Université de Strasbourg

ICUBE

IRMA
Institut de Recherche Mathématique Avancée

Thèse présentée pour obtenir le grade de docteur
de l'université de Strasbourg

❧

École doctorale de Mathématiques, Sciences de l'Information et de l'Ingénieur

Discipline : Informatique

Clusterisation incrémentale, multicritères de données hétérogènes pour la personnalisation d'expérience utilisateur

---

Par : **Emmanuelle Claeys**

Sous la direction de Pierre Gançarski, Professeur des universités, université de Strasbourg

Co-encadrée par Myriam Maumy-Bertrand, Maître de conférences, université de Strasbourg

Co-encadrée par Hubert Wassner, Chief data scientist à AB Tasty

Membres du jury:

- **Rapporteur externe :** Antoine Cornuèjols, Professeur des universités, AgroParisTech
- **Rapporteur externe :** Aurélien Garivier, Professeur des universités, E.N.S. de Lyon
- **Examinateur :** Jérémie Mary, Maître de conférences, Senior Research Staff, université de Lille, CRITEO
- **Examinateur :** Myriam Maumy-Bertrand, Maître de conférences, université de Strasbourg
- **Examinateur :** Cédric Wemmert, Professeur des universités, université de Strasbourg

Invité : Gilbert Saporta, Professeur émérite, C.N.A.M.

Date de soutenance : 12 Novembre 2019

Remerciements

Bien que l'expérience de la thèse soit un exercice solitaire, je n'ai jamais manqué de stimulations, de soutiens et d'encouragements pendant ces trois ans. Le temps des remerciements est donc venu pour remercier tous ceux qui m'ont permis de n'être jamais seule. Mes remerciements me rappellent que toute épreuve d'une vie ne peut être menée à bien que parce que l'on est épaulé, soutenu et parfois même rattrapé par d'autres. La liste est longue des personnes auxquelles je pense alors que j'écris ces lignes, des personnes qui ont, de près ou loin, sans toujours le savoir, contribué à l'accomplissement de cette thèse, mais surtout à ces trois ans incroyables.

Je remercie bien évidemment Pierre Gançarski qui a activement et patiemment dirigé ce travail. Je le remercie pour sa grande disponibilité et son exigence, pour les centaines de pages qu'il a lues, corrigées et discutées avec moi des heures durant. Je le remercie pour ses conseils et remarques avisés. Je le remercie enfin d'avoir su trouver le temps et les mots pour m'encourager.

Je remercie ma co-encadrante Myriam Maumy Bertrand, qui a fait preuve d'un encouragement sans faille, et qui a su comprendre mon état d'esprit pour me donner un sentiment d'accomplissement à la fois personnel mais aussi intellectuel. Je tiens également à la remercier de m'avoir fait rencontrer Pierre, le monde merveilleux des mathématiques, et pour m'avoir donné envie de faire une thèse lorsque j'étais son étudiante.

Je suis aussi particulièrement reconnaissant à Hubert Wassner qui a contribué à ce que cette thèse existe en acceptant de m'encadrer à AB Tasty. Je le remercie pour le soutien qu'il m'a toujours apporté, pour sa disponibilité, pour sa présence bienveillante et amicale. Auprès de lui et au gré des recherches que nous avons menées ensemble, j'ai énormément appris du métier de data scientist.
Merci à Rémi Aubert et Alix de Savigne de m'avoir fait confiance et d'avoir financé ces trois années de recherche.

Antoine Cornuéjol, Aurélien Garivier, Jeremie Mary et Cédric Wermmert ont accepté de lire et discuter ce travail. C'est un honneur qu'ils me font de faire partie de mon jury de soutenance et je les en remercie. Je tiens à remercier chaleureusement Gilbert Saporta pour toutes les discussions scientifiques et philosophiques que nous avons pu avoir en conférence et d'avoir accepté d'être membre invité du jury. Merci à Frédéric Bertrand

pour ses conseils avisés et sa confiance. Rencontrer de telles personnes rend les moments passés en conférences magiques et inoubliables.
Un grand merci à mes collègues d'AB Tasty, à commencer par mon incroyable Gaëtan sur qui j'ai toujours pu compter, Guillaume, Wissem, Fabien, Léo, Pawel, Sofiane, Cyril et toute l'équipe d'AB Tasty
Je remercie bien évidemment mes parents, qui m'ont encouragé dans mes choix. Vous êtes des parents formidables et l'on ne peut rêver mieux. Merci ! À ma sœur Géraldine, merci d'avoir fait figure d'exemple pour son courage et son acharnement au travail.
Cette thèse m'a permis de découvrir Strasbourg et les Strasbourgeois, merci à mes camarades doctorants de l'IRMA : Marie, Claire, mon président Philippe R, Nicolas, Guillaume K, Philippe M., François, Valdo, P.A., Fred, Thibaut, Gatien, Arthur, Alexander, Laura, Alix, Titin, Léo et tous les autres, ainsi que mes camarades de l'ICUBE Baptiste, Abdoul et Étienne. Courage les amis, ne lâchez pas et n'oubliez pas de profiter de la vie ! Merci à mes voisins formidables Albin et Jeff, ainsi que Clément, Lucie, Yann, Agathe, Élise, Aurélia, Alex, Charlotte. Merci pour toutes ces soirées de folies et d'avoir écouté mes histoires avec tendresse ! Merci à mes camarades de l'E.S.I.E.A. Sarah, Pich, Nico, Michel, Julie, Florian, Cédric, Franck et tous les autres qui, après leur stupeur face à mon choix professionnel, m'ont soutenu par leurs encouragements.

Merci à Thomas B, Emma et Baptiste B. pour leur amour inébranlable. Trois ans peuvent paraître courts et longs à la fois. Merci à Thomas D pour ton soutien surtout dans les derniers moments difficiles et pardon pour le Arch Enemy régulier. Au passage je remercie le groupe Rammstein qui a contribué à ma productivité sans le savoir. Merci à Guillaume D d'avoir détourné mon regard vers les étoiles. Merci à Jérémie d'avoir toujours su trouver les mots justes quand j'étais perdue, de m'avoir appris tant sur la science et sur le monde. Enfin merci à Léa pour son affection quotidienne (ta rencontre vaut tous les plus grands théorèmes).

Merci encore à toutes ces personnes précédemment citées ainsi qu'à tous les scientifiques dont les travaux furent inspirants pour mon travail de m'avoir permis de clôturer cette thèse avec une richesse inégalable.

Chapitre 0

Introduction

Problématique

Dans de nombreux domaines (santé, vente en ligne, production industrielle,...) concevoir *ex nihilo* une solution optimale répondant à un problème défini (trouver un protocole augmentant le taux de guérison, concevoir une page Web favorisant l'achat d'un ou plusieurs produits, définir un processus de fabrication débouchant sur une meilleure qualité des produits ...) est souvent très difficile voire impossible. Face à cette difficulté, les concepteurs (médecins, web designers, ingénieurs de production, ...) travaillent souvent de façon incrémentale par des améliorations successives d'une solution existante. Néanmoins, définir les modifications les plus pertinentes reste un problème difficile. Pour tenter d'y répondre, une solution adoptée de plus en plus fréquemment consiste à comparer concrètement différentes alternatives (appelées aussi variations) afin de déterminer celle(s) répondant le mieux au problème. L'idée est de mettre en œuvre réellement ces alternatives et de comparer les résultats obtenus, c'est-à-dire les gains respectifs obtenus par chacune des variations.
Le principe, mis en œuvre dans les méthodes d'A/B test [28, 70] est le suivant : à partir d'une solution existante dite originale et généralement nommée *variation A*, une ou plusieurs alternatives ou variations (communément appelées *variations B/C/D* ...) sont construites (modification dans la composition d'un médicament, ajout d'une image sur une page web, changement d'une étape de fabrication ...). Ces différentes variations sont alors mis en œuvre in vivo : des cohortes différentes se voient prescrire chacune un médicament différent, les visiteurs d'un site web voient chacun une seule variation de la page, différentes lignes de production sont utilisées en parallèle ...

Cependant, afin de garantir l'indépendance entre les réalisations des différentes variations, chaque décision d'*affectation* d'un *item* (patient, visiteur, objet à produire) à une variation, c'est-à-dire, par exemple le choix du médicament à prescrire à un patient, de la page web à afficher au visiteur ou de la ligne de production à enclencher, est irrévocable : le médicament prescrit à un patient sera toujours le même, un visiteur verra toujours la même page web à chacune de ses visites, un objet sera entièrement produit sur une seule chaîne. De cette contrainte forte découle le fait qu'il est impossible de savoir quel aurait été le gain obtenu en cas d'affectation de l'item à une autre variation. De plus, les sujets

soumis au test n'ont généralement pas connaissance d'être testés et ignorent qu'il existe des variations différentes.

À la fin de cette phase dite d'*exploration*, les performances de chaque variation sont alors comparées en fonction de l'objectif visé (état de santé des patients, ventes d'un produit, qualité des productions,...). Cette comparaison repose sur une ou plusieurs méthodes statistiques permettant de quantifier et/ou qualifier une différence de gain global comme par exemple, le cumul ou la moyenne des récompenses produites par chacune des variations. La meilleure variation peut alors être mise en *exploitation*.

R. Fisher, biologiste et statisticien britannique, est le premier à suggérer cette idée en 1925 [27]. Il propose d'en définir plus formellement le contexte en 1935 [28]. Ses travaux seront utilisés par la suite en médecine, pour comparer un traitements médical et un placebo. Parallèlement, W. R. Thompson proposa dans [70] une méthode de test qui viendra compléter les techniques de comparaisons proposées par R. Fisher.

Stratégies d'allocation

La stratégie d'allocation des items aux variations est au cœur du processus de test car c'est elle qui doit permettre de déterminer la meilleure variation à l'issue de la phase d'exploration.

Une première approche d'A/B test qualifiée de *statique* consiste à fixer en amont du test la durée de la phase d'exploration et les proportions relatives d'items à affecter à chaque variation. À l'issue de cette phase, les gains respectifs de chacune des variations sont alors comparés et il est ainsi possible de déterminer la meilleure variation.

Il s'avère que la durée de cette phase d'exploration est dans les faits, très difficile à fixer car elle dépend fortement de l'objectif du test, des variations testées et des items eux-mêmes. Or cette exploration peut présenter un coût pouvant être élevé. En effet, le fait de proposer trop longtemps une variation sous-optimale alors que la variation optimale (*i.e.*, la meilleure variation parmi les variations testées) aurait pu être choisie plus tôt débouche sur un manque à gagner (ou *regret*) pouvant être fort : à chaque fois que la méthode affecte une variation de façon sous-optimale, celle-ci produit un gain inférieur à celui qu'aurait produit la version optimale. Or cette dernière est malheureusement inconnue a priori, l'objectif de test étant par définition, de la déterminer.

Pour répondre à ce problème, le concept d'*allocation dynamique* a vu le jour. Il s'agit de permettre de modifier automatiquement les différents ratios lors de la phase d'exploration et ainsi de basculer plus vite vers la solution optimale. L'idée est de maintenir et mettre à jour les estimations de gain de chacune des variations et d'allouer les items en fonction de celles-ci. Il s'agit à la fois de privilégier la variation la plus prometteuse tout en continuant à affiner les estimations des espérances des autres variations en continuant à allouer des

items à des variations potentiellement sous-optimales.

Allocation dynamique et modèle de bandits

Pour mettre en place une allocation dynamique efficace, les concepteurs de méthodes d'A/B test se sont orientés vers des stratégies d'allocation basées sur des *modèles de bandits*. Cette notion de bandit a été introduite par Lai et Robbins, [48] sous le nom de bandit multi-bras (plus couramment désigné sous le terme de bandit). Il se définit comme un problème dans lequel un ensemble limité de ressources doit être réparti entre des choix (alternatives) d'une manière qui maximise le gain. Les propriétés de chaque choix ne sont que partiellement connues au moment de la répartition, et ne peuvent être mieux comprises qu'au fil du temps ou en allouant des ressources aux différents choix [36]. Intuitivement, par analogie avec les machines à sous d'un casino, il s'agit de choisir, pour un joueur, sur une machine à plusieurs bras, celui qui présente, pour lui, la meilleure espérance de gain. Pour cela, chaque fois qu'il a joué un bras et éventuellement récolté un gain, le joueur met à jour ses estimations de gain sur chaque bras de la machine. Son choix suivant tiendra compte de ces nouvelles estimations. Le but du joueur est de trouver le plus rapidement possible le meilleur bras, appelé *bras optimal* pour le jouer au maximum ou de minimiser le regret. De façon évidente, ce cadre théorique répond parfaitement au problème de l'allocation dynamique : les bras seront les variations, les parties seront les items et les espérances de gains seront calculées à partir des récompenses réellement obtenues. Les méthodes basées sur les bandits telles que Ucb ou Thompson Sampling [48, 70] ont rapidement prouvé leur efficacité dans bien des domaines.

Néanmoins, elles ne permettaient pas jusqu'à récemment de tenir compte des items eux-mêmes. En effet, ces approches sont *non informées* et ne prennent pas en compte les informations potentiellement disponibles sur les items eux-mêmes. Or, le contexte du patient, du visiteur ou de l'objet à produire est apparu comme indispensable à prendre en compte pour améliorer et réduire fortement la phase d'exploration. Ainsi, des méthodes *informées* ont vu le jour. Basées sur des *bandits contextuels*, elles permettent d'intégrer les *caractéristiques* des items (âge, sexe, navigateur utilisé, coût des matériaux, ...) directement dans le mécanisme d'affectation dynamique des variations. Les méthodes telles que KernelUCB [72] et LinUCB [20] ont montré l'intérêt mais aussi les limites de ces approches, dont en particulier

- un temps de latence, c'est-à-dire le délai nécessaire à l'algorithme pour allouer un item à une variation, trop important,
- et un déficit d'explicabilité des choix faits par l'algorithme

qui limitent leur utilisation concrète. Des méthodes plus récentes proposent donc d'utiliser des bandits non contextuels indépendants sur des groupes pré-définis.

Le **premier objectif** de cette thèse a été de proposer une méthode d'allocation dy-

namique informée basée sur une explicitation préalable des groupes autorisant une forte explicabilité des allocations effectuées.

Stationnarité et A/B test

Par ailleurs, il apparaît de plus en plus indispensable d'étudier l'influence du temps sur les tests et ce à différents niveaux. En effet, les méthodes actuelles sont souvent amenées à faire des hypothèses de stationnarité sur les distributions réelles des gains associées aux différentes variations (par exemple, la variation optimale est toujours la même), sur les caractéristiques globales de l'ensemble de items (par exemple, la proportion d'hommes et femmes est invariant) ou encore sur les caractéristiques des items eux-mêmes (par exemple, le poids d'un patient reste le même durant tout le protocole de soin). Malheureusement, ces hypothèses s'avèrent de plus en plus contraignantes et limitent leur applicabilité sur des A/B tests. Il est donc indispensable à la fois d'en étudier l'impact sur les tests et proposer de nouvelles méthodes autorisant un relâchement de telles contraintes de stationnarité.

Parallèlement, les informations temporelles liées aux comportements des items avant et pendant le test ne sont actuellement pas prises en compte lors de l'allocation d'un item à une variation. Or, l'utilisation d'informations temporelles comme par exemple, la liste des sites visités par un nouveau visiteur avant d'arriver sur le site hébergeant le test ou celle des pages parcourues et les options cliquées sur celui-ci avant d'arriver à la page testée devrait, à notre avis, améliorer fortement les mécanismes d'allocation.

Si l'étude de nouvelles méthodes d'A/B test permettant de relâcher les contraintes de stationnarité n'a pas pu malheureusement être pas abordée concrètement dans cette thèse, un **deuxième objectif** de celle-ci a été d'étudier et proposer une méthode d'allocation permettant de prendre en compte des informations temporelles sur les items.

Contributions

L'objectif global de cette thèse est de proposer une méthode générique d'A/B test permettant une allocation dynamique en temps réel capable de prendre en compte les caractéristiques des items qu'ils soient temporels ou non.

La première contribution porte sur la prise en compte des caractéristiques d'items lors de la phase d'exploitation. L'idée originale est de postuler qu'il existe dans la population des items des sous-groupes présentant chacun une sensibilité différente aux variations et donc que l'allocation dynamique doit se faire différemment dans chacun de ces sous-groupes. La méthodes Ctree-Ucb que nous proposons est constituée de deux phases. La première, dite *d'explicitation des groupes*, effectuée en pré-traitement (hors ligne) consiste à créer un *modèle de classement* des items en groupes distincts. Ce modèle est appris à partir des caractéristiques d'items ayant été affectés à la variation originale avant le test

et de leurs gains obtenus sur cette variation originale. La seconde phase, correspondant à la phase d'exploration elle-même, consiste à associer à chacun des groupes un bandit non contextuel chargé de l'allocation dynamique des nouveaux items qui auront été classés par le modèle dans ce groupe. Ainsi, un nouvel item sera classé dans un groupe en fonction de modèle prédéfini et de ses propres caractéristiques puis affecté à une variation par le bandit associé à ce groupe.

Les expériences que nous avons menées ont montré que cette méthode, en plus de déboucher sur une phase d'exploration raccourcie par rapport aux autres méthodes (tout en facilitant l'explication a posteriori des choix), permettait une allocation dynamique en temps réel.

La deuxième contribution est une extension de la méthode précédente aux caractéristiques temporelles. Il s'agit de mieux préciser les modèles de classement en intégrant la possibilité de prise en compte de données temporelles. Les méthode DBA-Ctree-Ucb et DBA-LinUcb permettent ainsi d'intégrer à la fois des caractéristiques temporelles historisées (c'est-à-dire liées aux comportement des items avant le test) et dynamiques (c'est-à-dire par exemple, saisie à la volée lors de la navigation d'un visiteur sur le site de test). Nos expériences ont montré que détecter un contexte évolutif et adapter le test en fonction de celui-ci offre un intérêt pour la problématique générale de l'A/B test, et plus particulièrement pour la mise en forme de pages WEB d'e-commerce.

Les contributions de cette thèse sont de différentes natures. Elles se veulent à la fois génériques en proposant des solutions applicables à tout domaine, théoriques en validant mathématiquement les propositions faites et applicatives en les mettant concrètement en œuvre.

En effet, cette thèse étant réalisée en partenariat avec la société AB Tasty, notre contribution repose à la fois sur des garanties théoriques par des propriétés statistiques, mais également par des résultats empiriques issus de jeux de données réelles publics ou venant (majoritairement) de cette société. Ainsi, afin de valider cette approche, des expérimentations ont été menées à la fois sur des données dites de benchmarks et sur des données issues de tests réels menés par AB Tasty.

La suite du mémoire présente d'abord la méthode Ctree-Ucb proposée, ses garanties théoriques et ses résultats expérimentaux.

La deuxième partie de ce mémoire propose une variante de la méthode Ctree-Ucb : DBA-Ctree-Ucb ainsi qu'une méthode alternative sans classement, DBA-LinUcb, qui permettent d'intégrer des aspects temporels dans les caractéristiques des items à l'aide d'une étape supplémentaire de clustering dans la phase d'explicitation des groupes.

Plan du mémoire

La première partie de ce mémoire introduit le cadre général de cette thèse :

— La section 1 présente le contexte de cette thèse et en particulier les caractéristiques et problématiques propres aux A/B tests dans le domaine du WEB d'e-commerce.
— La section 2 décrit plus précisément les principes et mécanismes de bandits mis en œuvre dans les méthodes d'A/B test.

La deuxième partie décrit plus précisément nos contributions

— La section 3 décrit la méthode Ctree-Ucb et la compare aux principales méthodes de l'état de l'art.
— La section 4 décrit la méthode DBA-Ctree-Ucb, extension de la méthode précédente et la méthode DBA-LinUcb, intégrant la possibilité de caractéristiques temporelles.

Enfin, une dernière section conclut ce travail et donne des pistes d'améliorations et des perspectives fortes correspondant aux nombreuses questions restant ouvertes.

Table des matières

Première partie

Cadre théorique et contexte applicatif

Chapitre 1

Contexte applicatif : A/B dans le WEB

1.1 Introduction

Les A/B tests sont aujourd'hui majoritairement employés dans le marketing digital où les variations sont plusieurs versions d'une page web, d'une application mobile, d'un e-mail ou d'une bannière publicitaire, comparées afin de choisir la plus efficace et de l'utiliser en production.

Cette thèse étant réalisée avec la société d'A/B test ABTasty, le contexte applicatif est le suivant. Les items (appelés *visiteurs*) soumis au test sont décrits par différentes caractéristiques, soit personnelles, soit liées à leur navigation sur le site. Les visiteurs testés peuvent être plus ou moins nombreux (d'une centaine à quelques millions).

Les variations A/B/C/D/... sont comparées selon un gain associé à une valeur numérique ou booléenne (appelée *récompense*) produite par chaque item testé (valeur d'achat, clic/absence de clic,...).

Aucune hypothèse n'est faite sur la différence entre les moyennes de gain des variations, c'est-à-dire que ces moyennes peuvent être très proches, importantes ou égales (une différence entre les moyennes variant de 0% à 20% par exemple). En revanche, nous imposons que :

— Les objectifs de tests variant d'un analyste à un autre, le système d'allocation dynamique proposé intègre différents types de gains (variable utilisée pour comparer les variations). Par exemple il peut être amené à comparer plusieurs variations selon un gain discret ou continu.

— Dans le contexte du test A/B, les items testés ont tendance à répondre à l'objectif plus ou moins souvent selon leurs propres caractéristiques et/ou leurs besoins. Le système doit pouvoir prendre en compte les caractéristiques d'un item pour proposer un choix d'affectation (A, B,...) optimal.

— Le système s'exécute en temps réel : lorsqu'il prend en entrée un item à tester, le système doit pouvoir donner un choix d'affectation dans un temps inférieur à 100 ou 200 millisecondes.

— Les items à tester peuvent être caractérisés (ou non) par des informations de type

statiques (par exemple le navigateur) et/ou temporelles (par exemple l'historique de navigation). Lorsque une caractéristique est temporelle, elle sera donnée sous forme de série temporelle de taille variable.

1.2 Société Tasty

AB Tasty est une entreprise française, basée à Paris, créée en 2008 par Alix de Sagazan et Rémi Aubert. Elle édite des solutions d'optimisation (dont les A/B tests) de conversions en mode SaaS, destinée aux sites e-commerce, aux sites d'actualités et aux éditeurs de services interactifs. La société est initialement créée sous le nom Liwio et opère une activité d'agence de conseil en web analytics, en se focalisant sur des prestations d'implémentation de solutions de mesure d'audience (ex : Google Anaytics) et de construction de tableaux de bord statistiques.

En parallèle de cette activité, la société développe depuis de 2012 des solutions d'A/B testing et de personnalisation du site web en mode SaaS, à destination des équipes marketing. La société pivote en 2012 pour se consacrer au métier d'éditeur de logiciels. Elle se développe alors rapidement sur le marché de l'A/B testing avec des clients grands comptes français. La société obtient le Pass French Tech délivré par Cap Digital, le pôle de compétitivité et de transformation numérique en 2015, 2016 et 2017. La société bénéficie également du label BPI Excellence, le réseau des entrepreneurs de croissance accompagnés par Bpifrance et est classée 23ème au classement Deloitte Technology Fast 50. A partir de 2015, la société ouvre ses premiers bureaux hors de France. La société remporte le grand prix "E-commerce Awards" pour sa solution AB Tasty for App, lors du salon Paris Retail Week 2015.

1.3 L'A/B Testing et le WEB

L'optimisation des conversions[1] via l'A/B testing est un axe d'amélioration de la rentabilité pour les entreprises avec pour objectif de générer plus de revenus à trafic constant. Soucieux de l'augmentation du marché en ligne et de l'importance de la visibilité d'une entreprise sur le web, les analystes possédant des site web cherchent à tirer le maximum de leurs trafics. Pourtant, le taux de conversion, particulièrement lorsqu'il est défini comme le rapport entre les visiteurs achetant sur le site par rapport au nombre total de visiteurs ayant été sur le site, est généralement faible (voir, très faible). La raison principale est que hors le produit lui-même, la conversion est un mécanisme complexe qui fait intervenir de nombreux éléments tels que le design du site, l'expérience utilisateur, la qualité de l'offre, la réputation du site web, ou encore les actions de la concurrence.

1. réalisation d'une action par des visiteurs et attendue par l'analyste

Par ailleurs, la visite sur un site internet étant moins engageante qu'une visite physique en magasin, les taux de conversion sur internet sont généralement faibles par rapport au magasin (de l'ordre de 2% sur internet pour 20 % en magasin), les visiteurs se rendant sur le site n'ayant pas forcément une intention d'achat ou réalisent plusieurs visites sur différents sites, alors qu'un déplacement dans un magasin physique est plus contraignant en terme de temps.

Souhaitant améliorer au mieux l'attrait de leur site auprès de leurs visiteurs, les e-commercants disposent de nombreux outils analytiques. Parmi les outils disponibles, les e-commercants réalisent des A/B tests pour améliorer leurs pages web ou applications mobiles. En effet, le développement du digital a offert de nouvelles perspectives en multipliant les possibilités de tests et de mesures des performances. Appliqué à un site web (voir Fig.1.1), l'A/B testing permet de tester plusieurs versions d'une page et de mesurer précisément les performances de chaque version sur des indicateurs tels que l'action ou la valeur d'achat. Les évolutions technologiques ont également permis le développement de solutions d'A/B testing dédiées, qui facilitent la mise en place de tels tests et l'analyse des résultats. C'est par exemple via une interface très simple d'utilisation que la société A/B tasty propose à ses utilisateurs de procéder à des A/B tests sur leurs pages web.

FIGURE 1.1 – Exemple d'A/B test consistant à comparer les performances de deux variation d'une même page web

Tous les sites internet ayant un minimum de trafic peuvent bénéficier de l'A/B testing car ils ont tous généralement au moins un objectif mesurable, qui est leur raison d'être (*a mimina* la consultation). Parmi les utilisateurs majoritaires d'A/B testing, on trouve les médias et les sites d'e-commerce. Dans le cas des médias les tests portent généralement sur le succès d'une catégorie de contenus (par exemple en s'assurant qu'ils obtiennent un maximum de clic). Dans le cas de l'e-commerce, l'analyste cible généralement l'efficacité commerciale d'un site en observant sa rentabilité. L'A/B testing permet alors de trouver quelle version est la plus performante grâce aux ventes effectuées. L'analyste évaluera notamment la page d'accueil, les éléments de fiche produit, mais aussi les éléments de commande (boutons, texte, etc).

Constante dans le temps et dans toutes les situations, la *personnalité* en tant que segment marketing est particulièrement importante pour les études sur d'e-commerce. Des recherches [61] ont montré que les consommateurs sont beaucoup plus susceptibles d'acheter un produit si les conditions qui l'entourent sont en phase avec leurs caractéristiques (localisation, culture, habitudes et besoins). En effet, pendant la navigation d'un visiteur, faire correspondre un message, un contenu, des images à son profil peut avoir des effets importants sur son envie d'acheter ou de cliquer.

Ces recherches en markting [57] ont par exemple permis de distinguer différents profils qui seront définit ici comme des *patterns* (motifs) de visiteurs. En sciences humaines, les patterns sont des modèles simplifiés d'une structure de comportement individuel ou collectif, établi à partir des observations répétées. Dans notre contexte, ces patterns permettent de décrire un ensemble de visiteurs en fonction de leur avancement dans le processus d'achat :

- Les observateurs : Ils ont un éventuel besoin sans en avoir d'idée réellement précise et cherchent à l'identifier en effectuant des recherches sur le site. Ils sont en quête d'informations.
- Les évaluateurs : Ils sont plus informés que les observateurs et cherchent à comparer différents produits. Ils cherchent également à collecter de l'information.
- Les acheteurs : Ils cherchent à passer commande sur le site. Leurs visites ont pour but principal d'acheter un article préalablement choisi.
- Les clients : Ils ont déjà acheté sur le site et recommenceront si le site répond à leurs besoins.

Par ailleurs, les auteurs du livre "Allways be testing" [57] ont proposé de nouveaux patterns de visiteurs selon leur propension à l'achat :

- Les *prospects parfaits* : ils savent exactement ce qu'ils recherchent, se dirigent naturellement vers les fiches produit et cherchent les conditions d'achats.
- Les *acheteurs potentiels* : ils savent plus ou moins ce qu'ils veulent, leur besoin est clairement défini mais ils n'ont pas encore établi avec certitude ce qu'ils vont choisir.
- Les *indécis* : ils font en quelque sorte du lèche vitrine. Ils n'ont pas de besoin clairement définis mais peut-être achèteront-ils quelque chose.
- Ceux qui sont *arrivés par hasard* sur le site et qui n'ont pas l'intention d'acheter.

Dans la suite du manuscrit, nous utiliserons ces dernières définitions pour définir les patterns de visiteurs. En effet, dans ce cadre, les visiteurs sont plus ou moins disposés à répondre à l'objectif de l'analyste de l'A/B Test (clic, achats, etc). Par exemple, un visiteur se rendant par erreur sur le site fera une ou deux visites et aura un historique très limité. Cependant, le visiteur « prospect parfait » ne vient que pour acheter et son historique sera lui aussi également limité. Une attention doit être portée sur le choix des caractéristiques. Décrire par exemple les visiteurs par une série de valeurs binaires

(présence/non présence sur le site depuis sa première visite) ne sera pas suffisant. Ce qui peut permettre, par exemple, de distinguer ces deux profils sera le temps passé sur le site lors de leurs précédentes visites mais aussi de celle en cours. Le prospect parfait aura passé plus de temps sur le site que le visiteur arrivé par erreur puisqu'il souhaite passer commande (l'autre quittant immédiatement le site).

1.4 Conclusion

Grâce aux nouvelles technologies, la mise en place et la récolte de données issues d'AB test deviennent de plus en plus faciles. Disposant d'outils permettant de collecter des données, l'analyste historise des caractéristiques dynamiques définissant les items testés (constituant son parcours d'achat). Avec de telles données, il parait naturel que l'analyste cherche à identifier pour quel pattern (qu'il connaît de par son expertise métier) ses variations testées sont optimales. Cependant, cette tâche est compliquée. Premièrement, comment construire les *historiques* ? Quelles données utiliser ? La série doit-elle être composée de valeurs numériques, ou peut elle prendre une forme plus simple comme une série de valeurs binaires décrivant la présence/non présence d'une caractéristique au cours du temps ? Pour répondre à une telle question, il est nécessaire de savoir comment sont décrits les patterns et comment les distinguer. De plus, l'analyse de séries temporelles dans les A/B tests est une pratique récente, les recherches sont encore peu nombreuses. Pourtant, les patterns réagissant différemment, il peut être intéressant pour l'analyste d'observer le résultat du test pour chaque profil temporel de visiteur.

Cette thèse propose de répondre à ces problématiques par différentes contributions. La première partie traite de la gestion du contexte (lorsqu'il est composé de caractéristiques statiques), du temps de réponse et propose une méthode d'allocation dynamique impliquant un pré-processus *offline* et une allocation dynamique en temps réel personnalisée. La deuxième partie de cette thèse propose une méthode d'intégration des informations temporelles dans le contexte des items afin d'améliorer l'allocation dynamique.

Chapitre 2

Allocation dynamique, contextes et bandits

2.1 Cadre théorique

2.1.1 Allocation dynamique et bandits

Les techniques traditionnelles d'A/B test sont généralement basées sur une allocation statique. Cela consiste, après avoir défini les variations, à assigner de façon alternative et irrévocable, les items à la variation A ou B et ce pendant une période donnée.

Le problème d'une telle approche est la *non adaptabilité* du test [49]. Par exemple, si une variation apparaît rapidement comme très performante par rapport à une autre, le test pourrait être conclu plus tôt. Lorsque l'analyste laisse tourner son test alors que la variation optimale est clairement identifiée par les observations empiriques, cela engendre un *regret*, c'est-à-dire une différence entre les gains que l'analyste aurait obtenu en utilisant plus tôt la meilleure variation et les gains obtenus à la fin du test. Pour répondre à ce problème, une solution est d'utiliser un modèle de bandit, c'est-à-dire un algorithme d'allocation dynamique dont l'objectif est de limiter le regret à la fin du test.

En effet, comme brièvement décrit en introduction, le modèle de bandit est un bon moyen de réduire le regret engendré par une allocation statique : à chaque instant t un item c_t est soumis au test. L'algorithme de bandit choisit un bras a dans l'ensemble (noté $\mathcal{A}$) des bras (chacun étant associé à une variation). Soit A_t le bras a choisi à l'instant t. Une récompense $X_{c_t,a=A_t}$ est alors obtenue (par exemple survie/décès, valeur d'achat, ...). Cette récompense est utilisée pour mettre à jour les gains de chaque bras. La caractéristique et l'avantage principal d'un algorithme de bandit est que le ratio d'allocation des items aux différentes variations est automatiquement ajusté au cours du temps en fonction des estimations : l'algorithme décide automatiquement quand une version doit être plus souvent sélectionnée qu'une autre. C'est un gain réel par rapport à l'allocation statique car cette allocation dynamique permet d'arrêter l'exploration lorsqu'elle n'est plus nécessaire. En effet, l'analyste peut arrêter son test, lorsque, par exemple, 90% des items sont affectés à une même variation.
La première partie de ce chapitre introduit le formalisme mathématique relatif aux modèles de bandit.

2.1.2 Le modèle théorique

La notion de bandit a été introduite par Lai et Robbins, [48] sous le nom de bandit multi bras (le terme bandit sera utilisé ici pour simplifier la lecture). Il se définit comme un problème dans lequel un ensemble limité de ressources doit être réparti entre des choix (alternatifs) d'une manière qui maximise le gain attendu. Les propriétés de chaque choix ne sont que partiellement connues au moment de la répartition, et ne peuvent être mieux comprises qu'au fil du temps ou en allouant des ressources au choix en question [36]. Intuitivement, par analogie avec les machines à sous de casino, il s'agit pour un joueur de choisir sur une machine à plusieurs bras celui qui peut potentiellement lui apporter le plus de gain en moyenne. Pour cela, après avoir joué et récolté le gain, le joueur met à jour ses estimations de gain sur chaque bras de la machine. Le but du joueur est de minimiser le regret en trouvant le meilleur bras, appelé *bras optimal* (noté a^*), avec le moins d'essais possible.

Soit $\mathcal{A}$ un ensemble de K bras [1] et a^* le bras optimal (inconnu *a priori*). Soit $X_{A_t,t}$ la récompense obtenue par le bras A_t sélectionné à l'itération t (correspondant au t-ième item). Le but d'un bandit est de trouver le bras a^* le plus rapidement possible afin de limiter le regret cumulé. Ce regret est égal à la somme des différences entre les récompenses qu'aurait obtenu le bras optimal a^* et celles obtenues par les bras choisis.

Le challenge pour l'analyste est de trouver la meilleure stratégie d'allocation dynamique π. Cependant, celle-ci dépend du contexte d'application, c'est-à-dire des récompenses potentielles de chaque item sur chaque bras. On modélise donc la récompense par une variable aléatoire X dont la distribution (ou loi) est inconnue. Pour cela, on s'intéresse à ce qui se passe à chaque itération. Si on note c_t l'item soumis à l'instant t, $A_t \in \mathcal{A}$ le bras choisi à l'instant t, alors on souhaite estimer la quantité $\nu_{A_t} = \mathbb{P}[X|A_t] : A_t \in \mathcal{A}$, où ν désigne la distribution de probabilité des récompenses X associées au bras A_t. L'objectif est d'estimer au mieux et le plus rapidement possible l'espérance du gain de chaque bras afin d'en déterminer l'optimal.

C'est dans ce but que le bandit met à jour les distributions en fonction des items soumis aux bras et des récompenses obtenues. Cependant, cette mise à jour est délicate car elle ne peut se faire que sur le bras sélectionné (dû à l'irrévocabilité).

L'algorithme 1 résume un algorithme de bandit. Il considère une durée de test infinie (c'est-à-dire avec un nombre infini d'items). En pratique, un critère d'arrêt est choisi en fonction du cas d'utilisation.

Une méthode rapide pour trouver le meilleur bras est de supposer que ce bras est le meilleur pour la majorité des items et donc d'appliquer une stratégie *non contextuelle* qui

1. avec $|\mathcal{A}| = K \in \mathbb{N}^+$ où $|\cdot|$ représente le cardinal et $\mathbb{N}^+$ l'ensemble des entiers naturels positifs

Algorithm 1 Algorithme de bandit

Require: Affect au moins un item au bras $a \in \mathcal{A}$

1: **for** t **do**
2: Reçoit un item c_t
3: Affecte c_t au bras A_t selon sa distribution ν_{A_t} et une stratégie π
4: Reçoit une récompense X_{c_t,A_t}
5: Mettre à jour ν_{A_t}

Output : Une séquence de choix et de récompenses

consiste à choisir le bras à allouer indépendamment des caractéristiques de l'item.

Ainsi, une stratégie π de bandit peut se définir en deux parties. La première partie suppose que les distribution de récompenses des bras suivent la même loi (comme par exemple la loi de Bernoulli, avec l'algorithme Thompson Sampling [70]) ou par une sous-gaussienne pour Ucb [6]). D'autres méthodes ne font aucune supposition, au prix d'un regret cumulé potentiellement important. L'autre partie d'une stratégie de bandit est le modèle mathématique mis en œuvre pour limiter le regret. Certaines méthodes choisissent par exemple le meilleur bras (avec les observations précédentes) selon une probabilité prédéfinie (comme l'algorithme Epsilon-Greedy [69]) ou adaptative (par exemple Softmax exploration [71]). Une stratégie de bandit est donc entièrement définie selon ces deux parties.

La section 2.2 présente les outils statistiques associés pour montrer les garanties théoriques d'une stratégie de bandit par rapport à une allocation statique, et ceci dans le cadre d'un une stratégie dite non contextuelle, c'est-à-dire que seules les récompenses sont prises en compte. La section 2.3 introduit la notion de contexte et ses conséquences sur les garanties théoriques s'il n'est pas pris en compte dans la stratégie de bandit. Les sections 2.4 et 2.5 présentent deux familles d'*approches contextuelles* dans lesquelles les caractéristiques des items sont considérées lors de l'allocation.

2.2 Stratégies d'allocation par bandit non contextuel

2.2.1 Outils statistiques

On considère ici que X est une variable aléatoire réelle ou discrète (qui prend ses valeurs dans $\mathbb{N}^*$). La distribution des récompenses d'un bras est caractérisée par un paramètre $\mu_a \in \mathbb{R}$, $a \in \mathcal{A}$, qui représente la moyenne de la variable X. Ces moyennes sont inconnues. Pour les déterminer, il est nécessaire d'utiliser un *estimateur* c'est-à-dire une statistique permettant d'évaluer ce paramètre inconnu, relatif à une loi de probabilité. Cet estimateur est construit à partir des observations vues jusqu'à présent. Ainsi, à un instant t, on considère pour chaque bras un estimateur $\hat{\mu}_a$ de sa moyenne.

Dans le cas où X est binaire, c'est-à-dire $X \in \{0; 1\}$, et $\mu_a \in [0, 1]$, les distributions de récompenses sont supposées suivre une loi de Bernoulli. La probabilité d'observer l'évènement $\{X = 1\}$ est alors égale à μ_a et la probabilité d'observer l'évènement $\{X = 0\}$ est égale à $1 - \mu_a$. L'intérêt de la loi de Bernoulli est de pouvoir modéliser n'importe quel cas applicatif où les récompenses sont des succès ou des échecs (par exemple survie/décès).

Soit X_t la récompense observée à l'instant t. Pour fournir un choix d'assignation A_t à chaque instant t, l'algorithme de bandit utilise une fonction F_t qui retourne un choix de bras en fonction des choix et récompenses passées.

$$A_{t+1} = F_t(A_1 X_1, \ldots, A_t X_t).$$

Cette fonction est arbitraire et n'est pas précisée ici. En effet, la définition de cette fonction varie selon l'algorithme de bandit utilisé. Néanmoins, quel que soit l'algorithme de bandit mis en œuvre, cette fonction dépendra uniquement des observations passées.

2.2.2 Maximiser les récompenses

Soit $\mu^* = \max_a \mu_a$ la moyenne du meilleur bras a^*. Le système cherche à maximiser la somme des récompenses obtenues :

$$X_1 + X_2 + \cdots + X_t + \cdots + X_T \qquad \text{où } T \text{ est le nombre total d'item à tester}$$

Cependant, les récompenses X étant des variables aléatoires, la somme $X_1 + X_2 + \cdots + X_T$ est également aléatoire. Cela signifie qu'en appliquant plusieurs fois une stratégie ayant une séquence de choix prédéfinie (par exemple alternativement, c'est-à-dire chaque bras tour à tour) sur des périodes différentes (par exemple à deux mois d'intervalle), l'analyste peut observer une somme de gain total différente pour une séquence de choix identique. L'expérience n'est ainsi pas reproductible et le critère numérique qui sera observé sera donc la moyenne de la somme des récompenses obtenues par l'algorithme divisée par T (nombre d'item à tester). C'est cette moyenne qu'il faut optimiser. Mathématiquement, l'espérance représente le gain que l'on peut espérer en moyenne et l'on va donc chercher à optimiser la quantité suivante :

$$\mathbb{E}\left[\sum_{t=1}^{T} X_t\right].$$

Cette estimation de l'espérance par la moyenne arithmétique est justifiée par la convergence en loi des récompenses X (par le théorème central limite [2], sachant que toute récompense X_t est une variable aléatoire). Les paramètres des bras sont inconnus a priori.

2. introduit en 1809 par Laplace

L'objectif est que la stratégie de bandit utilisée fasse aussi bien que la meilleure stratégie possible, c'est-à-dire une stratégie qui connaîtrait les paramètres des bras à l'avance et qui choisirait toujours le meilleur. Cette stratégie optimale est appelée la stratégie *oracle* [49]. Plus formellement, cette stratégie, dans une approche non contextuelle, donne l'espérance de gain égale à :

$$\mathbb{E}\Big[\sum_{t=1}^{T} X_t\Big] \simeq \underbrace{T\mu_{a^*}}_{\substack{\text{récompense} \\ \text{cumulée d'une} \\ \text{stratégie oracle}}}$$

Prenons l'exemple d'une variable X binaire. Une stratégie qui ne tire que le meilleur bras obtiendrait des récompenses successives X_t toutes suivant une loi de Bernoulli de paramètre égal à μ_{a^*} et ainsi, par linéarité de l'espérance :

$$\begin{aligned}\mathbb{E}\Big[\sum_{t=1}^{T} X_t\Big] &= \sum_{t=1}^{T} \mathbb{E}\Big[X_t\Big] \\ &= \sum_{t=1}^{T} \mu_{a^*} = T\mu_{a^*}. \end{aligned} \tag{2.1}$$

La comparaison avec cette stratégie permet de formaliser la notion de regret.

2.2.3 Le regret

Le regret est défini comme la différence entre ce qui est obtenu avec la stratégie optimale (ou oracle) de l'Éq. 2.1 et celle choisie par la stratégie d'allocation dynamique du bandit. Cependant, le regret va croître avec T puisque, plus il y a de choix à faire, plus il est possible de faire des choix de bras sous optimaux. Le regret R_T est donc une fonction croissante en T, et l'objectif est d'avoir le plus faible accroissement possible. Ce taux d'accroissement, noté $\frac{R_T}{T}$ peut s'écrire comme :

$$\frac{R_T}{T} = \mu_{a^*} - \frac{\mathbb{E}\Big[\sum_{t=1}^{T} X_t\Big]}{T}. \tag{2.2}$$

L'objectif est que $\frac{R_T}{T}$ tende vers 0 quand $T \to +\infty$. Cela sous-entend que le gain moyen converge asymptotiquement vers celui de la meilleure stratégie possible. L'objectif est donc :

$$\frac{R_T}{T} \underset{T+\infty}{\to} 0 \tag{2.3}$$

Pour simplifier la lecture des preuves théoriques, nous considérons dans cette section que le premier bras correspond à la plus grande moyenne de gain, le deuxième bras à la deuxième plus grande moyenne etc..., et que le k-ieme bras la plus faible. La figure 2.1

illustre une telle situation et permet d'introduire la quantité Δ_a (avec $a \neq 1$ et $< \Delta_a$) qui sera utilisée pour montrer la complexité du problème, avec Δ_2 l'écart entre μ_1 et μ_2, Δ_3 l'écart entre μ_1 et μ_3, etc ... :

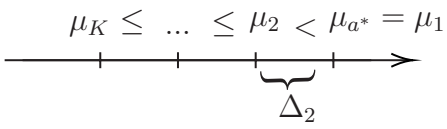


FIGURE 2.1 – Exemple de moyennes de bras décroissantes

Le regret peut alors s'écrire en faisant intervenir les écarts entre le bras optimal et les bras sous-optimaux :

$$\begin{aligned} R_T &= \sum_{a=2}^{K} (\mu_1 - \mu_a)\mathbb{E}[N_a(T)] \\ &= \sum_{a=2}^{K} \Delta_a \mathbb{E}[N_a(T)] \end{aligned}$$

où $N_a(T)$ est le nombre de tirages du bras a jusqu'à l'item T.

La minimisation du regret nécessite alors de :

— Tirer aussi peu que possible les bras sous-optimaux.

— Réaliser un compromis entre exploration et exploitation, c'est-à-dire assigner un nombre minimal d'items aux bras sous optimaux mais qui soit suffisamment grand pour pouvoir estimer correctement leurs moyennes de gain. Ainsi explorer revient à choisir un bras pour avoir un estimateur correct de son paramètre de distribution (ici la moyenne). Exploiter revient à choisir ce bras parce qu'il est supposé optimal.

2.2.4 Le compromis d'exploration/exploitation

Pour illustrer ce compromis d'exploration/exploitation, imaginons qu'un analyste ait T items testés à sa disposition, et que pour tout t, les variables X_t sont identiquement distribuées suivant une loi de Bernoulli de paramètre μ_a.

Stratégie 1 : Pure exploration L'analyste décide d'allouer uniformément ses items testés aux K versions de sa page. À la fin de l'expérience de test, l'analyste aura joué $\lfloor \frac{T}{K} \rfloor$ [3] fois chaque bras. Il obtiendra un regret égal à :

3. où $\lfloor . \rfloor$ désigne la partie entière inférieure de $\frac{T}{K}$

$$R_T = \frac{T}{K}\sum_{a=2}^{K}(\mu_1 - \mu_a)\mathbb{E}[N_a(T)]$$
$$= \frac{T}{K}\sum_{a=2}^{K}\Delta_a\mathbb{E}[N_a(T)]$$

Ce regret croît linéairement avec T et l'Eq. 2.3 n'est pas vérifiée. Cette stratégie n'est pas optimale puisqu'elle choisit $\lfloor\frac{T}{K}\rfloor$ fois le bras optimal mais aussi $(K-1)\frac{T}{K}$ les bras sous-optimaux.

Stratégie 2 : Pure exploitation Après avoir assigné un nombre m (avec $m \leqslant \frac{T}{K}$) fixé d'items à chaque bras, l'analyste choisit d'allouer systématiquement les items testés au bras qui lui apparaît comme optimal. Ce bras est le meilleur empiriquement après m items. Par conséquent

$$A_{t+1} = \underset{a}{\operatorname{argmax}}\,\hat{\mu}_a(t)$$

où $\hat{\mu}_a(t) = \frac{\text{somme des récompenses issues du bras a}}{\text{nombre de sélections du bras a}}$ est un estimateur de la moyenne inconnue $\mu_a(t)$. À un instant t, chaque bras a a reçu m items. Le problème d'une telle stratégie est qu'elle est dépendante de la valeur m choisie par l'analyste. Cette valeur peut être trop faible et rend alors la stratégie très mauvaise. Supposons en effet, que $m = 1$ et que pour le 1er item testé, le bras 1 est choisi et que $X_1 = 0$. Au second item testé, le bras 2 est choisi et la récompense $X_2 = 1$. Si le bras optimal est le bras 1, il sera écarté au profil d'un bras sous-optimal. En reprenant cet exemple, le regret serait :

$$R_T \geqslant \mathbb{P}[X_1 = 0|A_1 = 1]\mathbb{P}[X_2 = 1|A_1 = 2](\mu_1 - \mu_2)T$$
$$R_T \geqslant (1-\mu_1)(\mu_2)(\mu_1 - \mu_2)T$$

Si l'évènement $[X_1 = 0|A_1 = 1]$ et l'évènement $[X_2 = 1|A_1 = 2]$ se réalisent, alors le regret sera linéairement croissant avec T.

Stratégie 3 : Exploitation - Exploration Contrairement à la stratégie précédente, cette stratégie va chercher à calculer m de façon optimale, plutôt que de le fixer arbitrairement. Cet algorithme peut se voir comme un des premiers algorithmes introduisant la théorie des bandits [44]. Il repose sur des outils statistiques connus (comme la concentration de la mesure) pour proposer une exploration puis une exploitation [33, 44]. L'algorithme 2 résume cette approche pour un paramètre m fixé, spécifiant le nombre de tirages m à réaliser sur chaque bras pour l'exploration.

Soit $\mathbb{1}$ la fonction indicatrice d'un évènement. Pour calculer le regret d'une telle stra-

Algorithm 2 Explore puis exploite

Require: Une valeur $m \in 1, \ldots, \frac{T}{K}$
1: Choisir chaque bras m fois
2: Déterminer le meilleur bras empirique $\hat{a} = \text{argmax}_a \hat{\mu}_a$
3: Choisir $\hat{a}$ jusqu'à la fin du test :$A_{t+1} = \hat{a}$ avec $\forall t \leqslant Km$

Output : Une séquence de choix de bras et de récompenses

tégie, il faut commencer par définir le nombre de tirages d'un bras a sous-optimal [4] :

$$N_{a\neq 1}(T) = m + (T - Km)\mathbb{1}\{\hat{a} \neq 1\} \tag{2.4}$$

où : $(T - Km)\mathbb{1}\{\hat{a} \neq 1\}$ est le nombre de tirages supplémentaires qui seront réalisés si la condition $\{\hat{a} \neq 1\}$ est vraie. Il faut soustraire $(K - 1) * m$ les tirages des bras sous optimaux à la totalité des items testés : T.

Sachant que le nombre de tirages d'un bras sous optimal plus grand que m implique que sa moyenne empirique est supérieur à celle du bras 1 (meilleur bras dans cet exemple), l'Éq. 2.5 peut se ré-écrire comme :

$$N_{a\neq 1}(T) = m + (T - Km)\mathbb{1}\{\hat{\mu}_{a\neq 1}(Km) > \hat{\mu}_1(Km)\}$$

et son espérance :

$$\mathbb{E}[N_{a\neq 1}(T)] = m + (T - Km)\mathbb{P}[\hat{\mu}_{a\neq 1}(Km) > \hat{\mu}_1(Km)] \tag{2.5}$$

avec une valeur m suffisamment grande (qui sera précisée par la suite), l'évènement $\mathbb{P}[\hat{\mu}_{a\neq 1}(Km) > \hat{\mu}_1(Km)]$ devient peu probable. Soit $Y_{a,s}$: les récompenses successives du bras a. Soit $Z_1 = Y_{1,1} - Y_{a\neq 1,1}$, …, $Z_m = Y_{1,m} - Y_{a\neq 1,m}$.

$Z_1, \ldots, Z_m$ sont des variables aléatoires i.i.d. et d'après le théorème de la concentration de la mesure :

$$\frac{Z_1 + \cdots + Z_m}{m} \rightarrow_{m+\infty} \mu_a - \mu_1$$

Ainsi

$$\mathbb{P}[\hat{\mu}_{a\neq 1}(Km) > \hat{\mu}_1(Km)] = \mathbb{P}[\frac{Z_1 + \cdots + Z_m}{m} \geqslant 0]$$

4. Rappelons que pour simplifier la lecture, le bras 1 est défini ici comme a^*.

Pour garantir une borne théorique du regret, il est nécessaire de borner la probabilité que Z diffère de sa moyenne d'une certaine quantité ϵ. En d'autre terme, on cherche la borne supérieure de la distribution de probabilité de la forme :

$$\mathbb{P}[Z - \mathbb{E}[Z] \geqslant \epsilon] \quad \text{et} \quad \mathbb{P}[Z - \mathbb{E}[Z] \leqslant \epsilon]$$

En appliquant l'inégalité de Markov et en définissant $Y = |Z - \mathbb{E}[Z]|$ tel que Y est supposé intégrale au sens de Lesbegue : $|\mathbb{E}[Y]| < \infty$

Théorème 2.2.1 (Inégalité de Markov). *Soit Y une variable aléatoire strictement positive, pour tout $\epsilon > 0$, $Y\mathbb{1}\{Y \geqslant \epsilon\} \geqslant \epsilon\mathbb{1}\{Y \geqslant \epsilon\}$.*

$$\mathbb{P}[Y \geqslant \epsilon] \leqslant \frac{\mathbb{E}[Y\mathbb{1}\{Y \geqslant t\}]}{\epsilon} \leqslant \frac{\mathbb{E}[Y]}{\epsilon}$$

Soit ϕ une fonction non décroissante, non positive, définie sur un ensemble de valeurs réelles strictement positive $I \subset \mathbb{R}^+$ et Y une variable aléatoire prenant ses valeurs dans I implique que pour tout $\epsilon \in I$ et $\phi(\epsilon) > 0$, en utilisant l'inégalité de Chebyshev qui est le corollaire de l'inégalité de Markov :

Corollaire 2.2.1.1 (Inégalité de Tchebyshev)**.**

$$\mathbb{P}[|Z - \mathbb{E}[Z]| \geqslant \epsilon^q] \leqslant \frac{\mathbb{E}[|Z - \mathbb{E}[Z]|^q]}{\epsilon^q}$$

Avec q le moment d'ordre égale à un entier strictement positif et $\epsilon > 0$

Et dans notre cas $q = 2$

$$\mathbb{P}[|Z - \mathbb{E}[Z] \geqslant \epsilon|] \leqslant \frac{\mathbb{V}[Z]}{\epsilon^2}$$

Il faut ensuite faire intervenir la fonction caractéristique d'une variable aléatoire réelle Z pour définir de façon unique sa loi de probabilité ϕ. Par conséquent $\phi(\epsilon) = e^{\lambda\epsilon}$ avec λ une constante positive. Soit Z une variable aléatoire centrée réduite (donc $\mathbb{E}[Z] = 0$). L'inégalité de Markov implique que

$$\mathbb{P}[Z \geqslant \epsilon] \leqslant e^{-\lambda\epsilon}\mathbb{E}[e^{\lambda Z}]$$

Sachant que cette inégalité est vérifiée pour toutes les valeurs de $\lambda \geqslant 0$, on peut alors choisir λ tel qu'il minimise la borne supérieure de probabilité de l'évènement $\mathbb{P}[Z \geqslant \epsilon]$.

Pour cela, nous utilisons la méthode de Cramer-Chernoff qui utilise le logarithme de la fonction génératrice des moments ψ :

$$\psi_Z(\lambda) = \log \mathbb{E}e^{\lambda Z} \quad \lambda \geqslant 0$$

et introduisons

$$\psi_Z^*(\lambda) = \sup(\lambda - \psi_Z(\lambda))$$

L'inégalité de Chernoff est définie comme suit :

Théorème 2.2.2 (Inégalité de Chernoff)**.** *Soit Z une variable aléatoire centrée réduite,*

$$\mathbb{P}[Z \geqslant \epsilon] \leqslant e^{-\psi_Z^*(\epsilon)}$$

La résolution de $\psi_Z^*(\epsilon)$ permet de borner l'évènement $\mathbb{P}[Z \geqslant \epsilon]$ et de quantifier l'erreur d'un estimateur empirique par rapport à sa moyenne réelle. Selon la distribution de Z, cette borne peut être plus ou moins grande. Le lecteur peut retrouver le détail du calcul pour différentes distributions dans [11]. Le cas généralement admis est que Z est une variable aléatoire bornée dans $[a; b]$. Sous cette hypothèse :

- $\lambda = \frac{4\epsilon}{(b-a)^2}$
- $\psi_Z(\lambda) \leqslant \frac{\lambda^2(b-a)^2}{8}$
- $\psi_Z^*(\lambda) \leqslant \frac{t^2}{2\sigma^2}$ avec $\sigma^2 = \frac{(b-a)^2}{4}$
- $\mathbb{P}[Z \geqslant \epsilon] \leqslant e^{-\frac{\epsilon^2}{2\sigma^2}}$
- $\mathbb{P}[|Z| \geqslant \epsilon] \leqslant 2e^{-\frac{\epsilon^2}{2\sigma^2}}$

Soit :

$$\begin{aligned} \delta &= e^{-\frac{\epsilon^2}{2\sigma^2}} \\ \log(\delta) &= \frac{\epsilon^2}{2\sigma^2} \\ \epsilon &= \sqrt{2\sigma^2 \log(\frac{1}{\delta})} \end{aligned}$$

Ainsi, pour une erreur δ choisi, Z prend une valeur comprise dans l'intervalle $(-\sqrt{2\sigma^2 \log(\frac{1}{\delta})}; \sqrt{2\sigma^2 \log(\frac{1}{\delta})})$.

En reprenant l'Éq 2.5

$$\mathbb{P}[\hat{\mu}_{a\neq 1}(Km) > \hat{\mu}_1(Km)] = \mathbb{P}[\frac{Z_1 + \cdots + Z_m}{m} \geqslant 0]$$
$$\mathbb{P}[\hat{\mu}_{a\neq 1}(Km) - \hat{\mu}_1(Km) > \Delta] = \mathbb{P}[\frac{Z_1 + \cdots + Z_m}{m} \geqslant \Delta]$$

avec $\Delta = \mu_1 - \mu_{a\neq 1}$. En utilisant le lemme 2.2.2.1 ci-dessous, il est possible de garantir un regret théorique à la stratégie Explore puis Exploite.

Lemme 2.2.2.1. *Soit X, σ sous-Gaussienne, X_1, σ_1 sous-Gaussienne, X_2, σ_2 sous-Gaussienne. X_1 et X_2 sont supposés indépendants, alors*

- $\mathbb{E}[X] = 0$ *et* $\mathbb{V}[X] = \sigma^2$.
- cX *est* $|c|\sigma$*-sous Gaussienne pour tout* $c \in \mathbb{R}$ *(et c est une constante strictement positive).*
- $X_1 + X_2$ *est* $\sqrt{\sigma_1 + \sigma_2}$ *sous-gaussienne.*

Corollaire 2.2.2.2. *Soit $X_t - \mu$ une suite de variable aléatoire i.i.d. σ-sous gaussienne, alors pour tout $\epsilon \leqslant 0$ et $n \in \mathbb{N}^+$*

$$\mathbb{P}(\hat{\mu} \geqslant \mu + \epsilon) \leqslant \exp\Big(-\frac{n\epsilon^2}{2\sigma^2}\Big)$$

et

$$\mathbb{P}(\hat{\mu} \leqslant \mu - \epsilon) \leqslant \exp\Big(-\frac{n\epsilon^2}{2\sigma^2}\Big)$$

avec $\hat{\mu} = \frac{1}{n}\sum_{t=1}^{n} X_t$.

Preuve : D'après le lemme 2.2.2.1, $\hat{\mu} - \mu = \frac{X_t - \mu}{n}$ est σ sous gaussien, ce qui signifie que $\hat{\mu} - \mu$ a une distribution moins concentrée qu'une gaussienne[5] . En appliquant le théorème 2.2.2, on assure avec une probabilité $1 - \delta$ que

$$\mu \leqslant \hat{\mu} + \sqrt{\frac{2\sigma^2 \log(\frac{1}{\delta})}{n}} \tag{2.6}$$

5. De manière informelle, les queues d'une distribution sous-gaussienne sont dominées par les queues d'un distribution gaussienne.

et réciproquement

$$\mu \geqslant \hat{\mu} - \sqrt{\frac{2\sigma^2 \log(\frac{1}{\delta})}{n}} \tag{2.7}$$

Supposons que les distributions de chaque bras soient σ_a-sous gaussiennes, Z est $\frac{\sqrt{\sum_{a\in\mathcal{A}} \sigma_a^2}}{\sqrt{m}}$ Gaussienne (ou encore $\frac{\sigma_Z}{\sqrt{m}}$-sous gaussienne).

$$\mathbb{P}[Z \geqslant \Delta] \leqslant e^{-\frac{\Delta^2 m}{2\sigma_Z^2}}$$
$$\mathbb{P}[\hat{\mu}_{a\neq 1}(Km) \geqslant \hat{\mu}_1(Km)] \leqslant e^{-\frac{\Delta^2 m}{2\sigma_Z^2}}$$

Il est alors possible de quantifier le nombre d'essais de chaque bras. Redéfinissions plus généralement $\Delta_a = \mu_1 - \mu_a$,

$$\mathbb{E}[N_a(T)] \leqslant m + (T - Km) \sum_{a\in\mathcal{A}} \exp -\frac{\Delta_a^2 m}{2\sigma_Z^2}$$
$$R_T \leqslant \sum_{a\in\mathcal{A}} \Delta_a \mathbb{E}[N_a(T)]$$
$$R_T \leqslant m \sum_{a\in\mathcal{A}} \Delta_a + (T - Km) \sum_{a\in\mathcal{A}} \Delta_a \exp -\frac{\Delta_a^2 m}{2\sigma_Z^2}$$

abouti à une garantie théorique du regret.

Si l'on prend, par exemple, deux bras 1-sous-gaussiens, $\sigma_Z = \sqrt{2}$ le regret associé à un algorithme Explore puit Exploit sera de

$$R_T \leqslant m\Delta_2 + (T - 2m)\Delta_2 \exp\Big(-\frac{\Delta_2^2 m}{4}\Big). \tag{2.8}$$

Pour trouver la bonne valeur de m, il faut dériver l'inquéquation 2.8 et chercher son minimum. On retrouve par exemple pour une distribution 1-sous-gaussienne : $m = \frac{4}{\Delta_2^2} \log\left(\frac{\Delta_2}{4} T\right)$.

Conclusion : La valeur de m dépend de l'écart Δ_a. Plus cet écart est élevé, plus rapidement le meilleur bras sera identifié. Le problème est que cet écart est inconnu à priori. Cette stratégie est donc irréaliste. La section suivante propose une solution pour répondre à ce problème

2.2.5 Une borne adaptative : E.T.C.

Pour trouver le nombre d'items m nécessaire à l'exploration l'algorithme E.T.C. (*Explore Then Commit*) utilise une phase d'exploration à durée adaptative. Consistant à adapter dynamiquement la longueur de la phase d'exploration en fonction des récompenses observées, à chaque pas de temps, cet algorithme va prendre en compte l'écart entre les moyennes empiriques. Si cet écart est "suffisamment" grand, l'algorithme passe en phase d'exploitation. Il y a donc deux phases. La première est une exploration uniforme jusqu'à un instant τ qui sera le temps d'arrêt de la phase d'exploration. Une fois que le temps d'arrêt a été réalisé, l'algorithme passe à la deuxième phase : celle de l'exploitation. Selon les hypothèses de distribution, le lecteur peut trouver comment le calculer dans [33]. Un exemple de stratégie pour deux bras dont les récompenses suivent respectivement deux lois de Bernoulli de paramètres respectivement μ_1 et μ_2 :

$$\tau = \inf\left\{t \in \mathbb{N} : |\hat{\mu}_1(t) - \hat{\mu}_2(t)|\right\} > \sqrt{\frac{4\log(T/t)}{t}}$$

Ainsi, pour tout $t \in \{\tau, \dots, T\}$, $\hat{a} = \text{argmax}_a\, \hat{\mu}_a(\tau)$ et l'algorithme choisi ($A_{t+1} = \hat{a}$.

Dans [33], les auteurs garantissent une borne théorique du regret par le théorème suivant :

Théorème 2.2.3 (E.T.C.)**.** *Pour T items à tester, avec C une constante positive, l'algorithme E.T.C. a son regret borné par :*

$$R_T \leqslant \frac{2}{\Delta}\log(T) + C\sqrt{\log(T)}$$

Tout en étant adaptatif sur le temps d'arrêt et ne nécessitant pas la connaissance de Δ, E.T.C. possède le même taux de décroissance que la stratégie Explore Puis Exploite.

Le problème de cette méthode est que l'analyste doit connaître le nombre d'items T à tester en avance. La section suivante propose une autre stratégie, très utilisée, ne nécessitant pas la connaissance a priori de T.

2.2.6 Stratégie Ucb

La méthode Ucb est une méthode non contextuelle (i.e. non informée) basée sur une stratégie bayésienne optimiste (limites supérieures des estimations moyennes). Le principe de sa stratégie π est d'utiliser

— une surestimation de la moyenne empirique $\hat{\mu}_a$ pour chaque bras a,

— le nombre total d'items,

— leur allocation sur les différents bras,

pour affecter un nouvel item à un bras. Concrètement, un bras est choisi s'il est prometteur (parce que sa moyenne estimée est élevée) ou/et peu exploré.

En effet, les estimateurs $\hat{\mu}_{A_t}$ sont biaisés au début du test, en raison du petit nombre d'items considérés [44]. Pour contourner cette difficulté, Lai et Robbins, [48], proposent de calculer une surestimation de cette moyenne, appelée *limite de confiance supérieure*. L'acronyme Ucb veut dire Upper Confident Bound, comme la valeur la plus probablement supérieure à la valeur réelle de la moyenne. Les auteurs justifient leur proposition par une politique dite « optimiste face à l'incertitude ».

Cette valeur supérieure est la moyenne empirique obtenue estimée à un instant, plus un « bonus » d'exploration (aussi appelée intervalle de confiance). Elle dépend du nombre d'items assignés et leurs récompenses observées. Plus des items sont assignés à un bras, plus son bonus diminue.

Les algorithmes de type Ucb ont contribué très largement à la définition de méthodes d'allocation dynamique grâce à leurs performances surpassant une stratégie E.T.C. Ces algorithmes n'exploitent pas uniquement les moyennes empiriques, comme E.T.C. mais également des intervalles de confiance adaptatifs. Ucb calcule pour chaque moyenne $\hat{\mu_a}$ une borne supérieure de l'intervalle de confiance noté $U.C.B._a(t)$. Le principe est, qu'à chaque instant $t+1$, l'algorithme sélectionne $A_{t+1} = \text{argmax}_a\, U.C.B._a(t)$, c'est-à-dire le bras présentant la plus grande borne supérieure. La garantie théorique d'un tel algorithme revient à démontrer que la probabilité que la borne supérieure $U.C.B._a(t)$ soit plus grande que μ_a est grande que $(1-\delta)$. En d'autres termes, cela correspond à la probabilité que la surestimation $U.C.B._a(t)$ soit plus grande que la moyenne réelle μ_a.

Soit $(X_t)_{t=1}^n$ une séquence de variables aléatoires i.i.d. 1-sous gaussienne ayant pour moyenne μ et pour moyenne empirique $\hat{\mu} = \frac{1}{n}\sum_{t=1}^n X_t$. En utilisant l'équation (2.6) et l'équation (2.7) :

$$\mathbb{P}(\mu \geqslant \hat{\mu} + \sqrt{\frac{\log(\frac{1}{\delta})}{2n}}) \leqslant \delta \qquad \delta \in (0,1) \tag{2.9}$$

Considérons au $t$ème item, $N_a(t-1)$ observations pour le bras a et sa moyenne empirique : $\hat{\mu}(t-1)$. Pour construire l'algorithme Ucb proposé, Auer *et al* propose la borne supérieure de $\hat{\mu}(t-1)$ suivante :

$$U.C.B._a = \begin{cases} +\infty & \text{si} \quad N_a(t-1) = 0 \\ \hat{\mu}_a(t-1) + \sqrt{\dfrac{2\log(\frac{1}{\delta})}{N_a(t-1)}} & \text{sinon} \end{cases}$$

Le choix de δ est crucial. En effet un mauvais choix de cette valeur impliquerait que la borne supérieure de a^* serait inférieure à sa moyenne réelle μ_{a^*}. Dans un tel cas, l'algorithme de bandit risquerait de ne plus jouer le bras a^* et d'avoir un regret linéaire.

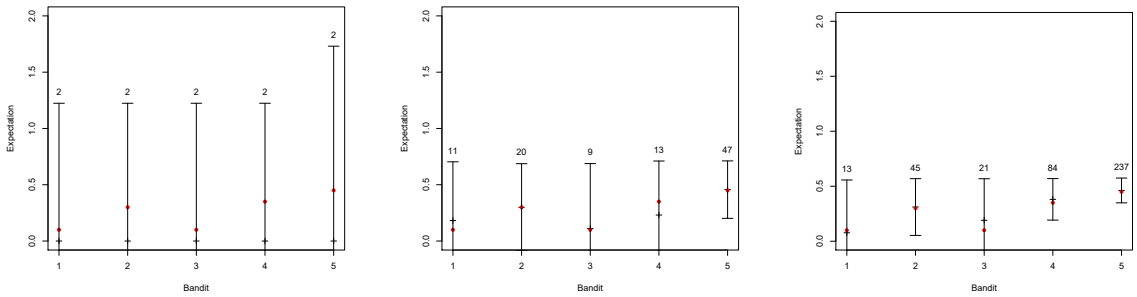
Lai et Robbins dans [48] ont suggéré de choisir une valeur de $\delta \leqslant \frac{1}{t}$ pour limiter la probabilité de sous estimer la moyenne réelle au cours du temps. L'algorithme en Annexe 6 montre une stratégie UCB possible, relative à l'algorithme UCB initialement proposé dans [7] par Auer *et al.*. Dans cet algorithme :

$$Upper_{\text{UCB}}(a,t) = \alpha\sqrt{\frac{2log(t)}{T_a(t)}} \tag{2.10}$$

où $T_a(t)$ est le nombre de fois que le bras a a été choisi et α est donné par l'analyste. Dans la version initiale de UCB, $\alpha = 1$ mais dans la pratique il est montré que le choix optimal de cette valeur dépend des distributions des bras [16].

Après chaque affectation, $\hat{\mu}_{A_t}$ est mis à jour et sa borne est réduite (voir équation 2.10). Comme l'intervalle de confiance dépend de $T_a(t)$, plus la valeur $T_a(t)$ est grande, moins la surestimation est importante. En effet, la surestimation de la moyenne du bras choisi A_t diminue, pour être égale à sa moyenne réelle. Les bornes supérieures des bras non choisis restent inchangées.

La figure 2.2 montre un exemple d'évolution de l'intervalle de confiance associée à cinq bras en fonction du nombre d'items soumis.



(a) Après 10 items testés (b) Après 100 items testés (c) Après 400 items testés

Les points rouges sont les moyennes réelles. Les croix noires sont les moyennes estimées.

FIGURE 2.2 – Exemple d'évolution des intervalles de confiance d'UCB

Pour prouver la convergence logarithmique du regret cumulé, [6,64] définissent la limite supérieure du regret cumulé par :

Théorème 2.2.4 (Regret d'une stratégie UCB)**.** *Considérons une stratégie de bandit* UCB *à* $|\mathcal{A}| = K$ *bras, dont les distribution de gain sont* 1 *sous gaussienne. Pour toute valeur* $\delta = \frac{1}{t}$*, le regret* R_t *à l'instant* t *est borné par*

$$\mathbb{E}[R_t] \leqslant \sqrt{8|\mathcal{A}|t(\log t + \frac{\pi^2}{3})} \tag{2.11}$$

En référence à l'équation 2.11, quand t tend vers $+\infty$, $\mathbb{E}[R_t]$ tend asymptotiquement vers une constante. Ne nécessitant que la moyenne des bras et le nombre d'items assignés à chacun d'eux, cet algorithme offre une faible complexité et permet à l'analyste d'obtenir un choix rapidement. Dans [14], le lecteur peut trouver plus de détails sur les propriétés de la stratégie Ucb. Des variantes de l'algorithme Ucb ont été proposées pour améliorer la borne théorique dans [5, 32, 34].

Limite d'une stratégie Ucb

La stratégie Ucb est efficace lorsque la distribution des récompenses est σ-gaussienne, hypothèse qui peut être vérifiée a posteriori par une allocation uniforme. Malheureusement, les expériences ont montré que cette hypothèse est rarement vérifiée. De fait, dans la réalité, la limite supérieure n'est toujours pas fiable. Par conséquent, pour trouver a^*, Ucb va nécessiter plus d'items pour trouver a^* bras. De plus, si la récompense présente des valeurs extrêmes, l'exploration sera aussi plus longue. En effet, ces deux situations impliquent une forte variance dans la distribution de gain des bras or le « bonus d'exploration » ayant une décroissance logarithmique, Ucb peut surestimer pendant longtemps les moyennes de bras sous-optimaux.

En revanche, Ucb ne nécessitant que $\hat{\mu_a}$ et $N_a(t)$, a une faible complexité algorithmique et ses résultats sont généralement bien compris par l'analyste d'un test A/B. Cette stratégie répond donc aux problèmes d'interprétation et de temps de réponse limité.

En résumé cette section a montré l'intérêt de l'allocation dynamique par rapport à une allocation statique définie en amont du test pour maximiser le gain et minimiser le regret à la fin du test. Une stratégie Explore puis Exploite permet de s'affranchir du temps d'arrêt mais présente un risque non négligeable de choix de mauvaises variations pour l'exploitation. Pour éviter une telle situation, la stratégie Ucb peut être appliquée, puisqu'elle permet à l'algorithme de re-proposer une variation qui n'a pas été proposée depuis longtemps.

2.3 Allocation dynamique et contexte

Dans de nombreux domaines applicatifs, les items soumis au test ont naturellement tendance à produire une récompense différente, selon leurs propres caractéristiques. De fait, ces items soumis au test peuvent ne pas être homogènes. Ainsi, par exemple, les décisions d'achat sur un site dédié au sport peuvent fortement dépendre du sexe ou de l'âge du visiteur, voire de sa taille ou de son poids. Cette non homogénéité de la population remet en cause l'hypothèse que les récompenses des items sont indépendantes et identiquement distribués (et donc que la distribution de probabilité autour de la récompense moyenne est normale). Dans le cas où la population est non homogène, c'est-à-dire

lorsque les valeurs de gains sont très dispersées autour de la moyenne et ne suivent pas une distribution gaussienne, les garanties théoriques de convergence de certains modèles de bandits non contextuels ne tiennent plus et leur performance en est détériorée. Trouver un unique a^* ne peut plus être réellement pertinent. La Figure 2.3 présente un exemple réel d'une telle distribution visiblement non gaussienne. Il s'agit de la distribution de la valeur des paniers après la consultation d'une page d'un site d'e-commerce.

Parallèlement, l'analyste peut être intéressé par des réponses à des questionnements plus complexes : une variation est-elle optimale pour tous les items ? Y a t-il une variation optimale pour certains d'entre eux et non-optimale pour d'autres ?. . ..

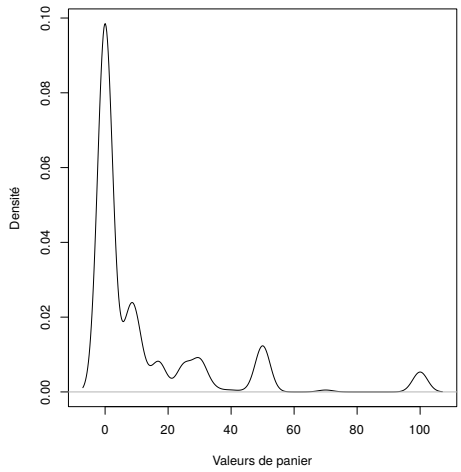


FIGURE 2.3 – Exemple d'une courbe de densité de valeurs de panier pour des 1322 visiteurs ayant vu une page web. L'hypothèse de normalité est rejetée (p-valeur < 0.05).

Supposons par exemple qu'il existe deux sous-groupes d'items ayant des réponses différentes aux variations du test. Par exemple, un traitement alternatif peut être efficace pour les personnes âgées et non pour les jeunes, une page ne peut être optimale que pour les utilisateurs de smartphones. Dans ce cas, le ν_a est très loin d'une distribution gaussienne. Si l'on suppose que ν_a est un mélange de gaussiennes, et que la récompense reçue est corrélée aux caractéristiques d'un item, il est possible d'appliquer une stratégie contextuelle. L'idée des stratégies d'allocation dite contextuelle (présentée ci-dessous) est que la récompense X_{c_t,A_t} dépend à la fois du bras attribué et des caractéristiques de l'item, c'est-à-dire de son *contexte*..

Pour cela, les méthodes contextuelles supposent qu'il existe des sous-groupes d'items présentant chacun une distribution de récompenses différentes. Néanmoins, si les expériences ont montré qu'ils existaient très fréquemment de tels groupes, il s'avère que les définir à partir de rien, avant le test, est souvent difficile. En effet, demander à l'analyste de les définir « manuellement » est souvent irréaliste, soit parce que cette opération peut être chronophage, soit parce qu'il n'a pas une vision claire de ceux-ci.

De fait deux approches contextuelles sont proposées dans la littérature :

— Des approches par *bandit contextuel* : respectant le principe d'un apprentissage partant de zéro, elles apprennent le lien entre le contexte et la récompense pendant le test, c'est-à-dire pendant l'allocation dynamique. Ce lien est généralement modélisé par K fonctions de régression (une pour chaque bras). Chaque fonction prend en entrée un contexte, pour prédire une récompense espérée. Les paramètres de cette fonction sont estimés au cours du temps (voir Section 2.4). En réalité, ces méthodes sont plus intéressées par la maximisation des gains que de l'explicabilité des choix auprès de l'analyste.

— Des approches en deux étapes où les groupes sont préalablement définis avant le test A/B (voir Section 2.5).

2.4 Stratégies d'allocation par bandits contextuels

2.4.1 Bandits contextuels

Dans un bandit contextuel, des récompenses sont supposées être générées à partir d'une fonction inconnue dépendante des contextes des items et du bras choisi. L'objectif est d'estimer cette fonction pendant le test. Par conséquent, des hypothèses sont faites sur le type de la fonction, comme par exemple sa linéarité.

Les bandits contextuels linéaires tels que Lin-Ucb estiment les paramètres des bras par une inversion de matrice. La remise à jour de leurs paramètres présente une complexité quadratique vis à vis du nombre de caractéristiques des items d, et ralenti donc le temps de réponse de l'algorithme [49]. Ces approches ont néanmoins démontré leurs performances théorique et pratique dans la réduction du regret cumulé [15]. De fait, l'algorithme Lin-Ucb est très utilisé de par ses performances et sa simplicité (sous condition d'un nombre de caractéristiques limité, la complexité de cet algorithme étant $\mathcal{O}(d^3)$ où d est le nombre de caractéristiques). La figure 2.4 montre le regret cumulé dans le temps t avec des données simulées générées avec une fonction de récompense linéaire. De cette figure, on peut voir que l'algorithme Lin-Ucb surpasse d'autres méthodes de bandit non contextuel tels que Thompson Sampling, uniform et Ucb.

Dans le cas où les hypothèses de linéarité ne sont pas vérifiées, d'autres techniques statistiques ont été proposées. Dans [26] les auteurs proposent une méthode basée sur un modèle linéaire généralisé (G.L.M.), appelé GLM-UCB permettant au modèle linéaire d'être relié à la récompense via une fonction de liaison (inverse). Cette généralisation permet d'envisager une classe plus large de problèmes et en particulier les cas où les récompenses sont des comptages ou des variables binaires utilisant, respectivement, une loi Poisson ou une régression logistique. Comme Lin-Ucb, GLM-UCB nécessite une inversion de matrice et peut être très coûteux en temps de réponse si le nombre de carac-

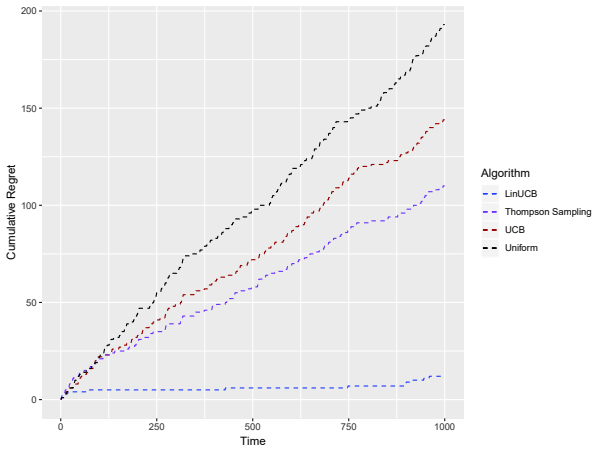


FIGURE 2.4 – Regret cumulé de LIN-UCB, UNIFORM et UCB sur des données simulées générées avec une fonction de récompense linéaire

téristiques décrivant les items est important.

Plus récemment, un algorithme de bandit contextuel basé sur les forêts aléatoires a été proposé via l'algorithme BANDITFOREST [30]. BANDITFOREST assigne les items aux bras de façon uniforme et aléatoire jusqu'à ce que tous les arbres soient complètement appris par rapport à un contexte particulier. Ceci a pour conséquence une assignation importante des items aux bras à faible moyenne de gain, ce qui détériore les performances empiriques de l'algorithme (regret cumulé important). De plus, dans [24], les auteurs remarquent que l'algorithme requiert quatre paramètres très spécifiques au contexte applicatif et nécessitent donc une expertise importante pour être établis. Deux paramètres influencent directement le niveau d'exploration, l'un contrôle la profondeur des arbres et l'autre détermine le nombre d'arbres dans la forêt. Les auteurs ont alors proposé l'algorithme TREEBOOTSTRAP qui ne nécessite aucun paramètre d'exploration préalable et choisit la profondeur automatiquement lors de la construction de l'arbre. Cependant, cet algorithme ne prend en compte que les récompenses binaires, ce qui peut limiter son application. Une autre alternative est l'algorithme KERNEL-UCB [72] qui fournit une modélisation non linéaire de la fonction de gain (comme GLM-UCB) en utilisant un espace de Hilbert du noyau de reproduction (RKHS). Malheureusement, KERNEL-UCB nécessite également un temps de réponse important pour fournir un choix et requiert la connaissance a priori du nombre total d'items à tester.

2.4.2 Limites des bandits contextuels

Les stratégies d'allocation par bandits contextuels représentent la relation entre le contexte et la variation par un modèle, dont les paramètres sont estimés au cours du temps. Ainsi, plus le modèle est complexe, plus le temps de réponse sera long. Malheureusement, un modèle à faible complexité peut ne pas être plus performant qu'une allocation statique lorsque les hypothèses faites ne correspondent pas à la réalité. De plus,

selon le choix de l'algorithme, ce modèle peut être facilement interprété par l'analyste (comme pour Lin-Ucb ou GLM-UCB), ou non (comme Kernel-Ucb, BanditForest ou TreeBootstrap).

C'est cette contrainte de temps de calcul et d'interprétablilité qui freine aujourd'hui certains utilisateurs d'AB test à appliquer l'allocation dynamique. En effet, il est par exemple inenvisageable pour un e-commerçant de ralentir l'affichage de sa page web, ou pour un industriel, sa chaîne de production. Ces derniers préféreront une allocation statique, au détriment du coût du test. A défaut d'un test optimisé, ils préféreront réaliser une analyse éventuelle post-test sur les données, pour détecter des groupes plus ou moins sensibles au test. Cependant, une telle analyse implique que l'analyste ait les compétences pour la réaliser. Cependant, en observant des analyses post-tests, il a été constaté que des groupes détectés sur les données collectées sur la variation A était très souvent aussi présents dans les données collectés sur la variation B. Par exemple, les visiteurs étant déjà venus sur le site, achètent plus que les visiteurs identifiés pour la première fois par le site. De même, des patients fragiles peuvent avoir une probabilité plus faible de survivre. Utiliser des groupes déterminés sur la variation A comme des groupes a priori est une approche contextuelle qui semble réaliste. Basées sur ces constatations, de nouvelles méthodes contextuelles par identification préalable de ces groupes ont vu le jour.

2.5 Stratégies d'allocation par pré-classement des items et multi bandits

2.5.1 Principes

Pour prendre en compte les caractéristiques des items et les sous-groupes implicites, les méthodes présentées précédemment telles que Lin-Ucb, proposent des approches basées sur des bandits contextuels où les groupes sont construits dynamiquement, c'est-à-dire pendant la phase d'exploration. Cependant, comme nous l'avons dit dans la section précédente, ces méthodes présentent deux inconvénients majeurs rendant leur mise en œuvre difficile :

— un temps de calcul pouvant être important lors de l'allocation d'un item. Dans de nombreux domaines, les contraintes de temps de réponse imposent que le système délivre un choix d'assignation dans un temps très court voir temps réel. Par exemple dans le WEB, le temps de réponse doit souvent être inférieur à 200 ms. En effet, il est évident qu'un temps de latence trop élevé a des conséquences négatives sur les visites sur le site. Or, la construction dynamique de cette segmentation complexifie le mécanisme d'allocation et influe fortement sur le temps de réponse à l'obtention d'un choix. De plus, ce temps de réponse augmente fortement avec le nombre

de caractéristiques décrivant les items. De fait, pour être concrètement mises en œuvre, ces méthode nécessitent souvent une sélection préalable des caractéristiques. Malheureusement, dans les faits, une telle sélection s'avère difficile à effectuer. Cette contrainte de temps réel peut aussi se retrouver dans d'autres problématiques industrielles, comme par exemple un processus industriel où il est impératif de ne pas ralentir la chaîne de production.

— des choix d'assignation peu interprétables. Or, dans bien des cas, l'analyste est aussi intéressé à identifier les groupes (s'il existent) ayant eu des réactions différentes, qu'elles soient positives, négatives ou neutres, aux différentes variations.

Par ailleurs, il est généralement admis qu'une des variations tesées, généralement associée à la variation (A) est la variation originale c'est-à-dire celle en exploitation avant le test. Des informations sur les gains réels produits par cette variation, en fonction des items soumis, sont disponibles ou peuvent l'être si nécessaire. Or, aucune méthode d'allocation dynamique contextuelle actuelle ne tire avantage de cette connaissance. Pourtant, il parait évident que ces informations devraient pouvoir permettre de mieux configurer les test A/B à venir. Ainsi, à notre avis, cette information devrait permettre

— de sélectionner les caractéristiques des items intéressantes pour définir les groupes, c'est-à-dire celles influençant les gains.

— de pré-définir des groupes en fonction de leur comportement vis-à-vis de la variation A et ainsi à la fois d'améliorer leur qualité, les choix d'allocation et aussi de réduire les temps de calcul lors de la phase d'allocation elle-même.

De plus, l'identification des tels groupes avant le test et leur analyse donnera à l'analyste des indications sur la composition du test lui-même et en particulier sur les modifications à apporter à A. Par exemple, sachant qu'il existe un groupe important d'utilisateurs sur mobile, est-il pertinent de proposer une version *responsive*[6] ? Sachant que les femmes réagissent mal au traitement original, comment l'améliorer ?

Des méthodes d'allocation dynamiques basées sur des groupes explicites, tels que Single-K-UCB (voir section 2.5.2) ont donc été proposées. Elles nécessitent de d'identifier des groupes au préalable de la phase d'exploration (voir section 2.5.3), et de définir une fonction de classement des items dans ces groupes (voir section 2.5.4).

2.5.2 Single-K-UCB

Dans [50], les auteurs proposent la méthode Single-K-UCB qui applique une stratégie non contextuelle à chaque groupe homogène d'items. Ils démontrent que si les groupes sont correctement construits, le regret cumulé au cours du test diminue significativement. Soit k un groupe possible parmi l'ensemble $\mathcal{K}$ des groupes dénombrable existant. Le regret

6. interface optimale quelle que soit le type d'appareil utilisé (téléphones mobiles, tablettes, liseuses, ordinateur).

cumulé de SINGLE-K-UCB est borné par $\sum_{k\in\mathcal{K}}\sum_{a\in\mathcal{A}}\frac{24\sigma^2\Delta_{a,k}\log T}{\Delta^{+2}_{a,k}}$ où $\Delta^{+2}_{a,k}$ est le gap minimal entre les moyennes d'un même bras mais pour différent groupes et tel que $\Delta^{+2}_{a,k}\geq\Delta_{a,k}$

Les auteurs considèrent que les groupes sont définis par leurs distributions de probabilité, déterminées par une variable latente. Une variable latente est une variable qui ne peut pas être mesurée directement, mais qui est supposée être à la base des variables observées. Considérant une fonction bijective f qui associe à chaque item via son contexte c_t un groupe k tel que $f(c_t)=k$. La distribution de probabilité des récompenses de tout bras a pour un groupe k : $\nu_{a,f(c)=k}$ est σ-Gaussienne (avec σ^2 la variance connue). Ils ne traitent cependant pas du choix de cette fonction f en pratique. De plus, ils supposent que les groupes sont connus et ne donnent aucune indication sur comment obtenir des groupes pertinents. Ils considèrent le problème théorique, dans un environnement particulier, ce qui limite fortement la garantie de performance lorsque la distribution des récompenses est inconnue dans ces groupes [58].

2.5.3 Identification des groupes d'appartenance

Mettre en œuvre une méthode basée sur des groupes nécessite d'utiliser un mécanisme de classement des nouveaux items. Il est nécessaire de plus, que chaque groupe ait une distribution de gains gaussienne.

Il est également indispensable de postuler qu'il existe un lien entre les caractéristiques des items et les récompenses obtenues. Cette hypothèse forte, déjà pressentie, s'est avérée vérifiée en pratique dans nos expériences, ce qui tend à montrer qu'elle est raisonnable en soi. Dans la suite, nous considérons donc que notre base de données (référée ici par $\mathcal{L}$) issue de l'exploitation passée de variation (A) contient pour chaque item, son contexte ainsi que la récompense obtenue, respecte cette hypothèse. L'identification de groupes au sein d'une population est en soi un problème ouvert régulièrement rencontrée en fouille de données et en statistiques.

Deux cas se présentent alors : soit les groupes (i.e. classe) et leurs récompenses associées sont connues a priori. Dans ce cas une approche supervisée est possible pour les définir. Soit il n'existe pas de connaissance a priori de ces groupes et une approche non supervisée est nécessaire.

Dans une approche supervisée, cela consiste, à partir des données existantes, à identifier directement des groupes et à construire un modèle de classement associé capable de classer automatiquement des nouveaux items dans les groupes ainsi définis. Il existe dans la littérature de nombreuses méthodes répondant à ce problème. Ainsi, des méthodes basées sur le principe Support Vecteurs Machine pourrait être utilisées. Malheureusement celles-ci ne seraient applicables que pour des récompenses binaires (Conversion/non conversion, survie/non survécue, ...) [39] et les résultats sont peu explicites et donc plus ou moins interprétables.

Dans une approche non supervisée, cela consiste à diviser (segmenter) la population en groupes homogènes à partir des caractéristiques qui décrivent les individus de cette population en fonction d'un objectif donné en amont puis d'extraire un modèle potentiel de classement par exemple à partir des centres des clusters proposés par K-means, les lois de distributions construites par EM ... Néanmoins, ce problème est réputé mal défini et mal posé. Il nécessite d'introduire de nombreux biais à la fois sur les objectifs et critères de regroupement et sur les méthodes d'optimisation de ces critères [21]. Par conséquent, de nombreuses méthodes ont été proposées dans ce sens, se différenciant notamment par l'objectif choisi pour dissocier les groupes [45]. Néanmoins, aucune de celles-ci ne nous a paru apte à générer des groupes respectant la contrainte de normalité des gains.

D'autres méthodes basées sur la construction d'arbres de régression semblent plus appropriées à une interprétation a posteriori des groupes. Cette approche a déjà été utilisée dans [25] dans le cas d'une récompense binaire ($X \in \{0; 1\}$). Elle a été également utilisée dans [29] pour la construction des groupes à travers plusieurs modèles construits par un mécanisme de *bootstrapping* ce qui ne permet qu'une faible interprétabilité des groupes. Les arbres de décision de type C4.5 [59] ou C.A.R.T. [13] ont montré de très bons résultats pour identifier des groupes homogènes avec une variable d'intérêt booléenne ou réelle. Ces méthodes utilisent l'entropie de la mesure (C4.5) ou encore le cœfficient de Gini (C.A.R.T.). Malheureusement, elles présentent le désavantage de sur-apprendre et ont un biais de sélection en faveur des caractéristiques numériques au détriment des caractéristiques catégoriels [67]. L'algorithme « Conditional Inference Trees » [41], à la différence de ces méthodes, utilise un test statistique différent en fonction de la nature de la caractéristique considérée, ce qui permet d'éviter ce biais de sélection. Les expériences montrent aussi que les arbres créés sont souvent fortement différents de ceux créés par les autres méthodes.

Pour décider de l'approche à mettre en oeuvre dans notre méthode, nous avons étudié (et testé) différentes méthodes de construction de groupes à savoir CART, C5.5 et CTREE. Une étude des caractéristiques de ces méthodes (et nos expérimentations) ont montré que CTREE présentait la meilleure adéquation avec notre objectif. En effet, il permet la création de groupes ayant une distribution de gain gaussiennne sans nécessiter d'élagage a posteriori par l'analyste. De plus, CTREE produit directement un indicateur probabiliste (p-valeur [7]) indiquant la fiabilité du modèle. Enfin, il autorise la présence de valeurs manquantes dans les caractéristiques des items traités. Nous l'avons donc retenu pour l'étape de création des groupes dans notre méthode. Ainsi, nous présentons plus en détail la méthode dans la section suivante.

7. la p-valeur, dans ce contexte, est la probabilité que la corrélation observée entre une caractéristique et les récompenses soit due au hasard

2.5.4 CTREE

À l'initialisation, tous les items sont dans un même groupe, associé à la racine de l'arbre. Un partitionnement récursif est alors appliqué à la racine. L'ensemble des caractéristiques $j \in \{1, \ldots, d\}$ pour chaque item est un vecteur aléatoire Y. L'hypothèse d'indépendance H_0 (avec un risque préfixé de type I : ϵ) entre chaque caractéristique Y_j et la récompense X est évaluée, pour tous les items présents dans l'ensemble considéré (initialement la racine puis le nœud lorsqu'un partitionnement est réalisé). Selon le résultat du test :

— Si l'hypothèse d'indépendance H_0 est rejetée, l'ensemble est divisé en deux groupes disjoints à l'aide de la caractéristique Y_{j^*} ayant la plus grande corrélation avec la récompense. Le choix de la valeur de Y_{j*} qui permettra de séparer l'ensemble sera celle qui maximise la divergence entre les distributions [8] des deux groupes.

— Si l'hypothèse d'indépendance H_0 est acceptée, pour toutes les caractéristiques Y_j, la récursivité s'arrête.

Les étapes de CTREE sont rapportées dans l'algorithme 3. Pour vérifier l'hypothèse de corrélation, de nombreux tests statistiques présents dans la littérature existent (par exemple A.N.O.V.A, le test de Spearman, Wilcoxon Man Witney, le test du Chi2, ... [42, 68]). Lors de l'utilisation de CTREE (et en particulier dans la méthode que nous proposons), ces tests seront spécifiés automatiquement et seront appliqués selon le type de caractéristique (continue, binaire, catégorielle) et de récompense (continue, binaire) observée. L'usage de CTREE nécessite de définir :

— un risque ϵ qui est la valeur minimale prise par la p-valeur pour rejeter H_0. Cette valeur est généralement égale à 0.05 ce qui correspond à un risque de 5% de considérer qu'une caractéristique n'est pas corrélée à la récompense alors qu'elle l'est.

— une fonction d'influence h c'est-à-dire une transformation sur les récompenses. Dans notre cas, nous travaillons avec la moyenne de récompense numérique et cette transformation n'est pas nécessaire, ainsi $h(X) = X$.

— La fonction de transformation g qui transforme une caractéristique catégorielle en vecteur composé de valeurs binaires, d'une dimension égale au nombre de modalité que prend cette caractéristique. Par exemple un caractéristique catégorielle décrivant l'age d'une visiteur prenant trois modalité (enfant, adulte et senior) sera transformé en trois caractéristiques binaire (prenant la valeur 0 ou 1 selon la modalité du visiteur). Pour les caractéristiques numériques g ne fait aucune transformation.

Un avantage majeur des arbres d'inférence conditionnelle est qu'ils ne requièrent aucun élagage et évitent ainsi le *sur-apprentissage* [66]. La sélection de la valeur de fractionnement est basée sur les p-values univariées [9], ce qui évite un biais de sélection en faveur des caractéristiques avec de nombreuses valeurs de fractionnement possibles. Par exemple

8. de récompense où de probabilité d'en avoir une si la récompense est binaire
9. c'est-à-dire testant le lien entre une seule caractéristique et la récompense

C.4.5 effectue une recherche exhaustive sur toutes les divisions possibles en maximisant une mesure d'information de l'impureté de nœud en sélectionnant la caractéristique montrant la meilleure division. Une telle approche sélectionnera alors en priorité les caractéristiques ayant de nombreuses scissions possibles.

Algorithm 3 CTREE algorithm

Require: — Un risque d'erreur $\epsilon \in]0, 1[$
— Un jeu de données de caractéristiques Y et leur réponse X.
— La fonction d'influence h de X.
— Une fonction de transformation g_j sur la caractéristique Y_j.
1: Calcule la statistique de test s_{j0} pour les données observées.
2: Permute plusieurs fois aléatoirement les observation dans le nœud.
3: Calcule la statistique de test s pour toutes les permutations.
4: Calcule la p-value (nombre de statistiques de test s, où $|s| > |s_0|$).
5: Corrige la p-values for tests multiples.
6: **if** H_0 accepté (p-value $> \epsilon$ pour tout Y_j). **then**
7: **return**
8: **else**
9: Sélectionne la caractéristique Y_j^* avec la plus forte association (p-value minimale).
10: Cherche la meilleure coupure Y_j^* (maximise la statistique de test s) et partitionne en deux sous ensemble les données.
11: Appliquer CTREE sur les deux sous-ensembles.
Output : Un partitionnement récursif.

Si une observation statistiquement significative a pu survenir par « hasard » en raison de la taille du sous ensemble observé, une correction Bonneferoni peut être appliquée [23]. En effet si plusieurs hypothèses sont testées, la probabilité de rejeter incorrectement une hypothèse nulle augmente. La correction de Bonferroni compense cette augmentation en testant chaque hypothèse avec un risque d'erreur ϵ adapté au nombre de tests statistiques ayant déjà été réalisés. Les tests intégrant des caractéristiques catégorielles peuvent nécessiter un temps de calcul très long lorsqu'une correction de Bonferroni est appliquée [40]. En effet, la segmentation testant toutes les combinaison de modalités, elle nécessiter plus de temps. Pour réduire ce temps, il est possible d'utiliser une correction de Bonferroni par la méthode de Monte Carlo [31]. Cependant, une telle correction comprend une partie aléatoire, qui peut faire varier la structure de l'arbre.

Deuxième partie

Une nouvelle approche pour une allocation dynamique temps réel contextuelle

Chapitre 3

Ctree-Ucb : une méthode d'allocation dynamique par explicitation de groupes

3.1 Ctree-Ucb : un nouvelle approche pour une allocation dynamique contextuelle temps réel.

Dans cette thèse nous nous plaçons dans le cas où les groupes ne sont pas connus a priori. Dans ce cadre, pour répondre à la double problématique d'intégration du contexte des items et du temps de réponse rapide, nous proposons une approche contextuelle basée sur l'explicitation préalable des groupes. Cette explicitation des groupes consiste, d'une part, à construire/extraire des groupes à partir des informations sur les items et des gains réels liées à l'exploitation de la variation A originale, et d'autre part, à définir un modèle de classement associé. Lors de la phase d'exploration du test, plusieurs bandits, chacun dédié à un de ces groupes pré-définis, seront utilisés. Ainsi, pour chaque nouvel item à tester, notre méthode le classera automatiquement dans un des groupes prédéfinis, le bandit associé à ce groupe se chargeant de lui affecter une variation.

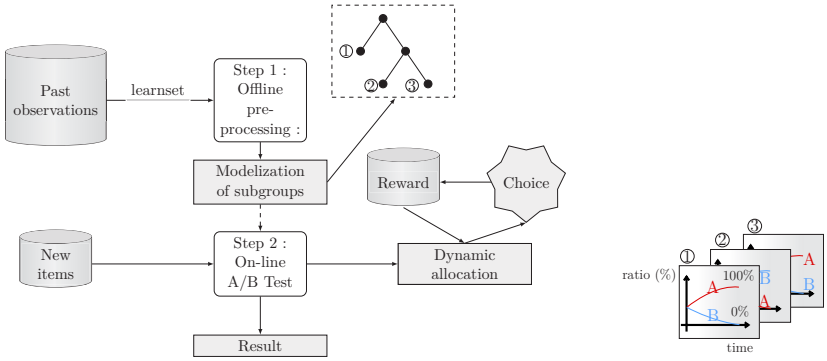


Figure 3.1 – Ctree-Ucb : Approche contextuelle d'un A/B test

Le schéma général de notre méthode Ctree-Ucb consiste en deux étapes (la figure 3.1 présente une vue schématisée du processus proposé). Les groupes définis en amont du test présenteront chacun, par construction, une distribution de probabilité (associée à la

récompense espérée) suivant une loi normale [1]. Dès lors, pendant la phase d'exploration, l'allocation dynamique des items associés à chacun des groupes pourra être assurée par un bandit non contextuel. L'analyste pourra aussi en déduire des liens plus explicites entre les variations optimales et les items pour ainsi mieux appréhender l'impact des modifications sur différents sous-ensembles d'items. Notre méthode répondra donc plus sûrement à la problématique de l'interprétabilité des résultats.

Bien évidement, la mise en œuvre d'une approche basée sur cette hypothèse ne pourra se faire que si les groupes issus de l'analyse du passé sont encore pertinents lors du test. Dans le cas du WEB d'e-commerce, [10] les auteurs indiquent que les groupes de visiteurs sont souvent persistants, indépendamment de la page affichée. Par exemple un visiteur peu intéressé par un article n'achètera pas quelque soit la version A ou B de la page affichée et réciproquement. Dans le domaine médical, un patient en mauvaise santé n'augmentera que de peu ses chances de guérison quelque soit le traitement.

3.1.1 Étape d'explicitation des groupes

Une première étape d'*explicitation des groupes*, réalisée hors ligne, en préalable de la phase d'exploration, consiste à identifier des groupes et un modèle de classement associé grâce à un mécanisme d'apprentissage dépendant directement des objectifs du test, c'est-à-dire du gain visé : l'identification des groupes sera faite en fonction des gains obtenus par les items par le passé. Plus précisément, nous mettons en œuvre une méthode d'inférence conditionnelle via la méthode CTREE. Cet apprentissage permet à la fois de proposer une modèle de classement qui sera utilisé lors de la phase d'exploration mais aussi de mettre en évidence les caractéristiques les plus corrélées à la récompense autorisant une meilleure interprétabilité des groupes. Concrètement, l'algorithme d'arbre d'inférence conditionnelle CTREE (cf.section 2.5.3) est appliqué sur $\mathcal{L}_n$ (où n défini le nombre d'items à disposition), pour identifier des groupes homogènes.

Concrètement, les groupes sont décrits par une récompense moyenne espérée conditionnelle à une ou plusieurs modalités de caractéristiques. A partir de ces informations, CTREE produit une fonction de classement (voir l'algorithme 3). Elle permettra de définir pour chaque nouvel item arrivant pendant la phase d'exploitation le groupe auquel il appartient et donc la récompense moyenne qu'il aurait obtenue sur A. Cette fonction f peut être considérée comme une fonction de régression non linéaire. Il est en effet supposé que la moyenne conditionnelle de la récompense X est liée par une fonction linéaire du vecteur de caractéristique Y via une fonction de lien h où $h\{E(X|Y, \mathbb{I}\{f(Y) = k\})\} = \alpha_k + \beta_k^\top Y$ (à noter que $\beta_k^\top$ signifie ici la transposé de β_k) , avec $\mathbb{I}\{f(Y) = k\}$, la fonction

1. Dans le cas ou la segmentation ne permet pas d'aboutir à une distribution gaussienne, le groupe sera considéré comme homogène

indicatrice associée au groupe, qui renvoie 1 si l'item appartient au groupe k et zéro autrement. À noter qu'ici α_k correspond ici au résidu [2] et non au bonus d'exploitation du modèle de bandit UCB. Les sections 3.2.1 et 3.2.2 donnent plus de détails sur les propriétés théoriques de cette méthode de régression.

3.1.2 Étape d'exploration

La seconde étape correspond à la phase d'exploration d'un A/B test. Dans cette phase, autant de bandits non contextuels sont définis qu'il y a de groupes. Lorsqu'un nouvel item doit être testé, il est préalablement classé dans un groupe via la fonction de classement f (ici produite par CTREE). Le bandit associé au groupe peut alors lui assigner la variation optimale en fonction de la stratégie choisie (UCB, THOMPSON SAMPLING, ...) en amont par l'analyste. Chacun de ces bandits est chargé d'identifier, le plus rapidement possible, la meilleure variation pour ce groupe. Leurs choix dépendront uniquement des récompenses observées dans leur groupe. Les bandits étant indépendants, chacun d'entre eux peut arrêter plus ou moins rapidement l'exploration selon la différence des moyennes réelles des variations pour leur groupe. Le test se termine lorsque tous les bandits exploitent systématiquement la meilleure variation, ou bien lorsque l'analyste le décide. L'algorithme 4 détaille le fonctionnement de CTREE-UCB.

Si l'étape 1 d'explicitation peut être relativement coûteuse en temps de calcul pour l'apprentissage, elle est effectuée hors-ligne en préalable du test lui même. Le mécanisme de classement d'un nouvel item est par contre quasi instantané une fois le modèle connu. De fait, parce que la complexité algorithmique d'un bandit non contextuel étant relativement faible (voir section 2.2.6), le temps d'assignation d'un item à une variation sera négligeable lors de la phase d'exploration. La contrainte sur le temps de réponse sera plus aisément satisfaite. Notre méthode répondra donc plus sûrement à la problématique de temps de calcul.

La section suivante propose une analyse théorique de CTREE-UCB afin de montrer la convergence du regret cumulé

3.2 Etude théorique de Ctree-Ucb

Nous détaillons dans la section 3.2.1 les tests statistiques réalisées par CTREE. Dans la section 3.2.2 nous présentons comment la variance de la distribution d'un groupe est réduite par rapport à la population globale. Enfin, dans la section 3.2.3 nous proposons une borne théorique pour l'algorithme CTREE-UCB.

2. résidu peut se voir comme la différence attendue entre la valeur observée et la valeur prédite par le modèle

Algorithm 4 Algorithme CTREE-UCB

Require: $\alpha > 0$, *Learn dataset*, $\epsilon \in [0, 1]$,

Génère avec *Learn dataset* un modèle d'arbre d'inférence conditionnel : f avec une erreur ϵ acceptée via l'algorithme CTREE.
loop
  $c_t \leftarrow$ un nouvel item avec un vecteur Y_t de caractéristique
  Affect c_t à un groupe k tel que $f(c_t) = k$
  if $T_{a,k}(t) = 0$ **then**
    $A_{t,k} = a$
  else
    $A_{t,k} = \text{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,k,t} + \alpha \sqrt{\frac{2 * log(\sum_{a \in \mathcal{A}} T_{a,k}(t))}{T_{a,k}(t)}}\}$
Assigne le bras $A_{t,y}$ à c_t
Reçoit $X_{c_t,A_t,k} \leftarrow$ la récompense du bras $A_{t,k}$
Met à jour $\hat{\mu}_{A_t,k}$ et $T_{A_t,k}(t)$

Output : Une séquence de choix et de récompenses pour chaque groupe k

3.2.1 Test statistiques

Cette section détaille le calcul de l'espérance et de la variance de gain pour chaque groupe. Pour chaque partition, les observations qui ne sont pas dans la partition à tester, obtiendront le poids $w = 0$ et sinon $w = 1$. La fonction g_j est une transformation de la j-eme caractéristique Y_j. La fonction h est la fonction d'influence, c'est-à-dire une transformation de la récompense X. Les fonctions h et g_j restent les même pour chaque partition, et sont considérées comme une étape de pré-traitement classiques aux algorithmes de *machine learning*. Il suffit donc de les choisir une fois avant le premier fractionnement (nous considérons, pour simplifier la lecture, qu'aucune transformation n'est réalisée). Pour justifier la réduction de la variance justifiée lors de l'étape suivante, nous définissions la statistique de test $\mathbf{T}_j$ pour une partition $S(\mathcal{L}, w)$ considérée (c'est-à-dire les cas où $w_i = 1$) :

Théorème 3.2.1 (Horthon et al, (2006)).

$$\boldsymbol{T}_j(\mathcal{L}_n, w) = vec\Big(\sum_{i=1}^{n} w_i g_i(Y_{ij} h(X_i, (X_1, ..., X_n))^\top\Big) \in \mathbb{R}$$

Le calcul de l'espérance conditionnelle $\mu_j \in \mathbb{R}$ et de la covariance $\Sigma_j \in \mathbb{R}$ de $\mathbf{T}_j$ sous l'hypothèse H_0 pour toutes les permutations possibles $\sigma \in S(\mathcal{L}_n, w)$ de la réponse est donnée par Strasser *et al.* [68] :

$$
\begin{aligned}
n_{\text{node}} &= \sum_{i=1}^{n} w_i \\
\mu_j &= \mathbb{E}(\mathbf{T}_j(\mathcal{L}_n, w)|S(\mathcal{L}_n, w)) = \text{vec}\Big(\Big(\sum_{i=1}^{n} w_i g_i(Y_{ji})\Big)\mathbb{E}[h|S(\mathcal{L}_n, w)]^\top\Big) \\
\Sigma_j &= \mathbb{V}(\mathbf{T}_j(\mathcal{L}_n, w)|S(\mathcal{L}_n, w)) \\
&= \frac{n_{\text{node}}}{n_{\text{node}} - 1}\mathbb{V}[h|S(\mathcal{L}_n, w)] \otimes \Big(\sum_{i=1}^{n} w_i g_i(Y_{ji}) \otimes w_i g_i(Y_{ji})^\top\Big) \\
&- \frac{n_{\text{node}}}{n_{\text{node}} - 1}\mathbb{V}[h|S(\mathcal{L}_n, w)] \otimes \Big(\sum_{i=1}^{n} w_i g_i(Y_{ji})\Big) \otimes \Big(w_i g_i(Y_{ji})^\top\Big)
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}(h|S(\mathcal{L}_n, w)) &= n_{\text{node}}^{-1} \sum_{i=1}^{n} w_i h(X_i, X_1 \dots X_n)) \in \mathbb{R}^q \\
\mathbb{V}(h|S(\mathcal{L}_n, w)) &= n_{\text{node}}^{-1} \sum_{i=1}^{n} w_i (h(X_i, (X_1 \dots X_n)) - \mathbb{E}(h|S(\mathcal{L}_n, w)))(h(X_i, (X_1 \dots X_n)) - \\
&\quad \mathbb{E}(h|S(\mathcal{L}_n, w)))^\top
\end{aligned}
$$

3.2.2 Réduction de la variance dans l'étape 1

Cette partie propose une démonstration pour établir un modèle de régression, où la réponse X et les caractéristiques Y_j (où $j = 1, ..., d$) appartiennent au domaine des réels. Soit n la taille totale des observations. Par souci de simplicité, il est supposé que la fonction d'influence h et la fonction de transformation g_j est la fonction d'identité, signifiant qu'aucune variable ne sera transformée. Ainsi $h = X_i$ et $g_j = Y_j$. La formulation $\sum_{i \in \text{node}}$ est la somme sur toutes les observations de la partition. Elle est identique à la somme de toutes les observations avec pondération, car seule l'observation dans le nœud courant a un poids $w = 1$ et les autres ont un poids $w = 0$. La statistique de test standardisée est définie par :

$$
\mathbf{T}_j(\mathcal{L}_n, w) = \sum_{i=1}^{n} w_i X_{j,i} Y_i = \sum_{i \in \text{node}} X_{j,i} Y_i.
$$

$n_{\text{node}} := \sum_{i=1}^{n} w_i$ est le nombre d'observations dans un nœud. $\hat{X}_{\text{node}}$ est la moyenne estimée du nœud. $\hat{Y}_{j,\text{node}}$ est la moyenne estimée des Y_j dans un nœud.

$(\Sigma)_{ll}$ est la l-ieme diagonale de la matrice de covariance. La statistique de test univariée s associe une statistique linéaire multivariée observée $\mathbf{t} \in \mathbb{R}^d$ dans $\mathbb{R}$ en passant par le

maximum des valeurs absolues de toutes statistiques linéaires normalisées :

$$s_{\max(\mathbf{t},\mu,\Sigma)} = \max_{l=1,\dots,d} \left| \frac{(\mathbf{t}-\mu)_l}{\sqrt{(\Sigma)_{ll}}} \right| = \left| \frac{\mathbf{t}-\mu}{\sqrt{\Sigma}} \right| .$$

$$\begin{aligned}
\mu_j &= \Big(\sum_{i=1}^{n} w_i Y_{ji}\Big)\mathbb{E}_{node}[h] = n_{node}\hat{Y}_{j,node}\hat{X}_{node} \\
\Sigma &= \frac{n_{node}}{n_{node}-1}\mathbb{V}_{node}(h)\sum_{i=1}^{n} w_i Y_{ji}^2 - \frac{1}{n_{node}-1}\mathbb{V}_{node}(h) n_{node}^2 \hat{Y}_{j,node}^2 \\
&= \frac{1}{n_{node}-1}\sum_{i\in node}(X_i - \hat{X}_{node})^2 \sum_{i=1}^{n} Y_{ji}^2 \\
&- \frac{1}{n_{node}-1}\frac{1}{n_{node}}\sum_{i\in node}(X_i - \hat{X}_{node})^2 n_{\text{node}}^2 \hat{Y}_{j,\text{node}}^2 \\
&= \frac{1}{n_{node}-1}\sum_{i\in node}(X_i - \hat{X}_{node})^2)(\sum_{i\in node} Y_{j,i}^2 - n_{\text{node}}\hat{Y}_{j,node}^2) \\
&= \frac{1}{n_{node}-1}(\sum_{i\in node}(X_i - \hat{X}_{node})^2)(\sum_{i\in node}(Y_{j,i} - \hat{Y}_{j,i})^2)).
\end{aligned}$$

Par conséquent :

$$s \propto \left| \frac{\sum_{i\in node} Y_{ji}X_i - n_{node}\hat{Y}_{j,node}\hat{X}_{node}}{\sqrt{(\sum_{i\in node}(X_{i,j} - \hat{X}_{node})^2)(\sum_{i\in node}(Y_{i,j} - \hat{Y}_{j,node})^2)}} . \right| \tag{3.1}$$

La statistique du test linéaire est proportionnelle au cœfficient de corrélation linéaire entre X et Y. L'étape suivante consiste à calculer les statistiques du test (cœfficient de corrélation multiplié par une constante). Ceci sera fait avec le test de permutation. Les tests de permutation sont des approches robustes, basées sur l'aléatoire et le ré-échantilonnage, qui permettent de réaliser des tests statistiques paramétriques, sans que la validité des résultats ne reposent sur des distributions théoriques. Pour se faire, après avoir calculé le cœfficient dans l'Eq 3.1, les récompenses sont mélangé aléatoirement sans changer les caractéristiques. Si le cœfficient avant permutations est extrême comparé aux cœfficients des permutations randomisées, la p-valeur sera très faible.
Le critère de division est la valeur où une partition E ou $\overline{E}$ choisie (inférieure ou égale pour les caractéristiques numériques, un ensemble pour les caractéristiques catégorielles) avec une caractéristique sélectionnée j^*. Cela conduit à deux ensembles différents de n_E (resp : $n_{\overline{E}}$) items pour la partition E (resp : $\overline{E}$). Les statistiques utilisées pour trouver la meilleure répartition sont les suivantes :

$$\mathbf{T}_{j*}^{E}(\mathcal{L}_n, w) = \sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\} X_i = \sum_{i:Y_{j^*,i}\in E} X_i = n_E X_i.$$

$$\mu_{j^*}^E = \sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\} \frac{1}{n_{node}} \sum_{i=1}^{n} w_i X_i = n_E \hat{X}_{node}. \tag{3.2}$$

$$\begin{aligned} \Sigma_{j^*}^E &= \frac{n_{node}}{n_{node}-1} \frac{1}{n_{node}} \sum_{i=1}^{n} w_i (X_i - \hat{X}_{node})^2 \sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\}^2 \\ &- \frac{n_{node}}{n_{node}-1} \frac{1}{n_{node}} \sum_{i=1}^{n} w_i (X_i - \hat{X}_{node})^2 (\sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\})^2 \\ &= \frac{1}{n_{node}-1} \sum_{i \in node} (X_i - \hat{X}_{node})^2 n_E (1 - \frac{n_E}{n_{node}}). \end{aligned} \tag{3.3}$$

Notons que $(1 - \frac{n_E}{n_{node}})$ est en fait la probabilité d'être affectée à E. Soit Z une valeur randomisée selon une distribution binomiale telle que $Z \sim \mathcal{B}(n_{node}, \frac{n_E}{n_{node}})$.

L'équation 3.3 peut être réécrite comme :

$$\begin{aligned} \Sigma_{j^*}^E &= \frac{1}{n_{node}-1} \sum_{i \in node} (X_i - \hat{X}_{node})^2 n_{node} \frac{n_E}{n_{node}} (1 - \frac{n_E}{n_{node}}) \\ &= \sigma_{X_{node}}^2 \sigma_Z^2. \end{aligned} \tag{3.4}$$

Ainsi la statistique de test standardisée est :

$$s_{\max}(\mathbf{T}_{j^*}^E, \mu_{j^*}^E, \Sigma_{j^*}^E) = \max_l | \frac{(\mathbf{t}^E - \mu)_l}{\sqrt{(\Sigma)_{ll}}} | = n_E | \frac{\hat{X}_E - \hat{X}_{node}}{\sqrt{\sigma_{X_{node}}^2 \sigma_Z^2}} | .$$

3.2.3 Borne théorique

Il est supposé que la fonction de prédiction assigne des items à un groupe k où sa récompense suit une distribution σ_k-gaussienne quelque soit le bras appliqué [3]. Définissons une récompense à l'instant t :$X_t = X_{A_t,k}$ et $T_{A_t,k}(t)$ le nombre de fois où le bras A_t du groupe k a été joué. Par définition, le regret cumulé après n essais est égale à :

$$R_{n,k} = \sum_{a \in \mathcal{A}} \Delta_{a,k} \mathbb{E}[T_{a,k}(n)]. \tag{3.5}$$

.

Le regret augmente de $\Delta_{a,k}$ lorsque :

(a) U.C.B$_{a \neq a^*}(k,t) > \mu_{a^*,k}$

(b) U.C.B$_{a=a^*}(k,t) < \mu_{a^*,k}$

3. Les expériences décrites dans la section d'expérimentation indiquent que cette hypothèse peut être confirmée dans la pratique.

Définissons $G_{a,k}$ l'événement (a) ou (b) par une constante $u_{a,y}$:

$$G_{a,k} = \{\mu_{a^*,k} < \min_{t\in[n]} \text{U.C.B}_{a=a^*}(k,t)\} \cap \{\hat{\mu}_{a,k,u_i} + \sigma_k\sqrt{\frac{1}{u_a}\log\frac{1}{\delta}} < \mu_{a^*,k}\}.$$

lorsque l'évènement $G_{a,k}$ apparaît, le regret convergent vers une valeur finie et si $G_{a,k}$ apparaît, alors $T_{a,k}(n) \leqslant u_{i,k}$. Soit $G_{a,k}^C$ l'évènement complémentaire.

$$\begin{aligned}\mathbb{E}[T_{a,k}(n)] &= \mathbb{E}[\mathbb{I}\{G_{a,k}\}T_{a,k}(n)] + \mathbb{E}[\mathbb{I}\{G_{a,k}^C\}T_{a,k}(n)] \\ &\leqslant u_{a,k} + \mathbb{P}[G_{a,k}^C]n.\end{aligned}$$

L'évènement complémentaire : $G_{a,k}^C$ est décomposé en deux parties :

$$G_{a,k}^C = \{\mu_{a^*,k} \geqslant \min_{t\in[n]} \text{U.C.B}_{a=a^*}(k,t)\} \cup \{\hat{\mu}_{a,k,u_a} + \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \mu_{a^*,k}\}.$$

$$\begin{aligned}\{\mu_{a^*,k} \geqslant \min_{t\in[n]} \text{U.C.B}(k,t)\} &\subset \cup_{s\in[n]}\{\mu_{a^*,k} \geqslant \hat{\mu}_{a^*,k,s} + \sigma_k\sqrt{\frac{2}{s}\log\frac{1}{\delta}}\} \\ \mathbb{P}[\mu_{a^*,k} \geqslant \min_{t\in[n]} \text{U.C.B}_{a=a^*}(k,t)] &\leqslant \sum_{s=1}^{n}\mathbb{P}[\mu_{a^*,k} \geqslant \hat{\mu}_{a^*,k,s} + \sigma_k\sqrt{\frac{2}{s}\log\frac{1}{\delta}}] \\ &\leqslant n\delta.\end{aligned}$$

Et donc $\mathbb{P}[G_{a,k}^C] \leqslant n\delta + \mathbb{P}[\hat{\mu}_{a,k,u_a} + \sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \mu_{a^*,k}]$. Rappelons que $\Delta_{a,k} = \mu_{a^*,k} - \mu_{a,k}$. Supposons une constante positive c où $\Delta_{a,k} - \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant c\Delta_{a,k}$:

$$\begin{aligned}\mathbb{P}[\hat{\mu}_{a,k,u_a} + \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \mu_{a^*,k}] &= \mathbb{P}[\hat{\mu}_{a,k,u_a} + \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \Delta_{a,k} + \mu_{a,k}] \\ &= \mathbb{P}[\hat{\mu}_{a,k,u_a} - \mu_{a,k} \geqslant \Delta_{a,k} - \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}}] \\ &\leqslant \mathbb{P}[\hat{\mu}_{a,k,u_a} - \mu_{a,k} \geqslant c\Delta_{a,k}] \\ &\leqslant \exp\left(-\frac{u_a c^2\Delta_{a,k}^2}{2\sigma_k^2}\right).\end{aligned} \tag{3.6}$$

La dernière ligne de l'équation 3.6 fait référence à une erreur bornée pour une variable aléatoire suivant une distribution σ_k-gaussienne. Ainsi $\mathbb{P}[G_a^C] \leqslant n\delta + \exp\left(-\frac{u_a c^2\Delta_{a,k}^2}{2\sigma_k^2}\right)$. Nous cherchons u_a satisfaisant :

$$\Delta_{a,k} - \sigma_y \sqrt{\frac{2}{u_a} \log \frac{1}{\delta}} \geqslant c\Delta_{a,k}$$
$$u_a \geqslant \frac{2\sigma_k^2 \log \frac{1}{\delta}}{\Delta_{a,k}^2 (1-c)^2}.$$

Si $\delta = \frac{1}{n^2}$

$$\mathbb{E}[T_{a,k}(n)] \leqslant \frac{2\sigma_k^2 \log \frac{1}{\delta}}{\Delta_{a,k}^2 (1-c)^2} + 1 + n \exp\left(-\frac{u_a c^2 \Delta_{a,k}^2}{2\sigma_k^2}\right) \tag{3.7}$$
$$\leqslant \frac{2\sigma_k^2 \log \frac{1}{\delta}}{\Delta_{a,k}^2 (1-c)^2} + 1 + n^{-1+\frac{c^2}{(1+c)^2}}.$$

La fonction $-1 + \frac{c^2}{(1+c)^2}$ est strictement négative pour n'importe quel $c < \frac{1}{2}$. L'équation 3.7 peut être ré-écrite :

$$\mathbb{E}[T_{a,k}(n)] \leqslant \frac{2\sigma_k^2 \log n^2}{\Delta_{a,k\,\frac{1}{2}}^2} + 2.$$

Pour tous groupe k le regret est borné par :

$$R_{n,k} = \sum_{a \in \mathcal{A}} \Delta_{a,k} \mathbb{E}[T_{a,k}(n)]$$
$$= \sum_{a \in \mathcal{A}} \frac{16\sigma_k^2 \log n^2}{\Delta_{a,k}} + 2 \sum_{a \in \mathcal{A}} \Delta_{a,k}.$$

Ceci définit finalement le regret cumulé pour tout groupe k basé sur les données d'apprentissage de $a = A$ (la variation originale à améliorer). Si l'arbre d'inférence conditionnelle définit un ensemble de $\mathcal{K}$ groupes admissibles où $\forall k \in \mathcal{K}$, $\mathcal{N}(\nu_{k,a=A}, \sigma_{k,a=A})$ est statistiquement différent (avec un risque accepté de ϵ) et $|\mathcal{A}| = K$, le regret cumulatif pendant le test A/B est

$$R_n^{\text{CTREE-UCB}} = \sum_{k \in \mathcal{K}} R_{n,k} = \sum_{k \in \mathcal{K}} \left(\sum_{a \in \mathcal{A}} \frac{16\sigma_k^2 \log n^2}{\Delta_{a,k}} + 2 \sum_{a \in \mathcal{A}} \Delta_{a,k} \right)$$
$$\leqslant K \max_{k \in \mathcal{K}} [\sigma_k^2] \left(\sum_{a \in \mathcal{A}} \frac{16 \log n^2}{\Delta_{a,k}} + 2 \sum_{a \in \mathcal{A}} \Delta_{a,y} \right).$$

3.3 Ctree-Ucb : expérimentations

Dans cette section nous montrons empiriquement l'intérêt de notre approche en comparant ses résultats en termes de regret cumulé à ceux obtenus par les méthodes d'allocation dynamique de l'état de l'art. Dans la phase d'explicitation des groupes, l'arbre de régression produit par CTREE met en évidence les récompenses espérée pour chaque groupe. Chaque feuille finale forme donc un groupe caractérisé par une moyenne espérée. Nous ne présentons ici que deux cas :

— Le cas de fonction de récompense continue par partie sur des données simulées
— Le cas de fonctions de récompense inconnues sur des données réelles issues de bases de données publiques ou produites par AB Tasty

Des expériences sur des cas plus simples (linéarité, distribution gaussienne, ...) pour lesquels toutes les méthodes (y compris Ctree-Ucb) sont performantes, ont aussi été menées mais ne seront pas présentées ici. Le lecteur intéressé trouvera les données utilisées et résultats obtenus sur notre site : `https://github.com/R-workshop-strasbourg/bandit4abtest`

3.3.1 Données simulées

Dans de nombreux domaines, l'analyste est confronté à des situations où le gain n'est pas linéairement lié aux caractéristiques des items. Par exemple, si le site offre un article pour 3 articles achetés, l'hypothèse de linéarité entre la caractéristique (quantité d'articles) et la récompense ne tient plus. Dans cet exemple, il existe un palier entre 3 et 4 articles dans la courbe des gains (on peut aussi considérer qu'il n'y a jamais achat de 3 articles). Le coût d'un service en ligne peut être fixe tant que le visiteur respecte son abonnement, mais il sera facturé dès qu'il le dépasse. Dans le domaine médical, si un traitement est efficace pour les jeunes enfants et les personnes âgées, mais déficient pour les adultes, le lien entre une amélioration de la santé et l'âge du patient, bien que continu, n'est plus linéaire. Ces cas, fréquemment rencontrés, ne sont malheureusement pas traitables par les méthodes classiques introduites ci-dessus, excepté par Kernel-Ucb mais qui nécessite alors une grande quantité de données avant de converger. En effet, les méthodes de bandits traditionnelles n'arrivent pas à identifier les variations optimales car elles ne peuvent présenter qu'un regret cumulatif linéaire. De faite leurs hypothèses ne sont plus vérifiées et ces stratégies sont alors incapables de garantir une convergence logarithmique du regret [16].

Nous proposons donc dans cette section de montrer, à travers des données simulées, que notre méthode est apte à le faire. Nous nous plaçons dans la cas où la fonction de liaison entre les récompenses et la caractéristique peut être modélisée par une fonction $f_s(x)$ continue par partie. Pour cela nous générons des données via une fonction continue par partie prenant en entrée 10000 valeurs d'une caractéristique x (générées aléatoire-

ment) associés à 20000 récompenses (10000 pour chaque variation). Cette simulation doit respecter les hypothèses faites précédemment (c'est-à-dire que la distribution de gain de A est corrélée à celle de B, et que la fonction de lien entre la caractéristique et la récompense est non linéaire). Nous générons ces récompense selon les moyennes de gains suivantes (le lecteur peut modifier ces moyennes dans le code [4]) :

$$\theta_A = (2, -1, 1.5, 0)$$
$$\theta_B = (1.5, -0.5, 1.25, 0)$$

où θ_A et θ_B sont les paramètres de la fonction $f_s(x)$ génératrice de gain continue par morceaux décrite ci-dessous [5].

$$X = \begin{Bmatrix} \text{if } 1 \leqslant x_1 < 2 & X_A = \theta_A[1] & X_B = \theta_B[1] \\ \text{if } 3 \leqslant x_1 < 4 & X_A = \theta_A[2] & X_B = \theta_B[2] \\ \text{if } x_1 \geqslant 4 & X_A = \theta_A[3] & X_B = \theta_B[3] \\ \text{if } x_1 < 1 & X_A = \theta_A[4] & X_B = \theta_B[4] \end{Bmatrix}$$

avec x_1 une caractéristique numérique générée aléatoirement.

3.3.2 Données réelles

Nous nous plaçons ici dans le cas où les données sont issues de jeux de données réelles et où aucune hypothèse n'est faîte a priori sur les distributions de gain.

Données réelles : Small MovieLens Les données utilisées ici permettent de tester notre méthode dans le cas où CTREE-UCB serait utilisé pour de la recommandation. Ces données proviennent du jeu de données public MovieLens [38] qui contient 9125 films décrits par 14 caractéristiques binaires (Aventures, Action, Comédie, Drame, Thriller, Romance, Science-Fiction, ...) et des notes associées (de 0 à 500) données par différents évaluateurs [6]. Pour simuler un A/B Test, nous associons :

- les films aux items
- les 5 évaluateurs de films aux 5 variations (notées A, B, ...E)
- les notes obtenues aux récompenses : la récompense, notée X_{A_t,c_t} avec $A_t \in a$ d'un film c_t pour la variation $a \in \{A, B, C, D, E\}$ sera la note donnée par l'évaluateur associé. Dans le cas où le film n'a pas été noté par l'évaluateur (ce qui peut être le

4. `https://github.com/R-workshop-strasbourg/bandit4abtest`

5. Ces valeurs ont été choisies arbitrairement et peuvent être modifiées dans le code.

6. Les notes étant généralement très proches, nous avons souhaité dans cette expérimentation augmenter artificiellement les différences entre les moyennes des différentes variation en multipliant les notes par 100. Ainsi nous limitons le nombre d'items testés (et donc la durée de l'AB Test). Ainsi, le lecteur pourra reproduire cette expérimentation facilement sur sa machine. Cette augmentation n'influence cependant pas l'arbre de segmentation de l'étape 1.

cas pour des films récents), la valeur manquante est estimée par la moyenne globale de ce film à partir des notes données par tous les évaluateurs du site MovieLens.

L'objectif est d'assigner chaque film à un évaluateur qui maximisera son évaluation. Dans ce jeux de données, un film (item), peut-être vu par plusieurs évaluateurs (variations). C'est donc une différence notable avec un réel AB Test où un item (visiteur, patient, ...) ne voit en pratique qu'une et une seule variation. En revanche, ce jeux de données nous permet de tester la pertinence de notre algorithme sur un autre cas d'usage (ici la recommandation).

Données A/B Tasty Les jeux de données considérés ici proviennent de six A/B tests différents réalisés en 2018 par la société A/B Tasty. S'ils sont totalement indépendants, ils ont tous suivis le même protocole et ont un objectif similaire. Ainsi, tous ces tests étaient basés sur une méthode d'allocation statique (en proportions égales) sur un mois. Chacun d'entre eux comparait deux pages web (référées ici par P_1 et P_2) ayant un visuel différent pour un même contenu. Pour chaque test, les visiteurs testés ont été identifiés par un identifiant visiteur : lors de sa première visite sur la page test, une description du visiteur (voir Tab. 3.1) [7] et son identifiant associé sont générés [8]. Cette description peut varier en fonction des données fournies par le site, et dépend de ce que l'analyste peut collecter (par exemple le navigateur est toujours présent, mais la localisation peut ne pas avoir été collectée). Ensuite, une variation (A ou B) lui était attribuée aléatoirement. Un cookie était paramétré pour mémoriser les caractéristiques [9], l'identifiant du visiteur et la version attribuée. Chaque fois qu'un visiteur revenait sur la page testée, la même variation était affichée. Enfin, toutes les actions (clics, achat, somme dépensée) des visiteurs lors de leurs visites ont été stockées.
En fonction de l'objectif de l'analyste (augmenter la valeur d'achat ou le taux de clics), la récompense d'un visiteur t correspond à la somme cumulée de ses éventuels achats, ou un clic/non clic, après avoir vu la page web test.

A la fin du test (*i.e.*, après T visiteurs observés pendant un mois) ces récompenses sont utilisées pour évaluer la meilleure variation entre P_1 et P_2.

3.3.3 Protocole d'expérimentation

La performance de CTREE-UCB a été comparée avec quatre stratégies d'allocation dynamique :

- Deux stratégies non contextuelles :

7. Si le système d'exploitation (resp. le navigateur) du visiteur n'est pas identifié, la caractéristique sera `ERREUR` (resp.`ERREUR_OS`)

8. Il n'y a aucun lien entre les informations liées visiteurs des différents tests même si dans certains cas, il peut s'agir de la même personne physique

9. Des caractéristiques supplémentaire peuvent être ajoutées via le site ou le user agent.

Table 3.1 – Caractéristiques des items (jeux de donnée A/B Tasty)

Type	Caractéristiques covariates (domaine de définition)
Continue	Visites ($\mathbb{N}$)
Catégorielle	Langage du navigateur (27), Type de navigateur : UserAgent (6), Appareil : Device (3), Système d'exploitation : OS (7)

  - uniform qui choisit les variations de façon alternative. Cette approche revient à faire une allocation statique et ne nécessite aucun paramètre.
  - Ucb décrit en Section 6.
- Deux stratégies par bandits contextuels :
  - Lin-Ucb qui choisit selon des hypothèses de linéarité entre les caractéristiques et la récompense, décrit en section 2.4.1.
  - Kernel-Ucb qui s'affranchit de l'hypothèse de linéaire, et modélise la relation entre les caractéristiques et la récompense par une régression non linéaire à noyau, décrit en section 2.4.1.

Chaque stratégie produit une séquence de choix. Ces séquences sont comparées avec celle d'un oracle qui choisirait systématiquement la variation qui maximise l'espérance conditionnelle (voir Section 2.4). Cet oracle est entraîné sur toutes les données disponibles. Le regret cumulée et moyen (différence par rapport à cette stratégie oracle pour chaque choix) est évalué pendant et à la fin de l'A/B test.

Pour montrer la performance de Ctree-Ucb, cette méthode est testée sur tous les jeux de données.

Tous les algorithmes sont implémentés en utilisant le langage *open-source* `R` sur Ubuntu 17.10, 64 bits. Ils fonctionnent sur un processeur Intel® Core™ i5-8250U CPU 8-processeurs fonctionnant à 1,60 GHz avec 7,5 Go de RAM. Tout le matériel (y compris les données, le framework de régression d'arbre conditionnel `CTREE` [43] et Ctree-Ucb) est disponible sur : `https://github.com/R-workshop-strasbourg/bandit4abtest`

Étape offline d'explicitation des groupes

Pour la méthode Ctree-Ucb, l'étape d'explicitation (voir Section 3.3.3) nécessite de disposer des résultats de l'exploitation d'une variation originale A. De fait, il n'existe pas de telle variation dans le cas des données MovieLens. Afin de minimiser (et évaluer) l'impact du choix d'une des variations comme variation initiale, et donc des groupes explicités, nous avons choisi de réaliser cinq fois l'expérience en sélectionnant à chaque fois une variation différente comme variation originale.

La fonction de régression conditionnelle `CTREE` en `R` [43] utilisée pour expliciter ces groupes intègre un mécanisme de validation croisée paramétré par un risque d'erreur maximum accepté noté ϵ. La sensibilité des résultats à ce paramètre a été étudiée ainsi que l'influence du nombre d'items pour l'apprentissage du modèle de classement.

Étape online d'allocation dynamique

L'évaluation de chaque algorithme sur des ensembles de données a consisté à comparer leurs regrets cumulatifs et moyens. Les résultats ont aussi été comparés avec ceux de l'oracle issus d'un modèle de régression non linéaire [10] qui apprend sur toutes les données. Ainsi, lors de l'affectation d'un item à une variation, le regret est évalué comme la différence entre la meilleure variation (entre toutes les variations possibles) et la variation de l'item choisi. Cette évaluation peut être considérée comme une différence entre les moyennes conditionnelles et repose sur la définition théorique présentée à la section 2.2.3.

Paramètres du test A/B

Tous les algorithmes (sauf UNIFORM) sont dérivés de l'algorithme standard UCB qui nécessite de définir un intervalle de confiance (voir équation 2.10, Section 6). Pour évaluer l'impact du bonus d'exploration α (nécessaire à toute méthode utilisant un intervalle de confiance) sur les résultats, nous avons effectué des expériences avec différentes valeurs de α, comprises entre 0,25 à 2,5, classiquement utilisées dans la littérature. Pour limiter le temps CPU consommé par l'algorithme KERNEL-UCB, plutôt que de réactualiser le modèle après chaque item testé, KERNEL-UCB nous remettons à jour ses estimations en mode *batch* pour chaque bras après 100 items testés.

Notes supplémentaires Pour les K variations $V_0, V_1, \ldots, V_K$, S_i est l'ensemble des items auxquels la variation V_i a été affectée, $T_i = |S_i|$ et $T = T_0 + T_1 + \cdots + T_K$ où $|.|$ désigne la cardinalité de cet ensemble.

$L = |\mathcal{L}|$ où $\mathcal{L}$ est l'ensemble utilisé pour apprendre l'arbre de régression conditionnelle. $T_{A/B} = |S_{A/B}|$ où $S_{A/B}$ est le jeu utilisé pour simuler le test A/B. ϵ est le paramètre de risque d'erreur accepté de `CTREE`.

Pour respecter l'hypothèse qu'avant le test A/B, l'analyste n'exploite que la variation originale (appelée ici V_A), l'arbre de régression ne peut être construit qu'à partir de l'ensemble des éléments S_A auxquels la variation V_A a été attribuée. Ainsi $\mathcal{L} \subset S_A$. Deux configurations ont été testées :

— $\text{Conf}_{30,70}$: $\mathcal{L} = 30\%$ de V_0, $S_{A/B} = \sum_i \{70\% \text{ de } S_i\}$

— $\text{Conf}_{100,100}$: $\mathcal{L} = V_0$, $S_{A/B} = \sum_i S_i$

Résumé des configurations des différentes expériences

La Tab. 3.2 présente les différentes configurations incluant tous les paramètres et leurs différentes valeurs utilisées.

10. Nous utiliserons CTREE pour obtenir ce modèle

TABLE 3.2 – Algorithm parameters

Data set	Algorithms	Parameters
MovieLens	UCB, LIN-UCB, KERNEL-UCB	$\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$
AB Tasty	CTREE-UCB	$V_A \in \{V_i\}$ Config. $\in \{\text{Conf}_{30,70}, \text{Conf}_{100,100}\}$ $\epsilon \in \{0.01, 0.05, 0.1\}$ $\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$

Visualisation des résultats

Lors de l'étape 1 d'identification des groupes, chaque groupe est ici représenté soit par sa boîte à moustaches (qui résume quelques indicateurs comme la médiane, les quartiles, minimum, maximum ou déciles) soit par sa moyenne uniquement. La moyenne du groupe est comprise dans la zone grisée de la boite à moustache. La boite à moustache est une option utilisée ici pour permettre à l'analyste de comparer les groupes cependant, seule la moyenne du groupe est considérée dans l'algorithme CTREE-UCB. Si l'analyste souhaite une représentation simplifiée, seule la moyenne (via un trait noir) sera affichée. Le nombre d'items (pendant la phase d'apprentissage) présents dans chaque groupe est indiqué via la variable n. L'analyste peut ainsi avoir une idée de la proportion d'items pour chaque groupe avant son test. Il peut également avoir un a priori sur le groupe qui maximise les récompenses pour la variation A.

Dans la suite, pour chaque expérience, un tableau résume les regrets cumulé et moyen de chaque algorithme. Le regret moyen permet d'évaluer l'évolution du regret cumulatif par rapport à l'ensemble des items testés (S). Les meilleures performances (regrets cumulés et moyens) seront mis en évidence par une police en gras dans ces tableaux.

3.4 Expérimentations

3.4.1 Données simulées

Nous sélectionnons 15% des données pour les associer aux récompenses obtenues dans l'étape d'explicitation des groupes. Ces données proviennent uniquement des récompenses de la variation A. 15% des données (récompenses associées à B) sont donc exclues de l'expérience. Ces données sont allouées au *Learn dataset* nécessaire à l'apprentissage de CTREE-UCB. 70% des données sont dédiés au test A/B lui même.

Étape offline d'explicitation des groupes

Les moyennes conditionnelle en fonction des valeurs x_1 défini pour simuler les données sont retrouvées par l'arbre de classement. La Fig.3.2 représente l'arbre de classement

appris sur les données d'apprentissage issu de A. 5 groupes sont identifiés : `Node 3`, `Node 4`, `Node 6`, `Node 8` and `Node 9`.

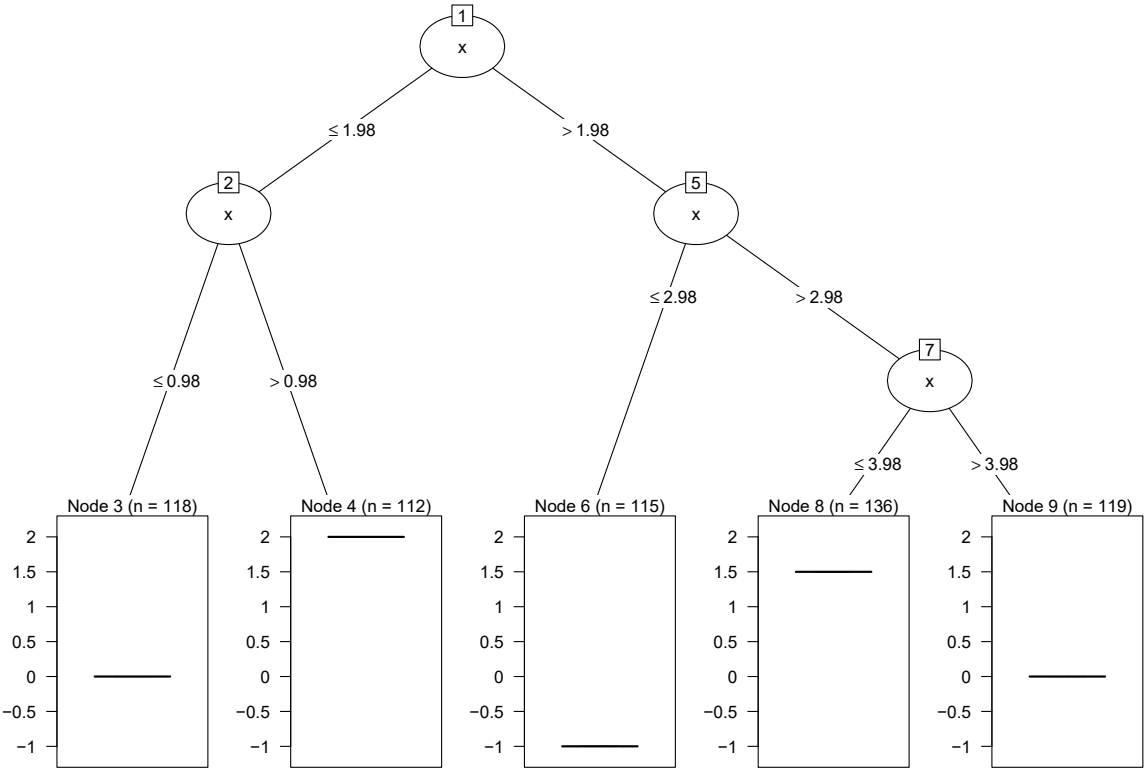


FIGURE 3.2 – Identification des groupes basés sur x dans l'étape de pré-traitement de CTREE-UCB (Config.$_{30,70}$, V_A : A, $\epsilon = 0.05$).

A/B test (Allocation dynamique)

Lors de la phase d'exploration par CTREE-UCB, l'allocation dynamique est réalisée pour chaque item à tester, selon le groupe auquel il appartient. Nous comparons les résultats obtenus par CTREE-UCB, LIN-UCB, UCB et UNIFORM. La Figure 3.3 montre le regret cumulé engendré par chaque algorithme au cours du test. Les Figures 3.4, 3.5, 3.6, 3.7 et 3.8 montrent le regret cumulé spécifique à chaque groupe. Le regret logarithmique de CTREE-UCB (voir Figure 3.3) montre que CTREE-UCB exploite toutes les variations optimales (selon les groupes) avant la fin du test.

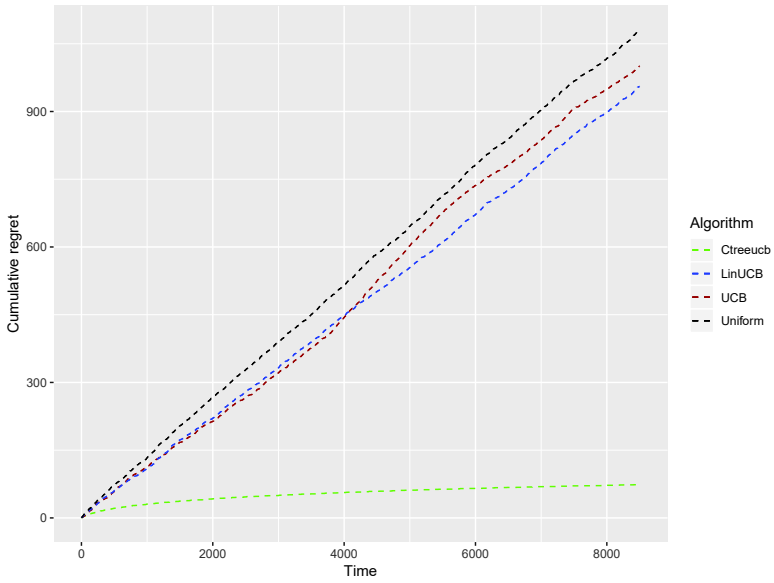


FIGURE 3.3 – Regret cumulé de CTREE-UCB, LIN-UCB, UNIFORM et UCB avec une fonction de récompense non linéaire

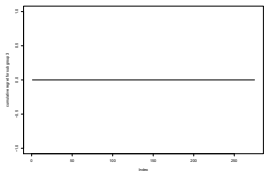


FIGURE 3.4 – Regret cumulé de CTREE-UCB pour le groupe `Node 3`

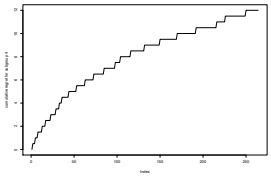


FIGURE 3.5 – Regret cumulé de CTREE-UCB pour le groupe `Node 4`

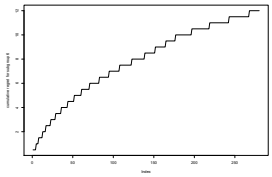


FIGURE 3.6 – Regret cumulé de CTREE-UCB pour le groupe `Node 6`

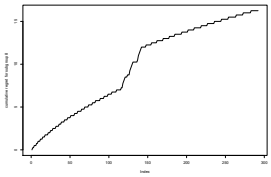


FIGURE 3.7 – Regret cumulé de CTREE-UCB pour le groupe `Node 8`

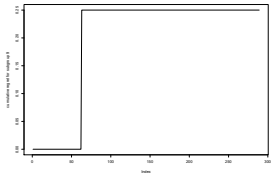


FIGURE 3.8 – Regret cumulé de CTREE-UCB pour le groupe `Node 9`

3.4.2 Movie data set

Étape offline d'explicitation des groupes

Sur la Figure 3.9, chaque feuille de l'arbre associe une moyenne et la variance d'un groupe identifié.

Pour classer un item, une procédure récursive est mise en oeuvre. Pour chaque nœud, l'item est placé dans l'un des deux sous ensemble en fonction d'une caractéristique. Par exemple pour le premier nœud (noté `Node#8` dans le graphique), si le genre du film est "Drama", le film sera placé dans le sous ensemble droit. Les feuille de l'arbre représentent les groupes identifiés par CTREE qui seront utilisés dans le modèle de classement de l'étape d'allocation dynamique.

À partir des récompenses données par l'évaluateur 1 avant le test, 9 feuilles ont été générées ce qui correspond à 9 groupes d'items avec des distributions de gains statistiquement différentes. Le Node#15 est le plus représenté avec 1133 items. Le groupe qui maximise les récompenses pour la variation A est celui du `Node#11`. On remarque aussi, par exemple, que l'évaluateur 1 donne, en général, une note plus importante si le film est de la catégorie "Film noir" (`Node#11`). La plus faible moyenne de gains est donnée pour des films de type "Comedy" (`Node#8`) ou "Action" (`Node#7`).

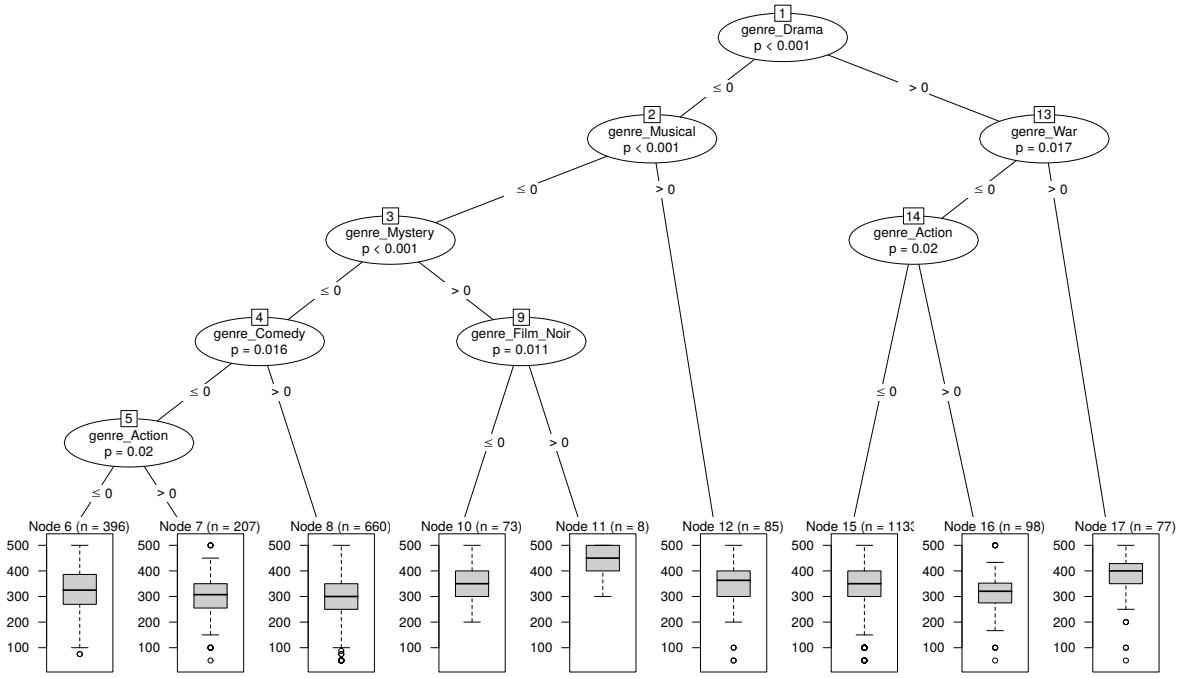


FIGURE 3.9 – Arbre d'inférence conditionnelle pour le jeu de données MovieLens (Config.$_{30,70}$, V_A : Reviewer 1, $\epsilon = 0.05$).

		Ctree-Ucb					Lin-Ucb	Kernel-Ucb	Ucb	uniform
		$V_A : \mathrm{R}_1$	$V_A : \mathrm{R}_2$	$V_A : \mathrm{R}_3$	$V_A : \mathrm{R}_4$	$V_A : \mathrm{R}_5$				
Configuration		$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$				
Config.$_{30,70}$	R_T	**19155**	21628	28458	27894	19976	22378	21239	20650	22808
	$\mathbb{E}[R_T]$	**3**	3.39	4.47	4.37	3.13	3.50	3.32	3.23	3.57
Config.$_{100,100}$	R_T	**27875**	34152	35679	35908	37679	68101	40848	31146	45643
	$\mathbb{E}[R_T]$	**3.05**	3.74	3.91	3.94	4.13	7.46	4.48	4.41	5

Table 3.3 – Influence des paramètres de segmentation sur le regret cumulé R_T et moyen $\mathbb{E}[R_T]$ pour le jeu de donnée MovieLens (avec $\alpha = 1$). La V_A indique le reviewer ayant été considéré comme la variation d'apprentissage (variation A)

A/B test (Allocation dynamique)

Dans le jeu de données MovieLens, de nombreux films n'ont pas été vus par certains évaluateurs et par conséquent, le remplacement des valeurs manquantes engendre des notes identiques entre plusieurs évaluateurs. Il y a donc des films pour lesquels le regret simple sera égal à zéro quelque soit l'évaluateur choisi. La différence entre les moyennes Δ_a, $\forall a \neq a^*$, étant très faible, trouver l'évaluateur qui maximise les récompenses (en accord avec un contexte) est difficile (voir Section 2.2.6). Une telle situation peut diminuer la performance des algorithmes de bandit testés et être comparable à une stratégie d'allocation statique (réalisée par l'algorithme uniform). Les résultats en terme de regret sont peu influencés par la valeur du paramètre ϵ et son impact sur les résultats sera traité dans les prochaines expériences. Donc, pour des raisons de lisibilité, dans la Table 3.3, sera rapportée seulement la valeur classique du risque d'erreur accepté, *i.e.* $\epsilon = 0.05$.

Ctree-Ucb présente le regret cumulé le plus faible (en gras dans la Tab. 3.3). L'algorithme Lin-Ucb présente de faibles performances. Une des explications pourrait être que l'hypothèse de linéarité requise par cet algorithme n'est pas respectée. Kernel-Ucb obtient un regret cumulatif comparable à une allocation statique (uniform). Plus de données sont probablement nécessaires pour terminer sa régression. Ctree-Ucb obtient un meilleur résultat avec l'évaluateur 1 (en gras) avec Config.$_{30,70}$ ou Config.$_{100,100}$. L'apprentissage sur l'évaluateur 3 (sur Config.$_{30,70}$) ou 5 (Config.$_{100,100}$) diminue ses performances. Nous supposons qu'ils impliquent un sur ou sous apprentissage en fonction de la configuration.

Pour obtenir de bons résultats avec Ctree-Ucb, quel que soit le paramètre V_A, l'étape hors ligne doit être effectuée sur une population représentative de celle qui sera testée.

La Fig. 3.10 présente le regret cumulé de chaque algorithme pour une configuration donnée. Le regret le plus faible au cours du test est celui de Ctree-Ucb (en vert). La stratégie Ucb (en marron) arrive en seconde position en terme de performance. Le regret cumulé le plus important est celui de uniform (en noir).

La Figure 3.11 montre comment α affecte le regret cumulatif. Quelque soit la valeur α, Ctree-Ucb produit les meilleurs résultats et garantit leur stabilité. Contrairement aux autres algorithmes, Kernel-Ucb fonctionne différemment selon α. Son regret cumulatif

semble fortement dépendant de ce paramètre. Choisir une valeur sous optimale pour α peut donc le rendre moins efficace que UNIFORM. (Fig. 3.11).

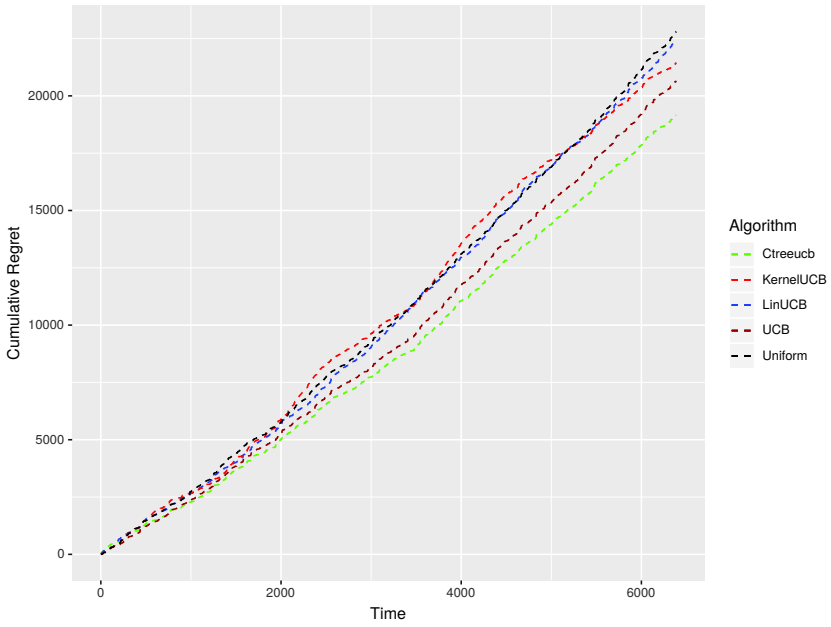


FIGURE 3.10 – Regret cumulé avec le jeu de données MovieLens (V_A : R_1, $Conf_{30,70}$, $\epsilon = 0.05$,$\alpha = 0.25$).

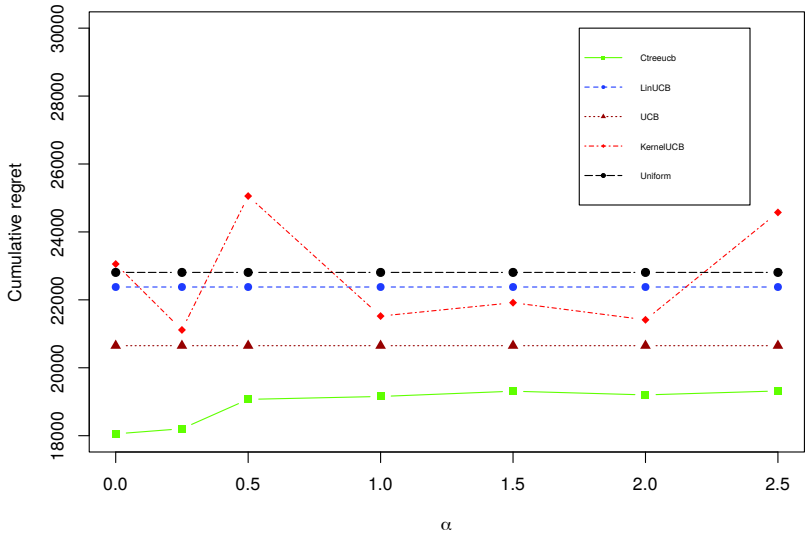


FIGURE 3.11 – Regret cumulé selon α avec le jeu de données MovieLens de CTREE-UCB (avec la configuration V_A : R_1, $Conf_{30,70}$, $\epsilon = 0.05$), LIN-UCB, UCB, KERNEL-UCB, et UNIFORM. Le paramètre α est observé selon 7 valeurs possibles. Le regret cumulé en accord avec ce paramètre est observé pour chaque algorithme.

3.4.3 Jeu de données AB Tasty 1

Étape offline d'explicitation des groupes

Le test concerne un site de vente dans le prêt à-porter et l'objectif est d'augmenter la valeur des achats de ses visiteurs. Dans la configuration Config.$_{30,70}$, 2543 visiteurs sont dédiés à la première étape d'apprentissage et 5934 visiteurs sont testés dans la deuxième étape. Sur la Config.$_{100,100}$ les 8477 visiteurs sont affectés à chacune des deux étapes. Sur la Figure 3.12, chaque feuille associe la valeur de la récompense espérée apprise à partir de la variation originale. Dans cet arbre, 10 groupes sont identifiés avec des distributions de gains statistiquement différentes. Les visiteurs dépensant le plus sont ceux qui ont déjà été sur le site 12 fois maximum et qui accèdent au site via leur Iphone ou via un ordinateur ayant un OS Linux (`Node`#5). La plus faible moyenne de gain est donnée pour des visiteurs ayant été sur le site plus de 96 fois et utilisant un Ipad ou un Iphone (`Node`#15). La caractéristique spécifiant le nombre de visites passées avant de voir la page testée a la plus forte corrélation avec les valeurs d'achats. Cependant, la valeur d'achat peut augmenter ou diminuer selon le user agent ou la langue du visiteur.

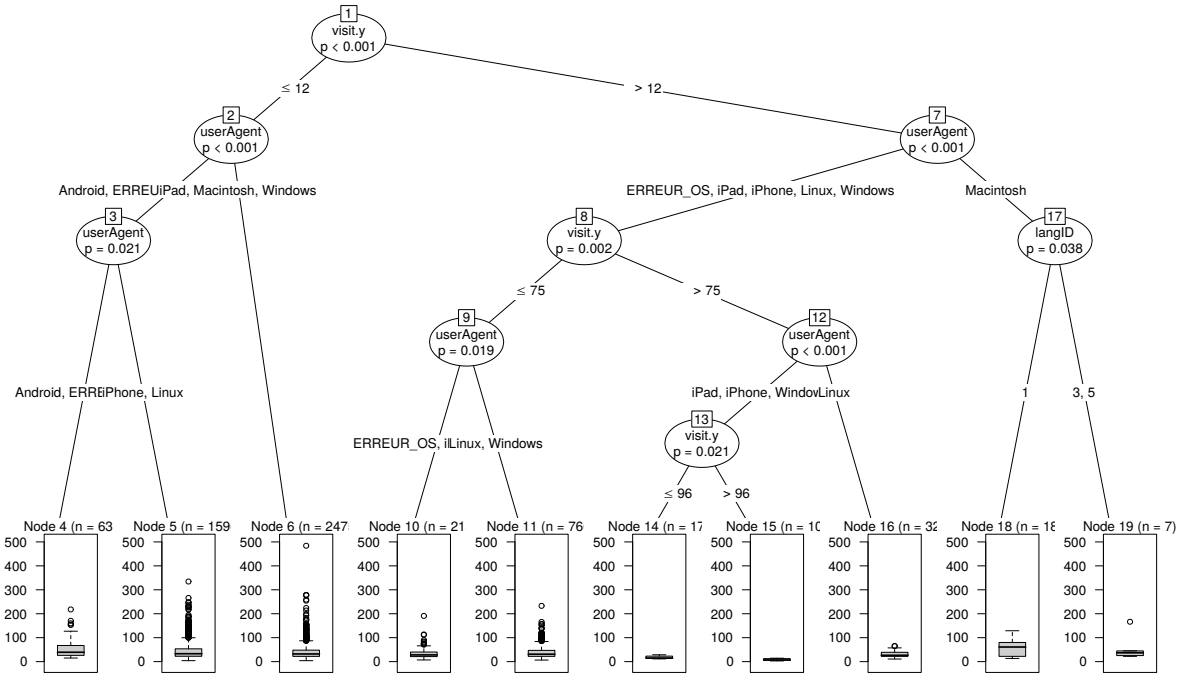


FIGURE 3.12 – Arbre d'inférence conditionnelle pour le jeu de données AB Tasty 1 à partir des récompenses données par la page 1 avant le test (Config.$_{30,70}, V\_A : P\_1, \epsilon = 0.05$).

A/B test (Allocation dynamique)

Avec Config.$_{30,70}$, tous les résultats fournis par CTREE-UCB (en gras) sont les meilleurs. Sur l'ensemble de données du jeu de données d'A/B Tasty 1, la Table 3.4 donnent le

		CTREE-UCB						LIN-UCB	KERNEL-UCB	UCB	UNIFORM
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
$\text{Conf}_{30,70}$	R_T	**100**	**100**	**100**	355	355	355	364	592	408	6610
	$\mathbb{E}[R_T]$	$\mathbf{1.68 * 10^{-2}}$	$\mathbf{1.68 * 10^{-2}}$	$\mathbf{1.68 * 10^{-2}}$	$\mathbf{5.59 * 10^{-2}}$	$\mathbf{5.59 * 10^{-2}}$	$\mathbf{5.59 * 10^{-2}}$	$\mathbf{6.11 * 10^{-2}}$	$\mathbf{9.97 * 10^{-2}}$	$6.87 * 10^{-2}$	1.11
$\text{Conf}_{100,100}$	R_T	152	152	152	67	67	67	24	448	241	8148
	$\mathbb{E}[R_T]$	$\mathbf{1.79 * 10^{-2}}$	$\mathbf{1.79 * 10^{-2}}$	$\mathbf{1.79 * 10^{-2}}$	$\mathbf{0.79 * 10^{-2}}$	$\mathbf{0.79 * 10^{-2}}$	$\mathbf{0.79 * 10^{-2}}$	$\mathbf{0.28 * 10^{-2}}$	$\mathbf{5.28 * 10^{-2}}$	$\mathbf{2.840.79 * 10^{-2}}$	0.96

TABLE 3.4 – Influence des paramètres de segmentation sur le regret cumulé et moyen avec le jeu de données AB Tasty 1 ($\alpha = 1$). La V_A indique la page ayant été considérée comme la variation d'apprentissage (variation A).

regret cumulé selon les différents paramètres (ϵ et V_A). Quelque soit la configuration, le paramètre ϵ ne modifie pas l'arborescence. Il est nécessaire, lors de l'étape 1 d'éviter un sous-apprentissage qui n'identifierait pas assez de groupes (confi $\text{Config.}_{30,70}, V\_A : P\_2$) ou un sur-apprentissage (confi $\text{Config.}_{100,100}, V\_A : P\_1$) ce qui peut légèrement diminuer la performance de l'algorithme (trop de groupes), mais qui reste plus efficace que UCB KERNEL-UCB et UNIFORM. Avec les paramètres appropriés CTREE-UCB apparaît comme l'algorithme qui atteint de bonnes performances.

La Fig. 3.13 présent le regret cumulé des différents algorithmes sous la configuration $\text{Conf}_{30,70}$, $V_A = P_1$ et$\epsilon = 0.05$. Le regret cumulé le plus faible au cours du test est celui de CTREE-UCB (en vert). La stratégie LIN-UCB (bleu) arrive en seconde position en terme de performance. Le regret cumulé le plus important est celui de UNIFORM (en noir). Toutes les stratégies d'allocation dynamique arrivent à converger vers un regret cumulé logarithmique. La période d'exploration s'arrête plus tôt (après 2000 visiteurs) avec notre méthode et l'on observe le gain par rapport à approche fréquentiste (UNIFORM) . Dans ce jeu de données, la variation A était optimale pour tous les profils de visiteurs, ce qui explique la convergence logarithmique de tous les algorithmes d'allocation dynamique.

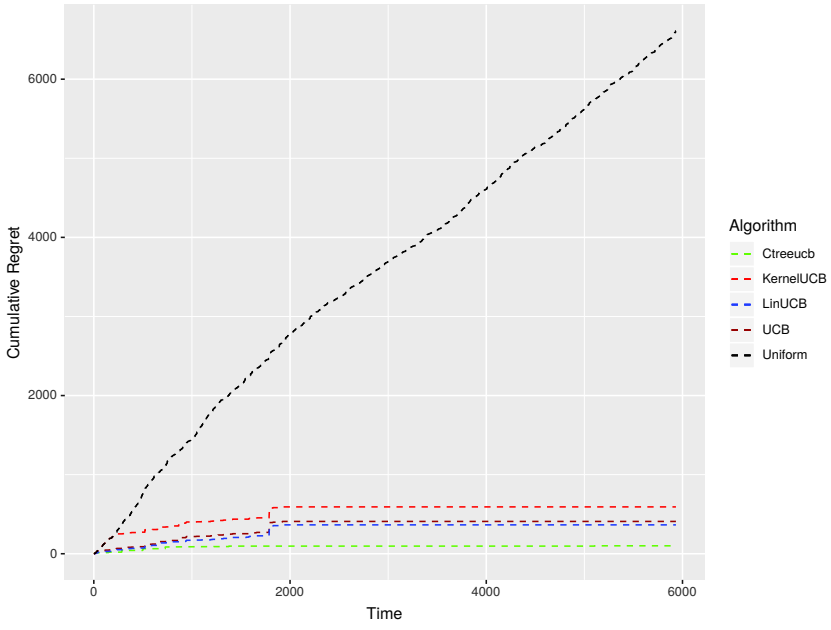


FIGURE 3.13 – Regret cumulé avec le jeu de données AB Tasty 1 ($\text{Conf}_{30,70}$ $V_A = P_1$,$\epsilon = 0.05$).

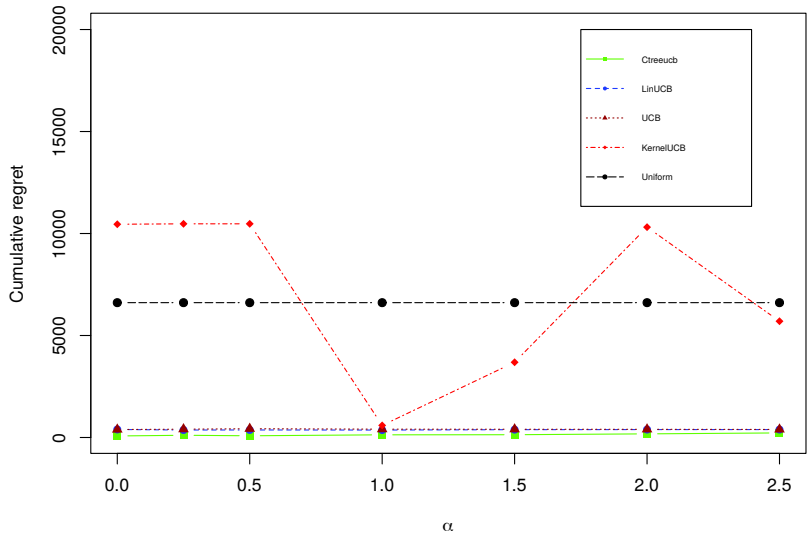


FIGURE 3.14 – Regret cumulé selon α avec le jeu de données AB Tasty 1 de CTREE-UCB ($V_A : P_1$, $\text{Conf}_{30,70}$, $\epsilon = 0.05$) LIN-UCB, UCB, KERNEL-UCB, et UNIFORM. Le paramètre α est observé selon 7 valeurs possibles. Le regret cumulé en accord avec ce paramètre est observé pour chaque algorithme.

3.4.4 Jeu de données AB Tasty 2

Étape offline d'explicitation des groupes

Le test concerne un site de vidéo streamming payant et l'objectif est d'augmenter la valeur des achats. Dans la $\text{Config.}_{30,70}$ 800 visiteurs sont dédiés à l'étape 1 d'apprentissage et 1865 visiteurs sont dédiés à l'étape 2 du test. Dans la $\text{Config.}_{100,100}$ les 2665 visiteurs sont dédiés aux étapes 1 et 2. Comme dans la précédente expérience, la Figure 3.15 présente l'arbre de classement qui identifie 7 groupes avec des moyennes de gains différentes. Les visiteurs dépensant le plus sont ceux qui n'ont été sur le site qu'une seule fois (Node4 et Node#7 et Node#8, Node#11). Chaque feuille associe une valeur d'achat espérée à chacun des groupes.

On remarque que les caractéristiques sélectionnées par CTREE pour définir les groupes sont le user agent et le nombre de visites. Par rapport à l'expérimentation précédente, une nouvelle caractéristique a été sélectionnée pour la classification (le navigateur Internet `name` dans la Figure 3.15). Notre hypothèse est que puisque les visiteurs achètent des vidéos en ligne, le navigateur est important car il est susceptible de faciliter le visionnage.

A/B test (Allocation dynamique)

La Table 3.5 donne le regret cumulé et moyen en fonction de différents paramètres (ϵ et V_A). Avec $V_A : P_1$, et $\text{Conf}_{30,70}$, l'arborescence n'est que légèrement modifiée (appari-

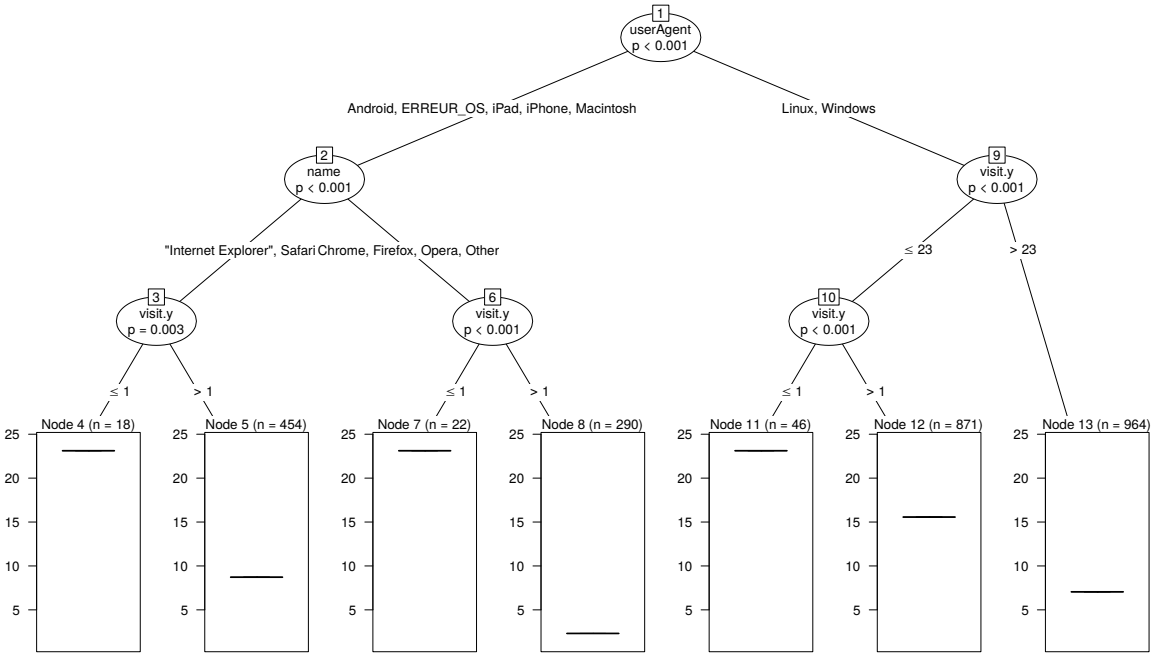


Figure 3.15 – Arbre d'inférence conditionnelle pour le jeu de données AB Tasty 2($\text{Config.}_{100,100}, V_A : P_1, \epsilon = 0.05$).

tion/disparition d'un ou deux groupes maximum). Avec la $\text{Conf}_{100,100}$ le paramètre ϵ n'a pas d'impact sur l'arborescence, voir Figure 3.17. Avec $V_A = P_1$ Ctree-Ucb obtient les meilleurs résultats quelque soit la configuration. Néanmoins, avec $V_A = P_2$, les meilleurs résultats seront donnés par Lin-Ucb. La Fig 3.16 donne le regret cumulatif selon α et met en évidence la performance robuste de Ctree-Ucb. En effet, le regret cumulé le plus faible au cours du test est celui de Ctree-Ucb (en vert). La stratégie Lin-Ucb (bleu) arrive en seconde position en terme de performance. Le regret cumulé le plus important est celui de Kernel-Ucb en rouge et Ucb en marron. Seule la stratégie Ctree-Ucb arrive à converger vers un regret cumulé logarithmique.

On remarque que Kernel-Ucb et Ucb ont un regret cumulé supérieur à uniform. Cela s'explique pour plusieurs raisons telles que par exemple, pour Ucb, la présence de valeurs extrêmes et pour Kernel-Ucb, un nombre d'items trop faible pour produire un modèle pertinent.

		Ctree-Ucb						Lin-Ucb	Kernel-Ucb	Ucb	uniform
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
$\text{Conf}_{30,70}$	R_T	**1251**	1263	1265	3110	3110	3110	3092	4469	4284	4279
	$\mathbb{E}[R_T]$	**0.67**	0.67	0.67	1.67	1.67	1.67	1.67	2.40	2.30	2.30
$\text{Conf}_{100,100}$	R_T	**1994**	**1994**	**1994**	3843	3843	3843	2457	7007	6114	6862
	$\mathbb{E}[R_T]$	**0.75**	**0.75**	**0.75**	1.44	1.44	1.44	0.92	2.63	2.30	2.57

Table 3.5 – Influence des paramètres de segmentation sur le regret cumulé et la moyenne avec l'ensemble de données AB tasty AB Tasty 2 ($\alpha = 1$)

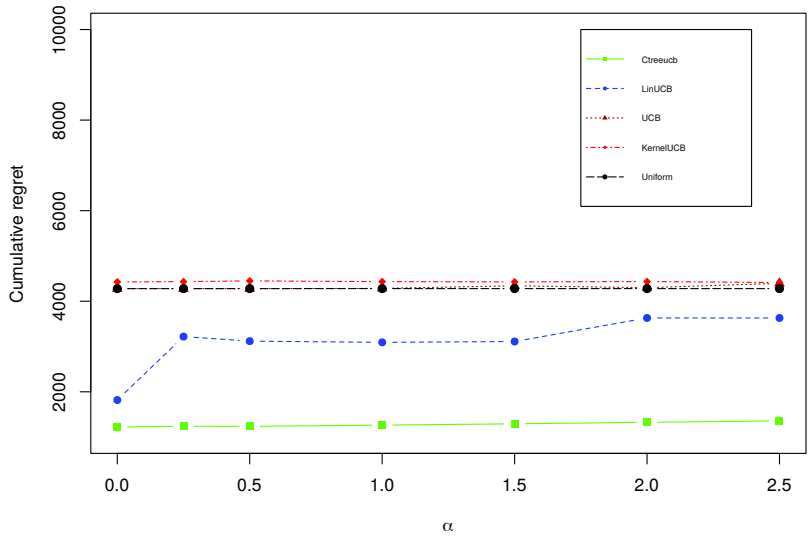


FIGURE 3.16 – Regret cumulé selon α avec le jeu de données d'AB Tasty 2 de CTREE-UCB ($V_A : P_1$, $\text{Conf}_{30,70}$, $\epsilon = 0.05$), LIN-UCB, UCB, KERNEL-UCB, et UNIFORM. Le paramètre α est observé selon 7 valeurs possibles. Le regret cumulé en accord avec ce paramètre est observé pour chaque algorithme.

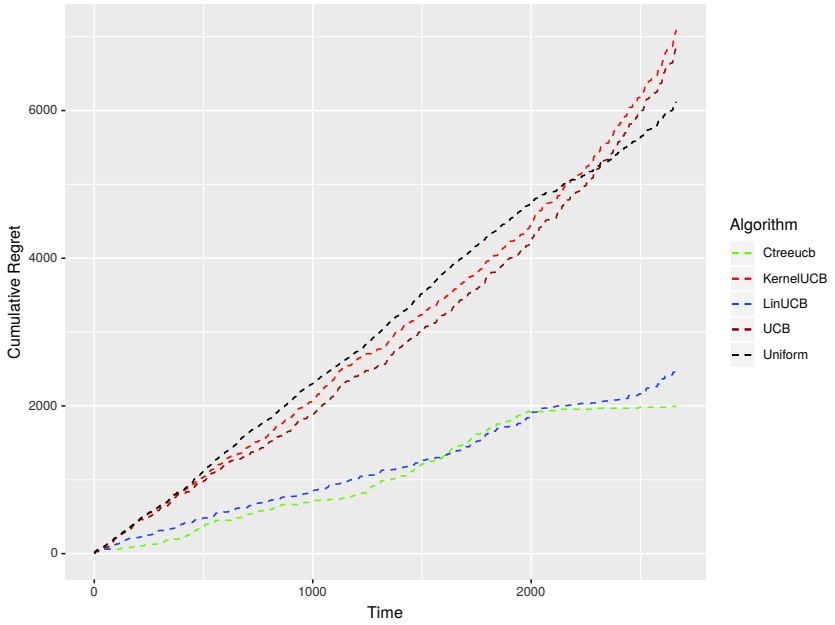
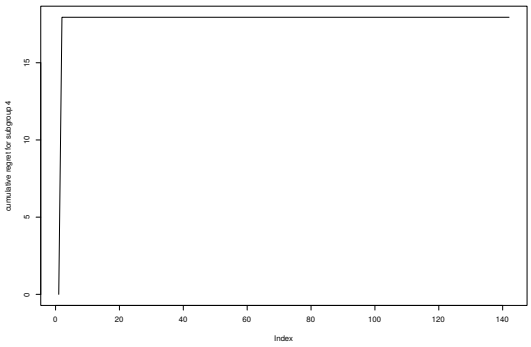


FIGURE 3.17 – Regret cumulé avec le jeu de données AB Tasty 2 ($\text{Conf}_{100,100}$ $V_A = P_1$,$\epsilon = 0.05$, $\alpha = 1$)

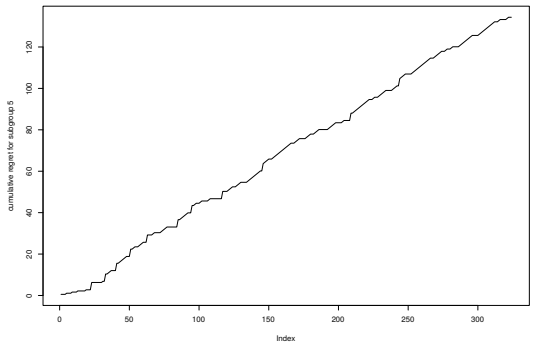
Analyse des groupes Le regret cumulé de CTREE-UCB (voir Figure 3.17) étant naturellement la somme des regrets cumulés de chaque bandit associé à son groupe, nous étudions plus en détail dans cette section les regrets cumulés des 7 groupes définis par l'arbre de classement. Les paramètres de CTREE-UCB sont : ($\text{Conf}_{100,100}$ $V_A = P_1$,$\epsilon = 0.05$,$\alpha = 1$) La Fig. 3.18 présente le regret cumulé de chaque groupe. Les noms de groupes font réfé-

rence à l'identifiant de la feuille dans l'arbre et non au n-ième groupe.

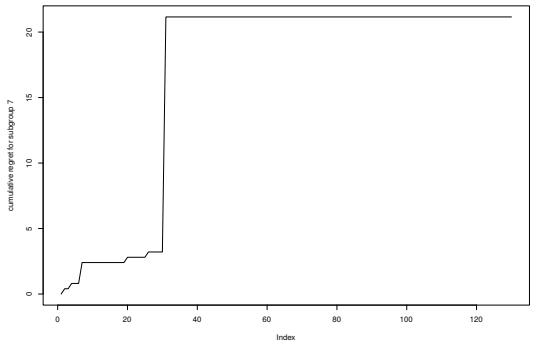
- Le regret cumulé du groupe #4 (voir Fig. 3.18a), est quasi constant tout au long du test A/B. Les deux premiers visiteurs, génèrent la totalité du regret (15), ce qui indique que les visiteurs restants ont été assignés à la variation maximisant les gains et qu'une différence forte existait entre les variations.
- Le regret cumulé du groupe #5 (voir Fig. 3.18b), augmente de façon linéairement tout au long du test. Pour ce groupe, la variation choisie pour l'exploration demande du temps ou n'est pas la meilleure pour tous les visiteurs. Différentes raisons peuvent l'expliquer : l'apprentissage issu de la page d'origine n'a pas permis d'identifier correctement tous les groupes existants dans l'ensemble de données du test ou les caractéristiques utilisées ne sont pas suffisantes pour obtenir une moyenne fiable pour ce groupe.
- Le regret cumulé du groupe #7 (voir Fig. 3.18c), tend asymptotiquement vers un regret égal à 20. L'exploitation pour ce groupe démarre très rapidement (moins de 40% des items dans ce groupe sont nécessaires pour identifier la variation optimale).
- Le regret cumulé du groupe #8 (voir Fig. 3.18d) est constant dans le temps. Le première visiteur a généré un regret, puis les visiteurs restants n'ont pas été sensibles au test.
- Le regret cumulé du groupe #11 (voir Fig. 3.18e), tend asymptotiquement vers un regret égale à 10. Contrairement au groupe #7 (voir Fig. 3.18c), l'exploration pour le groupe #7 nécessite plus d'items.
- Le regret cumulé du groupe #12 (voir Fig. 3.18f), augmente de façon linéairement tout au long du test comme celui du groupe #5. C'est pour ce groupe que le regret cumulé a été le plus élevé.
- Le regret cumulé du groupe #13 (voir Fig. 3.18g), est constant dans le temps. Ici les différences de gain entre les variations ont été très élevées et dès les premiers visiteurs, la variation optimale a été identifiée.



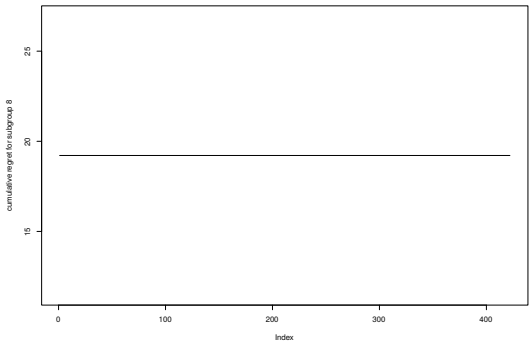
(a) Groupe #4



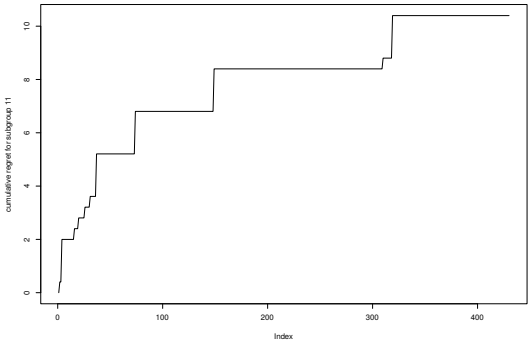
(b) Groupe #5



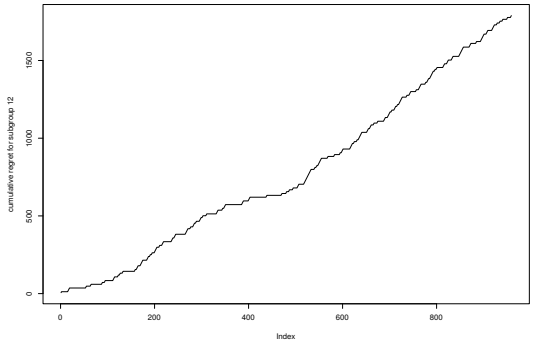
(c) Groupe #7



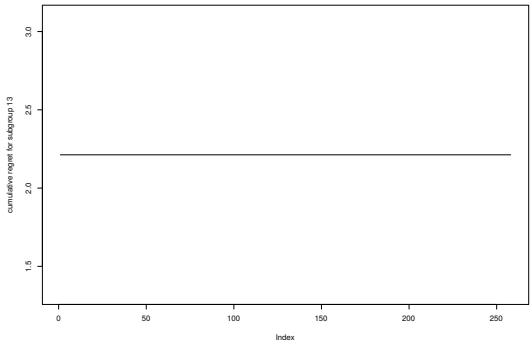
(d) Groupe #8



(e) Groupe #11



(f) Groupe #12



(g) Group #13

FIGURE 3.18 – Regret cumulé pour les groupes identifiés par CTREE-UCB avec le jeu de données AB tasty 2 ($\text{Conf}_{100,100}$ $V_A = P_1$,$\epsilon = 0.05$, $\alpha = 1$)

Temps de réponse Lorsque l'analyste a besoin d'un temps de réponse rapide (par exemple pour un test A/B pour comparer deux pages web, pour chaque visiteur) l'application d'AB Tasty doit afficher la page en moins de 100 milliseconde. L'algorithme doit donner un choix le plus rapidement possible.

La première étape de Ctree-Ucb est réalisée avant l'A/B test (en offline), l'analyste a donc le temps de la calculer. Ainsi, le temps alloué à la première étape n'est pas pris en compte. Seul le temps de calcul requis pour l'allocation dynamique, *i.e.* étape 2 (en ligne), est considéré. La table 3.6 présente le temps de calcul requis pour Ctree-Ucb et le comparons avec Lin-Ucb, Kernel-Ucb et Ucb sur Config.$_{100,100}$. Cette table présente également le temps de calcul le plus long parmi tous les groupes (groupe max). Comme les groupes de visiteurs sont testés séparément, il est possible de répartir le temps de calcul.

Après plusieurs expériences, le temps de réponse par visiteur pour Ctree-Ucb est toujours inférieur à une milliseconde. Kernel-Ucb nécessite le plus long temps de calcul. Ceci est expliqué par le calcul de la régression par noyau. Cependant, plus le nombre de caractéristiques augmente, plus le temps de réponse de Lin-Ucb et Kernel-Ucb s'accroît. En revanche, Ctree-Ucb fournit un temps d'exécution court quelque soit la valeur de d et permet son utilisation lorsque le temps de réponse est une contrainte de l'analyste.

	Ctree-Ucb	Lin-Ucb	Kernel-Ucb	Ucb
Temps total d'exécution (secondes)	0.504 (max group : 0.226)	0.622	16.013	0.096
Temps total d'exécution par visiteur (millisecondes)	0.18 (max group : 0.08)	0.23	6	0.03

Table 3.6 – Temps total de calcule pour chaque algorithme (Config.$_{100,100}$) sur le jeu de données d'A/B Tasty 2

3.4.5 Jeu de données AB Tasty 3

Étape offline d'explicitation des groupes

Le test concerne un site de vidéo straming payant et l'objectif est d'augmenter le taux de clics. Dans la Config.$_{30,70}$, 800 visiteurs sont dédiés à l'étape 1 d'apprentissage et 1865 visiteurs sont dédiés à l'étape 2 de test. Dans la Config.$_{100,100}$ les 2665 visiteurs sont dédiés aux étapes 1 et 2. La récompense est une valeur binaire (0 : pas de clic, 1 : clic). Sur la Figure 3.19, chaque feuille associe la valeur de la probabilité de clic espérée apprise à partir de la variation originale. Dans cet arbre, 5 groupes sont identifiés avec des probabilités de gains statistiquement différentes. Les visiteurs qui clic le plus sont ceux qui ont déjà été sur le site 16 fois maximum et qui accèdent au site via leur Iphone ou via un appareil non identifié (Node#8). La plus faible probabilité de clics est donnée pour des visiteurs ayant été sur le site moins de 3 fois et utilisant un autre type d'appareil (Node#4).

A/B test (Allocation dynamique)

		Ctree-Ucb						Lin-Ucb	Kernel-Ucb	Ucb	uniform
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
$\text{Conf}_{30,70}$	R_T	424	441	445	**387**	**387**	**387**	457	389	424	411
	$\mathbb{E}[R_T]$	0.23	0.24	0.24	**0.21**	**0.21**	**0.21**	0.25	0.21	0.23	0.22
$\text{Conf}_{100,100}$	R_T	658	658	658	**526**	**522**	560	558	581	589	588
	$\mathbb{E}[R_T]$	0.25	0.25	0.25	**0.20**	**0.20**	0.21	0.21	0.22	0.22	0.22

Table 3.7 – Influence des paramètres sur le regret cumulé R_T et moyen $\mathbb{E}[R_T]$ sur le jeu de donnée d'AB Tasty 3 ($\alpha = 1$)

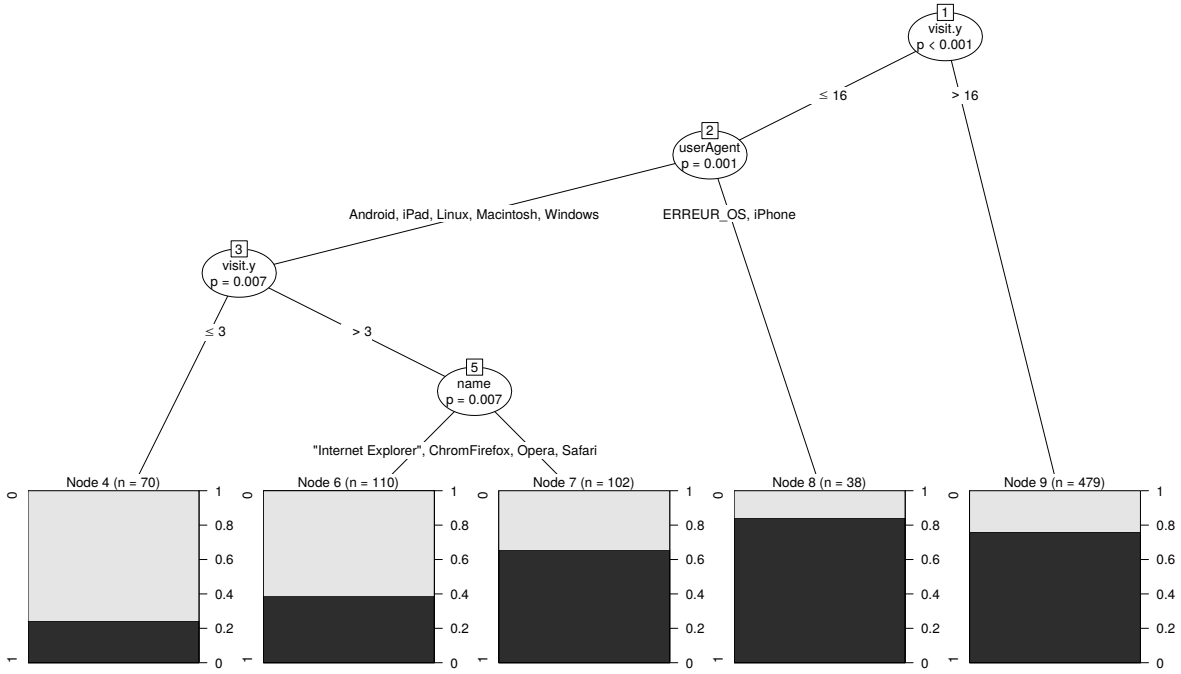


Figure 3.19 – Arbre d'inférence conditionnelle pour le jeu de données AB Tasty 3 ($\text{Config.}_{30,70}, V\_A : P\_2, \epsilon = 0.05$).

La Table 3.7 montre le regret cumulé et moyen de chaque algorithme. Si $V_A = P2$, Ctree-Ucb apparaît comme l'algorithme qui obtient les meilleures performances, indépendamment de $Config.$, α ou ϵ. En revanche si $V_A = P1$, Ctree-Ucb obtient une faible performance.

La Fig. 3.20 donne un exemple illustratif d'une configuration ($\text{Conf}_{30,70}$ $V_A = P_2$,$\epsilon = 0.05, \alpha = 1$). Le regret cumulé le plus faible au cours du test est celui de Ctree-Ucb (en vert). La stratégie uniform (en noir) arrive en seconde position en terme de performance. Le regret cumulé le plus important est celui de Lin-Ucb (bleu). Dans cet exemple, les probabilités de clics pour les deux variations sont très proches.

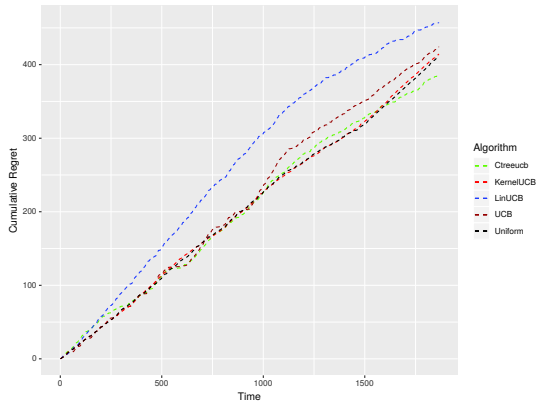


FIGURE 3.20 – Regret cumulé selon avec le jeu de données AB Tasty 3 ($\text{Conf}_{30,70}$ $V_A = P_2$,$\epsilon = 0.05, \alpha = 1$).

3.4.6 A/B Tasty database 4,5 et 6

Par souci de lisibilité, seuls les tableaux listant le regret cumulé et moyen de ces 3 jeux de données sont rapportés ici. La Table 3.8 détaille le contexte des différents A/B tests. Le symbole \$ indique que l'objectif de l'analyste est d'accroître la valeur d'achat, le symbole ▷ indique que l'objectif de l'analyste est d'augmenter le taux de clics. À l'exception des résultats rapportés dans la Tab.3.11, CTREE-UCB obtient le regret moyen le plus faible (elle arrive en seconde position dans cette dernière expérience). Même si les paramètres de CTREE-UCB ne sont pas optimaux (c'est-à-dire une configuration qui minimise le regret par rapport aux autres méthodes), les résultats pour 35 cas sur 36, sont généralement meilleurs ou équivalents à la stratégie UNIFORM.

Jeu de données	Secteur	Taille learnset		Taille testset		Objectif
		$\text{Conf}_{30,70}$	$\text{Conf}_{100,100}$	$\text{Conf}_{30,70}$	$\text{Conf}_{100,100}$	
4	Prêt à porter	3585	11952	8367	11952	\$
5	Prêt à porter	3585	11952	8367	11952	▷
6	Mobilier	1508	5028	3502	5028	▷

TABLE 3.8 – Description des jeux de données additionnels d'A/B Tasty

		CTREE-UCB						LIN-UCB	KERNEL-UCB	UCB	UNIFORM
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
$\text{Conf}_{30,70}$	R_T	287569	68684	72513	54926	**53014**	**54022**	73408	103543	76451	197661
	$\mathbb{E}[R_T]$	34.37	8.21	8.67	6.56	**6.34**	**6.46**	8.77	12.38	9.14	23.62
$\text{Conf}_{100,100}$	R_T	99786	106331	101874	**77049**	80537	**79954**	126586	178496	109535	279379
	$\mathbb{E}[R_T]$	8.35	8.90	8.52	**6.45**	6.74	**6.69**	10.59	14.93	9.16	23.38

TABLE 3.9 – Influence des paramètres sur le regret cumulé R_T et moyen $\mathbb{E}[R_T]$ sur le jeu de donnée d'AB Tasty 4 ($\alpha = 1$)

		Ctree-Ucb						Lin-Ucb	Kernel-Ucb	Ucb	uniform
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
$\text{Conf}_{30,70}$	R_T	1981	1981	**1674**	1981	1981	1981	2286	2034	1981	2031
	$\mathbb{E}[R_T]$	0.24	0.24	**0.20**	0.24	0.24	0.24	0.27	0.24	0.24	0.24
$\text{Conf}_{100,100}$	R_T	2917	2985	2883	2713	2707	**2564**	2657	2928	2606	2920
	$\mathbb{E}[R_T]$	0.24	0.25	0.24	0.23	0.23	**0.21**	0.22	0.24	0.22	0.24

Table 3.10 – Influence des paramètres sur le regret cumulé R_T et moyen $\mathbb{E}[R_T]$ sur le jeu de donnée d'AB Tasty 5 ($\alpha = 1$)

		Ctree-Ucb						Lin-Ucb	Kernel-Ucb	Ucb	uniform
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
$\text{Conf}_{30,70}$	R_T	712	712	712	620	620	620	**384**	653	712	653
	$\mathbb{E}[R_T]$	0.20	0.20	0.20	0.18	0.18	0.18	**0.11**	0.19	0.20	0.19
$\text{Conf}_{100,100}$	R_T	1047	1047	1051	977	970	960	**863**	947	1047	948
	$\mathbb{E}[R_T]$	0.21	0.21	0.21	0.19	0.19	0.19	**0.17**	0.19	0.21	0.19

Table 3.11 – Influence des paramètres sur le regret cumulé R_T et moyen $\mathbb{E}[R_T]$ sur le jeu de donnée d'AB Tasty 6 ($\alpha = 1$)

3.5 Analyse des expérimentations

Les expérimentations ont montré que Ctree-Ucb répondait aux objectifs de l'analyste à travers différents A/B tests concrets. À travers ces expériences, des caractéristiques continues, binaires et catégorielles et des récompenses continues ou binaires ont été considérées. Nous avons également observé le cas où $K > 2$, bien qu'en pratique, dans les A/B tests appliqués au web, l'analyste se limite à deux variations [11]. Ces expériences ont montré la performance de Ctree-Ucb à travers différents cas d'usage. Ctree-Ucb requiert 3 paramètres : Conf., V_A, ϵ et α. Nous pouvons conclure que :

- Un jeu de données partiel ($\text{Conf}_{30,70}$) pour la première étape est suffisant pour identifier des groupes représentatifs de la population testée, car les résultats obtenu sont comparables à $\text{Conf}_{100,100}$.
- Selon la variation originale V_A, à améliorer, les résultats de Ctree-Ucb sont susceptibles de varier. Quoiqu'il en soit, ces résultats restent relativement bons par rapport à des méthodes qui apprennent en ligne telles que Lin-Ucb et Kernel-Ucb, ou par rapport à une méthode qui ne réalise aucun classement (Ucb).
- Le paramètre ϵ (associé au risque de type I accepté dans les tests statistiques lors de la construction de l'arbre d'inférence) a peu d'influence sur les résultats. Une valeur par défaut ($\epsilon = 0.05$) peut être acceptée.
- Une valeur incorrecte de α peut détériorer fortement les résultats de Lin-Ucb et Kernel-Ucb alors que Ctree-Ucb montre plus de stabilité par rapport à ce paramètre.

La qualité des résultats de Ucb Kernel-Ucb et Lin-Ucb varie selon les types de données utilisées. Ces algorithmes peuvent devenir équivalents à une stratégie uniform

11. Chez AB Tasty, 70% des analystes ne testent que deux variations. Dans le cas contraire, le site de l'analyste doit avoir un trafic conséquent sur la page testée

lorsque leurs hypothèses ne sont pas respectées (par exemple la linéarité), ou que le paramètre α est inapproprié. L'analyste n'a alors aucun avantage à utiliser ces méthodes qui ralentissent le temps d'assignation à une variation. Kernel-Ucb se révèle être la méthode la plus difficile à paramétrer. Comme le cite Cesa-Bianchi *et al.* dans [19] *"En particulier, lorsque le nombre d'évaluations du noyau est limité, il y a des cas où aucun algorithme n'atteint des performances meilleures qu'une stratégie de sous-échantillonnage trivial, où la plupart des données sont jetées. De plus, aucun algorithme ne peut bien fonctionner lorsque le paramètre de régularisation est suffisamment petit ou lorsque la contrainte de norme est suffisamment grande."*. D'un autre côté, et particulièrement avec les données d'A/B tasty, Ctree-Ucb affiche les meilleures performances, notamment lorsque la récompense est numérique à valeur réelle.

3.6 Conclusion

Ce chapitre a détaillé la nouvelle méthode Ctree-Ucb comparable à une stratégie de bandit contextuel et a présenté ses résultats sur des données simulées et réelles. Ctree-Ucb obtient à la fin du test des regrets cumulé et moyen comparables voire inférieurs à ceux obtenus par des algorithmes traditionnels dans la littérature. Ctree-Ucb produit de bons résultats quelque soit les paramètres choisis. Cela devrait nous permettre de définir un paramétrage par défaut acceptable dans bien des cas (hypothèse que nous souhaitons valider par de nouvelles expérimentations).
Ce chapitre a également montré que le temps de calcul requis par Ctree-Ucb permet son intégration dans un environnement industriel où un temps de réponse rapide est nécessaire. De plus, des expériences ont montré que l'arbre d'inférence conditionnelle définit des groupes, tel que le regret cumulatif/moyen durant l'allocation dynamique fait partie des plus faibles parmi les méthodes comparées. En pratique, l'utilisation de Ctree-Ucb présente donc les avantages suivants :

- Les arbres d'inférences conditionnelles sont facilement interprétables par l'analyste (qui peut être un médecin, un expert marketing ...), ce qui lui permet de faire des hypothèses pour chaque groupe. Dans nos expériences l'analyste a identifié facilement des "personas" c'est-à-dire des profils-type d'items. Il est ainsi plus facile pour lui d'imaginer des variations.
- Les arbres d'inférences conditionnelles gèrent bien les données catégorielles. Ces caractéristiques peuvent être fournies par l'item testé (système d'exploitation, navigateur, etc...) ou complétées par des services externes (tels que Data Management Platform [12]) qui fournissent des informations complémentaires (sexe, âge, etc...).
- La création de l'arborescence sur les données de la variation originale peut être

12. Une Data Management Platform est une plateforme de gestion des données sur les visiteurs. Les données sur les visiteurs y sont stockées pour, par exemple, cibler la publicité en ligne.

rassurante pour l'analyste. Il n'y a pas de sentiment de risque puisque c'est la variation originale, déjà en production, qui est utilisée. La nouvelle variation (crainte, car potentiellement mauvaise) ne sera évaluée que dans un contexte de bandit.

— Parfois, les changements de variations sont mineurs. Ainsi, dans la pratique, le test n'aura alors qu'un impact limité sur quelques groupes. L'isolement des items dans un groupe limite les conséquences de récompenses dites "extrêmes" ou identique entre deux variations dues par exemple des acheteurs ayant dépensé une grande somme, des patients en état critique insensibles à tout traitement ...

— La génération de l'arbre de classement (via la méthode CTREE) ne séléctionne que des caractéristiques corrélées à la récompenses exclut celles qui ne présentent aucun intérêt et propose une approche de régression non linéaire . Il s'agit d'un compromis entre des approches qui apprennent *from scratch* mais qui peuvent accroître le regret et une approche globale qui limite l'interprétation du test A/B (une seule meilleure variation est considérée comme la meilleure) et conduit à un regret simple décroissant pendant le test.

— La construction de groupes, réalisée hors ligne, permet d'obtenir un temps de réponse comparable à une méthode non contextuelle et peut être diminuée avec un calcul distribué (en distribuant l'allocation de chaque groupe sur une serveur différente). Ainsi, Ctree-Ucb a un temps de réponse très faible comparé aux méthodes contextuelles traditionnelles. Il peut être implémenté pour tout cas d'utilisation dans lequel l'analyste a besoin d'une réponse rapide.

— La génération de groupes par un arbre d'inférence conditionnel est basée sur des tests statistiques qui acceptent les valeurs manquantes dans les caractéristiques. Des données manquantes peuvent se produire dans de nombreux cas pratiques et l'utilisation de Ctree-Ucb est une solution à de telles situations.

— Ctree-Ucb ne nécessite pas un nombre d'items important pour apprendre les groupes et accepte de fait des échantillons de petite taille (∼30 items) ou de taille moyenne (∼100 000 items).

— Les arbres d'inférence conditionnelle assurent un arbre de bonne taille proposé sans élagage supplémentaire ou validation croisée.

La bonne performance de Ctree-Ucb dans l'identification des groupes avec différentes variations originales suggère une corrélation entre la distribution des variations. Dans la pratique, les changements apportés par une variation sont généralement limités. Il est donc possible de créer des groupes sur la variation A et de supposer qu'ils sont similaires sur la variation B. Enfin, nos expériences sur les données partielles (30%) ou sur l'ensemble des données montrent que les groupes sont persistants dans le temps. De plus, l'identification des groupes peut aider à guider l'analyste sur la composition du test lui-même. Si le test n'était pas pertinent pour un groupe, un nouveau test peut être refait par la suite, plus spécifique.

Chapitre 4

Allocation dynamique et données temporelles

4.1 Introduction

Comme nous l'avons introduit précédemment, il apparaît de plus en plus indispensable d'étudier l'influence du temps sur les tests que ce soit au niveau global du test lui même (caractéristiques et hypothèses temporelles) ou au niveau plus local des items, c'est-à-dire de leurs caractéristiques et comportement lors du test. En effet, au niveau global, les méthodes actuelles font généralement des hypothèses de stationnarité sur les distributions réelles des gains associées aux différentes variations (par exemple, la variation optimale est toujours la même), sur les caractéristiques globales de l'ensemble de items (par exemple, le proportion d'hommes et femmes est invariant) ou encore que sur les caractéristiques des items eux-mêmes (par exemple, la propension à réagir à un médicament reste la même durant tout le protocole de soins). Malheureusement, ces hypothèses s'avèrent de plus en plus contraignantes car éloignées de la réalité. Il est donc indispensable à la fois d'en étudier l'impact sur les tests et proposer de nouvelles méthodes autorisant un relâchement de telles contraintes de stationnarité. Malheureusement, par manque de temps, l'étude de nouvelles méthodes d'A/B test permettant de relâcher les contraintes de stationnarité n'a pas pu être pas abordée concrètement dans cette thèse. Nous nous sommes uniquement intéressés au niveau local des items et en particulier aux apports théoriques et pratiques de la prise en compte des informations temporelles liées à leurs comportements avant et pendant le test *afin d'améliorer l'allocation des items aux variations*. En effet, grâce aux techniques de stockage de données massives, l'analyste a sa disposition de plus en plus souvent un grand nombre d'informations pour chaque item testé et en particulier de plus en plus de données temporelles. Ces données peuvent être de deux natures différentes :

— Des données temporelles historisées, concernant l'item, par exemple produites avant l'arrivée de celui-ci sur le site (historique de navigations antérieures sur ce site ou d'autres sites : liste des pages visitées, historiques d'achats, ...) ou dans le domaine médical avant la prise en charge du patient (liste des maladies, historique des traitements, évolution du poids sur les dernières années ...).

— Des données temporelles évolutives, générées dynamiquement à partir des actions

effectuées par les items lors du test par exemple entre son arrivée sur le site et son arrivée sur la page testée (clics, achats, ...). Ainsi par exemple, dans le cadre des sites e-marchants, grâce aux informations collectées à la volée, le responsable du site peut connaître pour chaque item le temps passé sur le site à chaque visite, le nombre et les pages visitées, ou encore les historiques d'achats depuis sa première connexion sur le site. Ces données sont appelées le *parcours d'achat*. Dans le domaine médical, ces données peuvent correspondre à des séries d'analyses successives : courbes de températures ou de pressions, liste et durée des différents traitements successifs depuis la prise en charge du patient.

La collecte de données temporelles historisées est parfois, en pratique, fortement limitée pour des raisons légales. Par exemple le partage d'information entre différents sites est interdit pour des sociétés externes telles qu'AB Tasty. Ces informations ne peuvent être collectées qu'au bon vouloir du visiteur, qui généralement est réticent à les communiquer. Dans nos travaux, nous nous sommes principalement intéressés aux données évolutives (voir Section 1.3) qui, elles, peuvent être générées "sans limitations". Analyser ces données temporelles plus finement semble intéressant afin d'en extraire des comportements partagés entre les items. En effet, cela devrait permettre de mieux expliquer les allocations aux différentes variations. Par exemple : le traitement B est visiblement plus efficace pour les personnes ayant un poids très instable que le traitement A, la variation B d'une page web n'est performante que pour les visiteurs se rendant fréquemment sur le site ...

Pour répondre à cet objectif d'amélioration de l'allocation, il nous semble indispensable de pouvoir intégrer ces informations temporelles évolutives directement dans le contexte (caractéristiques) des items. Malheureusement, les méthodes actuelles sont restrictives sur la structure des contextes manipulés : le contexte ne peut être défini que par un vecteur comportant des valeurs numériques. Cette restriction limite l'utilisation des données temporelles dont dispose l'analyste. Ainsi, il n'existe pas de méthode d'allocation dynamique dans la littérature permettant d'exploiter directement des séries temporelles comme caractéristiques de contexte pour réaliser une allocation dynamique.

Plutôt que de changer la méthode d'allocation dynamique nous proposons de transformer les caractéristiques évolutives en des caractéristiques unidimensionnelles (catégorielle par exemple). Ainsi, nous proposons dans ce chapitre une nouvelle méthodologie (et deux algorithmes associés) permettant une allocation dynamique contextuelle lorsque le contexte des items à tester intègre des caractéristiques temporelles sous forme de séries de valeurs binaires ou numériques.

La section 4.3 détaille le principe genéral de notre approche et notre contribution. La section 4.5 fixe le cadre d'expérimentation. Les section 4.4 et 4.5 présentent les résultats, suite aux expériences réalisées. La section 4.6 conclut sur ces résultats et propose des perspectives de recherche et de développement.

4.2 Données temporelles et contexte évolutif

Les algorithmes détaillés dans la section 2.4 et 3.1.2 présentent des restrictions fortes sur le format du contexte, celui-ci devant être présenté sous forme d'un vecteur à valeurs réelles ou catégorielles. De fait, ces restrictions limitent l'utilisation des algorithmes de bandits dans le cas où les caractéristiques à considérer portent sur des séries temporelles. Une solution pour contourner ce problème est alors d'aplatir les séries temporelles de chacune des caractéristiques temporelles. Deux approches sont alors possibles :

— Choisir la dimension du contexte telle qu'elle peut contenir tous les éléments différents dans toutes les séries. Pour chaque caractéristique temporelle $j \in \{1, ..., d\}$ des items $t \in \{1, ..., n\}$, l'ensemble des éléments temporels différents est extrait. Pour chacun de ces éléments, une nouvelle caractéristique est créée directement dans le contexte des items. Puis, pour chaque item, celle-ci est mise à 1 si cet élément temporel apparaît dans une série temporelle de l'item. La taille d'un vecteur de contexte sera égale au nombre d'éléments différents dans toutes les séries observées. Cette méthode est représentée dans la figure 4.1.

— Concaténer la taille de toutes les séries de tous les items. Pour chaque caractéristique temporelle $j \in \{1, ..., d\}$ et chaque item $t \in 1, ..., n$, la taille maximale $m_{t,j}$ de ses séries temporelles calculée. $m = \sum_1^d \text{argmax}_t\, m_t$ caractéristiques sont créées directement dans le contexte des items. Puis, pour chaque item, celle-ci prend la valeur dans la série temporelle de l'item, NA sinon. La taille d'un vecteur de contexte sera égale à la somme de tailles maximales des séries temporelles de chaque caractéristique des items. Cette méthode est représenté dans la figure 4.2.

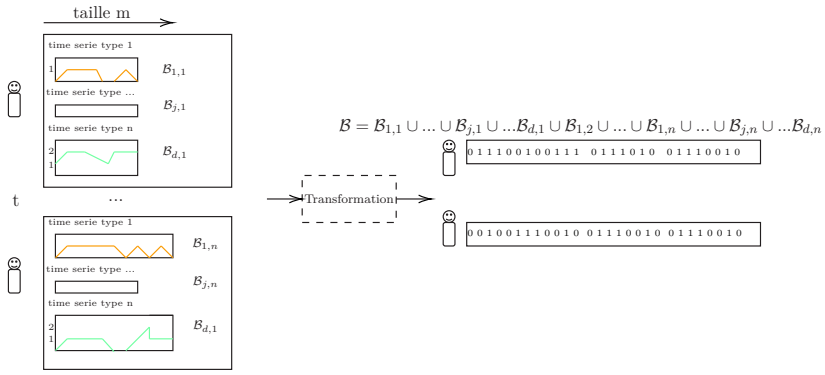


FIGURE 4.1 – Premier traitement possible des séries temporelles pour de l'allocation dynamique.

Ces approches posent plusieurs problèmes. D'une part, ces contextes pourront s'avérer creux si le nombre d'éléments temporels différents est élevé. D'autre part, et surtout, le modèle du bandit contextuel repose sur une modélisation statistique empirique prenant en entrée un contexte pour fournir une probabilité de succès ou une récompense moyenne. Or un contexte de grande taille introduit une grande variabilité sur les prédictions du

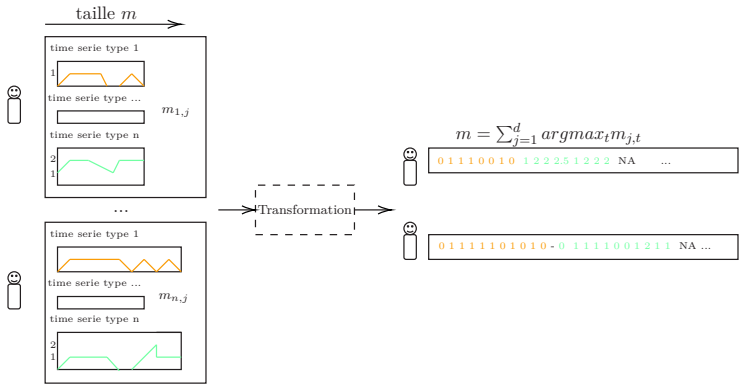


FIGURE 4.2 – Second traitement possible des séries temporelles pour de l'allocation dynamique.

modèle, ce qui nécessite de collecter beaucoup de données.

Pour répondre au problème de vecteur de contexte de très grande taille, [9] propose de construire empiriquement un modèle statistique de type LASSO pour l'intégrer dans une problématique de bandit. Cependant, cette approche nécessite que les caractéristiques soient indépendantes entre elles, ce qui, de façon évidente, n'est pas le cas lorsque des caractéristiques sont issues d'une même série temporelle. De fait, considérer une caractéristique temporelle est une tâche difficile sans *a priori* sur les données et il n'existe pas, à notre connaissance, d'algorithmes de bandits contextuels pouvant s'adapter à des données de taille variable. Une alternative possible est de réduire chaque caractéristique évolutive à une caractéristique catégorielle ou numérique représentative. C'est cette alternative qui caractérise notre approche.

4.3 Allocation dynamique basée sur des informations temporelles

4.3.1 Principe proposé

Dans notre approche, nous avons cherché à remplacer les séries temporelles, inexploitables telles quelles par un algorithme de bandit, par une caractéristique catégorielle. Idéalement, dans le cas du e-commerce par exemple, cette caractéristique pourrait représenter les patterns d'e-marketing proposés par Roukins (voir section 1.3) puisque ses travaux ont montrés qu'ils décrivaient bien les visiteurs par rapport à leur avancement dans le processus d'achat. Malheureusement, ces patterns sont difficiles à mettre évidence par l'expert car cela nécessiterait d'analyser manuellement les comportements passés des visiteurs afin d'en extraire des comportements caractéristiques. Nous proposons donc d'introduire dans notre méthode, un mécanisme d'identification automatique de ces *caractéristiques catégorielles représentatives*. Pour cela nous proposons de chercher des

séquences moyennes communes dans les séries temporelles supposées alors représenter les comportements caractéristiques des visiteurs. Pour cela nous ne considérerons pas la récompense car l'objectif ici n'est pas de la prédire mais d'identifier et de représenter des comportements différents.

Notre proposition consiste donc à définir un ensemble de groupes (par exemple par clustering) sur chacune des caractéristiques temporelles des items de la variation A. Puis dans la phase d'explicitation des groupes de Ctree-Ucb, de remplacer, pour chaque caractéristiques temporelle, chacune des série temporelles par le label du groupe (par exemple , le numéro du cluster) auquel elle appartient dans le clustering correspondant à la caractéristique concernée. Ces labels seront alors utilisés lors de la construction de l'arbre de classement. Enfin, lors de la phase d'exploitation, les caractéristiques temporelles des items à tester seront modifiées de manière identique avant l'opération de classement de l'item.

4.3.2 Des données à temps réelles à données catégorielles

Pour remplacer les séries par une caractéristique catégorielle, une première solution est de demander à l'analyste de labeliser chaque série manuellement. Cette solution est inenvisagable dès lors que le nombre de données devient important et qu'une allocation en temps réel est nécessaire. La deuxième solution consiste à remplacer une série par la moyenne des valeurs qu'elle contient. Une troisième solution est de d'abord construire des groupes automatiquement par exemple via une méthode de clustering, puis d'utiliser les labels (ou numéros) des clusters comme information catégorielle.

Dans notre travaux, nous avons mis en oeuvre cette dernière solution car elle nous semblait être la plus prometteuse. En effet, le résumé d'informations temporelles est un problème bien connue en fouille de données et les approches basées sur le clustering ont prouvé leur efficacité et pertinence dans ce cadre. Pour valider cette proposition, nous la comparerons entre autres à une approche basée sur la moyenne des séries.

Clustering Le clustering est une méthode permettant l'extraction d'informations structurelles (classes subjacentes) implicites dans les données. Le clustering de séries temporelles a donc un but exploratoire via une approche non supervisée. Il existe un grand nombre de méthodes de clustering par partitionnement ou hérarchiques, basées sur la similarité, la densité des données ou sur l'observation des distributions. Le lecteur peut trouver une présentation plus exhaustives de ces méthodes dans [1]. Dans notre cas, nous avons décidé d'utiliser une approche de partitionnement basée sur la similarité entre les items, car elle est très fortement utilisée en fouille de données et y a fait ses preuves. Néanmoins, toute autre méthode permettant l'extraction de clusters pourrait être utilisée. Pour mettre en œvre de une telle méthode, il nous a été nécessaire de fixer une mesure

de similarité [3]. En effet, dans les méthodes de clustering de séries temporelles, si l'on souhaite comparer des séries à échantillonnages irréguliers ou bien de tailles différentes, une attention particulière doit être portée sur le choix la mesure de similarité.

Choix de la mesure de similarité : Il existe un grand nombre de méthodes de mesure de similarité. Ainsi, la plus connue est la distance Euclidienne [12] qui consiste à calculer la somme des carrées des distances des éléments constitutifs des séquences considérées à chaque pas de temps. Si cette distance est très utilisée, elle ne peut l'être dans notre cas car elle impose que les séries aient toutes la même longueur. Or, dans nos travaux, nous nous plaçons dans le cas où l'observation des variables de contexte peut être irrégulière. Par exemple un visiteur peut se rendre plus fréquemment sur le site qu'un autre. Un patient peu avoir été pris en charge plus ou moins tôt.

La mesure de similarité *D.T.W. (Dynamic Time Warping)* est une métrique entre deux séries de tailles différentes largement reconnue comme pertinente pour de nombreux domaines d'application comme par exemple en vérification de signatures [2], en génétique [18], dans le suivi d'objets dans l'espace [35], en reconnaissance d'écriture [60] ou encore en télédétection [56]. D.T.W. construit un appariement des éléments des séquences en les associant / alignant. Cet ensemble d'associations respecte l'ordre total sur le séquencement des valeurs ; de façon imagée, les associations ne peuvent se croiser. La fonction objectif résolue de façon optimale par D.T.W. correspond à la minimisation de la somme des coûts des différentes associations. Soit $\delta(a_i, b_j)$ la similarité entre les éléments de deux séquences a et b, D.T.W. est définie par :

$$DTW(A_{1,\dots,i}, B_{1,\dots,j}) = \delta(a_i, b_j) + \min \left\{ \begin{array}{c} DTW(A_{1,\dots,i-1}, B_{1,\dots,j-1}) \\ DTW(A_{1,\dots,i}, B_{1,\dots,j-1}) \\ DTW(A_{1,\dots,i-1}, B_{1,\dots,j}) \end{array} \right\}$$

La figure 4.3 (source [55]) illustre le concept de la mesure de similarité D.T.W. Soit m la matrice de coûts d'association des toutes les sous-séries temporelles issues de A et B, représentée dans la figure 4.3, l'algorithme 5 défini le procédé de calcul de D.T.W..

En choisissant la métrique D.T.W, toutes les données sont utilisées même en présence d'un échantillonnage irrégulier, ce qui est notre cas. De plus, des comportements considérés très semblables voire identiques bien que décalés ou distordus dans le temps seront bien considérés comme semblable par D.T.W. [53]. Néanmoins, D.T.W reste une métrique sensible au bruit. Cependant, cela n'est pas gênant dans notre contexte dans la mesure où les données manipulées dans nos expériences sont supposées non bruitées.

Disposant d'une mesure de similarité, il est alors possible d'appliquer plusieurs algorithmes destinés à la fouille de données. Le lecteur peut trouver d'autres mesures selon les cas pratiques dans [1].

Algorithm 5 Algorithme DTW

Require: — Une séquence $A = \langle a_1, \dots a_S \rangle$.
— Une séquence $B = \langle b_1, \dots b_T \rangle$.
— Une matrice de coût m de taille $S * T$ initialisée à 0.

1: $m[1,1] \leftarrow \delta(a_1, b_1)$
2: **for** $j \in \{2, \dots, T\}$ **do**
3: $m[1,j] = m[1,j-1] + \delta(a_1, b_j)$
4: **for** $i \in \{2, \dots, S\}$ **do**
5: **for** $j \in \{2, \dots, T\}$ **do**
6: $$m[i,j] = \delta(a_i, b_j) + \min \left\{ \begin{matrix} m[i-1,j] \\ m[i,j-1] \\ m[i-1,j-1] \end{matrix} \right\}$$

Output : m

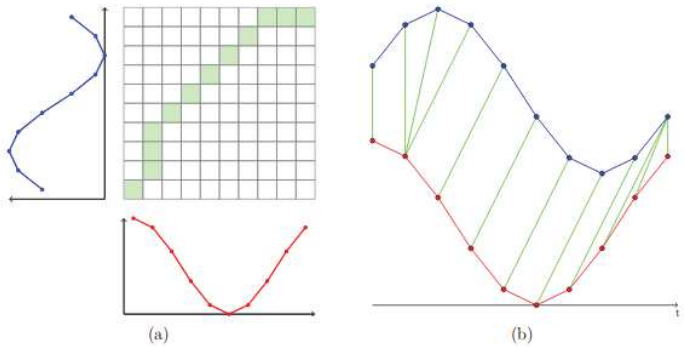


FIGURE 4.3 – Similarité de deux séries par DTW. (a) Matrice et chemin d'alignement optimal calculé par DTW. (b) Alignement résultant des deux séquences.

K-means Lorsque l'on dispose d'une mesure de similarité une première approche est la classification ascendante hiérarchique (C.H.A). Cette méthode, bien que courante en fouille de données, n'a malheureusement pas donné de résultats satisfaisants en pratique et ne sera donc pas abordée par la suite. Une autre méthode très utilisée en fouille de données est l'algorithme de partitionnement K-Means. Cependant celui-ci nécessite de disposer d'une méthode de calcul de la moyenne des observations [51].

La méthode de moyennage *D.B.A (Dtw Barycentric Averaging)* introduite par [54] a montré de bonnes performances dans ce cadre. D.B.A. a été conçue comme une méthode globale, en opposition aux stratégies par paire telles que NLAAF [37]. L'idée principale de D.B.A., inspirée de K-means, repose sur un raffinement itératif d'une séquence moyenne initiale (initialisée arbitrairement), afin de minimiser sa distance quadratique aux séquences à moyenner (SRC). Cette itération se fait en deux étapes :

— Association de chaque point de chaque séquence au centre (position) de la moyenne courante le plus proche
— Déplacement de chaque centre vers le barycentre des observations qui lui sont associés.

Cette méthode est aujourd'hui une des plus utilisée pour le clustering de séries temporelles utilisant la mesure de similarité D.T.W.

4.3.3 Notre proposition : DBA-Ctree-Ucb et DBA-LinUcb

Pour répondre au besoin de l'analyste d'intégrer les séries temporelles dans son algorithme d'allocation dynamique, cette thèse propose les algorithmes DBA-Ctree-Ucb et DBA-LinUcb. Ces algorithmes sont des extensions des algorithmes Ctree-Ucb et Lin-Ucb.

Nous rappelons que l'analyste est supposé disposer d'une variation originale (par exemple la page web A) et des historiques (par exemple des logs) des items ayant été soumis à cette variation originale, collectés avant qu'ils n'arrivent sur la page test. Ces données constituent les données d'apprentissage nécessaires aux algorithmes DBA-LinUcb et DBA-Ctree-Ucb.

La première étape des deux algorithmes proposés ici correspond à l'étape offline d'explicitation des groupes de Ctree-Ucb. La modification consiste à remplacer, dans tous les items connus, chaque série temporelle par un caractéristique catégorielle. Pour cela, les deux méthodes DBA-LinUcb et DBA-Ctree-Ucb déterminent, au préalable de l'apprentissage, les clusters des séries temporelles pour chacune des caractéristiques temporelles. Ces clusters sont définis par leurs centres (appelés *centroïdes*) utilisant la méthode K-means avec D.T.W. et D.B.A. Le modèle de classement est alors effectué sur les items connus dans lesquels chaque série temporelle a été remplacée par le numéro du cluster auquel elle appartient, c'est-à-dire le numéro du cluster dont le centre, dans le clustering correspondant à la caractéristique considérée, est le plus proche (via D.T.W) de la série temporelle.

La deuxième étape de ces deux algorithmes correspond à l'étape online d'exploration de Ctree-Ucb. La modification consiste à remplacer dans tous les items à tester, chaque série temporelle par le numéro du cluster auquel elle appartient. Ces informations sont alors directement utilisables par un modèle de bandit contextuel. Ainsi, dans la version DBA-LinUcb (resp. DBA-Ctree-Ucb) l'algorithme de bandit Lin-Ucb (resp. Ctree-Ucb) utilise cette information pour choisir la variation à affecter.

Le schéma général 3.1 résume notre approche. L'algorithme 6 (resp. 7) détaille les étapes de DBA-Ctree-Ucb (resp. DBA-LinUcb).

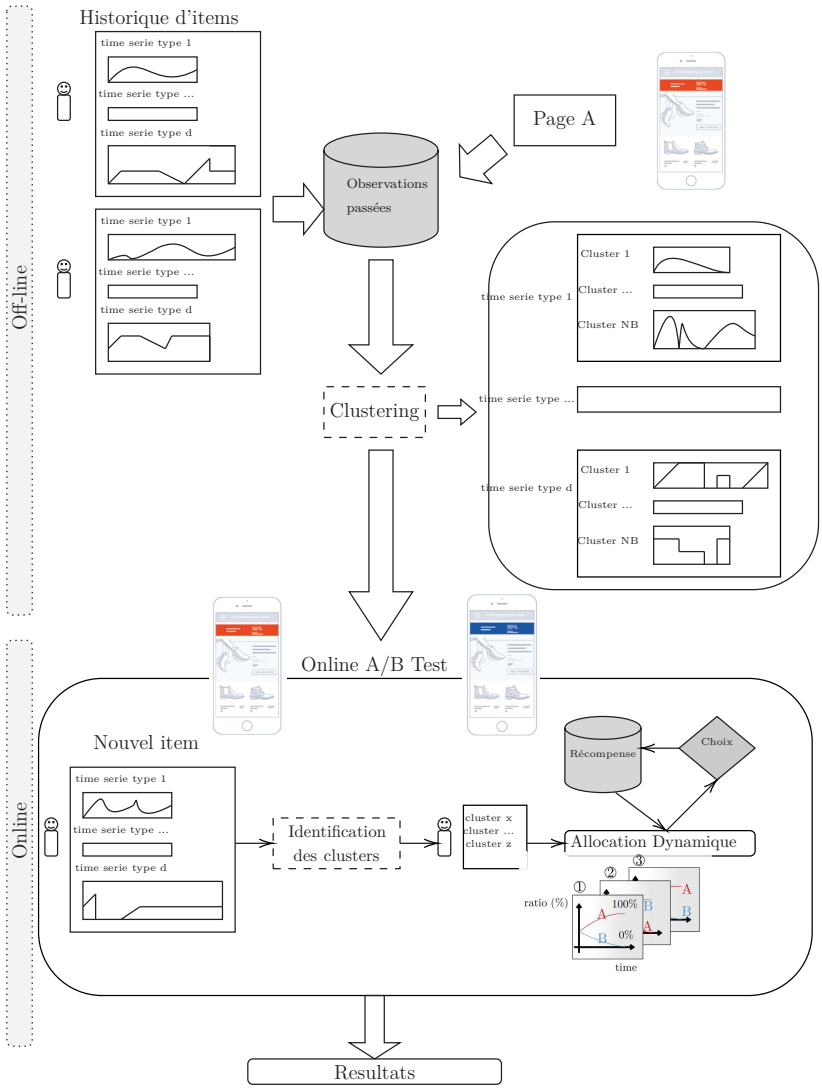


FIGURE 4.4 – Approche offline/online pour intégrer des séries temporelles aux A/B tests

Algorithm 6 Algorithme DBA-Ctree-Ucb

Require: $\alpha \in \mathbb{R}^+$

$\mathcal{L}$; Un jeux d'apprentissage issue de la variation original V_A
Nb_j : un nombre de clusters pour chaque type de série $j \in d$.
$Iter_{cluster}$: un nombre d'itérations.
$\epsilon \in [0, 1]$: un risque d'erreur accépté.

for Chaque type de série temporelles de $j \in d$ associé à $c_t \in \mathcal{L}$ **do**
    for $Iter_{cluster}$ itérations **do**
        Affecter les séries de j à un cluster selon le meilleur alignement (DTW) avec son centroïde
        Calculer les centroïds de chaque cluster
        Mettre à jour les centroïdes selon les séries appartenant aux clusters (D.B.A)
    end for
    Obtenir d centroïds
end for
Associer un cluster pour chaque série j des items $c_t \in \mathcal{L}$
Définir les clusters des items $c_t \in \mathcal{L}$ comme attributs
Générer un modèle d'arbre d'inférence conditionnel :f avec les items $c_t \in \mathcal{L}$ définis par leurs cluster et un risque ϵ accepté
loop
    Reçoit un item c_t (avec d série temporelles)
    Associer un cluster pour chaque série j de l'item c_t
    Définir les clusters de c_t comme ses attributs
    Affecter c_t à un groupe k tel que $f(c_t) = k$
    if $N_{a,k}(t) = 0$ **then**
        $a_{max,k} = a$
    else
        $a_{max,k} = \text{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,k,t} + \alpha \sqrt{\frac{2 * log(\sum_{a \in \mathcal{A}} N_{a,k}(t))}{N_{a,k}(t)}}\}$
Affecter l'item c_t au bras $A_t = a_{max,k}$
Recevoir une récompense X_{c_t,A_t}
Mettre à jour $\hat{\mu}_{A_t,k}$ et $N_{A_t,k}(t)$
end for

Output : Une séquence de choix de bras et de récompenses

Algorithm 7 Algorithme DBA-LinUcb

Require: $\alpha \in \mathbb{R}^+$

$\mathcal{L}$; Un jeu d'apprentissage issue de la variation originale V_A
$Nb_{cluster}$: un nombre de clusters. $Iter_{cluster}$: un nombre d'itération.
for Chaque type de série temporelles de $j \in d$ associé à $c_t \in \mathcal{L}$ **do**
    for $Iter_{cluster}$ itérations **do**
        Affecter les séries de j à un cluster selon le meilleur alignement (DTW) avec son centroïde
        Calculer les centroïds de chaque cluster
        Mettre à jour les centroïdes selon les séries appartenant aux clusters (D.B.A)
    end for
    Obtenir d centroïds
end for
for $a = 1 ; a \leqslant K$ **do**
Initialiser $A_a = I_d$, la matrice identité de taille d par d du bras a
Initialiser $b_a = 0_d$, un vecteur nul de taille $\sum_{j=1}^{d} Nb_j$

end for
loop
    Reçoit un item c_t (avec d série temporelles)
    Associer un cluster pour chaque série d de l'item c_t
    Définir les clusters de c_t comme ses attributs
    for $a = 1 ; a \leqslant K$ **do**
        Calculer les paramètres du bras par une régression rigide $\hat{\theta}_{t,a} = A_a^{-1} b_a$
        Estimer une récompense $\hat{\mu}_{a,t} = c_t^T \hat{\theta}_{a,t} + \alpha \sqrt{{c_t}^T A_a^{-1} c_t}$
    end for
    Choisir le bras A_t avec la récompense estimée $\hat{\mu}_{a,t}$ la plus élevée
    Recevoir une récompense X_{c_t,A_t}
    Mettre à jour $A_{At} = A_{At} + c_t {c_t}^T$
    Mettre à jour $b_{At} = b_{At} + X_{c_t,A_t} c_t$
    end for

Output : Une séquence de choix de bras et de récompenses

4.4 Cadre expérimental

L'expérimentation (sur Ubuntu 17.10, 64 bits.) et le langage (`R`) utilisé restent dans la même configuration que celle des expériences présentées en section 3.3. Pour réaliser le clustering, le framework `dtwclust` [65] est utilisé.

Soit $L = |\mathcal{L}|$ où $\mathcal{L}$ l'ensemble utilisé pour apprendre les clusters et l'arbre de classement. Soit $T_{A/B} = |S_{A/B}|$ où $S_{A/B}$ est le jeu utilisé pour simuler le test A/B. Pour respecter l'hypothèse qu'avant le test A/B l'analyste ne connaît que les récompenses obtenues par la variation originale (appelée ici V_A), le clustering ne peut être construit qu'à partir de l'ensemble des éléments S_A auxquels la variation V_A a été attribuée. Ainsi $\mathcal{L} \subset S_A$.

4.4.1 Les données

Pour tester l'apport des séries temporelles dans l'allocation dynamique et l'intérêt de notre méthode, les performances de DBA-LinUcb et DBA-Ctree-Ucb sont évalués sur plusieurs jeux de données simulées et réelles provenant de A/B Tasty ou de benchmark du domaine. Ces données sont décrites plus en détail dans la section 4.5 portant sur les expériences (en introduction de chacune d'elles).

4.4.2 Confrontation des méthodes

Afin de valider notre hypothèse (utiliser les données temporelles d'un item améliore la stratégie d'allocation dynamique), nous comparons à travers des expérimentations les résultats de DBA-LinUcb et DBA-Ctree-Ucb par rapport aux algorithmes n'utilisant que les moyennes de chaque série temporelle (voir la section 4.3.2).

Les performances de DBA-LinUcb et DBA-Ctree-Ucb sont donc comparées avec trois algorithmes :

- Un algorithme basé sur une stratégie non contextuelle qui n'observe pas le contexte des items avant de choisir la variation :
  - uniform choisit les variations de façon alternative. Cette approche revient à faire un test sans allocation dynamique et ne nécessite aucun paramètre.
- Deux algorithmes basés sur une stratégie contextuelle qui observent le contexte des items avant de choisir la variation :
  - Lin-Ucb qui choisi selon des hypothèses de linéarité entre les moyennes de chaque type de séries temporelles et la récompense.
  - Ctree-Ucb, décrit en section 3.3, qui s'affranchit de l'hypothèse de linéarité, et modèlise la relation entre les moyennes de chaque type de séries temporelles et la récompense par un arbre de classement.

La comparaison d'algorithmes de bandits se fait généralement par rapport au regret (voir section 2.2.3), qui est la différence entre une stratégie oracle (qui connaîtrait les

récompenses de chaque bras en avance) et la stratégie du bandit. Cette stratégie oracle est générée à partir d'un modèle apprenant sur l'intégralité des données. Malheureusement, se comparer à la stratégie oracle est ici impossible car cette stratégie serait différente selon le type de caractéristique utilisée. Dans notre cas Ctree-Ucb et Lin-Ucb n'utilisent que les moyennes des séries, contrairement à DBA-Ctree-Ucb et DBA-LinUcb qui utilisent des clusters. Si ces algorithmes sont comparés avec un modèle utilisant toutes les informations (cluters et moyennes), ces dernier n'ayant qu'une connaissance partielle du contexte par rapport à l'oracle, leur regret cumulé serait nécessairement linéaire. Les méthodes seront donc évaluées par rapport au gain moyen qu'elles ont obtenu à la fin du test.

4.4.3 Étape d'explicitation

L'étape préliminaire d'extraction des clusters et de remplacement des séries temporelles de DBA-LinUcb et DBA-Ctree-Ucb est cruciale car elle impacte tout le processus de test A/B. Nos expérimentations porterons particulièrement sur cette étape (choix du nombre de clusters ...). En particulier, nous étudierons son influence sur le gain moyen à la fin du test. Nous nous intéresserons aussi à l'interprétation des clusters. En effet, il serait intéressant de savoir par exemple, s'il existe un lien avec les clusters trouvés et les patterns introduits précédemment.

Choix du nombre de cluster

Les algorithmes DBA-LinUcb et DBA-Ctree-Ucb nécessitent de fixer un nombre de clusters Nb_j avec $j \in \{1...d\}$ pour chaque type de série. Choisir le bon nombre de clusters est un problème compliqué et bien connu en fouille de données. Pour répondre à ce problème, des indices (à minimiser ou maximiser) permettent d'indiquer à l'analyste la variabilité intra-classe ou inter-classe [4]. En fonction du nombre de clusters, nous considérons les indices suivants :

— Index de silhouette : Définit l'inertie interne en fonction des similarités par paires entre tous les points du cluster et tous les points dans l'autre cluster le plus proche [62]. À maximiser

— La mesure de Davies-Bouldin définit la compacité en fonction de la similarité des points dans le cluster à son centroïde et de la séparation en fonction des similarités entre les centroïdes les plus éloignés [47]. Un ajustement de cet indice a été proposé dans [46] . À minimiser

— La mesure de Calinski-Harabasz définit également l'inertie intra en fonction de la similarité des points d'un cluster à son centroïde et de la séparation en tant que similarité du centroïde de la grappe au centre de données [17]. À maximiser

— L'index de COP : Ratio entre les points d'un cluster et son centroïde [4] . À minimiser
— L'index de Dunn : Rapport entre la similarité maximale qui sépare deux éléments classés ensemble et la similarité minimale qui sépare deux éléments classés séparément. C'est un indice qui ne repose pas sur une similarité particulière et qui peut donc être utilisée dans une grande variété de situations [22]. À maximiser
— Le score : Basée sur une fonction exponentielle qui prend en compte l'inertie intra et inter classe des clusters [63]. À maximiser.

	Description	Critère
Sil	Silhouette index	max
SF	Score Function	max
CH	Calinski-Harabasz index	max
DB	Davies-Bouldin index	min
DBstar	Modified Davies-Bouldin index	min
D	Dunn index	max
COP	COP index	min

TABLE 4.1 – Résumé des indices considérés pour l'étape d'explicitation des groupes.

La table 4.1 résume les indices que nous prendrons en considération dans nos expérimentations. Ainsi via la méthode du coude, l'analyste sera plus à même de choisir le nombre de clusters. Dans nos expérimentations, nous choisissons le nombre de cluster en fonction des indices précédemment cités. Nous testerons donc plusieurs configurations. En fonction du gain obtenu à la fin du test, nous observerons quel indice aurait permis à l'analyste de choisir le bon paramétrage. Par ailleurs, dans toutes les expérimentations, nous représentons les séquences moyennes de chaque cluster pour un nombre Nb_j choisi. Nous résumerons le choix du nombre de cluster dans un tableau comparatif. Si un indice ne permet pas d'identifier un nombre de cluster à choisir, le symbole $\oslash$ sera utilisé dans ce tableau.

Clusters Cette étape étant dépendante des données utilisées en pré-collecte, nous avons aussi étudié également l'influence de la taille du jeu d'apprentissage et de test sur les résultats. Deux configurations ont été testées :

— $\text{Conf}_{30,70}$: $\mathcal{L} = 30\%$ de V_0, $S_{A/B} = \sum_i \{70\% \text{ de } S_i\}$
— $\text{Conf}_{100,100}$: $\mathcal{L} = V_0$, $S_{A/B} = \sum_i S_i$

Les séries temporelles étant collectées avant l'allocation, l'apprentissage des clusters ($\text{Conf}_{\text{clust})}$ est testé pour les configurations :

— $\text{Conf}_{\text{clust30,70}}$: $\mathcal{L}_c = 30\%$ de $\sum_i V_i$, $S_{A/B} = \sum_i \{70\% \text{ de } S_i\}$
— $\text{Conf}_{\text{clust100,100}}$: $\mathcal{L}_c = \sum_i V_i$, $S_{A/B} = \sum_i S_i$

Pour chaque type de série temporelle, l'impact du nombre de clusters Nb_j avec $j \in \{1, \ldots, d\}$, défini en amont du test, sera étudié.

Les algorithmes DBA-Ctree-Ucb et Ctree-Ucb nécessitent un paramètre supplémentaire ϵ, dont la sensibilité a été étudiée en section 3.3 et ne sera pas traitée ici (la valeur $\epsilon = 0,05$ sera choisie pour toutes les expériences). Ces algorithmes identifient des groupes à partir de la page originale A. Pour les N variations $V_0, V_1, \ldots, V_N$. S_i est l'ensemble des items auxquels la variation V_i a été affectée, $T_i = |S_i|$ et $T = T_0 + T_1 + \cdots + T_N$ où $|.|$ désigne la cardinalité de cet ensemble. Pour chaque ensemble de données, ont été considéré chaque variation comme une variation originale potentielle

Après la phase d'exploration, nous avons étudié sur l'ensemble des données les gains moyen des groupes identifiés par l'arbre de classement. Pour cela, nous générons deux modèles avec CTREE (l'un avec les clusters, l'autre avec les moyennes des séries). Le premier modèle sera réutilisé pour fournir une interprétation des groupes et en donner une signification métier en les associant dans un tableau comparatif aux patterns proposés par Roukins [61]. Nous ajoutons également la variation appliquée (A ou B) pour observer si une différence significative entre A et B est détectée par un test statistique.

4.4.4 Étape d'exploration de l'AB Test

Chaque algorithme produit pour chaque jeux de données une séquence de choix. Ces séquences sont comparées par rapport à leur moyenne de gains [1]. Cette évaluation diffère de la méthode d'évaluation présentée en section 3.3, reposant sur la notion de regret. En effet, le regret cumulée, défini en section 2.2.3, correspond à les choix d'une stratégie oracle (qui apprendrait sur toutes les données) et ceux de l'algorithme de bandit utilisés. Or, les oracles sont entraînés avec des caractéristiques différentes (cluster ou moyenne de série temporelle). Évaluer le regret cumulé n'est plus possible puisque les oracles produirons des choix différents. Nous nous intéresseront donc au gain moyen, qui, permet de comparer toutes les approches de la même façon selon l'objectif du test (c'est à dire maximiser les gains). Les garanties théoriques de toutes des algorithmes DBA-Ctree-Ucb (resp. DBA-LinUcb) restent identique à celles à Ctree-Ucb (resp. Lin-Ucb) puisque le type de contexte n'intervient pas dans la définition de la borne limitative du regret cumulé.

Pour chaque expérience, un tableau résume les gains moyens de chaque algorithme. Les meilleures performances (gains moyen) apparaissent en gras dans ces tableaux.

1. Dans ce contexte, le gain moyen est le *taux de clic* (C.T.R.) obtenu.

4.5 Analyse des résultats

La Tab. 4.2 résume tous les paramètres et leurs différentes valeurs observées.

Data set	Algorithms	Parameters
Localization Data Jeu de données AB Tasty 7 Jeu de données AB Tasty 8	Lin-Ucb	$\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$
	Ctree-Ucb	$V_A \in \{V_i\}$ Config. $\in \{\text{Conf}_{30,70}, \text{Conf}_{100,100}\}$ $\epsilon \in \{0.01, 0.05, 0.1\}$ $\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$
	DBA-LinUcb	$\text{Config.}_{\text{clust}} \in \{\text{Conf}_{\text{clust30,70}}, \text{Conf}_{\text{clust100,100}}\}$ $\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$ $Nb_j \in \{2, \ldots, 20\}$
	DBA-Ctree-Ucb	$V_A \in \{V_i\}$ Config. $\in \{\text{Conf}_{30,70}, \text{Conf}_{100,100}\}$ $\text{Config.}_{\text{clust}} \in \{\text{Conf}_{\text{clust30,70}}, \text{Conf}_{\text{clust100,100}}\}$ $\epsilon \in \{0.01, 0.05, 0.1\}$ $\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$ $Nb_j \in \{2, \ldots, 20\}$

Table 4.2 – Paramètres des algorithmes

4.5.1 Données simulées

a) Les données

Pour tester notre méthode, dans une premier temps, nous nous sommes placés dans le cas idéal où nous disposerions, pour chaque item, des récompenses obtenues selon les différentes variations. Or, de telles données réelles sont très rares. Ainsi, nous avons été amenés à simuler des données dans lesquelles, pour chaque item, les récompenses sont connues pour les deux variations.

Pour correspondre à notre objectif applicatif (l'AB Testing dans le web), nous souhaitons que cette simulation simple représente les patterns présents dans l'e-marketing, qui ont été proposés par Roukins [61] en section 1.3, c'est-à-dire que la récompense dépend à la fois du nombre de visites et du comment ces visites sont réparties dans le temps.

Pour cela, pour chaque item généré, son nombre de visites est tiré aléatoirement (entre 1 et 100) via une fonction cosinus. Puis pour chaque visite, une valeur (entre 1 et 100, qui pourrait correspondre au temps passé sur le site) est aléatoirement choisie. Enfin, une récompense est générée pour les deux variations en fonction de son nombre de visites et le temps maximal obtenu (voir tableau 4.3) où Q_1 et Q_3 sont respectivement le 1er et 3e quartile du nombre total de visites calculés à partir de tous les items et V_{max} est la valeur maximale dans la série ainsi générée (ce qui correspondrait au temps maximal que l'item a passé sur le site).

Nous avons généré ainsi 1650 items pour l'apprentissage des clusters et la construction de l'arbre de classement ($\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}}$) et 25850 items pour la phase d'explication

et d'exploration.

Label	Nombre de visites	V_max	μ_A	μ_B
Arrivé par erreur	<Q1	<20	1	4
Prospect parfait	<Q1	>70	25	20
Observateur	>Q1	20<max<70	3	2
Indécis	>Q3	>70	7	8
Autre			0.3	0.3

TABLE 4.3 – Récompense moyenne simulées selon le profil de l'item pour représenter un lien entre une caractéristique temporelle et une valeur d'achat espérée selon les patterns d'e-marketing.

b) Étape d'explicitation

La Fig. 4.5 montre les centroïdes identifiés lors de l'extraction des clusters par K-Means ($Nb_y = 10$)

L'arbre de classement de la phase d'apprentissage de DBA-CTREE-UCB est présenté dans la Fig. 4.6. Nous identifions 5 groupes avec des distributions de gains statistiquement différentes. Les items maximisant les récompense représentés par la feuille Node#4 dans l'arbre de classification, sont ceux dont la composante temporelle appartient aux clusters #6 et #7. La plus faible moyenne de gain représentés par la feuille Node#6 correspond à des items ayant leur composante temporelle associée aux clusters #3,#5, #8, et #9.

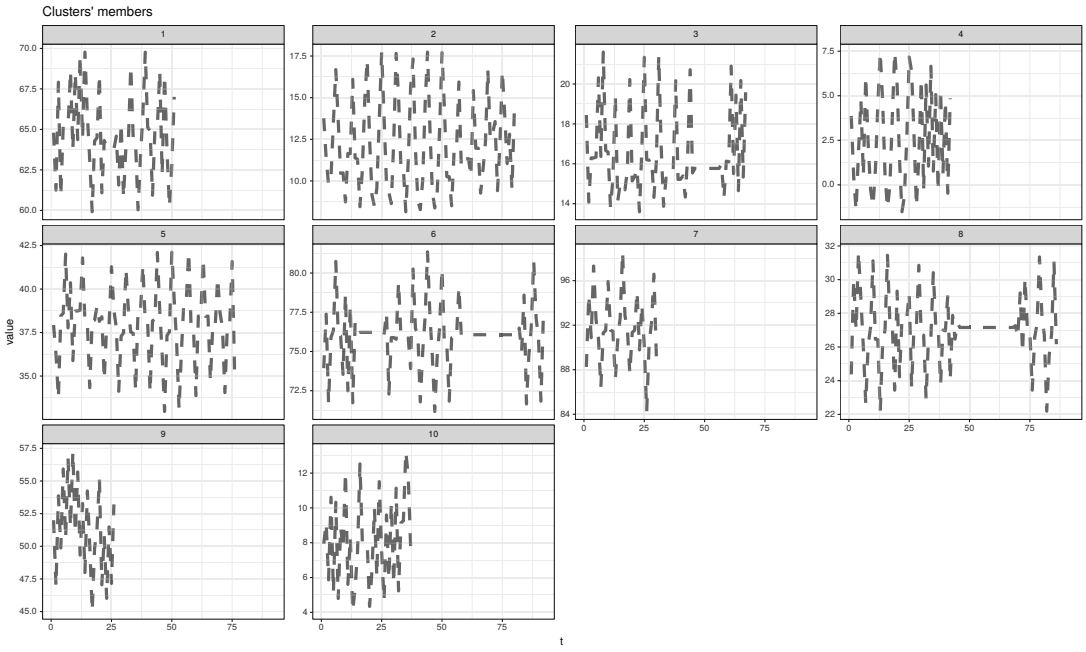


FIGURE 4.5 – Centroïdes des clusters identifiés dans la phase d'apprentissage. Les clusters sont appris sur 30% des données originales.

L'arbre de classement de la phase d'apprentissage de CTREE-UCB est présenté dans la Fig. 4.7. Trois groupes sont identifiés avec des distributions de gains statistiquement

différentes. Les items maximisant les récompenses sont ceux ayant une moyenne de durée de visite de 65.852 (Node#5). La plus faible moyenne de gain est donnée pour des items ayant une moyenne de visite de comprose entre 15.05 et 65.8565 (Node#4)

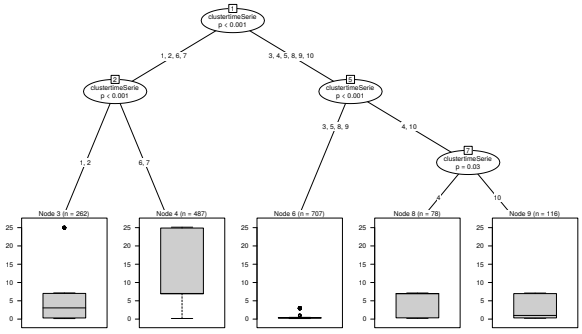


FIGURE 4.6 – Arbre de classement généré dans la phase d'apprentissage de DBA-CTREE-UCB des données simulées, à partir des récompenses données par la variation A (Config.$_{30,70}, V\_A : A, \epsilon = 0.05$).

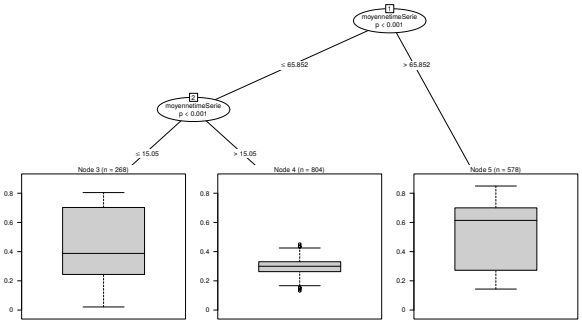


FIGURE 4.7 – Arbre de classement généré dans la phase d'apprentissage de CTREE-UCB des données simulées, à partir des récompenses données par la variation A (Config.$_{30,70}, V\_A : A, \epsilon = 0.05$).

c) Phase d'exploration de l'AB Test

Le gain cumulé des différents algorithmes au cours du temps est représenté dans la figure 4.8. Cette figure montre très clairement la supériorité des méthodes DBA-CTREE-UCB et DBA-CTREE-UCB par rapport aux approches traditionnelles n'utilisant que la moyenne de la série.

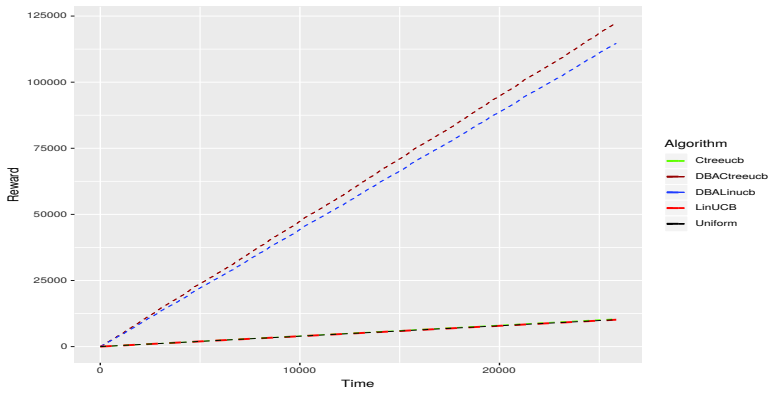


FIGURE 4.8 – Gain cumulé au cours du temps

4.5.2 Jeu de données AB Tasty 7

a) Les données

Nous utilisons dans nos expérimentations deux jeux de données collectées à partir de différents A/B tests sans allocation dynamique réalisés par AB Tasty sur des sites marchands en 2018. Ces jeux de données ont été sélectionnés pour leurs propriétés différentes : le premier jeu de données a une différence statistiquement significative entre les moyennes de A et B, pour le second, un test statistique traditionnel (test du Chi2) n'a pas permis de mettre en évidence une différence entre les moyennes de gain. En d'autre termes, les différences de gains par rapport à une stratégie UNIFORM seront plus ou moins visibles en fonction du jeux de données étudié. La collecte des caractéristiques évolutives étant encore difficile à réaliser techniquement pour la société AB Tasty, nous n'avons pas eu à notre disposition de jeux de données supplémentaires.

Les données incluent des historiques associés aux parcours d'achat des différents visiteurs qui ont navigué sur le site web et ont vu la page test concernée au moins une fois. Chacun d'entre eux est identifié par un identifiant unique : la première fois qu'il arrive sur le site un identifiant lui est généré. Lorsqu'il arrive sur la page test, une variation (A ou B) lui est attribuée (aléatoirement). Chaque fois qu'un visiteur revient sur la page testée, la même variation lui est affichée (irrévocabilité de l'allocation). Si un visiteur achète après avoir vu la page test, la récompense est de 1 et 0 sinon.

De sa première visite jusqu'à son arrivée sur la page test, sa navigation est enregistrée. L'analyste associe à chaque visiteur trois séries temporelles, de taille identique, égale au nombre de jours entre sa première et dernière visite avant de voir la page test :

- `presence_time_serie` : constituée de valeurs binaires. Pour chaque jour où le visiteur s'est rendu sur le site, une valeur 1 est définie, 0 s'il ne s'y est par rendu. Chaque série commence et finit donc par la valeur : 1. Le choix d'une observation "par jour" s'est justifiée par les expérimentations empiriques, les observations à une échelle plus fine (par exemple par heure) étant difficilement interprétables par l'analyste car elles génèrent des séries de très grandes tailles.
- `connexion_time_serie` : constituée de valeurs entières comprises dans $\{0, 1, 2, .., 23, 24\}$. Pour chaque jour où le visiteur s'est rendu sur le site, une valeur égale à son heure d'arrivée sur le site est définie, 0 s'il ne s'y est pas rendu. Si le visiteur vient plusieurs fois dans la même journée, c'est la première heure qui est sauvegardée.
- `time_spend_serie` : constituée de valeurs comprises dans $\mathbb{R}$. Pour chaque jour où le visiteur s'est rendu sur le site, une valeur égale à sa durée d'activité moyenne pendant sa visite (en microseconde) sur le site est définie, 0 s'il ne s'y est pas rendu sur le site.

Dans ces jeux de données AB Tasty, une seule récompense par item est connue. Pour évaluer les algorithmes précédemment sans avoir à remplacer les données manquantes,

nous appliquons un mécanisme de *rejection sampling* : lorsque l'algorithme choisit une variation qui n'a pas été réellement assignée à un visiteur, l'algorithme ignore ce visiteur pour passer au suivant. Bien que cette méthode permette de s'affranchir du remplacement des valeurs manquantes, elle peut diminuer drastiquement la taille de l'échantillon réellement testée.

Pour pallier le manque d'items, nous avons donc été amenés, dans certains cas, à augmenter artificiellement le jeu de données en le dupliquant afin de limiter l'influence du *rejection sampling* sur les résultats dans la phase d'exploration du test.

Ces données proviennent d'un site de vente dans le prêt à porter. L'objectif de l'AB Test est d'augmenter le taux de clics de la page testée. Pour apprendre les clusters et les groupes nous testons plusieurs configurations. L'une utilise 30% des visiteurs (3416) et l'autre utilise la totalité des visiteurs (11389).

b) Explicitation des groupes

Dans un premier nous avons déterminé le nombre Nb optimal pour chaque type de série à l'aide des critères de qualité usuelle utilisés en clustering de séries temporelles (voir section 4.4.3). La Tab. 4.4 présente ces critères sur l'ensemble des données associées à la série `time_spend_serie`. Les différentes valeurs des Nb qui semblent le plus pertinent selon ces indices sont présentés en gras dans le tableau. Ces indices indique un nombre $Nb_{\texttt{time\_spend\_serie}} \in \{3, 4, 5\}$ pour le critère de Silhouette (Sil), Calinski-Harabasz (CH), Davies-Bouldin simple (DB) et corrigé (DBStars), ou bien $Nb_{\texttt{time\_spend\_serie}} \in \{12, 14, 15, 16, 17\}$ si l'on privilégie l'indice de COP. L'indice Dunn (D) et la fonction de score (SF) ne renseigne aucune indication sur le $Nb_{\texttt{time\_spend\_serie}}$ à choisir, c'est-à-dire qu'aucune Nb se distingue particulièrement selon ces indices. En Annexe 6, est présentée une évaluation de ces indices sur un apprentissage intégral des données (c'est-à-dire sous la $\text{Conf}_{\text{clust100,100}}$). Les indices présentent sensiblement les mêmes résultats pour une $\text{Conf}_{\text{clust30,70}}$ ou une $\text{Conf}_{\text{clust100,100}}$. Les résultats selon ces indices pour les deux autres types de séries sont présentés dans la Tab 4.5 et la Tab. 4.6 La Tab. 4.7 rapporte les Nb optimaux selon les indices considérés.

Nb	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	8	19	20
Sil	**0.94**	0.92	0.91	0.89	0.90	0.86	0.85	0.81	0.84	0.80	0.70	0.81	0.82	0.79	0.78	0.70	0.74	0.75
SF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CH	**1952.67**	504.12	625.54	259.84	425.04	308.10	334.51	464.04	489.23	393.76	401.23	412.51	389.52	384.69	275.94	394.03	319.96	344.36
DB	1.30	**0.86**	1.54	1.66	1.56	1.71	1.62	1.79	1.46	1.96	1.41	1.64	1.80	1.77	1.77	1.71	3.24	1.63
DBstar	2.04	**1.57**	1.69	2.82	2.61	2.78	3.75	4.11	4.13	3.87	5.29	4.73	5.59	5.02	5.99	5.70	7.62	4.58
D	0.02	0.08	0.01	0.02	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
COP	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.02	0.02	0.02	**0.01**	**0.01**	**0.01**	**0.01**

TABLE 4.4 – Évaluation de différents indices de qualité des clusters sur la série `time_spend_time_serie` du jeu de données AB Tasty 7, avec la méthode de clustering D.B.A. ($\text{Conf}_{\text{clust30,70}}$). Les valeurs suggérées par les indices sont définies en gras.

Nb	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	8	19	20
Sil	0.72	0.82	0.91	0.94	0.91	0.96	0.91	0.96	**1.00**	0.99	0.99	0.99	0.99	0.99	**1.00**	0.99	0.99	0.99
SF	0.14	0.29	0.42	0.30	0.30	0.39	0.29	0.46	**0.54**	0.41	0.47	0.45	0.41	0.46	0.43	0.45	0.46	0.48
CH	9525.24	16083.45	23105.40	25858.17	9923.45	22940.25	7703.61	32409.46	**363045.71**	58316.76	50973.33	42868.30	60710.39	44209.85	115656.96	40320.92	56911.11	62829.10
DB	0.82	0.61	**0.33**	0.39														
DBstar	1.10	0.92	0.99	**0.71**	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
D	**0.00**	**0.00**	0.25	0.50	0.25	0.33	0.25	0.33	1.00	0.50	0.50	0.50	0.50	0.50	1.00	0.50	0.50	0.50
COP	0.36	0.10	0.06	0.03	0.05	0.02	0.05	0.02	**0.00**	0.01	**0.00**	**0.00**	0.01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

TABLE 4.5 – Évaluation de différents indices de qualité des clusters sur la série `presence_time_serie` du jeu de données AB Tasty 7, avec la méthode de clustering D.B.A. ($\text{Conf}_{\text{clust30,70}}$). Les valeurs suggérées par les indices sont définies en gras.

Nb	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	8	19	20
Sil	**0.38**	0.35	0.23	0.24	0.17	0.22	0.24	0.14	0.17	0.16	0.15	0.13	0.14	0.12	0.13	0.13	0.10	0.15
SF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CH	**4037.60**	1816.72	1846.03	1065.67	1010.60	812.10	862.74	1001.63	737.71	777.49	551.58	994.14	878.52	1043.83	425.85	545.27	624.64	603.50
DB	**1.50**	1.80	1.54	1.97	1.73	2.13	2.08	2.32	2.35	2.34	1.99	2.22	2.48	2.39	2.18	2.48	2.25	2.17
DBstar	**2.28**	2.37	2.48	2.89	2.52	2.65	3.17	4.68	3.50	3.94	3.27	5.10	5.04	4.56	4.64	5.01	5.68	4.75
D	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
COP	0.24	0.22	0.22	0.18	0.19	0.18	0.17	0.17	0.17	0.16	0.17	0.16	**0.14**	0.15	0.15	0.15	**0.14**	**0.14**

TABLE 4.6 – Évaluation à travers différents indices de qualité des clusters sur la série `connexion_time_time_serie` du jeu de données AB Tasty 7, avec la méthode de clustering D.B.A. ($\text{Conf}_{\text{clust30,70}}$). Les valeurs suggérées par les indices sont définies en gras.

	$Nb_{presence\_time\_serie}$	$Nb_{time\_spend\_serie}$	$Nb_{connexion\_time\_serie}$
Sil	11,17	3	3
SF	11	⊘	⊘
CH	11	3	3
DB	5	4	3
DBstar	6	4	3
D	11,17	4	⊘
COP	11,13,16,17,18,19,20	13,18,19	15,19,20

TABLE 4.7 – Synthèse du nombre de clusters selon différents indices qualité, avec la méthode de clustering D.B.A. ($\text{Conf}_{\text{clust30,70}}$)

Interprétabilité des clusters Nous observons ici centroïdes (séquence moyennes) des clusters selon deux configurations : les clusters sont appris sur 30% ou 100% de la base de données.

Sont présentés graphiquement dans cette section les centroïdes des clusters sous la configuration $Nb_{\texttt{presence\_time\_serie}} = 17$, $Nb_{\texttt{time\_spend\_serie}} = 4$ et $Nb_{\texttt{connexion\_time\_serie}} = 15$ à travers deux configurations :

— La première configuration est basée sur la configuration $\text{Conf}_{\text{clust30,70}}$ (voir Fig4.9, Fig. 4.11 et Fig. 4.13)

— La seconde configuration est basée sur la configuration $\text{Conf}_{\text{clust100,100}}$ (voir Fig4.10, Fig. 4.12 et Fig. 4.14).

En accord avec les *pattens* décrits par Roukins, dans la Fig. 4.9, avec une configuration $\text{Conf}_{\text{clust30,70}}$, nous pouvons identifier que les comportements d'indécis sont par exemple représentés par les clusters #3, #9 ou #14 de la série `presence_time_serie` (voir Fig.4.9). En effet, ces groupes font des visites de plus en plus rapprochées. En revanche, nous pouvons supposer qu'avec des visites plus espacées, comme celles représentées par clusters #6, #12 ou #13 de la série `presence_time_serie` les visiteurs sont des acheteurs po-

tentiels prenant le temps de la réflexion. A contrario, les visiteurs appartenant au groupe #17 de la série `presence_time_serie` ne sont, soit pas intéressés par le site, soit déjà engagés dans leurs processus d'achat.

La Fig. 4.10 montre les clusters identifiés via une configuration $\text{Conf}_{\text{clust}100,100}$. Les clusters de la séries `connexion_time_serie` permettent d'avoir une idée sur les horaires de connexion. Des visiteurs appartenant aux clusters #7 ou #8 par exemple se rend régulièrement vers 15h sur le site : il est intéressant d'observer les combinaisons de clusters pour chaque visiteur afin de différencier un visiteur arrivé par erreur d'un prospect parfait. En effet les prospects parfaits et ceux qui sont arrivés là par hasard sur le site sont difficilement distinguables en observant uniquement leurs parcours dans la série `presence_time_serie`. Leurs visites sont rapides et ils ne reviennent pas forcément sur le site. S'il l'on regarde en revanche le temps passé sur le site `time_spend_serie`, il est possible de distinguer ces deux profils (un visiteur arrivé par erreur quittera immédiatement le site). De la même façon, les acheteurs potentiels et les indécis peuvent revenir à la même fréquence sur le site. Pour les distinger, l'analyste peut supposer que ceux qui reviennent à la même heure régulièrement peuvent être considérés comme ceux visitant le site pour le loisir [61].

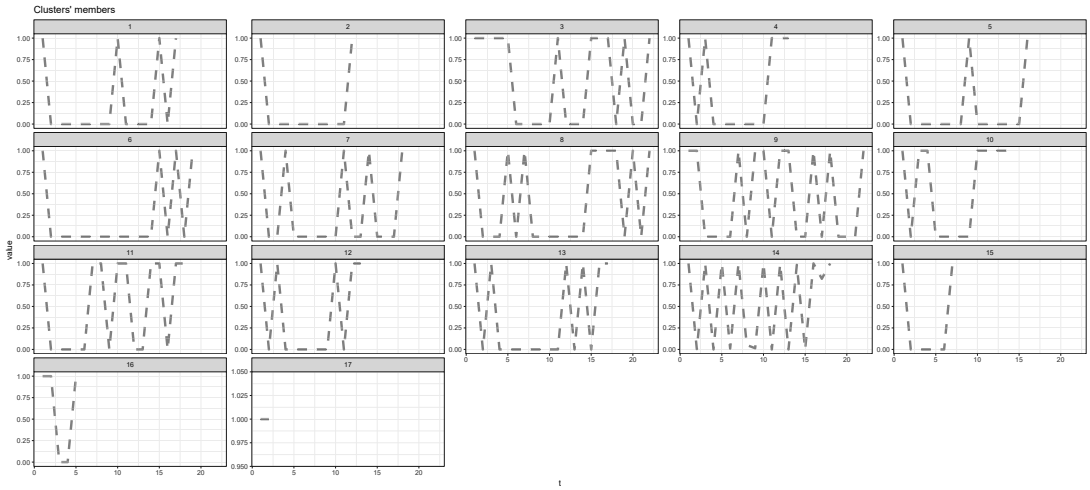


FIGURE 4.9 – Centroïdes des $Nb = 17$ clusters issus des séries `presence_time_serie`, $\text{Conf}_{\text{clust}30,70}$

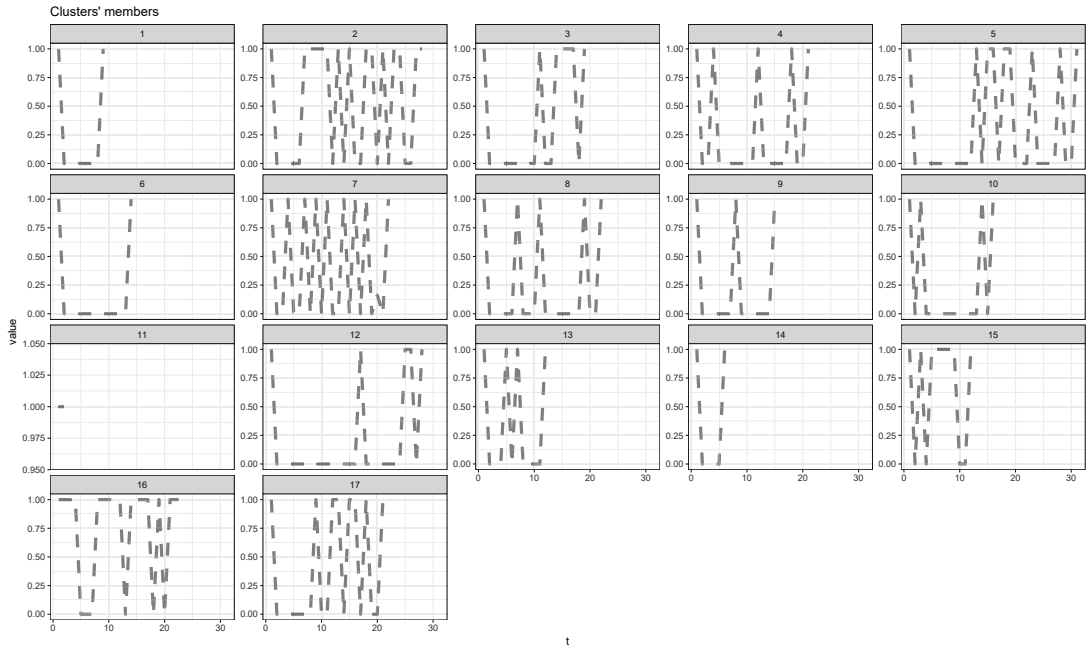


FIGURE 4.10 – Centroïdes des $Nb = 17$ clusters issus des séries `presence_time_serie`, $\text{Conf}_{\text{clust}100,100}$

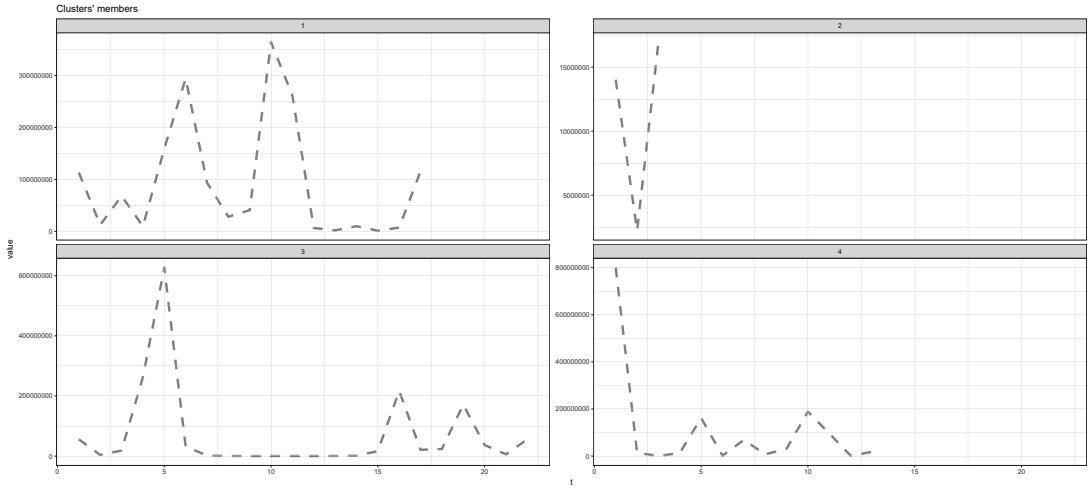


FIGURE 4.11 – Centroïdes des $Nb = 4$ clusters issus des séries `time_spend_serie`, $\text{Conf}_{\text{clust}30,70}$

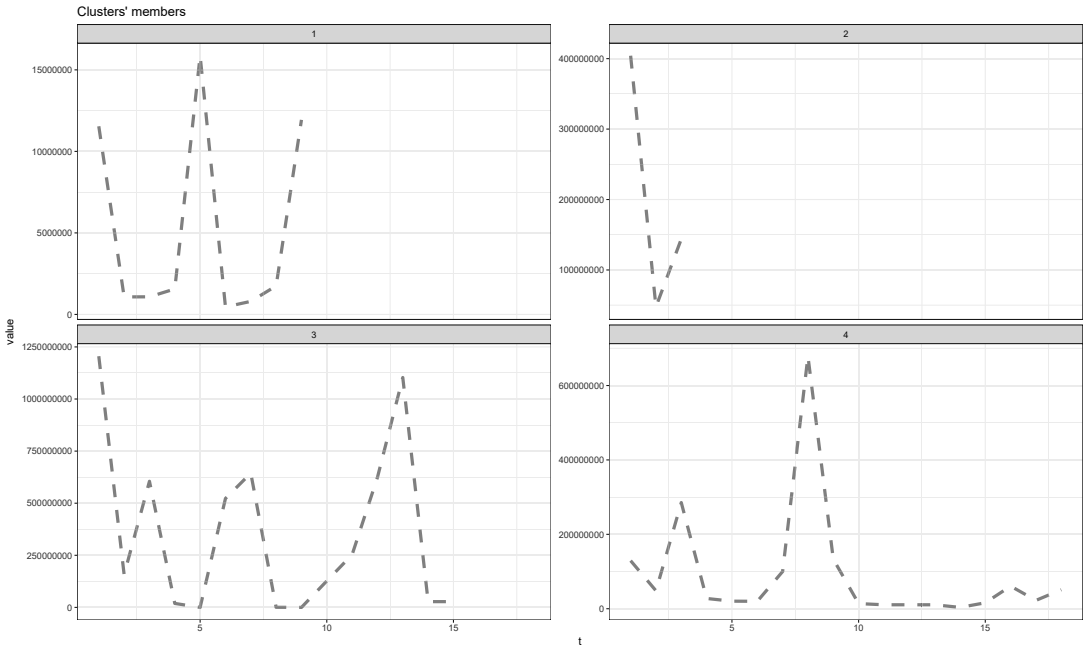


FIGURE 4.12 – Centroïdes des $Nb = 4$ clusters issus des séries `time_spend_serie`, $\text{Conf}_{\text{clust100,100}}$

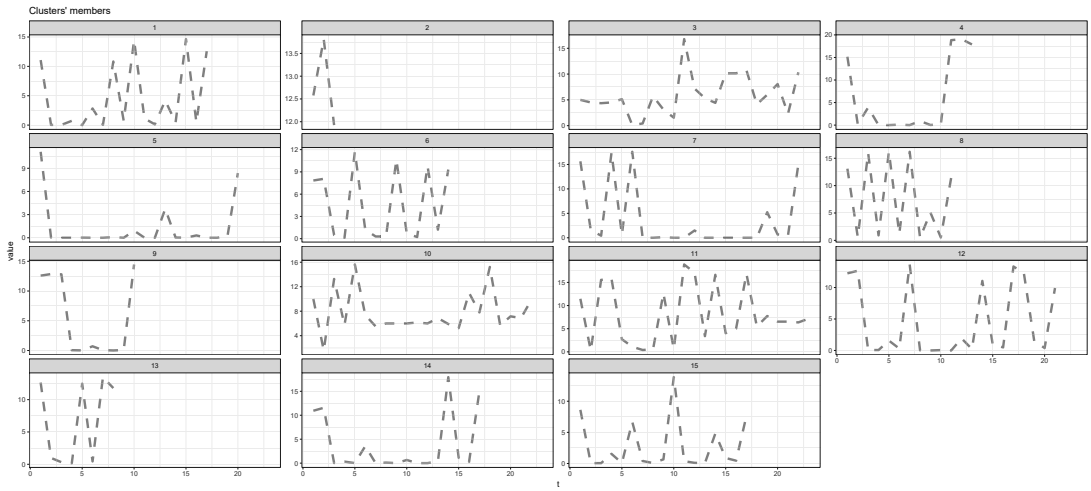


FIGURE 4.13 – Centroïdes de $Nb = 15$ clusters issus des séries connexion_time_serie, $\text{Conf}_{\text{clust30,70}}$

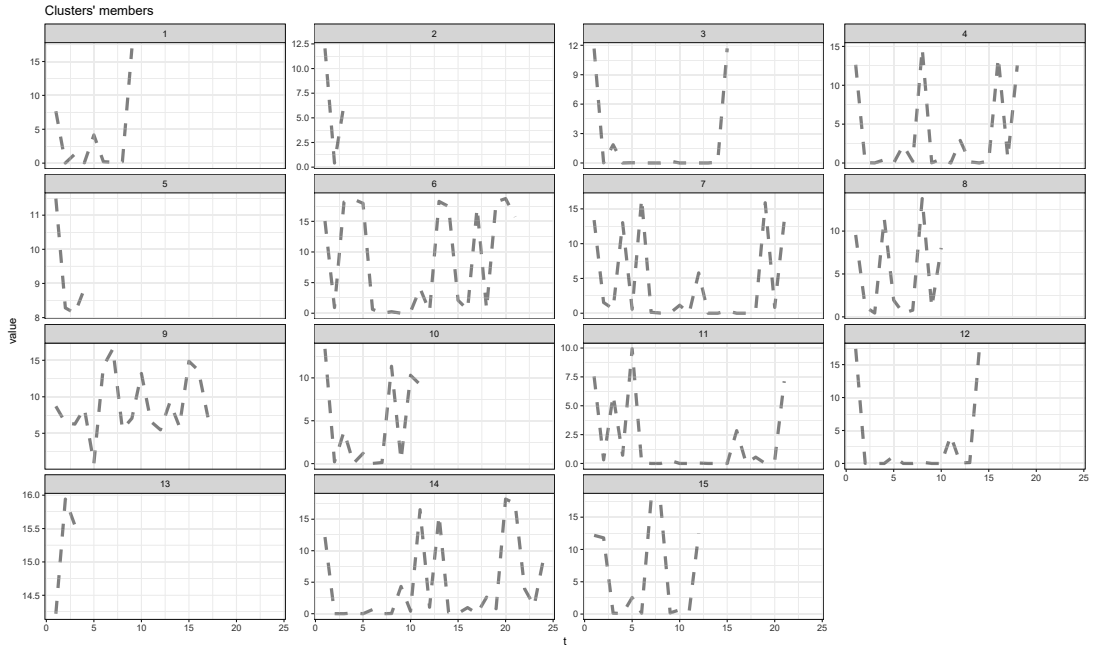


FIGURE 4.14 – Centroïdes de $Nb = 15$ clusters issus des séries connexion_time_serie, $\text{Conf}_{\text{clust}100,100}$

c) Étape d'exploration dans l'A/B test (Allocation dynamique)

Pour des raisons de lisibilité et d'interprétation des résultats, nous scindons en deux le compte rendu de nos expérimentations. La première partie compare DBA-Ctree-Ucb et Ctree-Ucb, la deuxième partie compare DBA-LinUcb et Lin-Ucb. Nous comparons ici les gains moyens (taux de clics) des algorithmes précédemment cités pour observer comment le nombre de clusters fait varier les résultats.

DBA-Ctree-Ucb vs Ctree-Ucb : Le taux de clics obtenu avec uniform (c'est-à-dire sans allocation dynamique) simulé à partir des données originales est de 11,74%. Dans les tests nous faisons varier le nombre de clusters, pour observer la sensibilité de DBA-Ctree-Ucb à ce paramètre et nous dupliquons progressivement la taille de l'échantillon pour éviter la variance induite par le *rejection sampling* (voir la section 4.5.2) .

La Tab. 4.8 présente de sensibilité de notre algorithme aux différentes configurations du nombre de clusters. On voit que les taux de clics associés à DBA-Ctree-Ucb dépassent systématiquement ceux de Ctree-Ucb n'utilisant que la moyenne des séries. Cependant, les résultats les plus élevés de DBA-Ctree-Ucb présentent une variance importante due au *rejection sampling*. Des résultats complémentaires portant sur plus d'items sont donc présentés dans la Tab. 4.9 (duplication par 2) et la Tab.4.10 (duplication par 5). En conclusion, le paramétrage : $Nb_{\texttt{presence\_time\_serie}} = 17$, $Nb_{\texttt{time\_spend\_serie}} = 3$ et $Nb_{\texttt{connexion\_time\_serie}} = 3$ semble être optimal que ce soit avec ou sans duplication du jeu de données.

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-Ctree-Ucb Nb cluster	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
5 ;3 ;3	13,34%	13,56%	12,94%	12,88%
5 ;4 ;15	13,96%	14,12%	11,90%	12,23%
11 ;3 ;3	14,15%	14,10%	14,62%	14,58%
11 ;4 ;15	13,85%	13,35%	13,85%	13,58%
17 ;3 ;3	14,23%	**15,46%**	**15,27%**	**15,34%**
17 ;4 ;3	**14,26%**	13,60%	14,87%	13,73%
Ctree-Ucb	11,40%	11,71%	12,24%	12,77%
uniform	11,74%	11,74%	11,74%	11,74%

Table 4.8 – Sensibilité au clustering pour le jeu de données AB Tasty 7 par l'algorithme DBA-Ctree-Ucb et comparaison avec Ctree-Ucb

DBA-LinUcb vs Lin-Ucb : Par analogie avec les expérimentations précédentes, les résultats de DBA-LinUcb et Lin-Ucb (n'utilisant que la moyenne) sont reportés en augmentant progressivement la duplication répété des items (voir Tab4.11, Tab4.12 et Tab.4.13). Les taux de clics associés à DBA-LinUcb dépassent Lin-Ucb si le bon

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-Ctree-Ucb Nb cluster	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
5 ;3 ;3	11,91%	13,48%	13,06%	13,03%
5 ;4 ;15	12,73%	12,74%	13,05%	12,17%
11 ;3 ;3	**14,30%**	**14,26%**	12,60%	12,75%
11 ;4 ;15	13,63%	13,27%	**13,85%**	**13,58%**
17 ;3 ;3	13,90%	13,80%	12,92%	13,02%
17 ;4 ;15	13,78%	13,26%	12,97%	13,33%
Ctree-Ucb	11,79%	11,93%	12,27%	12,62%

Table 4.9 – Sensibilité au clustering pour le jeu de données AB Tasty 7 par l'algorithme DBA-Ctree-Ucb et comparaison avec Ctree-Ucb, jeu de données dupliqué par 2 pour la phase de test

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-Ctree-Ucb Nb cluster	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
5 ;3 ;3	12,76%	12,31%	12,58%	**12,59%**
5 ;4 ;15	12,53%	12,47%	**12,61%**	**12,59%**
11 ;3 ;3	12,49%	12,45%	12,55%	12,57%
11 ;4 ;15	12,23%	12,14%	12,49%	12,46%
17 ;3 ;3	**12,76%**	**12,49%**	12,53%	12,48%
17 ;4 ;15	12,46%	12,43%	12,41%	12,76
Ctree-Ucb	12,28%	12,15%	12,28%	12,14%

Table 4.10 – Sensibilité au clustering pour le jeu de données AB Tasty 7 par l'algorithme DBA-Ctree-Ucb et comparaison avec Ctree-Ucb, jeu de données dupliqué par 5 pour la phase de test

nombre de clusters est choisi en amont. Cependant, les plus mauvaises performances de DBA-LinUcb restent très proches de celles obtenues par Lin-Ucb. Comme pour DBA-Ctree-Ucb le paramétrage : $Nb_{\texttt{presence\_time\_serie}} = 17$, $Nb_{\texttt{time\_spend\_serie}} = 3$ et $Nb_{\texttt{connexion\_time\_serie}} = 3$ semble être optimal à long terme. En revanche, l'apprentissage des clusters sur une base partielle ($\text{Conf}_{\text{clust}30,70}$) surpasse légèrement le gain obtenu par Lin-Ucb. Notre hypothèse est que DBA-LinUcb est très dépendant du nombre de clusters et qu'un nombre de clusters important implique un apprentissage plus long en raison de la modélisation empirique des paramètres de la fonction linéaire.

DBA-LinUcb Nb cluster	$Conf_{clust30,70}$	$Conf_{clust100,100}$
5 ;3 ;3	11,93%	12,01%
5 ;4 ;15	**12,77%**	**12,66%**
11 ;3 ;3	12,45%	12,06%
11 ;4 ;15	12,51%	12,07%
17 ;3 ;3	12,20%	12,59%
17 ;4 ;15	12,50%	12,17%
Lin-Ucb	12,57%	12,57%
uniform	11,74%	11,74%

Table 4.11 – Sensibilité au clustering pour le jeu de données AB Tasty 7 par l'algorithme DBA-LinUcb et comparaison avec Lin-Ucb

DBA-LinUcb Nb cluster	$Conf_{clust30,70}$	$Conf_{clust100,100}$
5 ;3 ;3	12,11%	12,28%
5 ;4 ;15	12,44%	**12,73%**
11 ;3 ;3	12,34%	12,28%
11 ;4 ;15	12,70%	12,15%
17 ;3 ;3	12,40%	12,57%
17 ;4 ;15	**12,73%**	12,49%
Lin-Ucb	12,50%	12,50%

Table 4.12 – Sensibilité au clustering pour le jeu de données AB Tasty 7 par l'algorithme DBA-LinUcb et comparaison avec Lin-Ucb, jeu de données dupliqué par 2 pour la phase de test

DBA-LinUcb Nb cluster	$Conf_{clust30,70}$	$Conf_{clust100,100}$
5 ;3 ;3	12,92%	12,61%
5 ;4 ;15	12,77%	**12,76%**
11 ;3 ;3	12,70%	12,42%
11 ;4 ;15	12,93%	12,44%
17 ;3 ;3	**13,07%**	12,65%
17415	12,61%	12,70%
Lin-Ucb	12,43%	12,43%

Table 4.13 – Sensibilité au clustering pour le jeu de données AB Tasty 7 par l'algorithme DBA-LinUcb et comparaison avec Lin-Ucb, jeu de données dupliqué par 5 pour la phase de test

d) Analyse des groupes après clustering

Nous analysons ici les clusters et leur taux de clics associé en entraînant CTREE sur l'intégralité des données. La figure 4.15 montre le résultat de CTREE apprenant sur l'intégralité des données avec $Nb_{\texttt{presence\_time\_serie}} = 17$, $Nb_{\texttt{time\_spend\_serie}} = 4$ et $Nb_{\texttt{connexion\_time\_serie}} = 15$. Nous identifions une sous-population impactée par le test, représentée par les clusters #`presence_time_serie` $= 16$, #`time_spend_serie` $\in \{2, 4, 5, 7, 9, 12, 13, 15\}$ et #`connexion_time_serie` $= 2$ représentant les groupes associés à la feuille `Node`15 et `Node`16. A contrario, la figure 4.16 montre qu'un CTREE n'apprenant qu'avec les moyennes n'identifie aucune différence entre les variations. Il est également intéressant de constater que les taux de clic les plus élevés sont trouvés à l'aide des clusters.

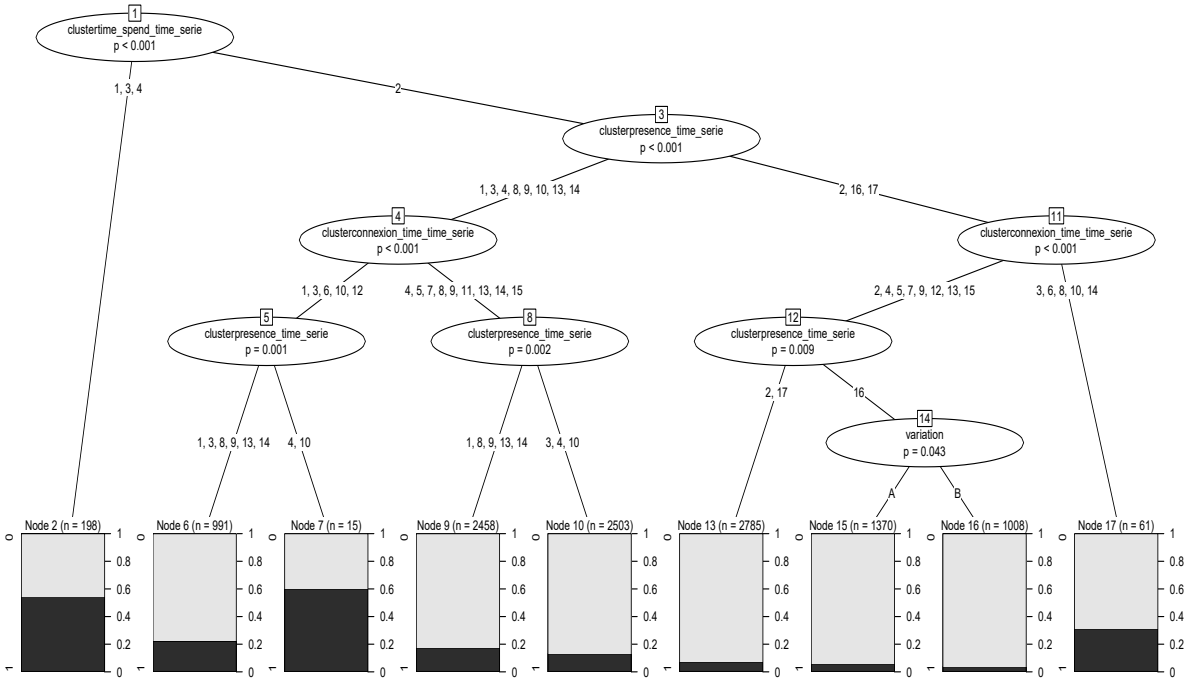


FIGURE 4.15 – Analyse de l'impact du test selon les cluster sur l'intégralité des données ($\text{Conf}_{\text{clust30,70}}, V_A = P_1$), AB Tasty 7, $Nb_{\texttt{presence\_time\_serie}} = 17$, $Nb_{\texttt{time\_spend\_serie}} = 4$ et $Nb_{\texttt{connexion\_time\_serie}} = 15$ (risque $\epsilon = 0.05$).

Nous rapportons dans le Tab. 4.14 les différents groupes identifiés par l'arbre de classement de la Fig. 4.15. En accord avec leurs profils évolutifs représentés par les centroides, nous proposons également dans ce tableau une correspondance en accord avec les patterns énoncés par les recherches en e-marketing. Nous considérons dans nos descriptions qu'un visiteur peut évoluer d'un pattern à un autre [2] : plusieurs patterns par groupes peuvent donc être trouvés.

2. D'après la définition du parcours visiteur type, [61] le visiteur peut par exemple passer d'indécis à acheteur potentiel

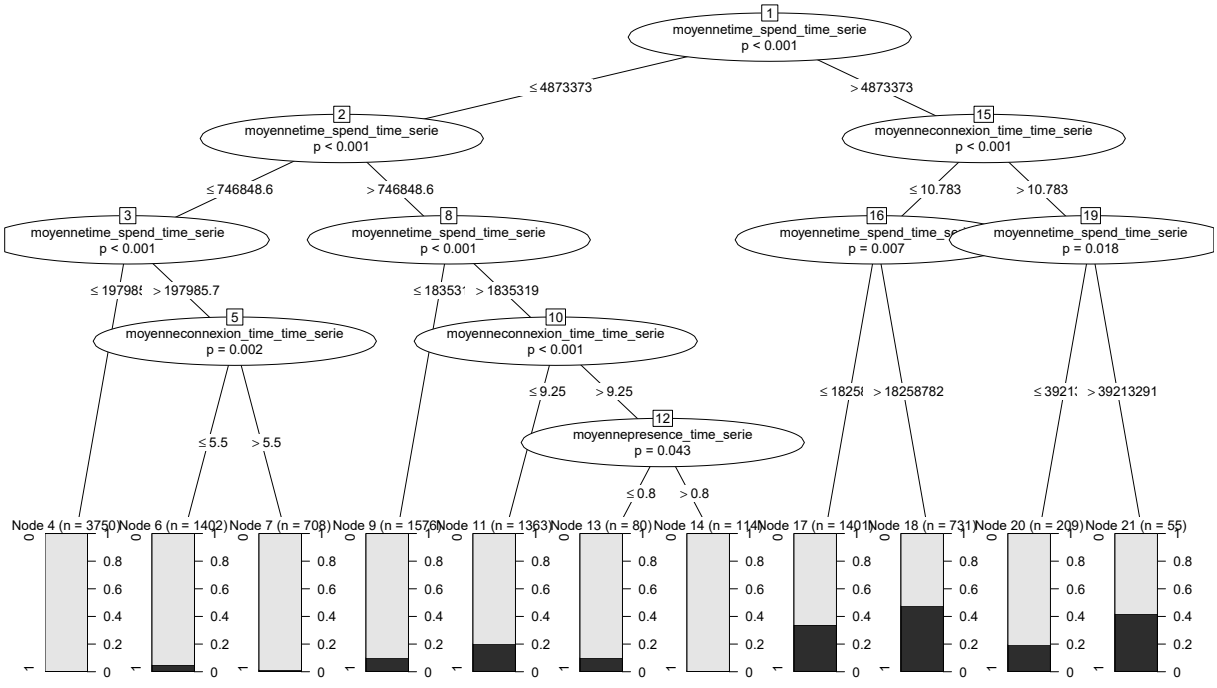


FIGURE 4.16 – Analyse de l'impact du test selon les moyennes ($\text{Conf}_{\text{clust30,70}}, V_A = P_1$), AB Tasty 7. Aucune sous population impactée par le test n'est détectée (risque $\epsilon = 0.05$).

d) Conclusion :

La méthode DBA-CTREE-UCB présente ici les résultats les plus intéressants en terme de gain moyen par rapport aux autres méthodes. Elle donne une interprétation claire des résultats par un arbre d'inférence et permet de rassembler des clusters ayant le même taux de clics. Fixer un nombre trop important de clusters ne semble pas faire baisser le gain. Les expériences montrent aussi que les clusters sont rassemblés dans un même groupe si les visiteurs du cluster ont le même taux de clics. C'est pourquoi, cette méthode parait être ici la plus adaptée. Un apprentissage des clusters sur 30% de la base de données est suffisant pour obtenir des résultats comparables à un apprentissage total. L'indice en accord avec la combinaison de clusters qui maximise le gain est l'indice de silhouette.

Node	#presence_time_serie	#time_spend_serie	#connexion_time_serie	label	CTR	Description
7		1,3,4		Acteur potentiel / prospect parfait	[50%-60%]	Une à deux session la première allant jusqu'à jusqu'à 13 minutes par page puis les suivantes sont environ de 3 minutes. Cherche à parfaire son besoin
6	1,3,8,9,13,14	2	1,3,6,10,12	Indécis / acheteur potentiel	[20%-25%]	Visites rapides. Se connecte régulièrement en matinée. Passe peut de temps sur les pages. Ses visites sont très régulières. Peut ne pas venir pensant quelque jours pour revenir très fréquemment.
7	4,10	2	1,3,6,10,12	Acheteur potentiel / prospect parfait	[50%-60%]	Visites rapides. Se connecte régulièrement en matinée. Fait une à deux visites, ne revient plus pendant quelque jour puis revient une fois.
9	1,8,9,13,14	2	4,5,7,8,9,11,13,14,15	Indécis à acheteur potentiel	[10%-20%]	Consultation des pages rapides. Se connecte régulièrement en matinée. Fait une à deux visites, ne revient plus pendant quelque jour puis revient deux à trois fois.
10	3,4,10	2	4,5,7,8,9,11,13,14,15	Indécis / acheteur potentiel	[10%-20%]	Consultation des pages rapides. Se connecte régulièrement en matinée. Fait des visites de façons successive avec éventuellement plusieurs jours sans revenir.
13	2,17	2	2,4,5,7,9,12,13,15	Arrivé par erreur	[0%-0.5%]	Consultation des pages rapides. Arrive à des heures différentes de la journée. Soit une visite. Soit deux visites avec une longue absence entre les deux.
15 et 16	16	2	2,4,5,7,9,12,13,15	Arrivé par erreur / indécis	[0%-0.5%]	Consultation des pages rapides. Arrive à des heures différentes de la journée. 3 visites rapprochées , un jour d'absence puis revient. Préfère la variation A.
17	2,16,17	2	3,6,8,10,14	Acheteur potentiel ou Prospect parfait	[30%-40%]	Consultation des pages rapides. Se connecte en matinée, pendant la nuit ou en début d'après midi. Soit une visite. Soit deux visites à trois visites.

TABLE 4.14 – Groupes identifiés par une segmentation post-test pour le jeu de données Modz

4.5.3 Jeu de données AB Tasty 8

a) Les données

Ces données, sont de même nature que celles du jeu de données A/B Tatst 7 et proviennent d'un site de vidéo streamming. L'objectif de l'AB Test est d'augmenter le taux de clics de la page testée. Pour apprendre les clusters et les groupes nous testons plusieurs configurations. L'une utilise 30% des visiteurs (1591) et l'autre utilise la totalité des visiteurs (5306).

b) Explicitation des groupes

Comme pour l'expérimentation précédente, nous avons déterminé le nombre Nb optimal pour chaque type de série à l'aide des critères de qualité usuelle utilisés en clustering de séries temporelles.

La Tab.4.15, la Tab. 4.16 et la Tab.4.17 présentent les résultats obtenus en ayant cette fois appris sur l'intégralité des données. En effet, la taille du jeu de données ne permettait pas ici d'avoir une idée fiable de ces indices avec uniquement l'échantillon d'apprentis-

sage. La synthèse de ces tableaux est rapportée dans la Tab. 4.18. Ces indices indiquent un nombre $Nb_{\texttt{presence\_time\_serie}} \in \{15, 17, 18, 19\}$ et un nombre $Nb_{\texttt{connexion\_time\_time\_serie}} \in \{4, 3, 16\}$. Tous les indices s'accordent pour définir $Nb_{\texttt{time\_spend\_serie}} = 3$. Comme dans l'expérimentation précédente, les indices semblent s'accorder pour un nombre important de clusters avec la série `presence_time_serie` et un nombre restreint pour la série `time_spend_serie`. Il est également remarqué que le $Nb_{\texttt{presence\_time\_serie}} = 17$ est suggéré, comme pour l'expérimentation précédente.

Nb	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	8	19	20
Sil	0.95	0.99	0.95	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	**1.00**	0.99	**1.00**	**1.00**	**1.00**	0.99
SF	0.52	0.57	0.44	0.57	0.56	0.56	0.50	0.54	0.55	0.51	0.46	0.61	0.63	0.52	**0.64**	0.62	**0.64**	0.49
CH	23967.69	94920.53	7392.26	92916.74	76481.60	73732.83	48440.85	54824.44	74328.78	35249.75	15426.34	113478.09	655642.97	59130.23	**878240.38**	390375.32	723453.11	40708.89
DB	0.47	0.21																
DBstar	0.48	**0.38**	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
D	**0.33**	0.50	**0.33**	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	**1.00**	0.50	**1.00**	**1.00**	1.00	0.50
COP	0.06	0.01	0.06	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	**0.00**	**0.00**	**0.00**	0.01

TABLE 4.15 – Évaluation de différents indices de qualité des clusters sur la série `presence_time_serie` du jeu de données AB Tasty 8, avec la méthode de clustering D.B.A.

Nb	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	8	19	20
Sil	**0.93**	0.84	0.84	0.87	0.80	0.68	0.55	0.82	0.74	0.72	0.76	0.73	0.70	0.69	0.64	0.64	0.30	0.59
SF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CH	**1098.40**	708.89	531.42	594.28	728.20	811.72	734.14	690.02	812.77	729.64	506.72	704.80	685.37	595.79	592.96	524.53	602.55	479.05
DB	**0.52**	0.91	2.84	0.83	1.28	0.92	1.20	1.46	1.09	1.26	1.84	1.05	1.29	1.86	1.29	1.65	1.52	1.45
DBstar	**0.75**	2.73	5.51	1.32	3.12	2.03	6.40	2.73	3.31	8.08	4.81	9.68	9.27	5.75	12.79	22.27	11.65	11.40
D	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
COP	0.04	0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

TABLE 4.16 – Évaluation de différents indices de qualité des clusters sur la série `time_spend_time_serie` du jeu de données AB Tasty 8, avec la méthode de clustering D.B.A.

Nb	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	8	19	20
Sil	**0.35**	0.29	0.33	0.27	0.24	0.22	0.24	0.17	0.25	0.20	0.22	0.24	0.26	0.19	0.24	0.21	0.22	0.19
SF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CH	**2551.46**	1783.64	1281.63	2280.90	1635.07	1383.83	1324.97	1128.67	1459.37	840.49	774.04	2066.44	1032.65	1860.81	1017.65	640.29	1072.69	852.35
DB	1.48	**1.32**	1.61	1.33	1.58	1.45	1.56	2.07	1.68	1.97	1.65	1.56	1.46	1.46	1.37	1.54	1.51	1.86
DBstar	1.67	**1.32**	2.40	1.61	1.95	2.43	3.05	3.04	2.82	3.53	2.76	2.78	2.91	2.87	2.75	2.97	3.43	3.58
D	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
COP	0.23	0.21	0.16	0.17	0.17	0.15	0.15	0.15	0.13	0.15	0.12	**0.11**	0.12	**0.11**	**0.11**	**0.11**	**0.11**	**0.11**

TABLE 4.17 – Évaluation de différents indices de qualité des clusters sur la série `connexion_time_time_serie` du jeu de données AB Tasty 8, avec la méthode de clustering D.B.A.

Interprétabilité des clusters Nous observons ici centroïdes des clusters selon deux configurations : les clusters sont appris sur 30% ou 100% de la base de données. Les centroïdes des clusters sous la configuration $Nb_{\texttt{presence\_time\_serie}} = 4$, $Nb_{\texttt{time\_spend\_serie}} = 3$ et $Nb_{\texttt{connexion\_time\_serie}} = 4$ à travers les configurations : $\text{Conf}_{\text{clust30,70}}$ (voir la Fig. 4.17, la Fig. 4.19, la Fig. 4.21) et pour la configuration $\text{Conf}_{\text{clust100,100}}$ (voir Fig. 4.18, la Fig. 4.20 et la Fig. 4.22).

Nous constations des similarités entre :

— les centroïdes du cluster #`presence_time_serie` = 3 ($\text{Conf}_{\text{clust30,70}}$) et #`presence_time_serie` = 4 ($\text{Conf}_{\text{clust100,100}}$).

	presence_time_serie	time_spend_serie	connexion_time_serie
Sil	15,17,15	3	3
SF	17,19	⊘	⊘
CH	17	3	3
DB	4	3	4
DBstar	4	3	4
D	3,5	⊘	⊘
COP	17,19,19	⊘	14,15,16,17,18,19,20

TABLE 4.18 – Synthèse du nombre de clusters selon différents indice qualité, avec la méthode de clustering D.B.A. ($\text{Conf}_{\text{clust100,100}}$) avec le jeu de données AB Tasty 8

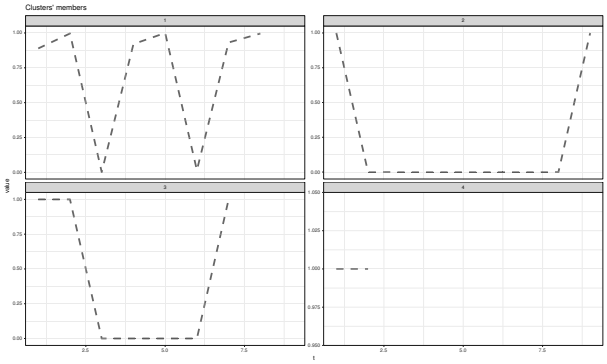


FIGURE 4.17 – Centroïdes de $Nb = 4$ clusters associés aux séries presence_time_serie, $\text{Conf}_{30,70}$ jeu de données AB Tasty 8

— les centroïdes du cluster #`presence_time_serie` = 4 ($\text{Conf}_{\text{clust30,70}}$) et #`presence_time_serie` = 3 ($\text{Conf}_{\text{clust100,100}}$).
— les centroïdes du cluster #`time_spend_serie` = 2 ($\text{Conf}_{\text{clust30,70}}$) et #`time_spend_serie` = 2 ($\text{Conf}_{\text{clust100,100}}$).
— les centroïdes du cluster #`connexion_time_serie` = 4 ($\text{Conf}_{\text{clust30,70}}$) et #`connexion_time_serie` = 3 ($\text{Conf}_{\text{clust100,100}}$).

Des différences sont cependant visibles, notamment sur la taille des séquences des centroïdes, entre les deux configurations.

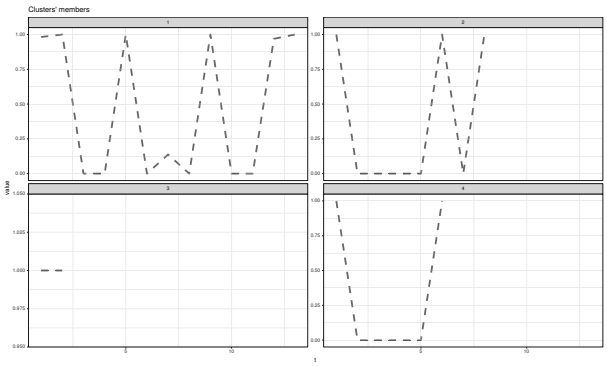


FIGURE 4.18 – Centroïdes de $Nb = 4$ clusters associés aux séries presence_time_serie, $\text{Conf}_{100,100}$ jeu de données AB Tasty 8

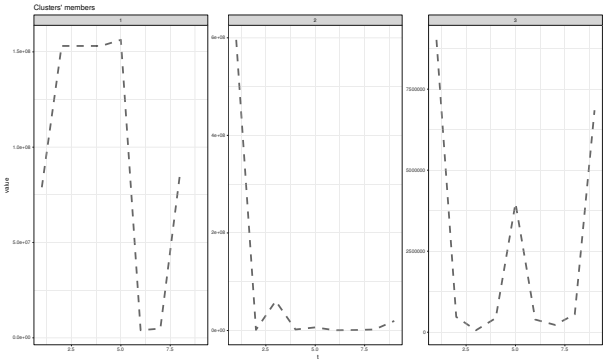


FIGURE 4.19 – Centroïdes de $Nb = 3$ clusters associés aux séries time_spend_serie, $\text{Conf}_{30,70}$ jeu de données AB Tasty 8

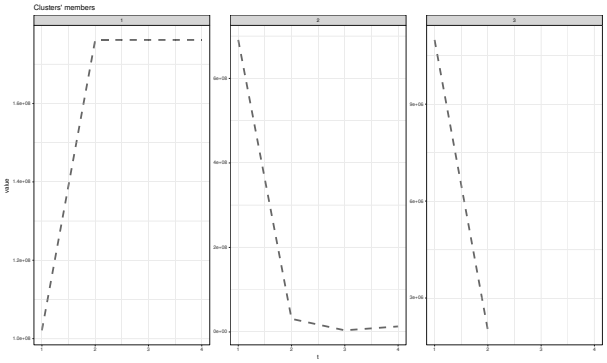


FIGURE 4.20 – Centroïdes de $Nb = 3$ clusters associés aux séries time_spend_serie, $\text{Conf}_{100,100}$ jeu de données AB Tasty 8

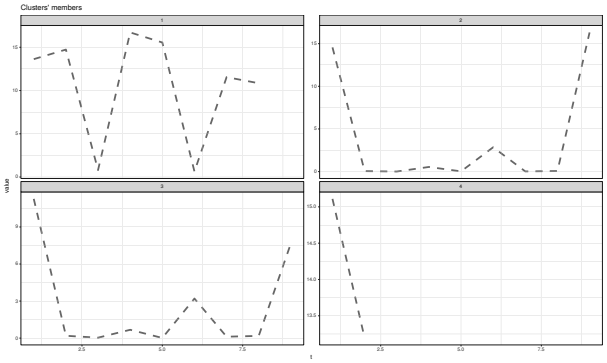


FIGURE 4.21 – Centroïdes de $Nb = 17$ clusters associés aux séries connexion_time_serie, $\text{Conf}_{30,70}$ jeu de données AB Tasty 8

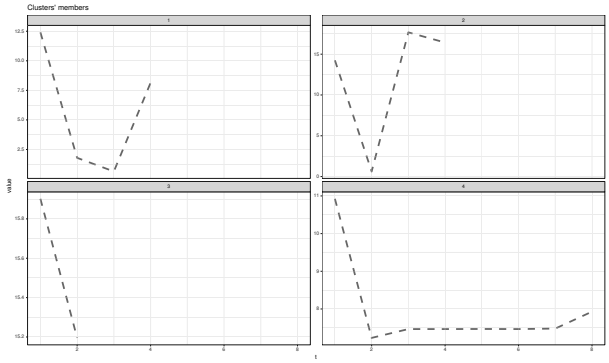


FIGURE 4.22 – Centroïdes de $Nb = 17$ clusters associés aux séries connexion_time_serie, $\text{Conf}_{100,100}$

c) Étape d'exploration dans l'A/B test (Allocation dynamique)

Les gains moyens (taux de clics) sont comparés ici pour les algorithmes précédemment cités afin d'observer comment le nombre de clusters fait varier les résultats.

DBA-Ctree-Ucb vs Ctree-Ucb : Le taux de clics moyen obtenu avec UNIFORM (c'est-à-dire sans allocation dynamique) est de 9,21%. L'objectif minimal est que les gains moyens obtenus par nos méthodes soient supérieurs à cette valeur. Cet objectif sera difficile à atteindre car la différence entre les deux variations est très faible. Comme pour le précédent jeu de données, l'impact du nombre de clusters est observé en dupliquant le jeu de données afin d'éviter la variance induite par le *rejection sampling*. Dans la Tab. 4.19, pour un jeu de données non dupliqué, si l'apprentissage des groupes est réalisé avec $V_A = P_2$, DBA-CTREE-UCB n'identifie aucun groupe distinctif et applique une stratégie UCB. Par conséquent, bien qu'elle dépasse CTREE-UCB avec $\text{Conf}_{30,70}$, elle ne bat pas CTREE-UCB si $\text{Conf}_{100,100}$. En revanche, pour un apprentissage avec $V_A = P_1$, le gain moyen est maximal. Lorsque qu'on duplique le jeu de données, nous constatons que lorsque DBA-CTREE-UCB n'identifie par de groupes distincts, il sous-performe CTREE-UCB, qui en identifie plusieurs, via les moyennes. Pour DBA-CTREE-UCB, l'apprentissage avec $V_A = P_2$ n'obtient des bons résultats qu'avec duplication, contrairement à CTREE-UCB qui obtient des bons résultats avec et sans duplication. En revanche, si nous choisissons judicieusement le nombre de clusters et la base d'apprentissage ($Nb_{\texttt{presence\_time\_time\_serie}} = 4$ ($\text{Conf}_{30,70}$, $Nb_{\texttt{time\_spend\_serie}} = 3$ et $Nb_{\texttt{connexion\_time\_serie}} = 3$, $V_A = P_1$) ou bien ($\text{Conf}_{100,100}$, $Nb_{\texttt{time\_spend\_serie}} = 17$ et $Nb_{\texttt{connexion\_time\_serie}} = 3$, $V_A = P_2$), les valeurs maximales de gain sont atteintes par DBA-CTREE-UCB avec et sans duplication du jeu de données.

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-CTREE-UCB Nb cluster	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
4 ;3 ;3	9,86%	**9,86%**	**9,92%**	9,05%
4 ;3 ;4	**10,12%**	**9,89%**	**9,92%**	9,05%
17 ;3 ;3	9,86%	**9,86%**	**9,92%**	9,30%
17 ;3 ;4	10,12%	**9,86%**	**9,92%**	9,30%
CTREE-UCB	9,59%	9,31%	9,81	**9,63%**
UNIFORM	9,21%.	9,21%.	9,21%.	9,21%.

TABLE 4.19 – Gain moyen (taux de clics) pour DBA-CTREE-UCB et CTREE-UCB

DBA-LinUcb vs Lin-Ucb : Une analyse sur l'ensemble des données (voir section suivante, dans la figure 4.24) a mis en évidence une corrélation linéaire entre la moyenne de la série `time_spend_serie` et la probabilité de cliquer. Par conséquent, dans cette situation, surpasser la performance de LIN-UCB sera difficile. Les résultats sont rapportés dans les tables Tab. 4.22, 4.23 et 4.24. La performance de LIN-UCB est particulièrement

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-Ctree-Ucb Nb cluster	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
4 ;3 ;3	**9,61%**	**9,61%**	9,69%	9,33%
4 ;3 ;4	9,60%	**9,61%**	9,69%	9,33%
17 ;3 ;3	**9,61%**	**9,61%**	9,69%	9,41%
17 ;3 ;4	9,60%	**9,61%**	9,69%	9,41%
Ctree-Ucb	9,49%	9,32%	**9,86%**	**9,70%**

Table 4.20 – Gain moyen (taux de clics) pour DBA-Ctree-Ucb et Ctree-Ucb (jeu de données dupliqué par 2)

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-Ctree-Ucb Nb cluster	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
4 ;3 ;3	9,19%	9,19%	9,24%	10,06%
4 ;3 ;4	9,52%	9,19%	9,24%	10,06%
17 ;3 ;3	9,19%	9,19%	9,24%	**10,06%**
17 ;3 ;4	9,49%	9,19%	9,24%	**10,06%**
Ctree-Ucb	**9,69%**	**9,76%**	**9,53%**	9,93%

Table 4.21 – Gain moyen (taux de clics) pour DBA-Ctree-Ucb et Ctree-Ucb (jeu de données dupliqué par 5)

visible lorsque le jeu de données n'est pas augmenté par duplication (voir la Tab. 4.22 et la Tab 4.23).

DBA-LinUcb Nb cluster	$\text{Conf}_{\text{clust}30,70}$	$\text{Conf}_{\text{clust}100,100}$
4 ;3 ;3	9,74%	9,69%
4 ;3 ;4	9,68%	9,72%
17 ;3 ;3	9,63%	9,84%
17 ;3 ;4	9,08%	9,32%
Lin-Ucb	**10,19%**	**10,19%**
uniform	9,21%	9,21%

Table 4.22 – Gain moyen (taux de clics) pour DBA-LinUcb et Lin-Ucb sur le jeu de données AB Tasty 8

DBA-LinUcb Nb cluster	$Conf_{clust30,70}$	$Conf_{clust100,100}$
4 ;3 ;3	9,68%	9,92%
4 ;3 ;4	9,67%	9,71%
17 ;3 ;3	9,58%	9,78%
17 ;3 ;4	9,43%	9,74%
Lin-Ucb	**9,98%**	**9,98%**

Table 4.23 – Gain moyen (taux de clics) pour DBA-LinUcb et Lin-Ucb (data-set dupliqué par 2)

DBA-LinUcb Nb cluster	$Conf_{clust30,70}$	$Conf_{clust100,100}$
4 ;3 ;3	9,84%	10,06%
4 ;3 ;4	**10,29%**	9,95%
17 ;3 ;3	9,52%	**10,07%**
17 ;3 ;4	9,96%	9,94%
Lin-Ucb	9,57%	9,57%

Table 4.24 – Gain moyen (taux de clics) pour DBA-LinUcb et Lin-Ucb (data-set dupliqué par 5)

c) Analyse des groupes après clustering

Nous analysons ici les groupes après clustering et leur taux de clics associé en entraînant CTREE sur l'intégralité des données.

Aucun des modèles (Fig. 4.23 et Fig 4.24) ne considère la variation comme une caractéristique faisant varier le taux de clics. Encore une fois, le groupe ayant le taux de clics le plus élevé est détecté par le modèle incluant les clusters.

Le modèle CTREE basé sur les moyennes met en évidence une corrélation linéaire positive entre la moyenne de `time_spend_serie` et la probabilité d'obtenir une récompense. Les autres caractéristiques n'ont pas été retenues dans la construction de l'arbre. Le modèle de la Fig. 4.23, fait intervenir à la fois les présences/non présences, et la durée des sessions.

Nous rapportons dans le Tab.4.25 les différents groupes identifiés par l'arbre de classement de la Fig. 4.23. En accord avec leurs profils évolutifs représentés par les centroides, nous proposons dans ce tableau une correspondance en accord avec les patterns.

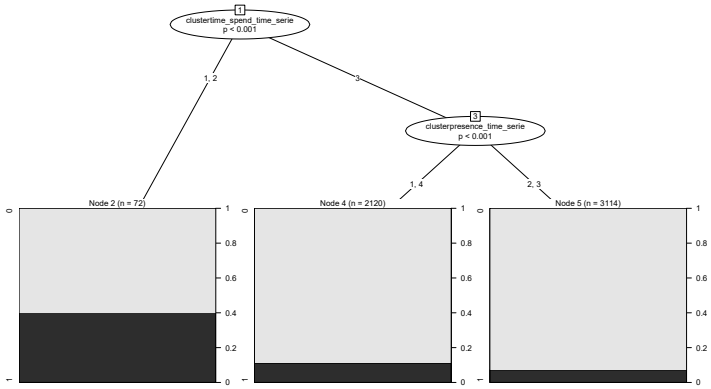


FIGURE 4.23 – Analyse de l'impact du test selon les clusters ($Config_{30,70}, V_A = P1$), AB Tasty 8, $Nb_{\texttt{presence\_time\_serie}} = 4$, $Nb_{\texttt{time\_spend\_serie}} = 3$ et $Nb_{\texttt{connexion\_time\_serie}} = 4$. (risque $\epsilon = 0.05$).

Node	#presence_time_serie	#time_spend_serie	#connexion_time_serie	label	taux de clics	Description
2			1,2	Acteur potentiel / prospect parfait	[35%-45%]	Reste de 2.30 minutes à 10 minutes sur une page.
4	1,4	3		Indécis ou arrivé par erreur	[5%-15%]	Temps par page très cours (présence de bot éventuel). Arrive une fois et ne revient plus, ou alors revient plusieurs fois à la même fréquence.
5	2,3	3		Arrivé par erreur	[0%-0.1%]	Temps par page très cours (présence de bot éventuel). Arrive une fois et ne reviens qu'une seule fois quelque jours plus tard.

TABLE 4.25 – Groupes identifiés par une segmentation post-test pour le jeu de données AB Tast 2.2

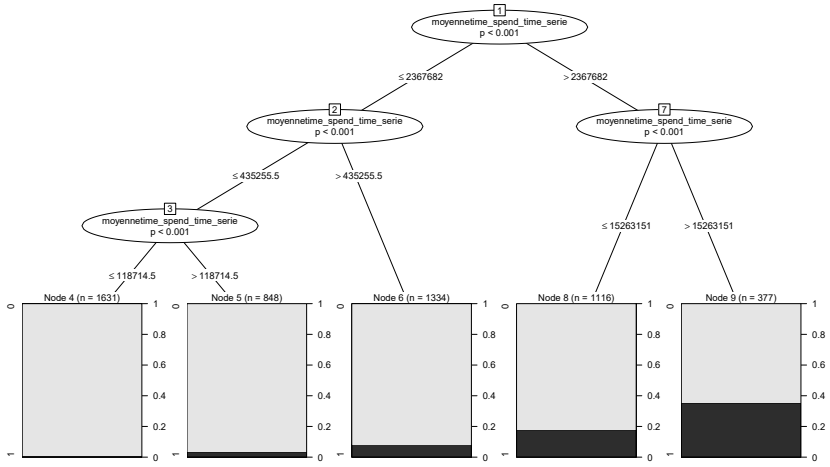


FIGURE 4.24 – Analyse de l'impact du test selon les moyennes avec le jeux de données AB Tasty 8 (risque $\epsilon = 0.05$).

d) Conclusion :

La méthode LIN-UCB présente les résultats les plus intéressants sans duplication notamment parce qu'elle exploite la corrélation importante entre le temps passé sur la page et la probabilité de cliquer. Pourtant, en dupliquant le jeu de données, c'est DBA-LINUCB qui présente le gain moyen le plus élevé. Ces résultats sont légèrement supérieurs à ceux de LIN-UCB et DBA-CTREE-UCB (Config$_{30,70}$, $V_0 = A$, $Nb_{\texttt{presence\_time\_serie}} = 4$, $Nb_{\texttt{time\_spend\_time\_serie}} = 3$ et $Nb_{\texttt{connexion\_time\_serie}} = 4$). Ici, la faible performance de DBA-CTREE-UCB et CTREE-UCB s'explique par la difficulté d'identifier des groupes au comportement statistiquement différents en utilisant uniquement la variation originale sur un échantillon réduit. Toutefois, lorsque DBA-CTREE-UCB identifie au moins deux groupes distincts dans l'étape offline, les résultats restent bien meilleurs que ceux obtenus par UNIFORM.

En raison de la taille du jeu de données original, un apprentissage des clusters sur 100% de la base de données permet d'avoir des résultats plus stables dans le temps. Les indices en accord avec la combinaison de clusters qui maximisent le gain (4,3,4 sans duplication, et 17,3,3 avec duplication) sont l'indice de silhouette et l'index de Davies Bouldin (simple et corrigé).

4.5.4 Localization Data

a) Les données

Le jeu de données “Localization Data for Posture Reconstruction” provient d'un site de jeux de données publics [8]. Il contient des données de localisation pour la reconstruction de la posture composées de 164860 mesures de position associées aux déplacements de 5 patients. Ce déplacement est donné suivant 3 axes (x,y, z) sur lesquels différentes mesures temporelles ont été faites. Chaque patient est ainsi représenté par 3 séries temporelles. Pour chaque mesure de chaque série, l'activité en cours est précisée (parmi 11).

L'objectif de notre A/B test est de voir s'il est possible de reconnaître l'activité d'un patient à partir de ses trois séries temporelles de mesures. Ainsi, dans notre AB test :

- les patients avec leurs trois séries sont les items ;
- Les variations à tester sont les différentes activités que nous avons regroupées en 3 types d'activités ("lying", "sitting down" et "sitting") pour limiter le nombre de variations à tester (11 variations nécessiteraient beaucoup trop d'items et donc un temps de calcul trop important) qui seront noté par A, B, C ;
- les récompenses sont l'appartenance à l'activité. Si le patient est effectivement bien en train de réaliser l'activité choisie par l'algorithme, la récompense est 1, 0 sinon. Chaque patient, décrit par une série temporelle, ne peut faire qu'une seule activité à la fois. L'objectif est de trouver le plus souvent la bonne activité.

Ce problème en pratique peut être vu comme une stratégie de classification lorsqu'on souhaite trouver l'activité d'une personne à un instant donné.

Le *rejection sampling* pour le jeu de données *Localization Data for Posture Reconstruction* n'est pas nécessaire. En effet, les récompenses pour tous les bras sont connues (pour chaque item, la bonne classe génère une récompense égale à 1, les autres classes génèrent une récompense égale à 0).

Néanmoins les patients étant au nombre de 5, il n'y aura donc que 5 items. Ce qui n'est bien évidemment pas suffisant pour la phase d'apprentissage des clusters et la phase d'A/B Test.

Nous avons donc artificiellement augmenté le nombre d'items à tester par duplication différentielle des patients en appliquant le protocole suivant :

- On choisi aléatoirement un patient.
- Ce patient est dupliqué.
- On choisi aléatoirement une période, c'est-à-dire un début et une fin de mesure.
- Dans le nouveau patient, on tronque la série associée à chacun des 3 capteurs à la période choisie.
- On lui ajoute l'activité à prédire au prochain pas de temps à partir de de son activité juste après la période.

Ainsi, 2000 patients ont été simulés

Pour apprendre les clusters et les groupes nous testons plusieurs configurations. L'une utilise 30% des patients (600) et l'autre utilise la totalité des visiteurs (2000).

b) Explicitation des groupes

Dans un premier nous avons déterminé le nombre Nb optimal pour chaque type de série à l'aide des critères de qualité usuelle utilisés en clustering de séries temporelles. Les Tab. 4.26, Tab. 4.27 et Tab. 4.28 présentent ces critères sur 30% des données associées aux séries `x`, `y` et `z`. La Tab. 4.29 rapporte les Nb optimaux selon les indices considérés.

	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_10	k_11	k_12	k_13	k_14	k_15	k_16	k_17	k_18	k_19	k_20
Sil	0.49	0.42	0.51	0.63	0.49	0.57	0.63	0.56	0.62	0.70	**0.71**	0.64	**0.71**	0.62	0.69	0.62	0.64	0.58
SF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CH	**542.23**	382.43	354.85	374.23	328.59	345.17	331.53	312.97	378.05	410.12	436.32	313.79	417.72	332.78	330.82	283.31	384.28	301.91
DB	0.95	1.14	1.32	1.00	0.83	0.84	0.57	0.69	0.96	0.58	0.71	0.70	0.81	0.64	0.88	0.90	**0.62**	0.98
DBstar	1.04	1.39	1.49	1.35	0.97	2.02	1.00	1.32	1.31	**0.77**	1.10	1.13	1.41	1.79	2.00	3.10	1.33	2.43
D	0.01	0.01	0.01	0.01	0.01	0.02	0.02	0.01	0.01	0.02	0.03	0.01	0.01	0.01	0.01	0.02	0.03	0.01
COP	0.13	0.10	0.09	0.08	0.07	0.07	0.06	0.05	0.04	0.03	0.03	0.04	0.03	0.03	0.03	0.03	**0.02**	0.03

TABLE 4.26 – Évaluation de différents indices de qualité des clusters sur la série `x` du jeu de données Localization Data, avec la méthode de clustering D.B.A. (Config.$_{30,70}$)

	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_10	k_11	k_12	k_13	k_14	k_15	k_16	k_17	k_18	k_19	k_20
Sil	0.18	0.36	0.37	0.34	0.49	0.39	0.54	0.47	0.58	0.57	0.54	**0.64**	0.57	0.52	0.58	0.61	**0.64**	0.62
SF	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CH	**300.10**	291.48	207.54	181.26	205.69	171.32	182.20	158.31	187.33	166.25	163.73	148.86	149.15	133.67	144.45	160.64	147.57	145.85
DB	1.79	1.17	1.55	1.54	1.19	1.03	1.24	1.31	1.07	0.99	1.19	**0.88**	0.93	0.89	1.00	1.14	1.10	0.98
DBstar	2.04	**1.20**	1.62	1.75	1.71	1.48	1.37	1.40	1.44	1.42	2.21	1.21	1.60	1.87	2.36	1.64	1.82	2.35
D	0.01	0.02	0.03	0.03	0.01	0.02	0.03	0.03	0.03	0.03	0.06	0.07	0.05	0.01	0.02	0.03	0.01	0.05
COP	0.35	0.29	0.25	0.25	0.18	0.17	0.15	0.14	0.12	0.11	0.09	0.10	0.10	0.10	0.10	**0.07**	**0.07**	0.08

TABLE 4.27 – Évaluation de différents indices de qualité des clusters sur la série `y` du jeu de données Localization Data, avec la méthode de clustering D.B.A. (Config.$_{30,70}$)

	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_10	k_11	k_12	k_13	k_14	k_15	k_16	k_17	k_18	k_19	k_20	
Sil	0.34	0.40	0.43	0.35	0.53	0.46	0.45	0.53	0.64	0.56	0.60	0.53	0.60	0.49	**0.61**	**0.61**	**0.61**	0.54	
SF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
CH	**338.99**	256.34	254.33	201.17	206.94	185.29	151.36	169.43	190.13	152.11	175.80	124.46	162.60	137.49	132.95	126.50	144.85	130.71	‰
DB	1.26	1.61	1.28	1.40	0.88	1.10	1.02	1.29	0.95	0.97	0.87	1.01	**0.79**	0.79	1.02	1.81	0.85	1.30	
DBstar	1.41	1.65	1.30	1.62	**1.05**	1.48	1.50	1.90	1.17	1.32	1.21	2.02	1.38	1.68	2.64	2.34	1.82	2.84	
D	0.03	0.00	0.02	0.01	0.00	0.00	0.02	0.03	0.03	0.02	0.05	0.00	0.01	0.00	0.00	0.02	0.04	0.02	
COP	0.42	0.28	0.25	0.25	0.20	0.19	0.19	0.17	0.13	0.13	0.11	0.14	0.10	0.11	0.11	0.09	**0.08**	0.09	

TABLE 4.28 – Évaluation de différents indices de qualité des clusters sur la série `z` du jeu de données Localization Data, avec la méthode de clustering D.B.A. (Config.$_{30,70}$)

Interprétabilité des clusters Nous observons ici les centroïdes des clusters selon deux configurations : les clusters sont appris sur 30% ou 100% de la base de données. Les centroïdes des clusters sous la configuration $Nb_x = 19$, $Nb_y = 18$, et $Nb_y = 19$ pour les trois capteurs sont représentés respectivement dans les figures 4.25 (x), 4.26 (y) et 4.27 (z) pour la Conf$_{clust,30,70}$ et dans les figures 4.28 (x), 4.26 (y) et 4.27 pour la Conf$_{clust,100,100}$.

Les centroïdes représentent les séries montrent quatre type de comportement :

— Décroissante.

— Croissante.

	x	y	z
Sil	13,15	19	17 , 18,19
SF	⊘	⊘	⊘
CH	3	3	3
DB	7	14	15
DBstar	7	4	7
D	⊘	⊘	⊘
COP	19	18	19

TABLE 4.29 – Synthèse du nombre de clusters selon différents indice qualité, avec la méthode de clustering D.B.A. ($\text{Conf}_{\text{clust30,70}}$) Localization Data

— Stationnaire.

— Oscillante.

Les creux dans les courbes peuvent être justifiés par un arrêt du patient ou bien un défaut dans la collecte. La forte similarité entre les deux configurations justifie leur performances comparables lors de l'allocation dynamique.

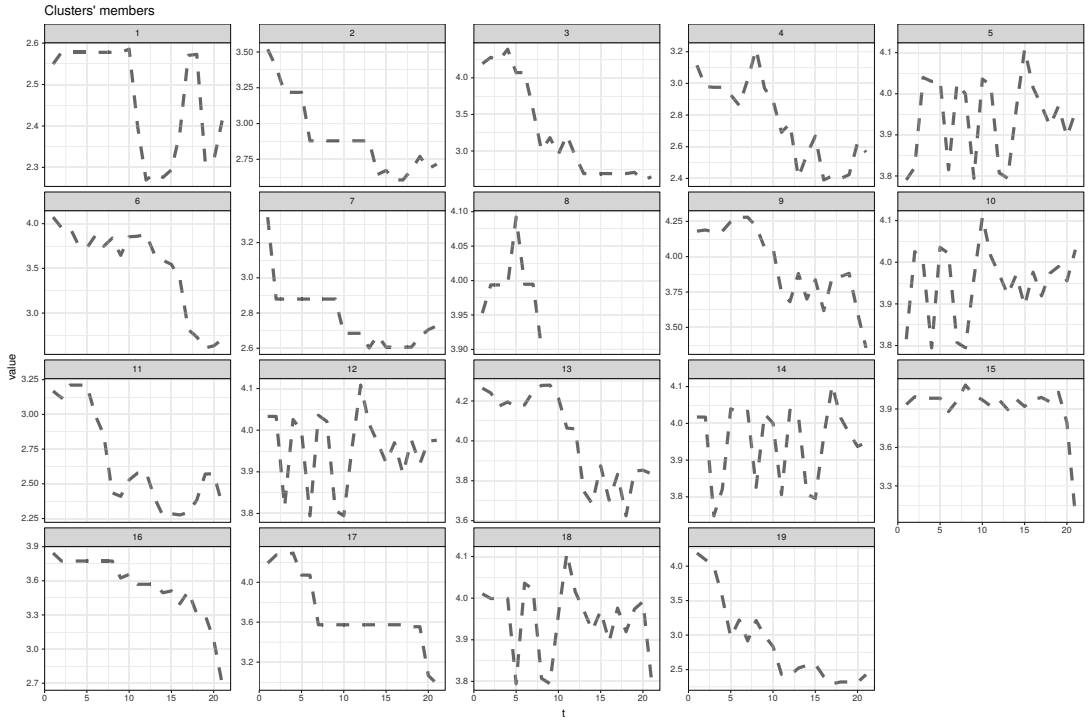


FIGURE 4.25 – Centroïdes de $Nb = 19$ clusters associés aux séries x, $\text{Conf}_{clust,30,70}$ Localization data

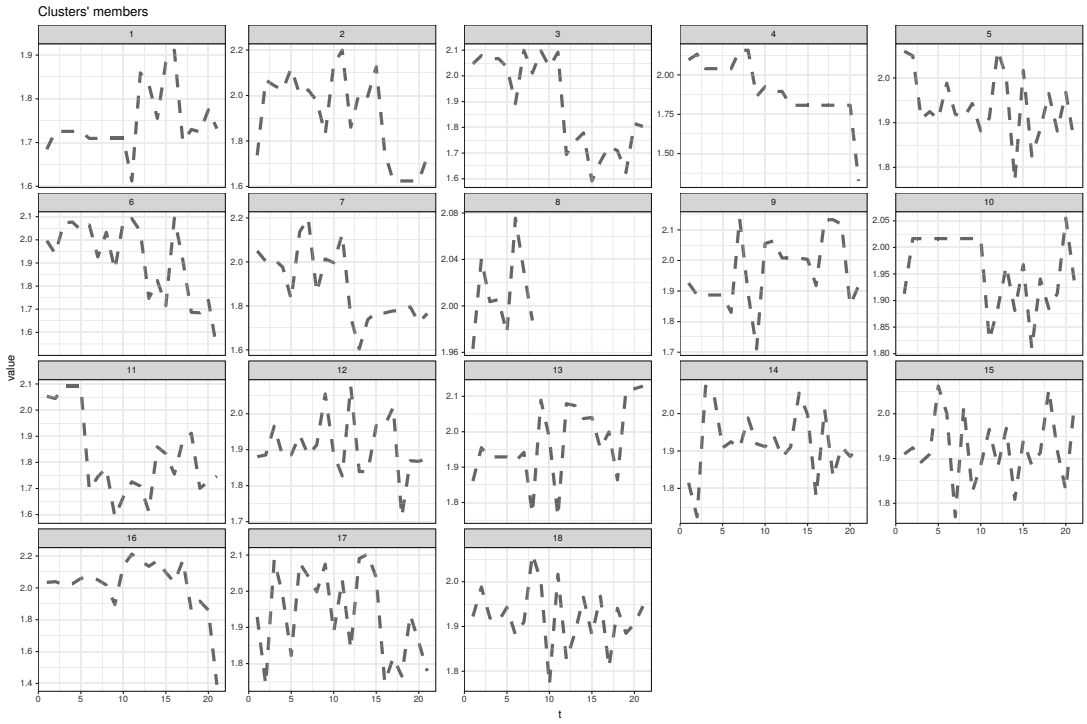


FIGURE 4.26 – Centroïdes de $Nb = 18$ clusters associés aux séries y, $\text{Conf}_{clust,30,70}$ Localization data

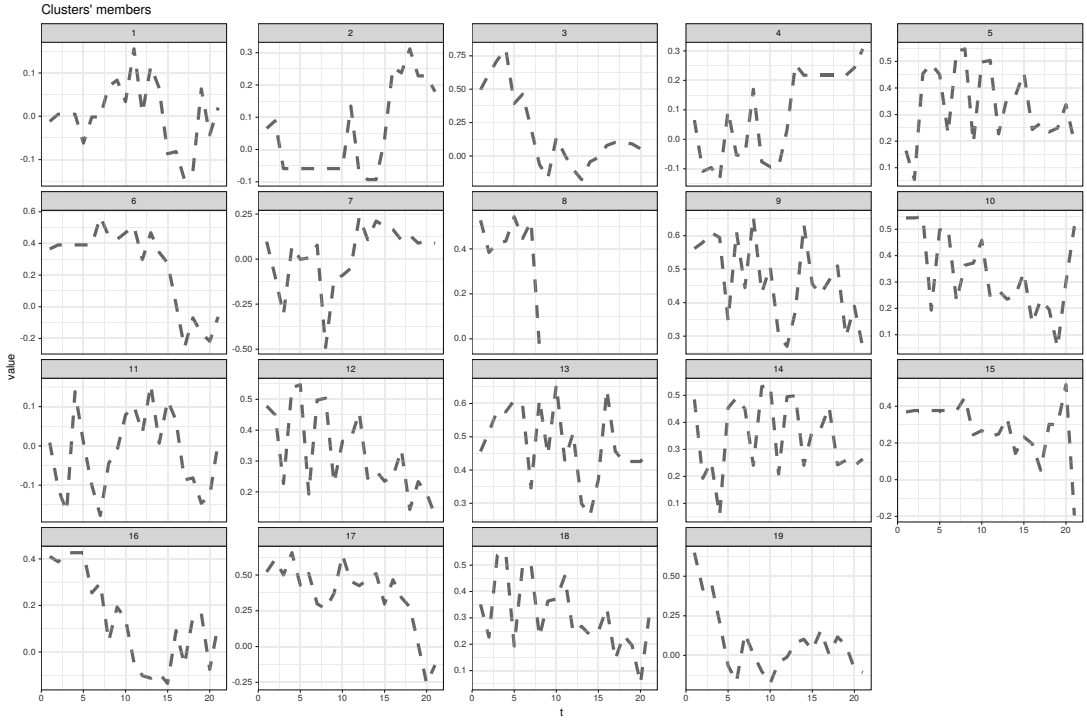


FIGURE 4.27 – Centroïdes de $Nb = 19$ clusters associés aux séries z, $\text{Conf}_{clust,30,70}$ Localization data

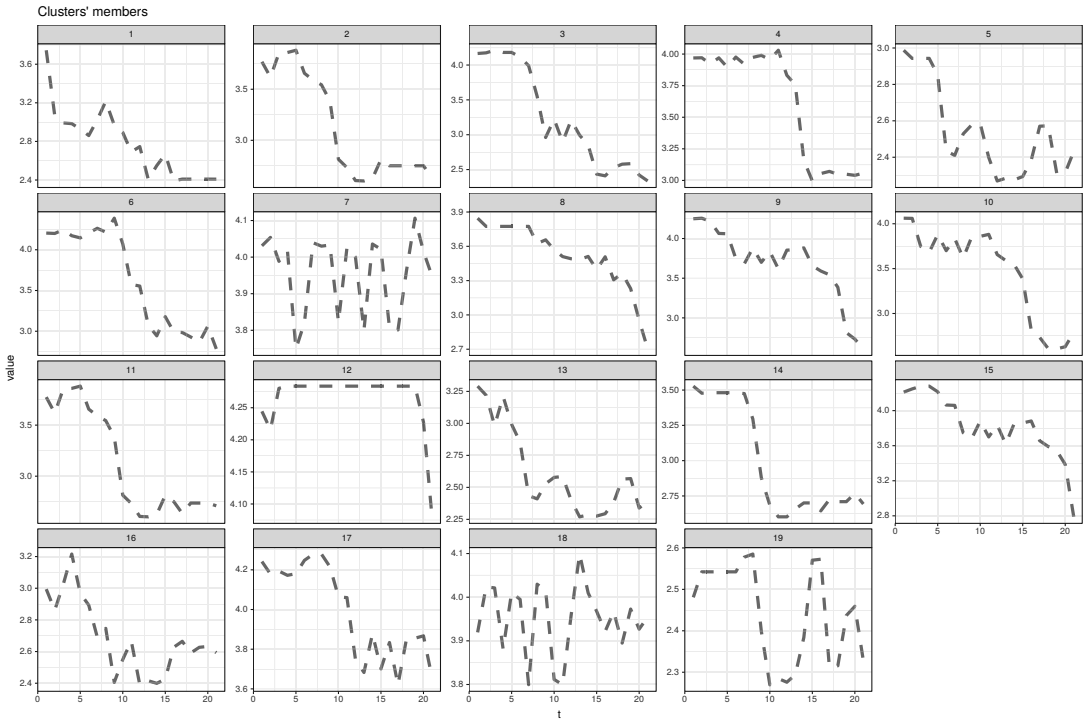


FIGURE 4.28 – Centroïdes de $Nb = 19$ clusters associés aux séries x, $\text{Conf}_{clust,100,100}$ Localization data

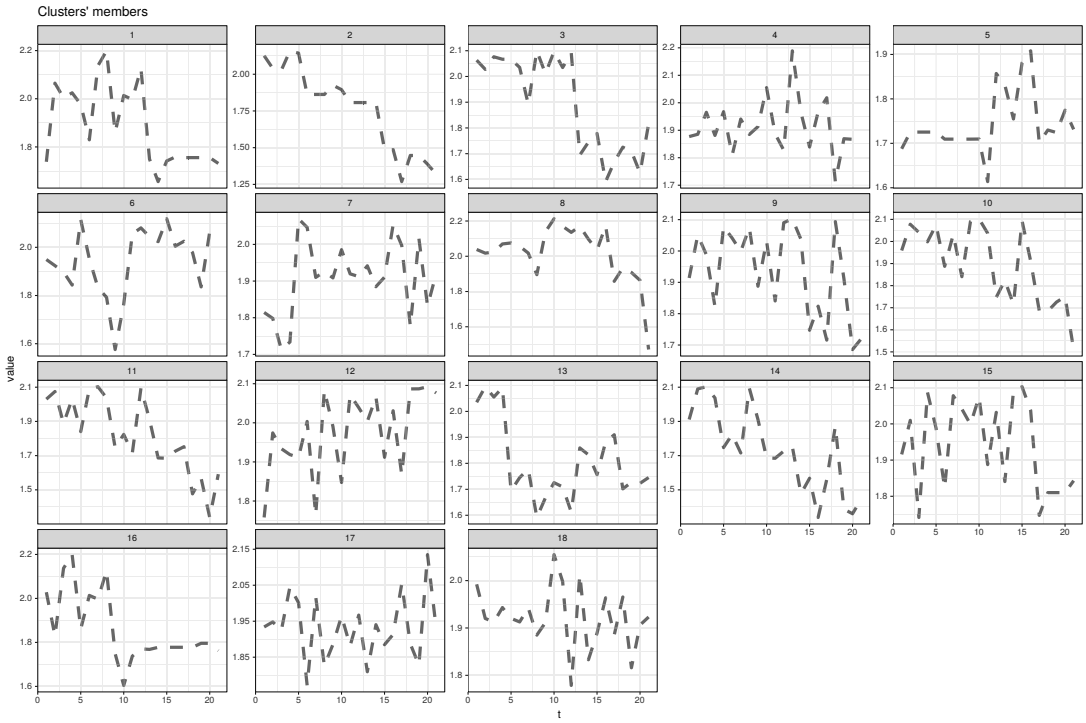


FIGURE 4.29 – Centroïdes de $Nb = 18$ clusters associés aux séries y, $\text{Conf}_{clust,100,100}$ Localization data

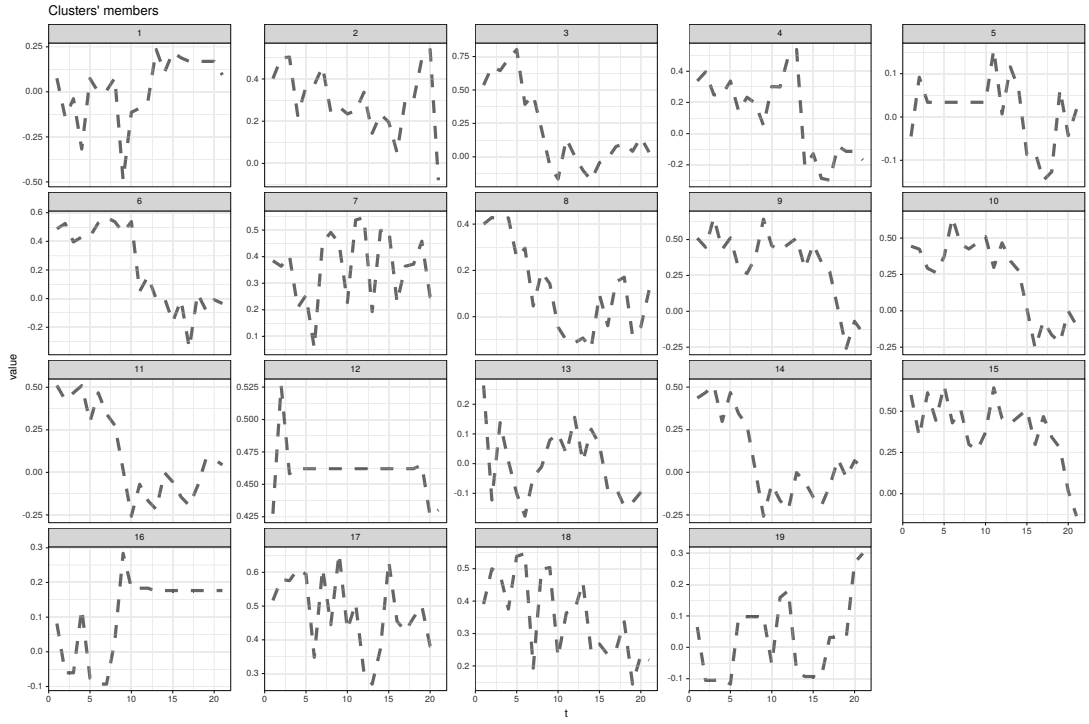


FIGURE 4.30 – Centroïdes de $Nb = 19$ clusters associés aux séries z, $\text{Conf}_{clust,100,100}$ Localization data

c) Étape d'exploration dans l'A/B test (Allocation dynamique)

Nous comparons ici les taux de bonne classification des algorithmes précédents pour observer l'influence du nombre de clusters sur les résultats.

DBA-Ctree-Ucb vs Ctree-Ucb Le taux de classification moyen pour une allocation UNIFORM est de 30%. L'objectif minimal est d'obtenir un gain moyen supérieur à cette valeur. La Tab. 4.30 présente les résultats des deux algorithmes dans les deux configurations en faisant varier la classe utilisée pour l'apprentissage. Les résultats de DBA-CTREE-UCB sont performants très majoritairement avec la $\text{Cong}_{100,100}$, un apprentissage partiel n'étant pas suffisant pour obtenir des résultats comparables à ceux CTREE-UCB qui apparaît être comme la meilleure des deux méthodes. Le paramétrage permettant de maximiser le gain pendant l'allocation dynamique est $Nb_x = 3$, $Nb_y = 3$ et $Nb_z = 17$.

	Conf3070 et confclust30/70			Conf100100 et confclust100		
DBA-CTREE-UCB Nb cluster (x,y,z)	$V_A = A_1$	$V_A = A_2$	$V_A = A_3$	$V_A = A_1$	$V_A = A_2$	$V_A = A_3$
13 ;19 ;3	881 0.629	617 0.441	610 43,6%	1416 70,8%	1618 80.9%	1575 78.8%
3 ;3 ;17	893 63,8%	821 0.586	935 **66,8%**	1413 70.7%	1488 74.4%	1618 80,9%
7 ;14 ;15	781 55,8%	80,2% 0.573	751 35,6%	1403 70,2%	1529 76,5%	1539 77,0%
19 ;18 ;19	731 52,2%	611 0.437	713 50,9%	1371 68,6%	1642 0.821	1566 78,3%
CTREE-UCB	1177 **84,1%**	1170 **83,6%**	971 62%	1753 **87,7%**	1832 **91,6%**	1582 **79,1%**
UNIFORM	455 32,5%	455 32,5%	455 32,5%	666 33,3%	666 33,3%	666 33,3%

TABLE 4.30 – Gain moyen pour DBA-CTREE-UCB et CTREE-UCB

DBA-LinUcb vs Lin-Ucb Les meilleurs résultats sont obtenus par DBA-LINUCB qui surpasse toutes les autres méthodes (voir figure 4.31). Les résultats sont particulièrement bons avec la $\text{Conf}_{\texttt{clust}100,100}$ et $Nb_x = 13$, $Nb_y = 19$ et $Nb_z = 3$ ou $Nb_x = 19$, $Nb_y = 18$ et $Nb_z = 19$. Quelque soit le paramétrage Nb du nombre de clusters, DBA-CTREE-UCB surpasse largement LIN-UCB.

DBA-LinUcb Nb cluster (x,y,z)	$Conf_{clust30,70}$	$Conf_{clust100,100}$
13 ;19 ;3	1064 76%	1851 92,6%
3 ;3 ;17	1144 **81,7%**	1752 87,6%
7 ;14 ;15	1029 73,5%	1792 89,6%
19 ;18 ;19	1045 74,6%	1855 **92,8%**
Lin-Ucb	927 66,2%	1297 64,9%
uniform	455 32,5%	666 33,3%

Table 4.31 – Gain moyen pour DBA-LinUcb et Lin-Ucb et uniform

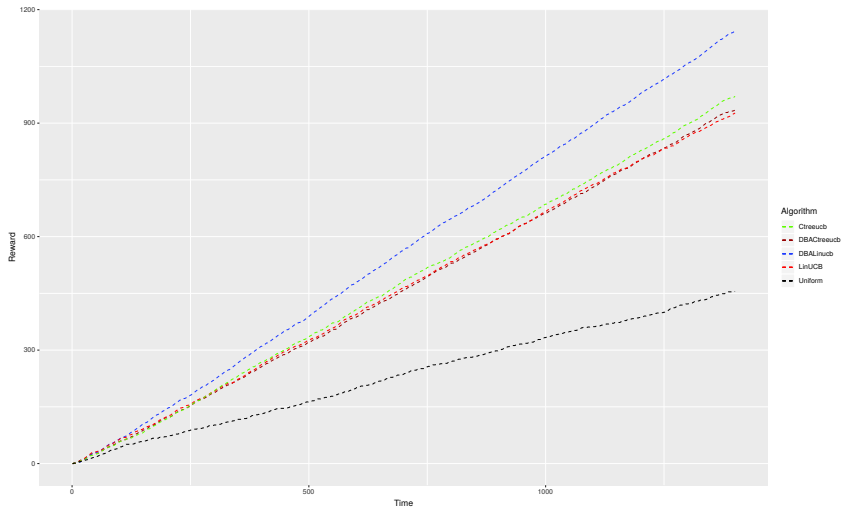


Figure 4.31 – Gain cumulé de DBA-Ctree-Ucb, DBA-LinUcb, Ctree-Ucb, Lin-Ucb et uniform avec $Conf_{30,70}$ et $V_A = A_1$

c) Analyse des groupes après clustering

Nous analysons ici les groupes après clustering et les activités des patients associées en entraînant CTREE sur l'intégralité des données. Nous observons si les clusters ou les moyennes des séries permettent d'identifier des groupes homogènes, c'est-à-dire ne présentant qu'une seule activité chacun.

La Fig. 4.32 présente l'abre généré par CTREE en utilisant les clusters. La Fig. 4.33 présente celui obtenu en utilisant que les moyennes. Dans la Fig. 4.32, 9 groupes sur les 14 identifiés prédisent une seule activité. Dans la Fig. 4.33, tous les groupes (14) identifiés prédisent une seule activité. En comparant avec les résultats de gain observés précédemment, il est difficile de battre un algorithme qui utilisera la moyenne. La justification que nous apportons est que la pré-classement via CTREE sur un apprentissage partiel en utilisant les clusters n'est pas pertinente dans ce cas et par conséquent, DBA-LinUcb où Ctree-Ucb présentent les meilleurs résultats.

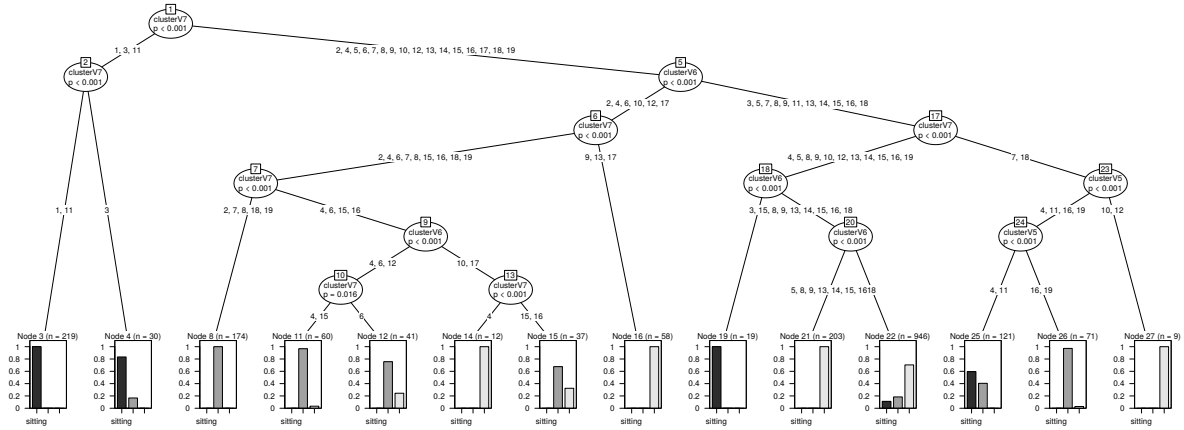


Figure 4.32 – Analyse de l'impact du test selon les clusters ($Config_{100,100}$), Localization data, $Nb_x = 19$, $y = 18$ et $z = 19$

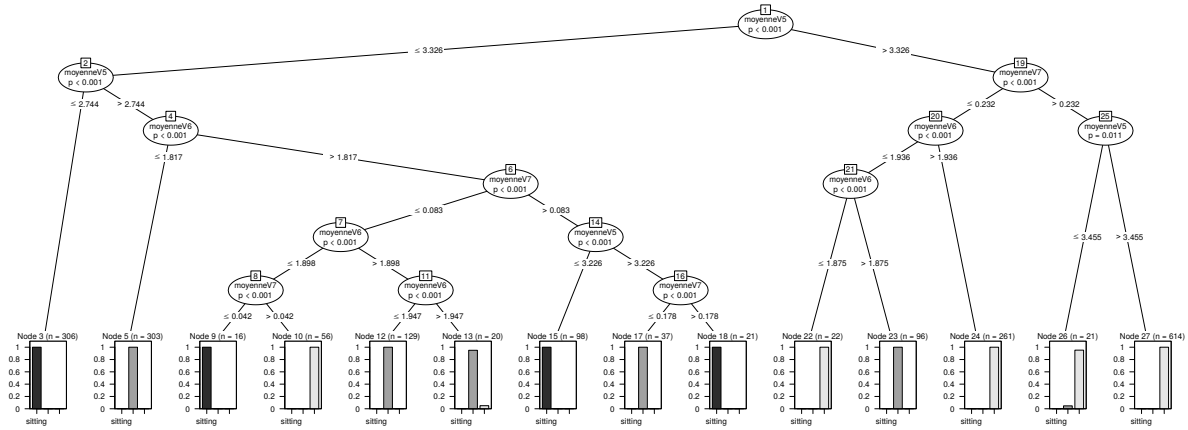


Figure 4.33 – Analyse de l'impact du test selon les moyennes ($Config_{100,100}$), Localization data, $x = 19$, $y = 18$ et $z = 19$

d) Conclusion

Dans cette expérimentation, DBA-LinUcb apparaît être la méthode la plus efficace si le nombre de clusters est correctement choisi. Ces résultats montrent que l'analyste peut utiliser DBA-LinUcb comme alternative si DBA-Ctree-Ucb n'obtient pas de bon résultats. Cette expérimentation montrent également que ces algorithmes peuvent être utilisés dans d'autres cas applicatifs (comme la prédiction de classe). L'indice vérifiant la vérité terrain (c'est-à-dire qui maximise le gain moyen) est l'indice COP.

4.6 Discussion

Nous avons constaté à travers nos expériences que l'intégration des séries temporelles sous forme de label de clusters dans le contexte des items permet d'augmenter fortement le gain pendant l'allocation dynamique. De plus, cette représentation des séries temporelles permet d'identifier plus facilement des patterns dans les items testés.

Nous avons aussi constaté que la méthode DBA-Ctree-Ucb rassemble des clusters dans des groupes si leur comportement face à l'objectif de test est identique. Fixer un nombre de clusters trop grand n'impacte que peu le modèle de classement car, s'ils ont une distribution de gain identique, ils seront classés dans le même groupe. Nos expériences montrent aussi que DBA-Ctree-Ucb permet de donner plus facilement une interprétation métier aux clusters. Par exemple, distinguer un visiteur "prospect parfait" d'un "visiteur arrivé par erreur" peut être difficile. En effet, leurs visites sont quasiment uniques avant d'arriver sur la page test. En couplant différents types de clusters (l'un rapportant les présences et l'autre le temps passé sur le site) et en prédisant leur probabilité de cliquer, nous pouvons discerner par exemple ces deux profils et savoir quelle est la variation la plus adaptée.

Par ailleurs, lorsque l'analyste dispose de séries décrivant chaque item, DBA-Ctree-Ucb semble être une méthode plus intéressante que Ctree-Ucb car nos expériences ont montré que les clusters permettent de construire des groupes plus homogènes (en termes de distribution de gain) que ceux obtenus en utilisant uniquement les moyennes des séries. En effet, dissocier les groupes lors de la construction de l'arbre par rapport à une variable continue, comme la moyenne d'une série rend l'arbre de classement très sensible aux valeurs extrêmes et donc aux données présentes dans l'apprentissage. Le modèle de classement utilisant les clusters évite cette sensibilité. On retrouve en effet, après avoir mélangé plusieurs fois les jeux de données, des arbres identiques.

Bien qu'aidant l'analyste dans le choix du nombre de clusters, les indices présents dans l'état de l'art peuvent se contredire et l'analyste peut alors ne pas savoir lequel choisir. En accord avec nos expérimentations, nous proposons une méthode de validation multiple

pour répondre à cette question cruciale :

— Les clusters doivent représenter des parcours caractéristiques des patterns énoncés par les recherches en e-marketing.

— Les clusters doivent maximiser le gain pendant l'allocation dynamique.

Avec ces deux critères, nous avons remarqué que l'indice de silhouette apparaissait comme le plus pertinent pour choisir le nombre de clusters lorsque l'on dispose d'une vérité-terrain (dans notre cas, les items, allocations et gains issus de l'exploitation de la variation originale A)

Nos expérimentations suggèrent que des profils types sont persistants à travers différents types d'e-commerce. Les études de marketing avaient déjà énoncé cette hypothèse, nous retrouvons effectivement dans deux expériences au contexte différent, un paramétrage identique pour définir le nombre de clusters Nb optimal. La série `presence_time_serie` par exemple, représente les entrées/sorties sur le site. C'est dans cette série que les comportements des visiteurs varient le plus, 17 profils étant suggérés dans les deux expérimentations. *A contrario*, les profils des séries `time_spend_serie` et `connexion_time_serie` sont peu diversifiés. En réalité, comme le disait [10], les gens se connectent généralement à la même heure et il semblerait qu'ils passent soit le même temps sur le site pour toutes leurs visites, soit qu'ils accroissent ou décroissent le temps passé au cours des visites. Ainsi, le paramétrage $Nb = 3$ clusters à fixer en paramètre pour ces deux séries semble pertinent en pratique et fait sens pour l'analyste.

Chapitre 5

Conclusion

L'approche proposée dans cette thèse a répondu aux objectifs cités en introduction. Le découpage en deux étapes (offline et online), ainsi qu'un apprentissage partiel sur la variation originale, a permis de transférer la connaissance acquise sur un jeu de données "passé" issu de l'exploitation de la variation originale A sur le test lui même. Ceci permet de mieux traiter le "nouveau jeu de données" obtenu lors de la phase d'exploration par affectation aux différentes variations (A, B, ...). Cette approche peut donc être vue comme une méthode de transfert learning [52]. En accord avec la définition des auteurs, le transfert learning est l'un des champs prometteur de recherche de l'apprentissage automatique et vise à transférer des connaissances d'une ou plusieurs tâches sources vers une ou plusieurs tâches cibles. Il peut être vu comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes.

Dans notre contexte, la connaissance en amont du test est un renseignement précieux pour l'analyste et permet d'optimiser l'AB Test. Au delà d'une simple explicitation de groupes permettant de limiter le coût du test, celle-ci aide directement l'analyste dans la composition de son test et lui permet de composer des variations en accord la population qui sera testée. L'utilisateur sera plus à même de définir des tests plus efficaces. Les méthodes que nous proposons présentent néanmoins des limites qu'il est important de présenter :

— La stationnarité : il est ici supposé que les distributions de gain pour chaque groupes sont stationnaires. Cela signifie que le temps n'influence pas les gains réels des variations. Or ceci est bien souvent non vérifié. Ainsi, par exemple, lors des soldes pour le e-commerce, les acheteurs potentiels ont tendance à attendre plutôt la fin de la période des soldes pour acheter).

— La corrélation entre les gains des différentes variations : la distribution de gain d'un groupe, que ce soit sur A ou B, suit la même loi (et la même variance) mais diffèrent éventuellement par leur moyenne. Si les groupes identifiés à partir de la variation A sont totalement différents de ceux identifiés sur B, le modèle de classement ne sera pas pertinent. Une solution possible à étudier serait de réactualiser dynamiquement les groupes et donc l'arbre de classement en fonction des items testés, des variations affectées et des gains obtenus lors de la phase d'exploration.

Enfin, des études plus poussées devront être menées afin de déterminer les garanties théo-

riques peuvent être faîte par DBA-Ctree-Ucb et DBA-LinUcb sachant que D.T.W n'est pas une distance mais une mesure de similarité.

Par ailleurs, des questionnements plus généraux sur les tests mériteraient d'être abordés tels que :

- **Types et qualité des données** : Quelles sont les informations que l'analyste peut collecter sur les items (âge, sexe, localisation, navigateur utilisé, ...) ? Ces données collectées sont elles conformes à la réalité des items ?
- **Homogénéité des items** : Les items observés avant, pendant et après le test peuvent-ils toujours être décris dans le même espace de caractéristiques et sinon, existe-il des solutions pour résoudre ce problème ?
- **Gains** : La différence de gain entre les variations influe fortement la performance des différentes méthodes. En effet, plus cette différences est faible, plus les techniques de comparaison de gains auront des difficultés à l'identifier. Comment évaluer a priori, même grossièrement, l'importance de cette différence afin de mieux choisir et paramétrer la méthode à utiliser ?
- **Reproductibilité vs variabilité** : L'expérience est-elle parfaitement reproductible, ou bien les résultats sont-ils sensibles à des phénomènes aléatoires ? Par exemple dans le processus de fabrication, les facteurs influant la qualité d'un objet sont généralement très contrôlés. *A contrario*, des personnes recevant un traitement ou une page web spécifique sont sensibles à des facteurs environnementaux inobservables pour l'analyste. Comment rendre compte de cette variabilité entre les récompenses observées pour des items ayant strictement le même contexte lorsqu'ils sont affectés à la même variation ? Il s'agit de la part d'aléatoire présente dans les résultats observés alors qu'une expérience reproductible ne souffre d'aucune variabilité.
- **Latence** : Dans le cas où l'analyste dispose de plus de temps pour assigner un item à une variation lors de l'allocation dynamique, quelles modifications peuvent être apportées à Ctree-Ucb, DBA-Ctree-Ucb et DBA-LinUcb pour encore les améliorer ?

Parallèlement d'autres domaines d'application des méthodes d'allocations dynamique pourraient être explorer, comme par exemple la recommandation. Dans ce cas, l'analyste s'intéresse uniquement à allouer ses items à (aux) la (les) variation(s) qui maximise(nt) les gains, sans s'intéresser à l'identification de la meilleure variation. Dans ce cas l'interprétation des résultats peut être inutile pour l'analyste, puisqu'il ne cherche pas à savoir pourquoi telle ou telle variation est optimale. De plus, les variations peuvent être de nature complètement différentes car correspondant à des articles voire des familles d'articles différents.

Enfin, des méthodes en statistique, dites « contrefactuels » [10] proposent d'estimer les gain de chaque variation avec peu d'expériences (c'est à dire peu d'items). Ces méthodes ont été très majoritairement abordées sous un angle théoriques et pourront, dans le futur, être potentiellement comparées aux algorithmes basés sur les bandits dans les AB tests

Toutes ces interrogations sont de pistes de recherche que nous tâcherons d'aborder dans nos travaux futurs.

Chapitre 6

Annexe

Notation

- $\mathcal{A}$ désigne l'ensemble des bras existant
- $|\mathcal{A}| = K$ désigne le nombre de bras possible.
- $a \in \{1, 2, \ldots, K\}$ est un bras parmi K.
- ν_{A_t} désigne la distribution de probabilité des récompenses X associées au bras A_t
- $\mu^* = \max_a \mu_a$ est la moyenne du bras maximisant les récompenses en moyenne.
- $a^* = \text{argmax}_a \, \mu_a$.
- A_t est le bras choisi à l'instant t.
- X_{A_t} variable i.i.d qui représente une récompense observée du bras A_t.
- $N_a(t)$ est le nombre de tirages du bras a jusqu'à l'instant t.
- π est une stratégie d'allocation dynamique.
- T désigne le nombre total d'item testés.
- k est un groupe d'items.
- $\mathcal{K}$ désigne l'ensemble des groupes existants.
- c_t est un vecteur décrivant le contexte d'un item à l'instant t (également dénoté plus généralement par Y).
- d désigne la dimension du vecteur de contexte.
- Nb_s est un nombre de clusters existant pour les série s.

Ucb algorithm

Critères de qualité pour le clustering de séries temporelles

Évaluation à travers différents indices de qualité des clusters avec $\text{Conf}_{100,100}$
Apprentissage sur l'intégralité des données

Génération des séries temporelle par simulation La fonction `sample.int(a,b)` tire aléatoirement un entier entre `a` et `b`. La variable x correspond à un pas de temps et la variable y la valeure associée.

```
x = 0:sample.int(100,1)                    # sequence of integers
```

Algorithm 8 UCB algorithm

Require: $\alpha > 0$
Require: Affecter au moins un item au bras a
1: **loop**
2: $c_t \leftarrow$ un nouvel item
3: $A_t = \text{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,t} + \alpha \sqrt{\frac{2 * log(t)}{N_a(t)}}\}$
4: Affecter A_t à c_t
5: $X_{c_t,A_t} \leftarrow$ la récompense du bras A_t
6: Mettre à jour $\hat{\mu}_{A_t}$ et $N_{A_t}(t)$
Output : Une séquence de choix et de récompenses

Nb	3	4	5	6	7	8	9	10	11	12	3	14	15	16	17	8	19	20
Sil	**0.35**	0.25	0.25	0.22	0.21	0.23	0.24	0.25	0.23	**0.27**	0.25	**0.29**	0.23	0.24	0.27	0.25	0.24	0.24
SF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CH	**793.40**	**1349.15**	**847.18**	1076.30	399.59	575.18	576.29	837.95	895.26	395.33	305.50	453.94	321.44	291.83	327.16	391.81	602.07	560.29
DB	1.70	**0.90**	1.30	1.49	1.63	1.32	**1.09**	1.15	1.25	1.17	1.50	1.23	1.21	1.25	1.18	1.28	**1.11**	1.23
DBstar	2.04	**1.07**	1.77	1.79	2.09	1.68	1.63	**1.56**	2.54	2.28	2.32	2.04	2.48	2.99	2.34	3.32	2.46	3.10
D	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
COP	0.26	0.18	0.17	0.16	0.14	0.14	0.14	0.13	0.12	0.10	0.11	0.10	**0.09**	**0.09**	**0.09**	0.09	0.09	0.09

TABLE 6.1 – Évaluation à travers différents indices de qualité des clusters sur la série connexion_time du jeu de données Modz, avec la méthode de clustering D.B.A. ($\text{Conf}_{100,100}$)

```
    y = parameter*cos(x+sample.int(10,1) ) + sample.int(100,1)
# value
```

Liste des tableaux

Table des figures

Bibliographie

[1] Sr Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Wah. Time-series clustering - a decade review. *Information Systems*, 53, 05 2015.

[2] Piyush Shanker Agram and A. N. Rajagopalan. Off-line signature verification using dtw. *Pattern Recognition Letters*, 28 :1407–1414, 2007.

[3] Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, pages 69–84, London, UK, UK, 1993. Springer-Verlag.

[4] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús Pérez, and Iñigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46 :243–256, 01 2013.

[5] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. *Math. Oper. Res.*, 39(1) :31–45, February 2014.

[6] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2) :235–256, May 2002.

[7] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1) :48–77, 2002.

[8] E. Dovgan M. Lustrek M. Gams B. Kaluza, V. Mirchevska. UCI machine learning repository, an agent-based approach to care in independent living, 2010.

[9] Hamsa Bastani and Mohsen Bayati. Online decision-making with high-dimensional covariates. *SSRN Electronic Journal*, 01 2015.

[10] Léon Bottou, Jonas Peters, Joaquin Qui nonero Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems : The example of computational advertising. *Journal of Machine Learning Research*, 14 :3207–3260, 2013.

[11] Stéphane Boucheron, Gabor Lugosi, and Pascal Massart. *Concentration inequalities : a non asymptotic theory of independence*. Oxford University Press, 2013.

[12] George Edward Pelham Box and Gwilym M. Jenkins. *Time Series Analysis : Forecasting and Control*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 3rd edition, 1994.

[13] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[14] S. Bubeck, V. Perchet, and P. Rigollet. Bounded regret in stochastic multi-armed bandits. *ArXiv e-prints*, February 2013.

[15] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1) :1–122, 2012.

[16] Giuseppe Burtini, Jason Loeppky, and Ramon Lawrence. A survey of online experiment design with the stochastic multi-armed bandit. *CoRR*, abs/1510.00757, 2015.

[17] Tadeusz Caliński and Harabasz JA. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3 :1–27, 01 1974.

[18] Alberto Casagrande, Kevin Casey, Rachele Falchi, Carla Piazza, Benedetto Ruperti, Giannina Vizzotto, and Bud Mishra. Translating time-course gene expression profiles into semi-algebraic hybrid automata via dimensionality reduction. In Hirokazu Anai, Katsuhisa Horimoto, and Temur Kutsia, editors, *Algebraic Biology*, pages 51–65, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[19] Nicolò Cesa-Bianchi, Yishay Mansour, and Ohad Shamir. On the complexity of learning with kernels. *CoRR*, abs/1411.1158, 2014.

[20] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 208–214, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[21] A. Cornuejols, C. Wemmert, P. Gan, and Y. Bennani. Collaborative clustering : Why, when, what and how. *Information Fusion*, 39 :81–95, Apr 2017.

[22] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3) :32–57, 1973.

[23] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293) :52–64, 1961.

[24] Adam N. Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. A practical method for solving contextual bandit problems using decision trees. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*, 2017.

[25] Adam N. Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. A practical method for solving contextual bandit problems using decision trees. *CoRR*, abs/1706.04687, 2017.

[26] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits : The generalized linear case. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 586–594. Curran Associates, Inc., 2010.

[27] R.A. Fisher. *Statistical methods for research workers.* Edinburgh Oliver & Boyd, 1925.

[28] R.A. Fisher. *The design of experiments. 1935.* Oliver and Boyd, Edinburgh, 1935.

[29] Dylan J. Foster, Alekh Agarwal, Miroslav Dudík, Haipeng Luo, and Robert E. Schapire. Practical contextual bandits with regression oracles. In *ICML*, 2018.

[30] Raphaël Féraud, Robin Allesiardo, Tanguy Urvoy, and Fabrice Clérot. Random forest for the contextual bandit problem. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 93–101, Cadiz, Spain, 09–11 May 2016. PMLR.

[31] Axel Gandy and Georg Hahn. A Framework for Monte Carlo based Multiple Testing. *arXiv e-prints*, page arXiv :1402.3019, Feb 2014.

[32] Aurélien Garivier, Hédi Hadiji, Pierre Menard, and Gilles Stoltz. KL-UCB-switch : optimal regret bounds for stochastic bandits from both a distribution-dependent and a distribution-free viewpoints. working paper or preprint, May 2018.

[33] Aurélien Garivier, Emilie Kaufmann, and Tor Lattimore. On Explore-Then-Commit Strategies. In *NIPS*, volume 29 of *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain, December 2016.

[34] Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In Sham M. Kakade and Ulrike von Luxburg, editors, *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19 of *Proceedings of Machine Learning Research*, pages 359–376, Budapest, Hungary, 09–11 Jun 2011. PMLR.

[35] D. M. Gavrila and L. S. Davis. Towards 3-d model-based tracking and recognition of human movement : a multi-view approach. In *In International Workshop on Automatic Face- and Gesture-Recognition. IEEE Computer Society*, pages 272–277, 1995.

[36] J.C. Gittins and D.M. Jones. A dynamic allocation index for the sequential design of experiments. In J. Gani, editor, *Progress in Statistics*, pages 241–266. North-Holland, Amsterdam, 1974.

[37] L Gupta, Dennis Molfese, Ravi Tammana, and Panagiotis Simos. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE transactions on bio-medical engineering*, 43 :348–56, 05 1996.

[38] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets : History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4) :19 :1–19 :19, December 2015.

[39] Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13(4) :18–28, July 1998.

[40] Torsten Hothorn, Kurt Hornik, Mark van de Wiel, and Achim Zeileis. Implementing a class of permutation tests : The coin package. *Journal of Statistical Software, Articles*, 28(8) :1–23, 2008.

[41] Torsten Hothorn, Kurt Hornik, Mark A van de Wiel, and Achim Zeileis. A lego system for conditional inference. *The American Statistician*, 60(3) :257–263, 2006.

[42] Torsten Hothorn, Kurt Hornik, Mark A van de Wiel, and Achim Zeileis. A lego system for conditional inference. *The American Statistician*, 60(3) :257–263, 2006.

[43] Torsten Hothorn, Universität München, Kurt Hornik, Wirtschaftsuniversität Wien, Achim Zeileis, and Wirtschaftsuniversität Wien. party : A laboratory for recursive partytioning.

[44] M N Katehakis and H Robbins. Sequential choice from several populations. *Proceedings of the National Academy of Sciences*, 92(19) :8584–8585, 1995.

[45] A. Ketterlin, P. Gançarski, and J. J. Korczak. Conceptual clustering in structured databases : A practical approach. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, KDD'95, pages 180–185. AAAI Press, 1995.

[46] Minho Kim and R.S. Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26 :2353–2363, 11 2005.

[47] David L. Davies and Don Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1 :224 – 227, 05 1979.

[48] T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1) :4 – 22, 1985.

[49] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press (preprint), 2019.

[50] Odalric-Ambrym Maillard and Shie Mannor. Latent Bandits., January 2014. Extended version of the paper accepted to ICML 2014 (paper and supplementary material).

[51] Vit Niennattrakul and Chotirat Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. pages 733–738, 01 2007.

[52] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10) :1345–1359, October 2010.

[53] François Petitjean. Description des alignements formés par DTW. working paper or preprint, August 2011.

[54] FRANÇOIS PETITJEAN, FLORENT MASSEGLIA, PIERRE GANÇARSKI, and GERMAIN FORESTIER. Discovering significant evolution patterns from satellite image time series. *International Journal of Neural Systems*, 21(06) :475–489, 2011. PMID : 22131300.

[55] François Petitjean. *Dynamic time warping : apports théoriques pour l'analyse de données temporelles : application à la classification de séries temporelles d'images satellites*. PhD thesis, 2012. Thèse de doctorat dirigée par Gançarski, Pierre Informatique Strasbourg 2012.

[56] François Petitjean, Alain Ketterlin, and Pierre Gancarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44 :678–, 03 2011.

[57] Nadia Pomirleanu, John Schibrowsky, James Peltier, and Alexander Nill. A review of internet marketing research over the past 20 years and future research direction. *Journal of Research in Interactive Marketing*, 7, 08 2013.

[58] Yi Qi, Qingyun Wu, Hongning Wang, Jie Tang, and Maosong Sun. Bandit learning with implicit feedback. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7276–7286. Curran Associates, Inc., 2018.

[59] J. Ross Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[60] Toni M. Rath and R. Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series. 2003.

[61] Serge Roukine. *Améliorer ses taux de conversion web : Vers la performance des sites web au-delà du webmarketing*. 2011.

[62] Peter Rousseeuw. Rousseeuw, p.j. : Silhouettes : A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. *Journal of Computational and Applied Mathematics*, 20 :53–65, 11 1987.

[63] Sandro Saitta, Benny Raphael, and Ian F. C. Smith. A bounded index for cluster validity. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, pages 174–187, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[64] A. Salomon, J.-Y. Audibert, and I. El Alaoui. Regret lower bounds and extended Upper Confidence Bounds policies in stochastic multi-armed bandit problem. *ArXiv e-prints*, December 2011.

[65] Alexis Sarda-Espinosa. dtwclust : Time series clustering along with optimizations for the dynamic time warping distance.

[66] Lisa Schlosser, Torsten Hothorn, and Achim Zeileis. The power of unbiased recursive partitioning : A unifying view of ctree, mob, and guide. 2019.

[67] Y.-S Shih. A note on split selection bias in classification trees. *Computational Statistics & Data Analysis*, 45(3) :457–466, 2004.

[68] Helmut Strasser and Christian Weber. On the asymptotic theory of permutation statistics. 1999.

[69] Richard S. Sutton and Andrew G. Barto. Reinforcement learning : An introduction. *IEEE Transactions on Neural Networks*, 16 :285–286, 1998.

[70] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4) :285–294, 1933.

[71] Michel Tokic and Günther Palm. Value-difference based exploration : Adaptive control between epsilon-greedy and softmax. In Joscha Bach and Stefan Edelkamp, editors, *KI 2011 : Advances in Artificial Intelligence*, pages 335–346, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[72] Michal Valko, Nathan Korda, Rémi Munos, Ilias Flaounas, and Nello Cristianini. Finite-time analysis of kernelised contextual bandits. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 654–663, Arlington, Virginia, United States, 2013. AUAI Press.