 si $T1 \leq 3$; 0.3 si $T1\epsilon[4;7]$; 0.6 si $T1\epsilon[8;10]$; 0.9 si $T1\epsilon[11;16]$; 1.2 si $T1 \geq 17$).

Le score d'un enfant peut donc aller de 0 à 29.2, leur répartition est représentée sur le graphique ci-dessous.

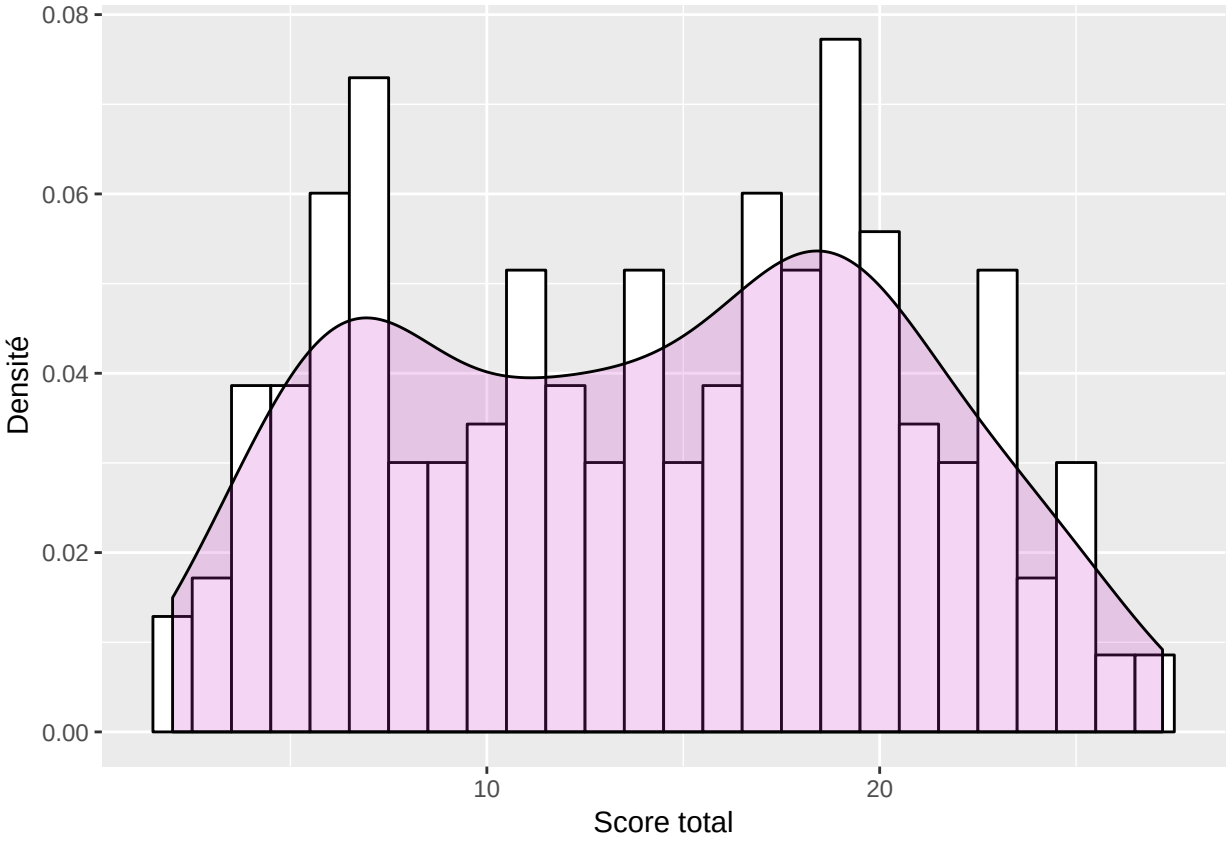


Figure 12: Répartition des scores des élèves au test

Vous pouvez voir deux pics de concentration autours des scores de 7 et de 19. Les résultats de nos élèves ne suivent pas une distribution normale. En voyant ce graphique, ceux qui cherchent absolument une différence pourraient espérer que ces deux pics correspondent chacun à une pédagogie pour en voir une plus efficace pour ce test. Ce n'est pas du tout le cas.

En effet, le graphique juste en dessous montre qu'il y a bien une forte densité autours de 7 et 19 chez les élèves conventionnels et Montessoriens. La seule différence est que nous observons un petit creux de densité entre 7 et 19 chez les élèves conventionnels quand la densité augmente légèrement chez les Montessoriens.

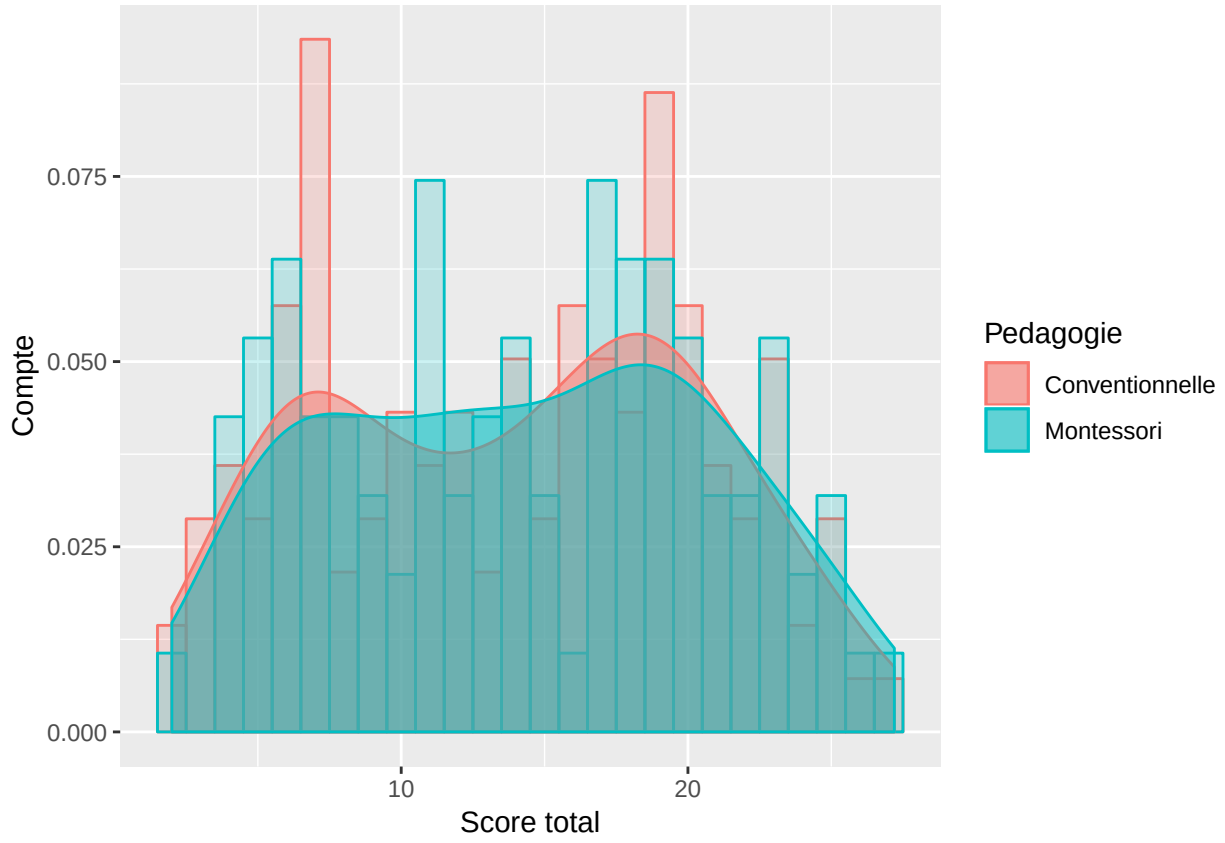


Figure 13: Répartition par pédagogie des scores des élèves au test

6.2 Réalisation du test

Notre test d'équivalence se base sur l'écart entre la moyenne des résultats des élèves conventionnels au test et celle des élèves suivants un enseignement Montessori.

Pour réaliser ce test d'équivalence, nous avons besoin de définir deux bornes (une inférieure et une supérieure). Si l'intervalle de confiance (à 95%) de l'écart entre les deux moyennes franchit au moins l'une des deux bornes, alors nous ne pourrons pas conclue à l'équivalence des deux échantillons (donc l'équivalence de résultat entre les deux méthodes d'éducation). En revanche, nous pourrons conclure que les deux méthodes sont aussi efficace l'une que l'autre pour réussir le test cognitif qui a été soumis si l'intervalle ne franchit pas l'une des bornes.

Pour choisir ces deux bornes, nous nous sommes fiés à une "règle" qui dirait que deux élèves ont à priori un niveau différent sur une notion particulière s'ils ont deux points sur 20 d'écart à un contrôle portant sur cette notion.

Nous adaptons donc cet écart au test soumis aux élèves : avec la transformation réalisée sur la question 1, ce test peut donner une note maximale de 29. Un écart de 2.9 serait donc révélateur d'une différence de niveau sur les notions abordées.

Une fois le test réalisé (résultats juste après), nous obtenons une IC 95% de notre écart à égal à [-2.025;1.417] et une p-valeur $< 0.2\%$ c'est à dire que nous pouvons rejeter l'hypothèse de différence entre les deux échantillons. Nous concluons donc de ce test **que suivre la pédagogie Montessori ou la pédagogie conventionnelle n'aura pas d'effet en moyenne section de maternelle sur l'assimilation des thèmes**

abordés dans le test soumis par l'équipe du projet Cogmont

Les résultats du test statistique :

```
## TOST results:
## t-value lower bound: 2.97    p-value lower bound: 0.002
## t-value upper bound: -3.67   p-value upper bound: 0.0002
## degrees of freedom : 231
##
## Equivalence bounds (raw scores):
## low eqbound: -2.9
## high eqbound: 2.9
##
## TOST confidence interval:
## lower bound 95% CI: -2.025
## upper bound 95% CI:  1.417
##
## NHST confidence interval:
## lower bound 97.5% CI: -2.275
## upper bound 97.5% CI:  1.666
##
## Equivalence Test Result:
## The equivalence test was significant, t(231) = 2.971, p = 0.00164, given equivalence bounds of -2.90

##

##
## Null Hypothesis Test Result:
## The null hypothesis test was non-significant, t(231) = -0.348, p = 0.728, given an alpha of 0.025.

##

##
## Based on the equivalence test and the null-hypothesis test combined, we can conclude that the observe

##
```

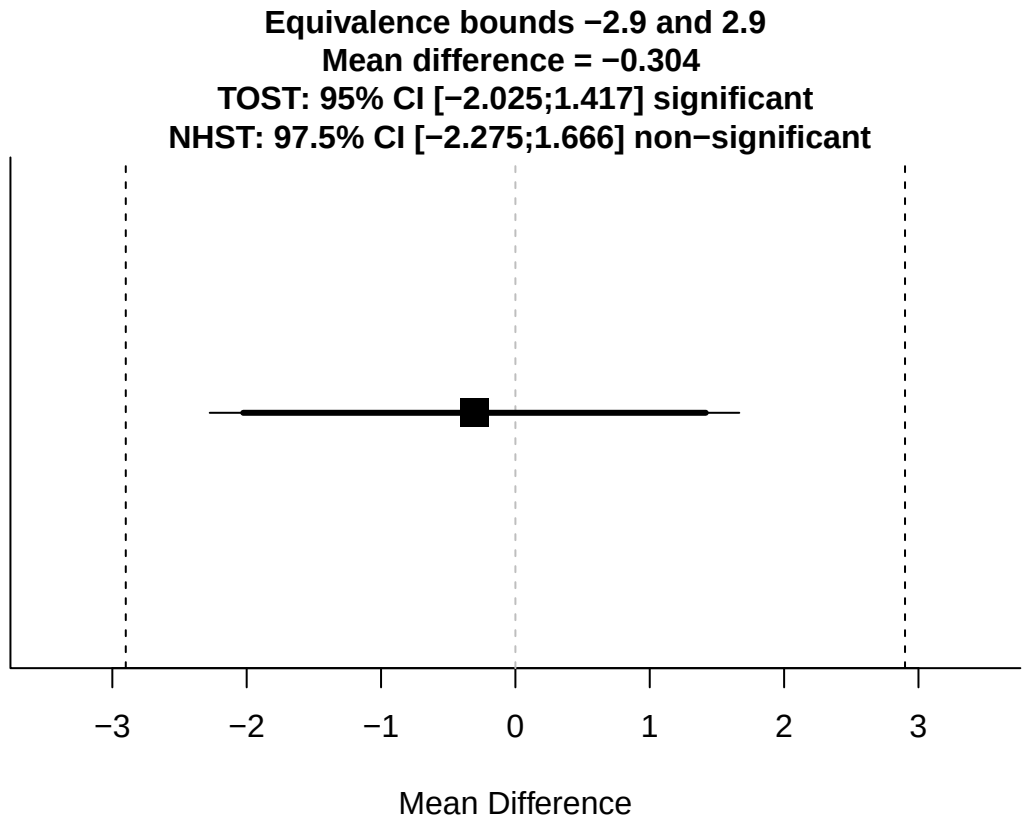


Figure 14: Résultat du test d'équivalence des moyennes des scores au test soumis par l'équipe

Conclusion

Afin d'étudier les éventuelles différences ou non en mathématiques entre les élèves suivants une pédagogie Montessorienne et ceux suivants Conventionnelle, nous avons commencé par une brève étude exploratoire permettant de comprendre nos données et de repérer les possibles liens entre les variables. Par la suite une analyse prédictive sur les réponses aux questions en fonction de la pédagogie aurait pu permettre de différencier ces deux pédagogies. Toutefois, les résultats de cette analyse sont peu concluants. Seuls 2 modèles différencient les 2 pédagogies : la modélisation des réponses aux questions concernant la création d'une collection équipotente et la reconnaissance d'un chiffre particulier. Il serait intéressant de continuer cette étude avec un nombre d'individus plus important voir si de nouveaux liens apparaissent.

Annexes

Application Shiny

Onglet des Classifactions ascendantes hiérarchiques des variables

Cet onglet montre les dendrogrammes de nos classifications hiérarchiques. Nous avons la possibilité de choisir la classification ainsi que les regroupements de variables selon le nombre de classes que nous désirons.

Onglet de l'Analyse en composantes principale

Affiche les ACP réalisées sur nos donnés. Il comprend le graphique de l'inertie, le graphique des corrélation, ainsi que la projection des individus et des variables sur le plan séléctionné. Ici, nous pouvons choisir les données sur lesquelles l'ACP est réalisée, ainsi que les axes représentés.

Onglet de l'Analyse en composantes multiples

Ici, on a l'analyse en composantes multiples. Nous pouvons choisir les axes de représentation du graphique et le nombre de modalités qui vont contribuer à la construction de l'ACM..

Onglet des règles d'association

Cet onglet présente les règles d'association observées dans notre jeu de données. Ici, nous avons la possibilité de choisir support, confidence, le lift (force de la règle), la taille de longueur de la règle. Mous pouvons aussi choisir quelles variables sont prises en compte ou non.

Onglet de forêt d'arbres décisionnels

Et dans le dernier onglet, nous pouvons voir les arbres décisionnels et deux différents modèles de forêts aléatoires. Pour les arbres décisionnels, on a le choix entre différents jeux de données. En ce qui concerne les modèles de forêt, modèle Rf concerne tous les variables initiales, et modèle Rf1 concerne les variables créées (audela, outil, objets, classe T1).

Sorties régression logistique mixte

T21 : T21m <- glmm(T21 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.re
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T21 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.3057     3.0754  -1.075   0.2824
## age           1.1383     0.6709   1.696   0.0898 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.4276     0.3807   1.123      0.131
```

T21mm <- glmm(T21 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T21 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.3057     3.0754  -1.075   0.2824
## age           1.1383     0.6709   1.696   0.0898 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.4276     0.3807   1.123      0.131
```

L'âge est significatif

T22 : T22m <- glmm(T22 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.re
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T22 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.8772     2.1247  -2.295   0.0217 *
## PedagogieMontessori   0.1927     0.2904   0.664   0.5069
## age                   1.0333     0.4555   2.268   0.0233 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.06170    0.03924   1.572     0.0579 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T22mm <- glmm(T22 ~ age , random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
```

```
## glmm(fixed = T22 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.7064     2.1177  -2.222   0.0263 *
## age           1.0168     0.4565   2.228   0.0259 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.0960     0.1224   0.784      0.216
```

L'âge est significatif

T23 : T23m <- glmm(T23 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.re
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T23 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -6.9440     2.5369  -2.737   0.0062 **
## PedagogieMontessori   0.0219     0.3247   0.067   0.9462
## age                   1.2203     0.5384   2.267   0.0234 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##            Estimate Std. Error z value Pr(>|z|)/2
## newClasse 9.303e-06  6.041e-06    1.54     0.0618 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T23mm <- glmm(T23 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T23 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
```

```
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -6.931      2.528  -2.741  0.00612 **
## age            1.220      0.538   2.267  0.02338 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##            Estimate Std. Error z value Pr(>|z|)/2
## newClasse 7.641e-05  2.229e-05   3.428   0.000305 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

L'âge est significatif

T31 : T31m <- glmm(T31 ~ Pedagogie + age , random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.re
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T31 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.0403     2.1396  -1.888   0.0590 .
## PedagogieMontessori  -0.2905     0.2882  -1.008   0.3135
## age                   0.9967     0.4617   2.159   0.0309 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03464    0.02968   1.167      0.122
```

T31mm <- glmm(T31 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T31 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
```

```
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.2006     2.1331  -1.969   0.0489 *
## age           1.0047     0.4614   2.177   0.0295 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03192    0.02776    1.15      0.125
```

L'âge est significatif

T32 : T32m <- glmm(T32 ~ Pedagogie+age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T32 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.8056     2.5029  -1.920   0.0549 .
## PedagogieMontessori  -0.2458     0.3725  -0.660   0.5095
## age                   0.7875     0.5332   1.477   0.1397
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1327     0.1093   1.214      0.112
```

T32mm <- glmm(T32 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T32 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  -4.7710     2.4503  -1.947   0.0515 .
## age           0.7566     0.5236   1.445   0.1485
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03926    0.01266     3.1   0.000969 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aucune variable n'est significative.

T41a : T41amm <- glmm(T41a ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=d
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41a ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.5675     2.0932  -1.227    0.220
## age           0.4839     0.4502   1.075    0.282
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.07025    0.07588   0.926      0.177
```

T41amm <- glmm(T41a ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41a ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.5675     2.0932  -1.227    0.220
## age           0.4839     0.4502   1.075    0.282
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.07025    0.07588   0.926      0.177
```

Aucune variable n'est significative.

T41b : T41bm <- glmm(T41b ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=do
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41b ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -3.5797     2.2675  -1.579    0.114
## PedagogieMontessori   0.2549     0.3456   0.737    0.461
## age                   0.5952     0.4840   1.230    0.219
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1913     0.1499   1.276      0.101
```

T41bmm <- glmm(T41b ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41b ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.4196     2.2721  -1.505    0.132
## age           0.5856     0.4871   1.202    0.229
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.2097     0.2553   0.821      0.206
```

Aucune variable n'est significative.

T41c : T41cm <- glmm(T41c ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41c ~ Pedagogie + age, random = list(~0 + newClasse),
```

```
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.9182     2.0559  -0.933    0.351
## PedagogieMontessori  -0.3739     0.2784  -1.343    0.179
## age                   0.4215     0.4414   0.955    0.340
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.02070    0.03398   0.609      0.271
```

T41cmm <- glmm(T41c ~ Pedagogie, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41c ~ Pedagogie, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.04459    0.16968   0.263    0.793
## PedagogieMontessori -0.38695    0.26947  -1.436    0.151
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##            Estimate Std. Error z value Pr(>|z|)/2
## newClasse 6.002e-05  1.823e-05   3.293   0.000495 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aucune variable n'est significative.

T41d : T41dm <- glmm(T41d ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=do family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41d ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
```

```
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.1638     2.1550  -1.932   0.0533 .
## PedagogieMontessori  -0.1122     0.3139  -0.357   0.7209
## age                   0.8861     0.4610   1.922   0.0546 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1332     0.1420   0.939      0.174
```

T41dmm <- glmm(T41d ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T41d ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.299      2.188  -1.965   0.0494 *
## age            0.904      0.469   1.928   0.0539 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1533     0.1789   0.857      0.196
```

Aucune variable n'est significative.

T42a : T42am <- glmm(T42a ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42a ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -7.1116     2.4335  -2.922  0.00347 **
```

```
## PedagogieMontessori   0.2878     0.3212   0.896  0.37015
## age                   1.2844     0.5166   2.486  0.01290 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.05656    0.05744   0.985      0.162
```

T42amm <- glmm(T42a ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42a ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.1062     2.4867  -2.858  0.00427 **
## age           1.3070     0.5286   2.473  0.01342 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1205     0.1724   0.699      0.242
```

L'âge est significatif

T42b : T42bm <- glmm(T42b ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=do family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42b ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -7.42220    3.09394  -2.399   0.0164 *
## PedagogieMontessori  0.01163    0.55303   0.021   0.9832
## age                  1.20527    0.64742   1.862   0.0627 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.8206     0.5341   1.536     0.0622 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T42bmm <- glmm(T42b ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42b ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.4216     3.0780  -2.411   0.0159 *
## age           1.2063     0.6483   1.861   0.0628 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.8151     0.5947   1.371     0.0853 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aucune variable n'est significative.

T42c : T42cm <- glmm(T42c ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42c ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -2.6699     2.4445  -1.092    0.275
## PedagogieMontessori   0.1571     0.3323   0.473    0.636
## age                   0.2975     0.5231   0.569    0.570
##
##
```

```
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.04451    0.05953   0.748      0.227
```

T42cmm <- glmm(T42c ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42c ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.5814     2.4313  -1.062    0.288
## age           0.2922     0.5221   0.560    0.576
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03916    0.03128   1.252      0.105
```

Aucune variable n'est significative.

T42d : T42dm<- glmm(T42d ~ Pedagogie + age , random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=do family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42d ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.37244    2.44434  -0.971    0.332
## PedagogieMontessori  0.04849    0.40156   0.121    0.904
## age                  0.27897    0.52131   0.535    0.593
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.3783     0.2877   1.315     0.0942 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T42dmm <- glmm(T42d ~ age , random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T42d ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.3462     2.4323  -0.965    0.335
## age           0.2782     0.5213   0.534    0.594
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.3801     0.2868   1.326     0.0925 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aucune variable n'est significative.

T51 : T51m <- glmm(T51 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.re
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T51 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -6.1835     2.7664  -2.235   0.0254 *
## PedagogieMontessori  -0.8113     0.4430  -1.831   0.0670 .
## age                   1.0720     0.5864   1.828   0.0675 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.2842     0.1426   1.993     0.0231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T51mm <- glmm(T51 ~ Pedagogie, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
```

```
## glmm(fixed = T51 ~ Pedagogie, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.1884     0.2428  -4.894 9.88e-07 ***
## PedagogieMontessori  -0.8875     0.4269  -2.079   0.0376 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.3518     0.2132    1.65     0.0494 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La Pedagogie est significative

T52 : T52m<-glmm(T52 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T52 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -8.6671     5.3099  -1.632    0.103
## PedagogieMontessori  -0.5095     0.7397  -0.689    0.491
## age                   1.2100     1.1185   1.082    0.279
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.2739     0.1367   2.004     0.0226 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T52mm<-glmm(T52 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T52 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
```

```
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -9.351      5.197  -1.799    0.072 .
## age            1.313      1.096   1.198    0.231
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1555     0.0488   3.187   0.000719 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aucune variable n'est significative.

T61 : T61m<-glmm(T61 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T61 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.003691   4.796375  -0.001    0.999
## PedagogieMontessori -0.733890   0.677868  -1.083    0.279
## age                  0.752913   1.039026   0.725    0.469
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.3544     0.3962   0.895      0.186
```

T61mm<-glmm(T61 ~ Pedagogie, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T61 ~ Pedagogie, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
```

```
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           3.4102     0.4691   7.270  3.6e-13 ***
## PedagogieMontessori  -0.7500     0.6134  -1.223    0.221
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.18073    0.07671   2.356    0.00924 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aucune variable n'est significative.

T62 : T62m<-glmm(T62 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T62 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.69942    2.79674   0.250    0.803
## PedagogieMontessori  0.07761    0.48539   0.160    0.873
## age                   0.19510    0.59867   0.326    0.745
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.5374     0.4585   1.172      0.121
```

T62mm<-glmm(T62 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T62 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)   0.7672     2.7853   0.275    0.783
## age           0.1912     0.5991   0.319    0.750
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.5695     0.4208   1.353      0.088 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aucune variable n'est significative.

T71 : T71m<-glmm(T71 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T71 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.63670    2.34639  -0.271    0.786
## PedagogieMontessori -0.04589    0.38660  -0.119    0.906
## age                 -0.03305    0.50111  -0.066    0.947
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.3297     0.3408   0.967      0.167
```

T71mm<-glmm(T71 ~ Pedagogie, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T71 ~ Pedagogie, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.78584    0.26575  -2.957  0.00311 **
## PedagogieMontessori -0.05269    0.38761  -0.136  0.89187
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.3408     0.2838   1.201      0.115
```

Aucune variable n'est significative.

T72 : T72m<-glmm(T72 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T72 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.4420     2.2819  -1.947   0.0516 .
## PedagogieMontessori  -0.3388     0.3047  -1.112   0.2661
## age                   0.7857     0.4875   1.612   0.1071
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse 0.011139   0.004219    2.64    0.00414 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T72mm<-glmm(T72 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T72 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.7274     2.2917  -2.063   0.0391 *
## age           0.8207     0.4905   1.673   0.0943 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
```

```
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03163    0.02474   1.279      0.101
```

Aucune variable n'est significative.

T81 : Impossible de lancer la T81

T82 : T82m<-glmm(T82 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T82 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -7.4145     2.4656  -3.007 0.002637 **
## PedagogieMontessori  -0.1923     0.3216  -0.598 0.550001
## age                   1.8590     0.5386   3.451 0.000558 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03821    0.04497    0.85      0.198
```

T82mm<-glmm(T82 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T82 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.6877     2.5044  -3.070 0.002143 **
## age           1.9020     0.5487   3.466 0.000528 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.08476    0.13629   0.622      0.267
```

L'âge est significatif

T83 : T83m<-glmm(T83 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T83 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -5.2337     2.5875  -2.023   0.0431 *
## PedagogieMontessori  -0.1066     0.3595  -0.297   0.7668
## age                   1.4033     0.5654   2.482   0.0131 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1468     0.2398   0.612       0.27
```

T83mm<-glmm(T83 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T83 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.7031     2.5235  -2.260  0.02382 *
## age           1.4911     0.5472   2.725  0.00643 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.2620     0.1552   1.687     0.0458 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

L'âge est significatif

T84 : T84m<-glmm(T84 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T84 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -5.3306     2.2462  -2.373  0.01763 *
## PedagogieMontessori   0.3000     0.2983   1.006  0.31449
## age                   1.2818     0.4858   2.639  0.00833 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.02287    0.01575   1.452     0.0732 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T84mm<-glmm(T84 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg,
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T84 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.1165     2.2360  -2.288  0.02212 *
## age           1.2634     0.4854   2.603  0.00925 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03857    0.03824   1.009      0.157
```

L'âge est significatif

T85 : T85m<-glmm(T85 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg
family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T85 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -5.1371     2.3333  -2.202   0.0277 *
## PedagogieMontessori   0.2723     0.3303   0.824   0.4098
## age                   1.2048     0.5026   2.397   0.0165 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1400     0.2487   0.563      0.287
```

T85mm<-glmm(T85 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T85 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.9911     2.2996  -2.170   0.0300 *
## age           1.1992     0.4996   2.401   0.0164 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.1556     0.2365   0.658      0.255
```

L'âge est significatif

T86 : T86m<-glmm(T86 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T86 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
```

```
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -7.6429     2.2227  -3.439 0.000585 ***
## PedagogieMontessori   0.4833     0.2968   1.629 0.103378
## age                   1.6708     0.4782   3.494 0.000476 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03358    0.03797   0.884      0.188
```

T86mm<-glmm(T86 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T86 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.4088     2.1828  -3.394 0.000688 ***
## age           1.6634     0.4723   3.522 0.000428 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.06699    0.02505   2.674    0.00375 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

L'âge est significatif

T87 : T87m<-glmm(T87 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T87 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
```

```
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -5.1607     2.1435  -2.408   0.0161 *
## PedagogieMontessori   0.5017     0.2872   1.747   0.0807 .
## age                   1.0959     0.4596   2.385   0.0171 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03229    0.03426   0.943      0.173
```

T87mm<-glmm(T87 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T87 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.7777     2.0897  -2.286   0.0222 *
## age           1.0556     0.4509   2.341   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.02187    0.01809   1.209      0.113
```

L'âge est significatif

T88 : T88m<-glmm(T88 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T88 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
```

```
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -5.2898     2.2323  -2.370   0.0178 *
## PedagogieMontessori   0.5928     0.3128   1.895   0.0581 .
## age                   1.0940     0.4762   2.297   0.0216 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.09598    0.19195     0.5      0.309
```

T88mm<-glmm(T88 ~ age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg, family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T88 ~ age, random = list(~0 + newClasse), varcomps.names = c("newClasse"),
##     data = don.reg, family.glmm = binomial.glmm, m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.1752     2.1998  -2.353   0.0186 *
## age           1.1270     0.4739   2.378   0.0174 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse   0.2527     0.3231   0.782      0.217
```

L'âge est significatif (age + pedagogie presque)

T89 : T89m<-glmm(T89 ~ Pedagogie + age, random = list(~ 0 + newClasse), varcomps.names=c("newClasse"),data=don.reg family.glmm = binomial.glmm, m=10^4, debug=TRUE)

```
##
## Call:
## glmm(fixed = T89 ~ Pedagogie + age, random = list(~0 + newClasse),
##     varcomps.names = c("newClasse"), data = don.reg, family.glmm = binomial.glmm,
##     m = 10^4, debug = TRUE)
##
##
## Link is: "logit (log odds)"
##
## Fixed Effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -6.3577     2.2006  -2.889  0.00386 **
## PedagogieMontessori   0.6404     0.2888   2.218  0.02658 *
## age                   1.2867     0.4698   2.739  0.00617 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Variance Components for Random Effects (P-values are one-tailed):
##           Estimate Std. Error z value Pr(>|z|)/2
## newClasse  0.03508    0.04559    0.77      0.221
```

L'âge + la pédagogie sont significatifs