Thèse
de doctorat
de l'UTT

Yoann VALERO

Intelligence artificielle pour la modélisation de parcours individuels

Champ disciplinaire :
Sciences pour l'Ingénieur

2023TROY0037 **Année 2023**

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR
de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

en SCIENCES POUR L'INGENIEUR

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Yoann VALERO

le 6 novembre 2023

Intelligence artificielle pour la modélisation de parcours individuels

JURY

M. Stéphane CHRÉTIEN	PROFESSEUR DES UNIVERSITES	Président
Mme Aurélie FISCHER	MAITRE DE CONFERENCES - HDR	Rapporteure
M. Gilbert SAPORTA	PROFESSEUR EMERITE CNAM	Rapporteur
Mme Christine KERIBIN	MAITRE DE CONFERENCES - HDR	Examinatrice
M. Nicolas LACHICHE	PROFESSEUR DES UNIVERSITES	Examinateur
M. Nicolas WICKER	PROFESSEUR DES UNIVERSITES	Examinateur
M. Frédéric BERTRAND	PROFESSEUR DES UNIVERSITES	Directeur de thèse
Mme Myriam MAUMY-BERTRAND	MAITRE DE CONFERENCES - HDR	Directrice de thèse

Personnalités invitées

M. Simon PIOCHE	DIRECTOR OF PRODUCT MANAGEMENT QAD PROCESS INTELLIGENCE
M. François ROSSET	HEAD OF AI QAD PROCESS INTELLIGENCE

Remerciements

Mes remerciements sont tout d'abord adressés à mes rapporteurs, Gilbert Saporta et Aurélie Fischer, pour avoir accepté si promptement de rapporter ma thèse. Je remercie également Christine Keribin, Nicolas Wicker, Nicolas Lachiche et Stéphane Chrétien d'avoir accepté d'être examinateurs dans mon jury de thèse.

Avec une fierté non dissimulée, je remercie chaleureusement mes directeurs de thèse, Frédéric Bertrand et Myriam Maumy. Vous m'avez permis d'atteindre un niveau d'études que je n'osais considérer il y a seulement quatre ans. Le sujet que vous m'avez proposé, novateur et ouvert sur tellement d'horizons ; la passion de la recherche ; l'ouverture à l'international... La liste serait bien trop longue pour pouvoir être exhaustive dans ce manuscrit. Frédéric, nos échanges scientifiques et la richesse de tes idées ont été un *impetus* considérable pour ma créativité scientifique. Myriam, nos discussions sur la nature de la statistique, de la science, de l'éthique, de la responsabilité du chercheur, ont formé mon esprit à la fois à l'ouverture et à la rigueur réfléchies. Nos discussions tous ensemble m'ont formé pour faire de moi le professionnel que je suis aujourd'hui. Je n'ai jamais autant appris et aimé apprendre. Pour cela, et tout le reste encore, merci.

Reclus dans mon appartement lors du confinement, arrivé sitôt après mon embauche, j'ai pourtant été accueilli à bras ouvert et inclus si agréablement par l'équipe de Livejourney, qui a fait le grand plongeon avec moi dans une thèse qui était aussi nouvelle pour moi que pour eux. En particulier, je souhaite remercier Simon Pioche et François Rosset. Simon, ta confiance en mes capacités, mon autonomie, et tes encouragements tout au long de la thèse ont fait de ce doctorat un vrai plaisir à poursuivre. François, tu m'as tout appris sur la fouille de processus, tu m'as initié à Python, et traité comme un égal chaque seconde de ce doctorat. Merci à François Arnaud, Denis Casselle, William Moustrou et Damien Thirion pour leur suivi de mes travaux et leur *maestria* pour avoir incorporé avec brio mes outils dans le logiciel et poussé ma réflexion vers le client, au-delà de la recherche. Merci à tous pour ce sentiment de réel confort, d'appartenance et de reconnaissance dans cette entreprise, dans laquelle j'ai appris à allier recherche et production avec plaisir et efficacité.

Impossible d'oublier les collègues du LIST3N : Mélanie Piot, Benoit Vuillemin et Yufei Gong, vous avez été là depuis la première heure, jusqu'à la dernière. Preuve que les épreuves créent la camaraderie, et que le réconfort la renforce. Je remercie Insun Choi, voix de la

raison dans beaucoup de nos joviales dérives. Merci également à Bernadette André et Véronique Banse, absolus piliers du secrétariat du laboratoire, à qui je dois la simplicité et la flexibilité de chacune de mes missions, ici comme à l'étranger. Je remercie Lionel Amodeo et Antoine Grall pour leur bienveillance à la direction du laboratoire et de l'École Doctorale respectivement. Je remercie sincèrement les membres de mon conseil de suivi individuel : Edith Grall, Blaise Kevin Guepie, Faicel Hnaien et Christian Derquenne pour leur suivi, leurs idées de qualité et nos échanges lors des différentes phases de suivi. Dans un contexte académique plus large, je remercie Franck Velikonia pour avoir perçu chez moi une appétence pour les mathématiques, et Michel Bourguet pour l'avoir définitivement fait éclore. De même, je remercie Frédéric Jean-Pierre pour son amour des sciences au sens large qui m'a animé depuis le lycée. Il m'est important de remercier Elisabeth Remm, première à flairer chez moi dès la licence un réel potentiel en Statistique.

Enfin, je tiens à remercier ma famille et mes amis. Merci Kévin et Anaïs pour votre soutien indéfectible et les week-ends entiers à réellement déconnecter, soigner ma psyché. Famille, amis, vous êtes autant l'un que l'autre. Merci Nathalie, dont le courage face à mes avalanches de réflexions et d'introspections me fut à la fois salutaire et parfaitement mystérieux. Merci à mes oncles et tantes, Juliette, Alain, Jeannine et Bernard, et à ma cousine Catherine, car une famille proche et soudée est pour moi le ciment du bien-être. Merci à Julianne, ma filleule, pour ta force d'esprit, ta joie de vivre et ta compréhension, qui rendent les moments passés ensemble réellement relaxants. Puissent tes projets te mener où ton potentiel se dessine, avec passion et épanouissement.
Stéphane et Béatrice, en plus de me soutenir vous avez été là dans les moments les plus improbables, venus me récupérer jusqu'en Suisse quand rien ne semblait se dérouler comme prévu. Merci Hubert pour ta bonne humeur, et Séverine pour ton mentorat et pour avoir été la première à me mettre la puce à l'oreille vis-à-vis d'un futur doctorat. Camille, Chloé, Émeline, Anthony et Alexandre, merci pour ces moments passés ensemble depuis, ma foi, presque toujours ! On se regarde tous grandir, je ne peux qu'avoir hâte de voir où vos pas vous mèneront ensuite.
Merci à ma famille de Toulouse, qui malgré la distance garde un contact constant. Clara, je n'ai aucune estimation finie de la limite supérieure de ta passion et de ta compétence pour ta discipline, et je ne peux qu'anticiper le moment où toi aussi, tu seras Docteur. Plus lointaine encore, merci mamie pour ton soutien constant et ta sagesse, j'ai à cœur de pérenniser les valeurs que tu m'as inculquées, sans faillir.
Le point d'orgue est réservé à mon père et à ma mère : d'abord, merci à vous deux pour les conditions idéales que vous m'avez fournies tout au long de cette thèse, m'offrant un confort inégalable dans le présent et me permettant de me construire un avenir réellement serein. Vous avez été là à chaque instant, peu importe la difficulté, mon tempérament, mes hauts comme mes bas. Vous m'avez permis de sortir grandi, accompli, de cette expérience unique. Vous avez été fiers de moi tout du long, et cela, c'est quelque chose qui me rend fier, moi.

Résumé

Les processus d'entreprises sont créés afin de transformer des éléments d'entrée en éléments de sortie de sorte à ce que ces éléments de sortie contribuent aux opérations de l'entreprise. Ces éléments sont généralement appelés « unités » (par exemple : un colis dans un processus de livraison). Lors de leurs différents mouvements dans un processus donné, ces unités sont enregistrées informatiquement dans des journaux d'événements. *In fine*, ces journaux d'événements contiennent un ensemble d'événements retraçant les parcours de ces unités dans ces processus, de façon horodatée. Les outils classiques d'analyse et de fouille de ces processus permettent leur modélisation dans une optique informative sur leur passé et leur présent, voire sur leur futur. Les modèles typiques tels que les réseaux de Petri permettent des prédictions, mais celles-ci concernent avant tout l'ensemble du processus plutôt que les unités elles-mêmes : des événements sont prédits, sans pouvoir être reliés à des unités précises. Pour une entreprise cherchant à maximiser sa compréhension et son anticipation du futur de ses processus, la possibilité d'examiner les prédictions sur chaque unité est obligatoire.
Plusieurs approches, en particulier en apprentissage profond, proposent la prédiction de parcours de nouvelles unités, mais les modèles existants ne maximisent pas l'utilisation de la donnée, adoptant majoritairement des approches analogues au traitement du langage naturel.
Dans ce contexte, cette thèse a pour premier objectif de compléter l'utilisation de la donnée grâce à la création de variables décrivant les processus dans leur globalité, puis une manière de les incorporer à un modèle prédictif adapté afin d'obtenir de meilleures prédictions des parcours des unités.
Ensuite, l'utilisation de ces variables conjointement au modèle créé dans ce but prédictif est étendue à un aspect simulatoire, menant ensuite à une analyse de sensibilité du modèle relativement à ces variables.
Le dernier objectif, connexe, concerne l'établissement d'une méthode permettant de contrôler la dérive de la donnée dans les journaux d'événements.

Dans un premier temps, la notion de journal d'événements ainsi que ses différents angles et échelles d'analyse sont exposés. Cinq jeux de données réels et publics y sont décrits dans le détail, ceux-ci étant utilisés dans l'intégralité de cette thèse.
Ensuite, les variables globales au centre de cette thèse sont définies et étudiées sur la donnée

réelle, tant dans leur apparence que dans leurs corrélations.
Le modèle prédictif utilisé, itération et amélioration d'un modèle pré-existant et consistant en le meilleur candidat pour la prédiction de séquences entières d'événements, est expliqué dans le détail, de même que les changements d'architecture nécessaires pour l'utilisation des variables globales précédemment créées. Il y est montré une augmentation notable de la précision du modèle par la simple incorporation de ces variables, tant dans les prédictions de séquences d'actions effectuées par les unités, que dans les prédictions des durées de ces actions.
Cette modélisation, à présent basée sur ces variables, mène naturellement à la simulation à partir de ces variables. Afin d'exposer les capacités simulatoires du modèle ainsi que l'efficacité de la méthode permettant d'arriver à cette fin, la simulation est illustrée grâce à une analyse de sensibilité basée sur les plans d'expériences et l'échantillonnage équilibré des variables globales créées dans cette thèse.
Enfin, nous explorons l'établissement d'une méthode permettant une définition de la dérive conceptuelle dans les journaux d'événements, ainsi que sa quantification et la possibilité d'établir des alertes à l'aide de cartes de contrôle. Cette approche est scindée en deux sous-parties : la première se base exclusivement sur la donnée, afin de détecter la dérive conceptuelle de la donnée brute. La seconde concerne la dérive conceptuelle relativement au modèle prédictif établi dans cette thèse. La première méthode est utilisable dans tout journal d'événements, et la deuxième avec tout modèle prédictif.

L'ensemble des avancées dans cette thèse visent en l'établissement d'un environnement prédictif complet, simple dans son implémentation, performant en termes de puissance prédictive et d'aisance dans la phase d'apprentissage, ainsi que contrôlable de façon simple, exhaustive et facilement interprétable.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Motivations

1.1.1 Définition d'un processus

Un processus, dans sa définition de la norme ISO 9000 :2015, section 3.4.1, est défini comme un « *ensemble d'activités corrélées ou en interaction qui utilise des éléments d'entrée pour produire un résultat escompté* ». On peut citer, comme exemples de processus :

— **Construction d'une pièce industrielle**
Entrée : matières premières
Activités : construction grâce aux machines et systèmes disponibles
Sortie : pièce usinée

— **Facturation**
Entrée : achat d'un bien ou d'un service
Activités : étapes permettant le paiement final du bien ou du service
Sortie : facture émise et payée

— **Processus hospitalier**
Entrée : nouveau patient
Activités : passages dans les différents services
Sortie : sortie du patient

— **Transports en commun**
Entrée : transport au départ
Activités : stations / gares / arrêts
Sortie : transport au terminus

Ces processus sont de natures différentes, remplissent une multiplicité d'objectifs, et sont d'une complexité particulièrement variée : un processus de transport en commun a tendance à être rectiligne dans son exécution, le véhicule suivant un chemin prédéfini, tandis qu'un processus hospitalier contient toute la complexité médicale des patients à traiter. De même, un processus ne nécessite pas toujours d'entités physiques en entrée ou en sortie, tout comme les activités ne sont pas nécessairement des objets ou étapes physiques / matérielles telles qu'une machine ou un centre de tri : une démonstration mathématique, en prenant le cas d'une démonstration par récurrence par exemple, a pour entrée une proposition, comme activités les étapes d'initialisation, d'hérédité et de conclusion, puis en sortie la véracité de la proposition. Notons par ailleurs que le détail des activités devient discutable : l'initialisation, l'hérédité et la conclusion contiennent des sous-étapes de calcul et de raisonnement. Poussant l'exemple plus avant, une démonstration par récurrence peut n'être qu'une étape d'une démonstration plus large, la récurrence entière pouvant alors constituer une activité dans une suite de sous-démonstrations. Ainsi, une séquence de processus peut elle-même constituer un processus, et un processus peut être considéré comme une activité dans cette séquence. Les termes fluctuent selon le contexte et l'échelle d'étude souhaités.
On en déduit que le terme « processus » englobe toute forme de chaîne d'exécution organisée ayant un but final bien défini, sans formaliser le détail des activités, le caractère imbriqué de potentiels sous-processus, ni même une série de processus.

L'une des difficultés concernant l'ampleur de cette définition est la notion d'unité, également appelée instance de processus. En effet, il est évident lors de la manufacture d'une pièce industrielle, que la pièce usinée constitue une unité du processus concerné. Dans un processus hospitalier, l'unité est le patient. Pour un transport en commun, il s'agit du véhicule. En revanche, pour une démonstration mathématique, la notion d'unité, correspondant ici à notre réflexion, devient abstraite.

Cette thèse se focalise ainsi sur un sous-ensemble de processus, spécifique aux entreprises et organisations : les processus d'affaires, également appelés processus métiers.

1.1.2 Processus métiers

Un processus métier est un processus dont la raison d'être est de mener à bien les affaires d'une organisation, d'après la norme ISO/IEC 19510 :2013.
Les processus listés dans la section 1.1.1 correspondent donc tous à des processus métiers, contrairement à une démonstration mathématique (bien que l'on pourrait argumenter dans le sens de démonstrations dans une visée de recherche et développement).

L'avantage des processus métiers est qu'ils sont ainsi définis de manière fixe par l'entreprise qui les met en place : les problèmes d'échelle, de contexte, de séquences de processus ou de processus imbriqués sont ainsi limités, bien que toujours possibles.

Une façon normée, justement décrite par la norme ISO/IEC 19510 :2013, de visualiser un processus métier, réside en son modèle BPMN (*Business Process Model and Notation*). Il s'agit d'une carte représentant les départs possibles dans le processus, les activités reliées de sorte à créer des chemins, jusqu'aux sorties possibles.

Ces processus sont le plus souvent enregistrés informatiquement, créant un type de données appelé un « journal d'événements ». Comme son nom l'indique, un journal d'événements est un ensemble d'événements. Ceux-ci sont *a minima* composés de trois variables :

— une variable d'identifiants uniques, permettant d'identifier les unités circulant dans un processus donné,

— une variable d'activités, détaillant les activités par lesquelles les unités circulent,

— une variable d'horodatages, spécifiant à quelle date(s) et heure(s) une unité est passée par une activité donnée.

Un événement est ainsi caractérisé, de façon nécessaire et suffisante, par un triplet répondant aux questions « qui ? », « où ? » et « quand ? ». Un journal d'événements, qui forme un ensemble de ces triplets sous la forme d'un tableau où chaque ligne représente un événement, peut être rangé dans différents ordres : par exemple par ordre chronologique. Le rangement le plus courant est l'ordre chronologique par unité : tous les événements caractérisés par une même unité sont rangés les uns à la suite des autres chronologiquement, puis les unités elles-mêmes sont rangées chronologiquement selon l'horodatage de leur première activité.

Tableau 1.1 – Exemple factice de journal d'événements contenant deux unités et un seul horodatage

Unité	**Identifiant**	**Activité**	**Date et heure de début**
Unité 1	Id_1	A_1	2023-03-31 18:37:00
Unité 1	Id_1	A_2	2023-04-02 12:10:00
Unité 1	Id_1	A_3	2023-04-02 17:58:00
Unité 2	Id_1	A_1	2022-12-31 20:01:00
Unité 2	Id_1	A_3	2023-01-05 06:36:00

Le tableau 1.1 montre quelques lignes d'un tel journal d'événements. Nous y observons les variables d'identifiant, d'activité et d'horodatage, un événement constituant une ligne du tableau. Les événements sont ordonnés entre eux chronologiquement par unité.
La séquence chronologique des événements d'une même unité constitue son *parcours*. Ainsi, nous avons deux unités Unité 1 et Unité 2 dont les parcours sont caractérisés par la séquence d'activités A_1, A_2, A_3 et A_1, A_3 respectivement.

Il est possible d'avoir des journaux d'événements comportant deux colonnes d'horodatages, comme dans le tableau 1.2. Dans ce cas, le premier indique la date et l'heure de début d'un événement, et le deuxième en indique sa date et son heure de fin.

Tableau 1.2 – Exemple factice de journal d'événements avec deux unités et deux horodatages

Unité	**Identifiant**	**Activité**	**Début**	**Fin**
Unité 1	Id_1	A_1	2023-03-31 18:37:00	2023-04-01 17:00:00
Unité 1	Id_1	A_2	2023-04-02 12:10:00	2023-04-14 15:12:00
Unité 1	Id_1	A_3	2023-04-12 12:58:00	2023-04-17 21:28:00
Unité 2	Id_2	A_1	2022-12-31 20:01:00	2023-01-02 15:46:00
Unité 2	Id_2	A_3	2023-01-05 06:36:00	2023-01-06 04:20:00

C'est ici que le titre de cette thèse prend son sens : l'objectif est d'être capable de modéliser les parcours des unités d'un processus métier, en particulier à l'aide d'apprentissage profond. Entre autres, nous devons être capables de prédire les événements restants pour une unité dont le parcours n'est pas complet. À noter que l'incomplétude d'un parcours n'impliquera jamais, dans cette thèse, d'événements passés manquants, mais bien d'événements futurs encore non observés.

1.2 Contexte en entreprise

L'entreprise Your Data Consulting, aujourd'hui QAD Process Intelligence suite à son rachat par l'entreprise californienne QAD, est à l'origine du produit Livejourney™ (abrégé "LJ"). Il s'agit d'un logiciel d'analyse de processus métiers, permettant leur étude en termes de temps d'exécutions, de flux d'unités, et de l'agencement des activités. Il permet la définition d'indicateurs clefs de performances des processus métiers ainsi que leur suivi relativement aux angles d'études cités ci-avant grâce à différents outils et tableaux de bords.

La figure 1.1 montre ce qu'un utilisateur apercevrait en premier dans le logiciel suite à l'import d'un journal d'événements. Nous y distinguons des activités dans les cases grises, des arcs reliant les activités entre elles selon les flux présents dans le journal d'événements, ainsi que des billes qui circulent entre les activités représentant les unités circulant dans le processus.

Livejourney™ , ici dans sa version 5.0.8 déployée le 12/07/2023, offre la possibilité d'analyser un processus selon des caractéristiques au choix de l'utilisateur à l'aide de *« tags »*, vocabulaire propre à Livejourney™ et n'apparaissant pas dans le reste de ce manuscrit, permettant d'isoler les unités correspondant aux *tags* choisis, de circuler entre les *tags*, et même de les superposer. La figure 1.2 donne un aperçu de l'analytique des différents types

de parcours, des plus fréquents au moins fréquents. Nous y voyons leur séquence d'activités respective, le nombre d'étapes, leur durée totale ainsi que la fréquence de ces parcours dans le journal d'événements. Ce module, appelé le « traceur », permet également de sélectionner des parcours contenant certaines séquences d'activités au choix de l'utilisateur grâce au champ situé en-bas de l'écran. Le cadran en-bas à droite s'adapte automatiquement et affiche le nombre d'unités contenant les séquences choisies, ainsi que des statistiques positionnelles et de dispersion de leurs durées.

Enfin, nous avons sur la figure 1.3 un module permettant l'identification d'anomalies (dans le sens de non-conformité au modèle de processus théorisé en amont par une entreprise cliente) telles que des boucles : 33,60% des unités passent par l'activité "PRET", et parmi elles, 38,96% repassent immédiatement par la même activité. Il s'agit de l'étape la plus fréquente suivant l'activité "PRET", montrant une anomalie fréquente dans le processus en ce qui concerne cette activité.

Le logiciel propose donc un ensemble d'outils poussés d'analytique du présent et du passé d'un processus. Vient alors l'intérêt applicatif de la thèse, : la particularité de Livejourney™ est de mettre l'accent sur l'apprentissage machine et l'apprentissage profond [LBH15] dans une optique prédictive, afin d'apporter sa force analytique au futur des processus, prédit par un modèle fait sur-mesure pour le logiciel.

À cet égard, la surveillance prédictive des processus métier (*PBPM* : *Predictive Business Process Monitoring*) vise à prédire l'avenir d'une unité en cours dans un processus donné, en particulier les processus métier. La prédiction de cet avenir est généralement divisée en deux catégories : la prédiction d'événements et la prédiction d'issues spécifiques. D'une part, la prédiction d'issues vise à prédire si une unité va exhiber une certaine caractéristique à un moment donné de son parcours. Ces caractéristiques peuvent être retard / non retard, guérie / non guérie, acceptée / rejetée et ainsi de suite, les caractéristiques n'étant d'ailleurs pas nécessairement binaires. La prédiction d'événements, quant à elle, vise à générer le prochain événement, ou la prochaine série d'événements, qui doit encore se produire dans le parcours d'une unité dans un processus donné.

Cette thèse se concentre sur la prédiction d'événements dans les chapitres 4 et 5, et surtout la prédiction de la séquence restante d'événements à l'aide de l'apprentissage profond. La prédiction d'un événement, dans le contexte du *PBPM*, peut avoir plusieurs significations. En effet, un événement est caractérisé par l'unité pour laquelle il se produit, ainsi que par l'activité par laquelle l'unité va passer et la date et l'heure de début de cette activité, avec éventuellement d'autres variables descriptives.

Cette thèse a pour finalité applicative la mise en place d'un environnement prédictif simple d'utilisation et efficace sur plusieurs dimensions : temps total d'entraînement, convergence de l'apprentissage, efficacité prédictive, flexibilité à la donnée, simulation d'unités (chapitre 6) et contrôle de la dérive de la donnée (chapitre 7).

FIGURE 1.1 – Carte d'un processus dans Livejourney™

FIGURE 1.2 – Analytique des parcours dans Livejourney™

FIGURE 1.3 – Module de détection et alerte des anomalies dans Livejourney™

1.3 Contributions

Plusieurs approches d'apprentissage profond ont été développées dans une optique prédictive dans l'état de l'art, comme développé en chapitre 3. Cependant, la donnée elle-même ne semble pas y avoir été exploitée dans son entièreté : certaines variables, implicites, mais entièrement contenues dans la donnée, contiennent une quantité non négligeable d'information qui ne semble pas être captée par les modèles actuels. D'ailleurs, la plupart de ces modèles ne tiennent aucunement compte des variables descriptives potentiellement présentes dans les journaux d'événements, et n'incluent donc pas non-plus de *feature engineering* particulier.

Sachant ce contexte, ainsi que les objectifs professionnels relatifs à Livejourney™, cette thèse adresse quatre sujets principaux, chacun ayant donné lieu à des innovations technologiques, propriétaires de Livejourney™, et académiques :

1. la création de variables dites de « *peuplement* » [VBM22] ayant pour but le simple comptage du nombre d'unités présentes dans les différentes activités d'un processus à chaque temps enregistré. Ces variables, séries temporelles à valeurs discrètes, sont calculables pour tout journal d'événements et contiennent une quantité d'information permettant l'ensemble des méthodes développées par la suite dans cette thèse, même en l'absence de variables descriptives, en plus d'augmenter notablement la qualité des prédictions de parcours d'unités.

2. L'amélioration du modèle prédictif proposé dans [TR20] par le changement de la fonction de coût d'apprentissage, et celui de son architecture afin de bénéficier de l'utilisation des *peuplements* et autres covariables possibles [VBM22].

3. La création d'une méthode permettant de simuler un parcours « type » en fonction de conditions de départ. La création du *peuplement* permet de fournir à notre modèle prédictif des conditions de départ concernant la façon dont un processus est peuplé d'unités. Le modèle est alors capable de donner un parcours entier, de la première activité à la dernière, en fonction de ces *peuplements*. Cette méthode permet par ailleurs l'inclusion des autres variables descriptives disponibles le cas échéant.

4. Le modèle choisi et développé étant particulièrement difficile à expliquer à cause de son architecture antagoniste, la simulation donne lieu à une forme d'analyse de sensibilité. En effet, grâce au *peuplement*, il est possible de générer des plans d'expériences factoriels, permettant d'explorer les combinaisons possibles de *peuplements* maximaux et minimaux, ou d'échantillonner directement les *peuplements*, afin d'explorer quelles combinaisons donnent quelles prédictions.

5. Enfin, il est d'intérêt de pouvoir définir quand la donnée semble dériver. Or les données de processus peuvent dériver d'une myriade de façons plus ou moins complexes. De plus, la dérive de la donnée peut être capturée efficacement par le modèle prédictif, ne nécessitant pas nécessairement de ré-entraînement. Ainsi, deux méthodes distinctes ont été créées afin de définir et détecter de façon formelle la dérive de données de processus à l'aide de cartes de contrôle, avec et sans modèle prédictif [VBM23].

Cette thèse est scindée en 5 chapitres supplémentaires. Tout d'abord, des préliminaires posent les bases théoriques et appliquées, permettant de formaliser le contexte du doctorat et d'expliquer les facettes de la donnée de processus, en plus de présenter les données utilisées pour illustrer les recherches effectuées. Ensuite, un chapitre dédié au peuplement de processus décrit l'intérêt de sa création, avant de passer à sa définition formelle suivie de son exploration dans les différents journaux d'événements utilisés. Le chapitre suivant se focalise sur la modélisation prédictive : le modèle choisi et théorisé, la transformation des données, l'utilisation des peuplements (et potentielles covariables), ainsi que le coût d'entraînement. Les performances de ce modèle sont comparées à d'autres modèles proposant les meilleures performances prédictives pour des séquences d'événements.
La simulation et l'analyse de sensibilité sont concaténées en un seul chapitre, l'une emmenant l'autre naturellement et les résultats de la simulation étant d'autant mieux justifiés par l'analyse de sensibilité.
Enfin, le dernier chapitre décrit la méthodologie développée pour la détection et la quantification de la dérive des données *via* des cartes de contrôle.

1.4 Communications et *proceedings*

Des communications en conférences internationales ont été effectuées et ont parfois donné lieu à des publications dans des chapitres des ouvrages issus de ces conférences, ou dans leurs *proceedings*. Ces communications et publications sont, dans l'ordre chronologique :

Communications orales en personne en conférences internationales

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Introduction to Process Mining for the Improvement of Patient's Journeys*. Dans Smart Healthcare International Conference 2021, Troyes (France).

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Use of Process Crowding in Conditional WGAN for Remaining Process Events Prediction*. Dans Symposium on Data Science and Statistics 2022, Pittsburgh (États-Unis).

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Adding Covariables and Learning Rules to GAN for Process Units' Suffix Predictions*. Dans European Network for Business and Industrial Statistics 2023, Trondheim (Norvège).

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Feature Engineering Approach for Learning and Predicting Process Units with Two Timestamps*. Dans Joint Statistical Meetings 2022, Washington D.C. (États-Unis).

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Reinforcement learning for next best action recommendation in process data*. Dans International Conference on Computational Statistics 2022, Bologne (Italie).

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Predictive Process Model Explainability : The Case of Crowding and Design of Experiments*. Dans Eut+ Workshop on Statistical Data Science 2023, Darmstadt (Allemagne).

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Concept Drift in Process Data : A Control Chart Approach and Methodology*. Dans : Joint Statistical Meetings 2023, Toronto (Canada).

***Proceedings* dans des conférences internationales**

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Introduction to Process Mining for the Improvement of Patient's Journeys*. Dans les chapitres de Smart Healthcare International Conference 2021.

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Use of Process Crowding in Conditional WGAN for Remaining Process Events Prediction*. Dans Joint statistical Meetings 2022 Proceedings.

— Yoann Valero, Frédéric Bertrand, Myriam Maumy. *Concept Drift in Process Data : A Control Chart Approach and Methodology*. Dans Joint Statistical Meetings 2023 Proceedings (à paraître).

Une communication orale en conférence internationale sur le sujet de la *success story* industrielle et commerciale de Livejourney™ vis-à-vis de son utilisation de l'intelligence artificielle a été effectuée à la conférence internationale ECMI 2023 à Wrocław, en Pologne, en juin 2023.

1.5 Logiciels et librairies utilisés

Dans cette thèse, un ensemble de logiciels, langages de programmation et librairies ont été utilisés. Les images ont été générées sur R [R C21] à l'aide de la librairie `ggplot2` [Wic16]. Les palettes de couleurs pour les graphiques utilisant des classes sont issues de la librairie `RColorBrewer` [Neu22], et les palettes illustrant des valeurs continues sont issues de la librairie `viridis` [Gar+21]. Le code pour la modélisation prédictive est réalisé sur Python 3.9, grâce à la librairie `PyTorch`[Pas+19b].

Chapitre 2

Contextes formel et pratique

2.1 Définitions

Comme illustré en chapitre 1, les données les plus couramment enregistrées et utilisées pour modéliser des processus se présentent sous la forme de journaux d'événements. Nous définissons ci-après quelques concepts qui sont utilisés tout au long de cette thèse, principalement tels que définis dans [BSD21], [DAB19], [ERF17] et [TR20].

Les unités qui circulent dans un processus reçoivent un identifiant unique et peuvent passer par un ensemble fini d'activités possibles. Ainsi, soit $\mathcal{C}$ l'ensemble des identifiants, $\mathcal{A}$ l'ensemble de toutes les activités possibles et $\mathcal{T}$ l'ensemble des horodatages, qui dans notre cas est équivalent à $\mathbb{R}_+$.

Définition 1 (**Événement**). *[Tax+17; DAB19] Un* événement $e_i \in \mathcal{E} = \mathcal{C} \times \mathcal{A} \times \mathcal{T} \neq \varnothing$ *est un n-uplet e_i comprenant au moins 3 composants : $e_i = (c_i, a_i, t_i)$, où $c_i \in \mathcal{C}$, $a_i \in \mathcal{A}$, et $t_i \in \mathcal{T}$ est son horodatage marquant la date et l'heure de début de a_i. Un événement peut contenir d'autres attributs, mais ceux-ci ne sont pas pris en compte dans ce manuscrit.*

Nous pouvons ensuite définir les fonctions $\pi_{\mathcal{C}}(e_i) = c_i$, $\pi_{\mathcal{A}}(e_i) = a_i$ et $\pi_{\mathcal{T}}(e_i) = t_i$ pour extraire l'identifiant, l'activité et l'horodatage d'un événement donné [Tax+17] [ERF17] [TR20] [DAB19]. Ces fonctions sont simplement des projections canoniques.

Définition 2 (**Instance de processus**). *[Tax+17] [DAB19] Une* instance de processus*, également appelée* parcours*, est une séquence finie non vide d'événements notée $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$, $e_i \in \mathcal{E}$ et $n \in \mathbb{N}^\star$, de sorte que $\pi_{\mathcal{T}}(e_i) \leqslant \pi_{\mathcal{T}}(e_{i+1})$ et $\pi_{\mathcal{C}}(e_i) = \pi_{\mathcal{C}}(e_{i+1})$, $\forall i \in [\![1; n-1]\!]$ si $n \geq 2$. En d'autres termes, une instance de processus est la séquence de tous les événements qui partagent le même identifiant, ordonnés chronologiquement. Pour un parcours σ donné, il est conventionnel de noter $|\sigma| = n$ pour dénoter la longueur de ce parcours en termes de nombre d'événements [PS20] [DAB19].*

Définition 3 (**Journal d'événements**). *[DAB19] Un* journal d'événements *L, également appelé* log*, est un ensemble d'instances de processus, tel que chaque événement qu'il contient est unique. Formellement, un journal d'événements est donc une collection d'instances de processus $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$. Afin d'uniformiser les notations, sa taille en termes d'instances de processus est notée $|L|_\sigma$ et sa taille en termes d'événements est notée $|L|_e$, avec $|L|_\sigma \leqslant |L|_e$.*

TABLEAU 2.1 – Trois instances de processus du journal d'événements *Traffic fines*

Identifiant	**Activité**	**Date**
A1	Create Fine	2006-07-24
A1	Send Fine	2006-12-05
A100	Create Fine	2006-08-02
A100	Send Fine	2006-12-12
A100	Insert Fine Notification	2007-01-15
A100	Add penalty	2007-03-16
A100	Send for Credit Collection	2009-03-30
A10000	Create Fine	2007-03-09
A10000	Send Fine	2007-07-17
A10000	Insert Fine Notification	2007-08-02
A10000	Add penalty	2007-10-01
A10000	Payment	2008-09-09

Le tableau 2.1 présente les trois premières instances de processus du journal d'événements *Traffic fines* de la librairie R[R C21] `eventdataR`[Jan22b], qui fait partie du *framework* `bupaR` [Jan22a]. Ces trois premières instances de processus constituent un journal d'événements de longueurs $|L|_\sigma = 3$ et $|L|_e = 12$, chaque ligne constituant un événement.

Définition 4 (**k-préfixe / suffixe**). *[TR20] Soit un parcours $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$. Un* k-préfixe *$\sigma_{\leqslant k}$ est une sous-séquence non vide de σ telle que $\sigma_{\leqslant k} = \langle e_1, e_2, \ldots, e_k \rangle$, $k < n$. Le suffixe de $\sigma_{\leqslant k}$ est donc $\sigma_{>k} = \langle e_{k+1}, e_{k+2}, \ldots, e_n \rangle$.*

Nous avons maintenant une définition de différents niveaux d'analyse dans un journal d'événements : de l'événement à l'instance de processus jusqu'au journal d'événements complet. Il y a cependant d'autres niveaux d'analyse possibles, qui prennent en particulier leur sens dans un contexte appliqué. Il convient donc d'introduire les cinq jeux de données qui seront utilisés tout long de cette thèse, ainsi que des ajouts en termes de terminologie pratique.

2.2 Données réelles

2.2.1 Jeux de données

Plusieurs jeux de données réelles sont utilisés dans cette thèse. Il semble commode de les exposer dans les préliminaires dans la mesure où ils permettent du même coup d'introduire quelques notions utilisées purement dans le domaine applicatif, qui sont néanmoins importantes tout le long de cette thèse.

Ainsi, cette sous-section est destinée à un simple descriptif contextuel des jeux de données. Est ensuite expliquée une certaine terminologie pratique illustrée par des exemples issus de cette donnée réelle, avant de terminer par un descriptif des jeux de données à partir des notions illustrées ici, et d'autres statistiques positionnelles.

Le choix des cinq jeux de données utilisés lors de cette thèse s'appuie sur les données utilisées dans les articles [Tax+17] et [TR20] :

— **Helpdesk** [1] : journal d'événements enregistrant les événements liés à un processus de gestion des tickets dans un service d'assistance d'une entreprise italienne, de janvier 2 010 à novembre 2 012.

— **Traffic fines** [2] : journal d'événements réels d'un système d'information gérant les amendes routières. Les horodatages vont de juin 2 006 à mars 2 012.

— **BPI2012(W) et BPI2012** [3] : journal d'événements d'un processus de demande de prêt dans un institut financier néerlandais, contenant trois sous-processus. L'un d'entre eux, noté *W*, était utilisé par [Tax+17] et [TR20] et a donc été testé à part. Il couvre les dates d'octobre 2 011 à mars 2 012.

— **BPI2017** [4] : journal d'événements contenant des données relatives à un processus de demande de prêt dans le même institut financier néerlandais que dans BPI2012, mais établi entre janvier 2 016 et février 2 017.

2.2.2 Terminologie pratique

Traces, échelles

Au-delà des définitions de la section 2.1, la notion de *trace* est également importante dans l'étude des journaux d'événements. Également appelée *instance distincte de processus* (de l'anglais *distinct process instance*[De +13]), il s'agit de la séquence d'activités représentant tous les parcours possédant exactement cette même séquence d'activités, sans tenir

1. `https://data.4tu.nl/repository/collection:eventlogsreal`
2. `https://data.4tu.nl/articles/dataset/Road_Traffic_Fine_Management_Process/12683249`
3. `https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204`
4. `https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884`

compte des durées. Ainsi, un journal d'événements peut comporter $n \in \mathbb{N}^\star$ parcours et une unique trace, pour peu que tous les parcours aient la même séquence d'activités. Notons $|L|_T$ le nombre de traces présentes dans un journal d'événements L donné.
Le rapport du nombre de traces sur le nombre de parcours $\frac{|L|_T}{|L|_\sigma}$ est un indicateur simple et parlant de la complexité d'un processus, ou du moins de la rigidité de son exécution. En effet, un tel ratio avoisinant 0 indique que tous les parcours suivent globalement le même chemin, la variation résidant principalement dans les durées. Au contraire, un tel ratio de 1 indique que chaque parcours est unique sur au moins une de ses étapes.

Ainsi, un journal d'événements peut être étudié selon plusieurs niveaux de granularité : on parle d'« échelles ». On peut par exemple effectuer des études à l'échelle des parcours, des traces, ou du journal d'événements entier.

Intérêt

Prenons le journal d'événements *Traffic fines* pour illustrer la nature de ces différentes échelles en y étudiant la fréquence des activités.

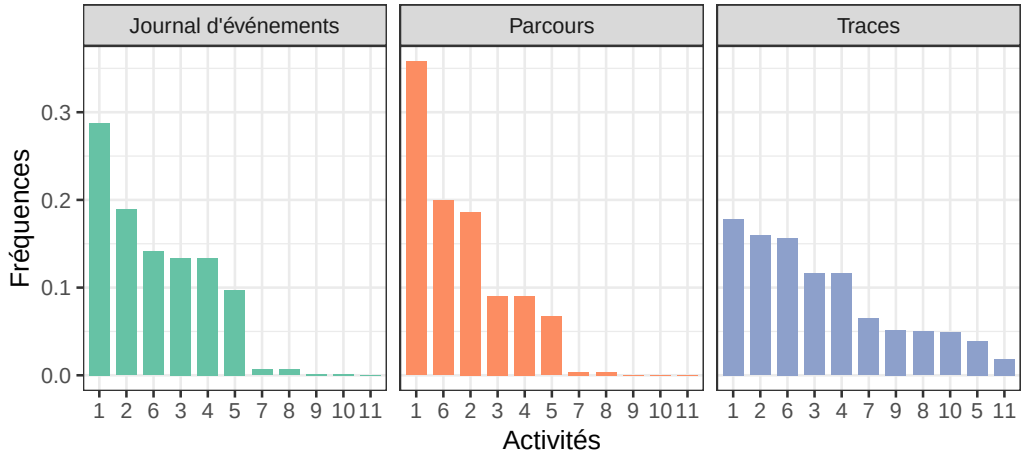

FIGURE 2.1 – Fréquences des activités à l'échelle du journal d'événements, des parcours et des traces dans *Traffic fines*, classées dans l'ordre décroissant

La figure 2.1 montre les fréquences des activités calculées aux différentes échelles mentionnées ci-avant, classées dans l'ordre décroissant. Le graphique de gauche montre leurs fréquences dans le journal d'événements. Nous y observons l'activité 1 qui y est représentée en plus grand nombre, suivie de l'activité 2, puis de la 6, et ainsi de suite.
Dans le graphique du milieu, à l'échelle des parcours, la fréquence d'une activité est calculée au sein de son parcours, puis on calcule la fréquence moyenne de cette activité dans tous les parcours. Nous observons, assez naturellement, un ordre identique à celui trouvé à l'échelle du journal d'événements, bien que les fréquences aient changé puisqu'il s'agit

cette fois d'une moyenne sur les parcours.
Enfin, un calcul analogue à celui effectué à l'échelle des parcours a été effectué à l'échelle des traces. Les cinq activités les plus fréquentes restent inchangées, mais les six activités moins fréquentes ne possèdent plus le même ordre. Cela indique en réalité une sur-représentation de certaines traces dans les parcours, montrant que la fréquence des activités n'est pas la même si l'on s'intéresse à leur apparition « brute » dans le journal d'événements ou à leur apparition dans la typologie des parcours.

Certains journaux d'événements, tels que *BPI2017*, ont un déséquilibre conséquent. À tel point que l'activité la plus fréquente change selon que l'on étudie l'échelle des parcours ou l'échelle des traces, comme visible en figure 2.2. Nous y remarquons en effet sur le graphique du milieu, à l'échelle des parcours, que les trois activités les plus fréquentes sont, dans l'ordre décroissant, la 10, la 15, puis la 4. Tandis qu'à l'échelle des traces, ce même podium appartient respectivement aux activités 15, 10 et 13. L'activité 4 n'y est que cinquième plus fréquente.

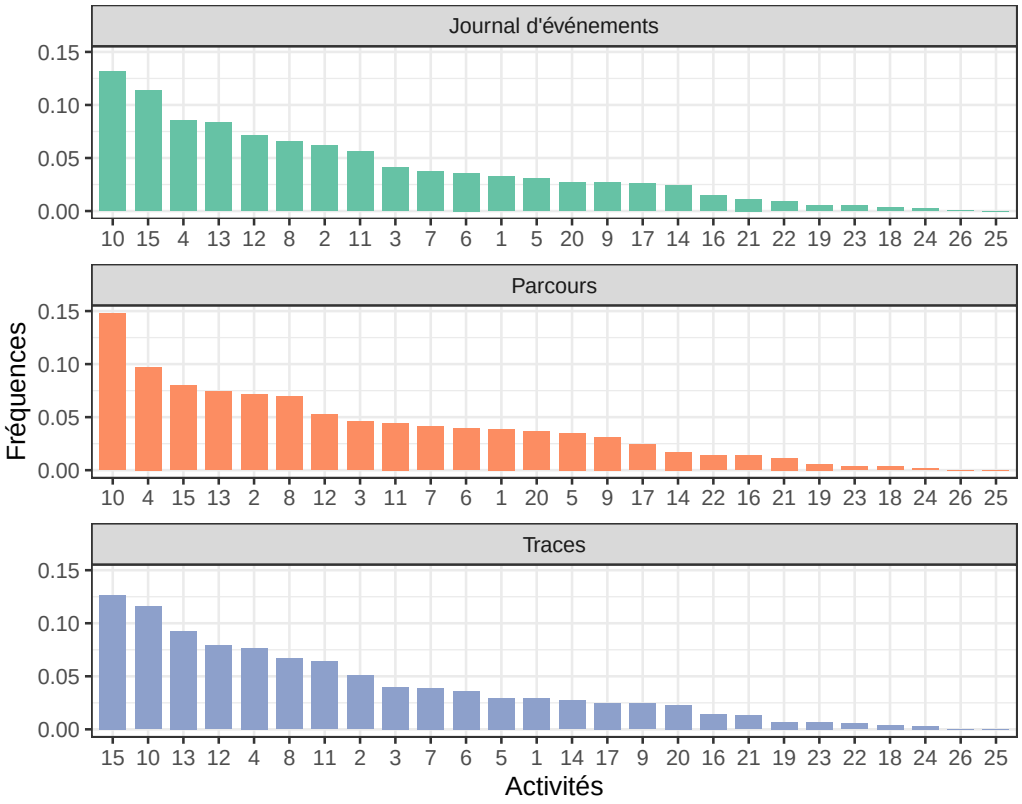

FIGURE 2.2 – Fréquences des activités à l'échelle du journal d'événements, des parcours et des traces dans *BPI2017*

Ainsi, l'idée de modéliser ces parcours pose également la question de leurs conditions d'apparition : en effet, modéliser un parcours individuel est une chose, mais modéliser

un ensemble de parcours individuels qui, ensemble, doivent permettre de retrouver les différences entre échelles, en est une autre.
Cette problématique de différences entre échelles, symptomatique de traces représentées de façon déséquilibrée, a motivé certaines décisions prises au cours de cette thèse qui y seront expliquées à l'endroit adéquat.

Par ailleurs, le déséquilibre des traces dans les journaux d'événements utilisés dans cette thèse sera exploré dans ce chapitre après quelques statistiques positionnelles descriptives de ces derniers.

2.2.3 Description des cinq jeux de données

Nous possédons à présent des définitions de base et un formalisme, ainsi qu'une idée approfondie des différences d'échelles dans les journaux d'événements. Cette section décrit de façon plus exhaustive les journaux d'événements utilisés, par des statistiques positionnelles visibles dans le tableau 2.2. Soit $\mathcal{A}$ l'ensemble des activités d'un journal d'événements L. Soit $\mathcal{C}ard(\mathcal{A}) = |\mathcal{A}|$. Un parcours dans L est noté σ et la longueur de σ est $|\sigma|$. Les jeux de données réels utilisés peuvent alors être décrits d'après les statistiques du tableau 2.2.

Tableau 2.2 – Statistiques descriptives positionnelles des journaux d'événements utilisés

Journaux d'événements	**Événements**	**Parcours (σ)**	$\lvert\mathcal{A}\rvert$	$\lvert\sigma\rvert$ **Min / Moyenne / Max**	**Temps total (jours) Moyen / Max**
Helpdesk	13 710	3 804	9	1 / 3,60 / 14	13,75 / 59,85
Traffic fines	34 724	10 000	11	2 / 3,47 / 9	296,13 / 1 956
BPI2012 (W)	72 413	9 657	6	1 / 7,49 / 74	13,13 / 91,08
BPI2012	262 200	13 087	23	3 / 20,03 / 175	22,17 / 91,45
BPI2017	1 048 575	27 499	26	10 / 38,13 / 180	22,05 / 169,11

Au-delà de la multiplicité des contextes entourant les journaux d'événements utilisés, Nous voyons également une diversité notable en termes de complexités : *Helpdesk* contient le plus petit nombre d'événements et de parcours, avec des parcours comprenant en moyenne 3,60 événements pour un minimum de 1 et un maximum de 14. Les durées totales de parcours gravitent autour de 13,75 jours pour un maximum de 59,85 jours.
Traffic fines, bien que paraissant analogue en termes de nombre d'activités, et longueur des parcours, possède en réalité une complexité cachée concernant les durées : 296,13 jours en moyenne, pour un maximum de plus de 5 ans. De plus, il s'agit d'un journal d'événements regroupant des amendes routières, dont le paiement (ou non paiement) est assez largement décidé par les personnes verbalisées et /ou les instances juridiques correspondantes.
BPI2012 (W), partie de *BPI2012*, ne contient que 6 activités mais comptabilise 9 657 parcours de longueur moyenne de 7,49 événements, pour un maximum de 74 événements.

La complexité de ce journal d'événements va donc résider dans les répétitions d'activités ou sous-séquences d'activités. Cette complexité est décuplée dans *BPI2012*, qui montre les mêmes tendances avec un total de 23 activités et 13 087 parcours.
Enfin, *BPI2017* est de loin le plus massif avec plus d'un million d'événements et 27 499 parcours, un total de 26 activités possibles et un nombre moyen de 38,13 événements par parcours. Les durées varient également de façon notable, la moyenne des durées totales étant à 22,05 jours et le maximum à 169,11 jours.

Attardons-nous sur les répétitions dans les journaux d'événements. Puisqu'il existe plusieurs sortes de répétitions, nous avons décidé d'utiliser la notion de répétition maximale telle qu'exposée dans [BDH13] : dans une séquence w de symboles, une répétition maximale est une sous-séquence qui apparaît plus d'une fois dans w, telle que chacune de ses extensions, à gauche et à droite, apparaît moins de fois. Les séquences observées sont les séquences d'activités dans les parcours : pour un parcours $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$, sa séquence d'activités est la séquence $\langle \pi_{\mathcal{A}}(e_1), \pi_{\mathcal{A}}(e_2), \ldots, \pi_{\mathcal{A}}(e_n) \rangle$, dont nous pouvons étudier les répétitions maximales.

Tableau 2.3 – Nombre de répétitions maximales distinctes dans les journaux d'événements utilisés

Journal d'événements	**Répétitions maximales**
Helpdesk	33
Traffic fines	2
BPI2012 (W)	295
BPI2012	639
BPI2017	1 470

Nous voyons dans le tableau 2.3 que *BPI2012 (W)* possède 295 répétitions uniques, trouvées de multiples fois à travers ses différentes traces, expliquant immédiatement la grande quantité de parcours par rapport au très petit nombre d'activités. Le plus grand nombre de répétitions uniques appartient à *BPI2017*, avec 1470 répétitions maximales différentes.

Nous disposons donc de journaux d'événements présentant un éventail large de complexités et caractéristiques notables en termes d'activités, durées, longueur des parcours, répétitions, et nature.

Enfin, nous pouvons regarder le nombre de traces par rapport au nombre de parcours, qui donne immédiatement une idée de la difficulté de la tâche de modélisation des parcours.

TABLEAU 2.4 – Quantité de parcours comparée à la quantité de traces dans les journaux d'événements utilisés

Échelle	***Helpdesk***	***Traffic fines***	***BPI2012 (W)***	***BPI2012***	***BPI2017***
Parcours	3 804	10 000	9 657	13 087	27 499
Traces	154	44	2 263	4 366	14 226

Dans le tableau 2.4, nous pouvons voir que *Helpdesk* et *Traffic fines* ont une quantité de parcours considérablement plus grande que leur quantité de traces, alors qu'un journal d'événements comme *BPI2017* en a seulement deux fois plus.

La modélisation doit donc être capable de tenir compte de ces différentes formes de complexité. Pour finir l'exploration de la donnée réelle, regardons la problématique du déséquilibre des effectifs de façon plus poussée.

2.2.4 Effectifs déséquilibrés

Nous avons vu dans la section 2.2.2 que les fréquences d'activités observées changent en fonction de l'échelle choisie dans un journal d'événements, en particulier à cause d'un déséquilibre des effectifs : certaines traces sont représentées de façon disproportionnée. Il est intéressant de voir dans quelle mesure ce phénomène se produit. Pour cela, les traces ont été récupérées et comptées, puis leur somme cumulée a été calculée. Ces sommes cumulées, analogues à des fonctions de répartition empiriques, sont exposées en figure 2.3.

Dans la figure 2.3, pour *Helpdesk* et *Traffinc fines*, une dizaine de traces représentent la vaste majorité des parcours observés. Les autres jeux de données atteignent plutôt des dizaines voire une centaine de traces apparaissant plus de deux fois. La difficulté pour *BPI2012 (W)*, *BPI2012* et *BPI2017* réside donc dans la proportion de traces n'apparaissant qu'une fois, c'est-à-dire des parcours totalement uniques sur la période observée.
Ces parcours peuvent poser problème dans la mesure où les conditions de leur émergence ne sont visibles qu'une seule fois, il n'est donc pas trivial de les modéliser sans passer par un sur-apprentissage immédiat, ou sans les ignorer complètement.

TABLEAU 2.5 – Fréquence de traces n'apparaissant qu'une fois dans les journaux d'événements utilisés

Journal d'événements	**Parcours totalement uniques**
Helpdesk	56,49%
Traffic fines	31,82%
BPI2012 (W)	75,56%
BPI2012	85,98%
BPI2017	87,48%

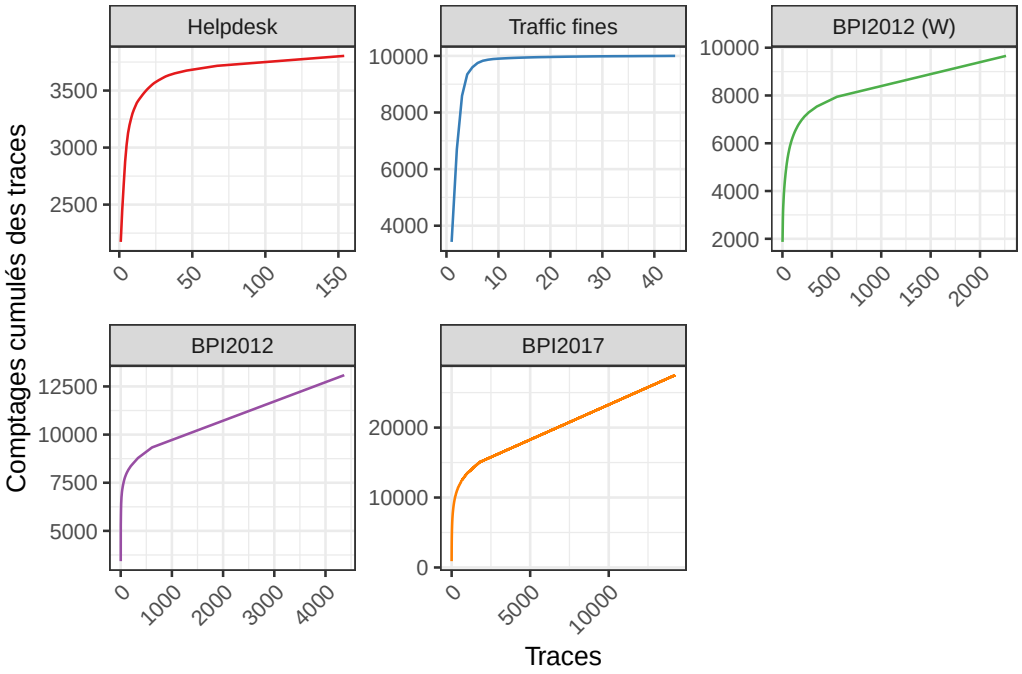

FIGURE 2.3 – Comptages cumulés des traces dans *Helpdesk*, *Traffic fines*, *BPI2012 (W)*, *BPI2012* et *BPI2017*

Le tableau 2.5 montre que pour ces trois journaux d'événements, la vaste majorité des traces observées sont des occurrences uniques : 3 traces sur 4 le sont dans *BPI2012 (W)*, 6 traces sur 7 dans *BPI2012* et 7 traces sur 8 dans *BPI2017*.
Cela dit, les traces plus fréquentes, bien que très minoritaires, constituent systématiquement au moins la moitié des parcours observés, dans des conditions souvent variables.

2.2.5 Visualisation des processus

Nous pouvons également étudier une représentation graphique des processus utilisés dans cette thèse, en particulier grâce aux cartes de processus générées grâce à la librairie R `processmapR` [Jan22c]. Afin de pouvoir générer de telles cartes, les activités normalement gardées sous forme de nombres ont été converties en lettres majuscules (l'activité 1 devient A, 2 devient B, *et caetera*). Cet encodage est commode dans la mesure où aucun journal d'événements ne contient plus de 26 activités différentes.

Nous observons sur la figure 2.4 la carte du processus *Helpdesk*. Nous y voyons, comme dans les autres figures contenant une carte de processus, un début et une fin factices des parcours « *Start* » et « *End* » qui ne sont pas dans la donnée. Le but de ce rajout est de posséder un unique début et une unique fin pour la carte, malgré la possibilité de plusieurs activités réelles de début et de fin.
Nous pouvons observer dans les cellules les activités, le nombre d'unités y étant passé, en

comptant les répétitions, des flèches indiquant les transitions entre activités ainsi que les unités ayant effectué ces transitions. Ainsi, en prenant l'exemple de l'activité A, 3 644 unités parmi les 3 804 présentes y ont commencé leur parcours, 105 y passent depuis l'activité D, et une y passe depuis B. De plus, 394 passages de A à elle-même ont été comptés, pour un total de 4 144 passages d'unités dans l'activité A. Nous y distinguons une myriade de trace possibles, la plus fréquente étant en gras avec la séquence d'activités $\langle A, B, C\rangle$, correspondant à la séquence $\langle 1, 2, 3\rangle$ dans le journal d'événements d'origine. La trace $\langle 1, 2, 3\rangle$ est donc la trace majoritaire dans *Helpdesk*.

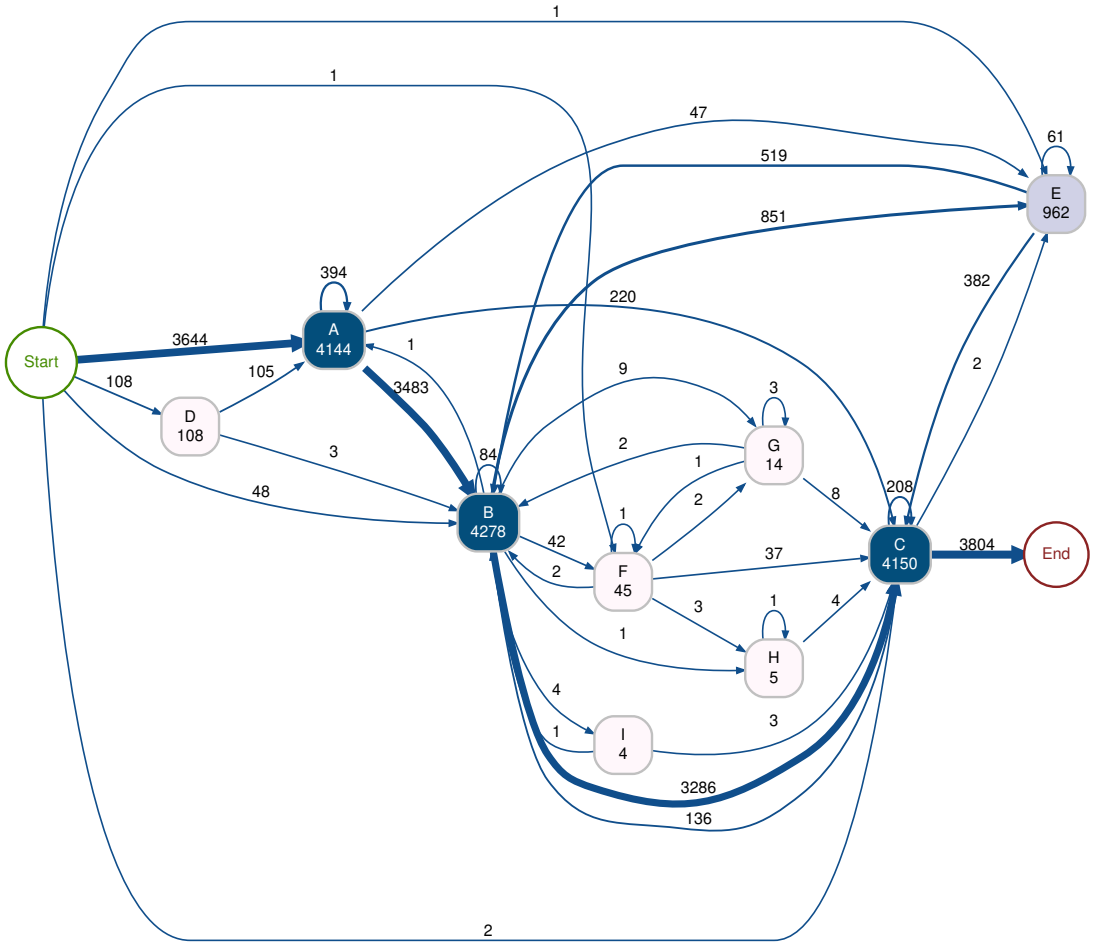

FIGURE 2.4 – Carte du processus *Helpdesk*

Sur la figure 2.5 qui représente la carte du processus *Traffic fines*, la trace majoritaire est plus difficile à distinguer. Nous distinguons en effet une allant de A vers B avec un comptage de 6 557, puis une transition de B à C avec un comptage de 4 633, mais la transition de C à D tombe à 4 417, puis 3 288 pour D vers E. Or la trace $\langle A, F\rangle$ (menant ensuite vers *End*) donne une majorité de comptages avec un total de 3 443. Ainsi, la trace $\langle A, F\rangle$, correspondant à la trace $\langle 1, 6\rangle$ dans le journal d'événements d'origine, est majoritaire, suivie par $\langle A, B, C, D, E\rangle$, c'est à dire $\langle 1, 2, 3, 4, 5\rangle$.

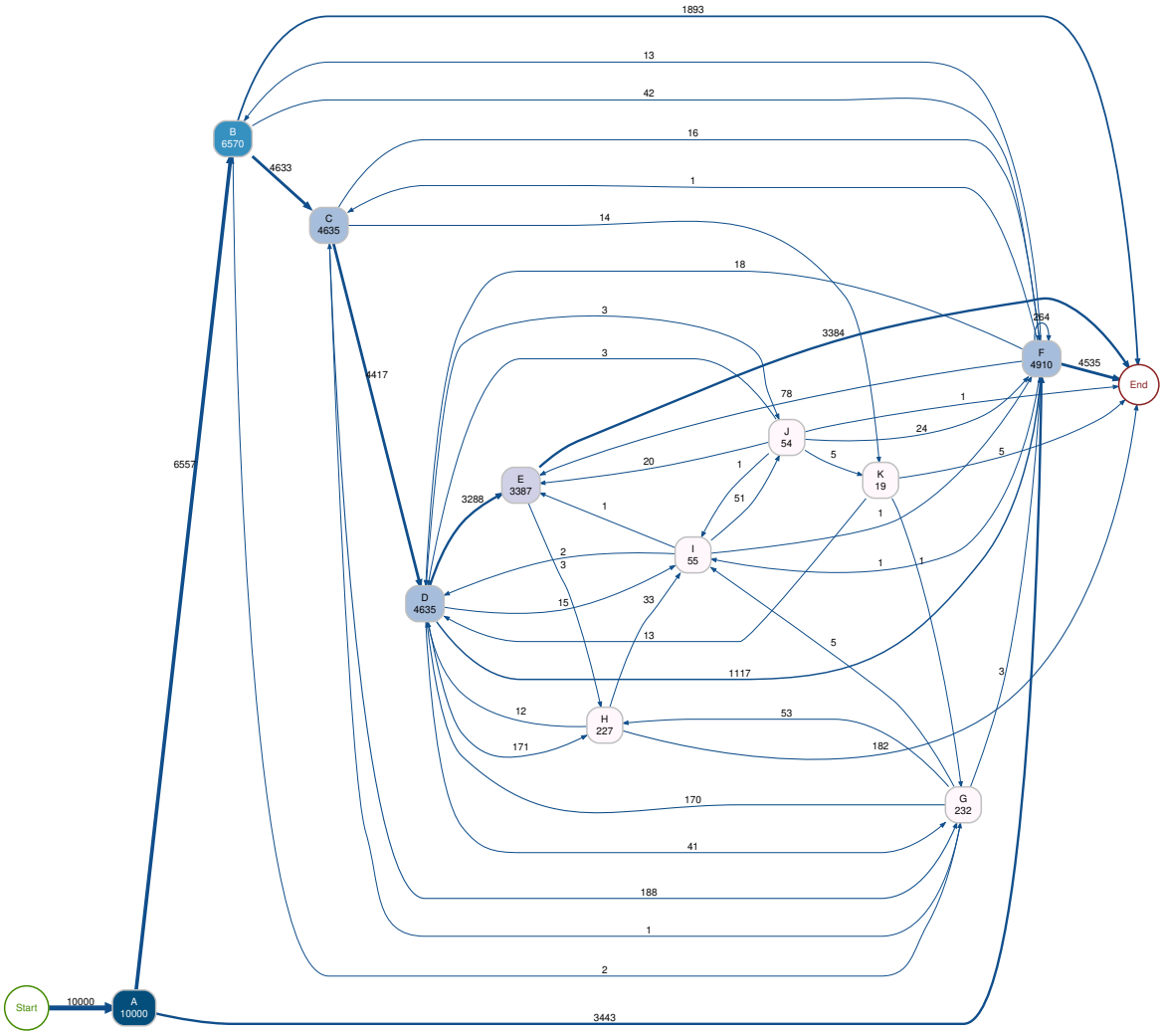

FIGURE 2.5 – Carte du processus *Traffic fines*

En passant à la figure 2.6, qui présente la carte de *BPI2012 (W)*, le processus semble de prime abord moins complexe. Cependant, le nombre de traces 15 fois supérieur à *Helpdesk* et 51 fois supérieur à *Traffic fines* indique le contraire. Nous observons justement, par le tableau 2.3, que les transitions les plus proéminentes correspondent aux boucles des activités sur elles-mêmes : nous comptons par exemple 17 960 répétitions sur B, 16 594 répétitions sur A et 9 219 répétitions sur E, ce qui indique que bon nombre de traces possèdent plusieurs de ces répétitions. Il est ainsi ardu de déterminer la trace majoritaire à partir de cette carte. En effet, nous ne pouvons déterminer le début et la fin de traces les plus fréquents : l'activité A pour le début et C pour la fin, avec des comptages de 4 852 et 2 751 respectivement. La trace majoritaire se trouve être D, donc 4 dans le journal d'événements d'origine.

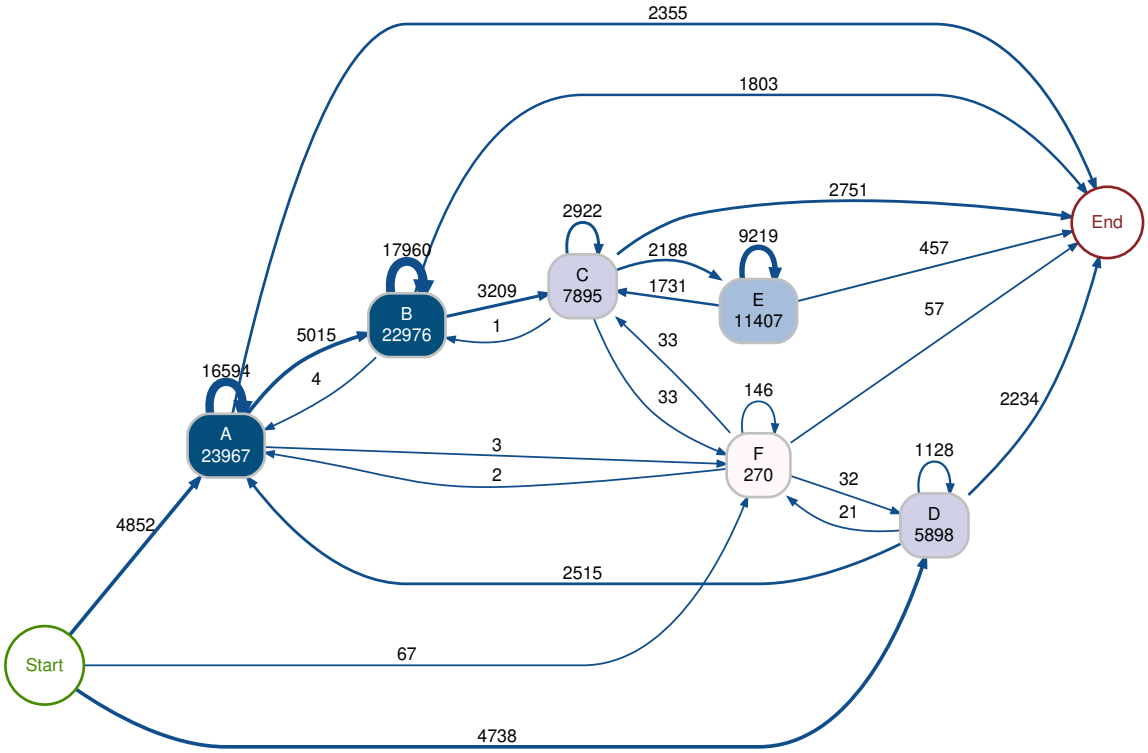

FIGURE 2.6 – Carte du processus *BPI2012 (W)*

Les derniers jeux de données, *BPI2012* et *BPI2017*, donnent des cartes de processus illisibles qui ont donc été omises. Afin de malgré tout permettre une certaine vision de ces processus, le tableau 2.6 contient le nombre de transitions entre activités dans chaque journal d'événements, ainsi que la trace majoritaire extraite par un code `R` comptant et classant les séquences d'activités distinctes selon leur fréquence. Il y a un saut considérable entre les trois journaux *Helpdesk*, *Traffic fines* et *BPI2012(W)* pour lesquels les cartes de processus sont montrées ici, et *BPI2012* et *BPI2017* : ces deux derniers contiennent respectivement 135 et 209 transitions entre activités observées, respectivement. Nous y voyons d'ailleurs que *BPI2017* possède une trace majoritaire particulièrement longue, ce qui correspond aux observations du tableau 2.2, dans lesquels nous voyons que la trace de taille minimale contient tout de même 10 événements.

TABLEAU 2.6 – Nombre de transitions entre activités observées dans les journaux d'événements utilisés et trace majoritaire

Log	**Transitions**	**Trace majoritaire**
Helpdesk	40	$\langle 1, 2, 3 \rangle$
Traffic fines	47	$\langle 1, 6 \rangle$
BPI2012 (W)	28	$\langle 4 \rangle$
BPI2012	135	$\langle 1, 2, 18 \rangle$
BPI2017	209	$\langle 1, 2, 2, 2, 2, 3, 4, 5, 6, 7, 8, 4, 4, 9, 10, 10, 10, 10, 22, 20, 20 \rangle$

Ce tour d'horizon pose les bases de ces recherches en termes de données, complexité et problématiques liées à cette donnée. Avant de passer aux recherches propres à cette thèse, effectuons une exploration de l'état de l'art afin de poser le paysage entourant ces recherches.

Chapitre 3

État de l'art

3.1 Approches classiques

Plusieurs outils libres sont à disposition sur des logiciels tels que R pour analyser les données présentées sous formes de séquences. On pense par exemple à la librairie `TraMineR`[Gab+11], qui propose un ensemble d'outils d'analyses graphiques des séquences.

Par exemple, la figure 3.1 montre une visualisation des parcours dans *Traffic fines*. Les activités sont représentées sur l'axe des ordonnées, et les étapes des parcours sont en abscisse. Les cases sont donc des points sur la grille $\{1, \ldots, |\mathcal{A}|\} \times \left\{1, \ldots, \max_{\sigma \in L}(|\sigma|)\right\}$. Un parcours commence dans une case d'abscisse 1, puis évolue jusqu'à atteindre son activité de fin après un nombre arbitraire d'étapes, jusqu'à un maximum observé de 9 dans le cas de *Traffic fines*. Chaque couleur représente un type différent de parcours (donc une trace), et la taille des carrés dans chaque case est proportionnelle à la fréquence des traces correspondantes dans le journal d'événements.

On peut également évoquer les méthodes impliquant la notion d'entropie, en particulier l'entropie transversale dans le cas de séquences. L'entropie transversale est simplement l'entropie de Shannon[Sha48], issue de la théorie de l'information, calculée sur les étapes communes des différents parcours. Soit une variable aléatoire X discrète, possédant $J \in \mathbb{N}^\star$ modalités, chacune ayant une probabilité p_j associée. L'entropie de Shannon de la variable aléatoire X est notée :

$$H_b(X) = -\sum_{j=1}^{J} p_j \log_b(p_j),$$

où b est la base du logarithme. Ainsi, l'entropie donne une mesure la quantité d'information fournie par la variable aléatoire X lorsqu'elle se réalise.
Prenons le cas où X possède $J \in \mathbb{N}^\star$ modalités, mais seul un $j \in \{1, \ldots, J\}$ est tel que $p_j = 1$. On a donc une seule modalité de probabilité 1, toutes les autres ont une probabilité

nulle. Dans ce cas, la réalisation de X est non informative, puisque son résultat est connu d'avance, nous n'avons pas besoin d'observer X pour déterminer la valeur que prendra sa réalisation.
En revanche, si X suit une loi uniforme discrète, on a $p_j = \frac{1}{J}\ \forall j \in \{1, \ldots, J\}$, et il est impossible de pencher pour la réalisation d'une modalité par rapport à n'importe quelle autre : ce n'est que lorsque X se réalise que l'on obtient l'information, l'entropie de X est maximale.
On imagine un cas intermédiaire où un j serait tel que p_j serait grand. Dans ce cas, on aurait un certain degré de certitude sur la valeur que prendra X lors de sa réalisation, son entropie n'est pas nulle mais pas maximale non-plus.

Il est ainsi commode, lors du calcul d'une entropie, de la diviser par sa valeur maximale théorique. Celle-ci est atteinte, comme rapidement évoqué précédemment, dans le cas où X suivrait une loi uniforme discrète. Dans ce cas, l'entropie maximale vaut $\log_b(J)$. On a alors une entropie normalisée :

$$-\frac{1}{\log_b(J)} \sum_{j=1}^{J} p_j \log_b(p_j)$$

Ainsi, plus l'entropie normalisée est proche de 1, plus la variable aléatoire observée s'approche d'une loi uniforme discrète. Plus l'entropie normalisée est proche de 0, plus la variable aléatoire observée est proche d'une variable aléatoire dégénérée, à valeur constante presque sûrement.

L'entropie transversale, calculable sur une séquence, correspond à l'entropie calculée sur une étape donnée de plusieurs séquences. Si l'on prend le cas de parcours dans un journal d'événements, l'entropie transversale de l'étape 1 consiste à récupérer toutes les activités de départ des parcours disponibles, ce qui correspond à des réalisations d'une variable aléatoire à valeurs dans $\mathcal{A}$, et à en calculer l'entropie. On répète l'opération pour l'étape suivante des parcours, et ainsi de suite, jusqu'à avoir calculé l'entropie de la dernière étape du ou des parcours les plus longs.

La figure 3.2 montre de telles entropies transversales normalisées calculées sur les parcours de *BPI2012*, en base e. Les rectangles colorés correspondent aux différentes activités, et la ligne bleu ciel correspond à l'entropie transversale : on y voit une entropie stabilisée à environ 0,7 sur une majorité des étapes, indiquant une multiplicité des activités possibles dans une majorité des parcours des unités observées.

Il est à noter que ces méthodes sont exploratoires, et n'ont pas vocation à constituer une méthode prédictive sur des séquences.

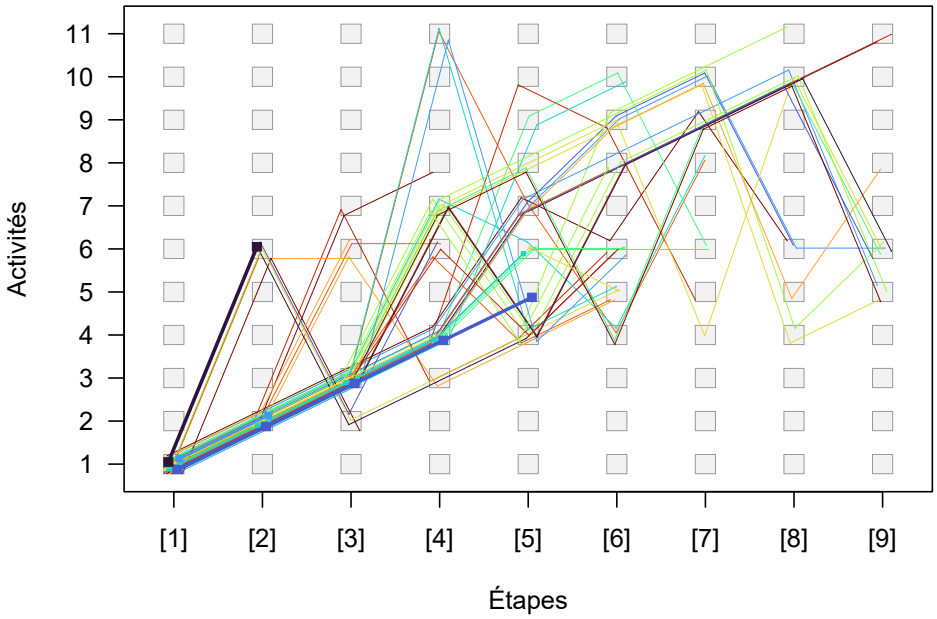

FIGURE 3.1 – Visualisation des parcours sous la forme étapes×activités dans *Traffic fines*

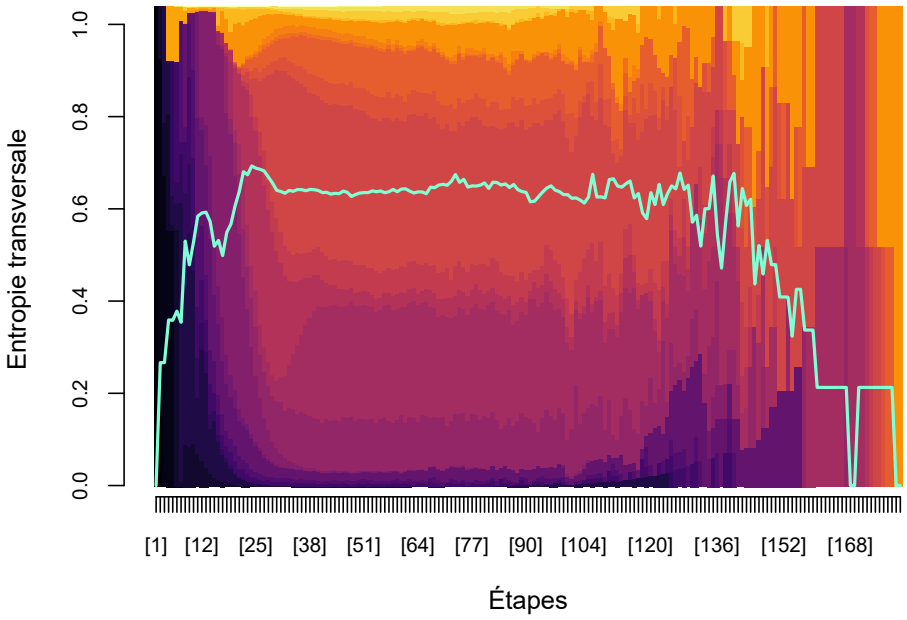

FIGURE 3.2 – Entropie transversale des différentes étapes des parcours observés dans *BPI2017*

3.2 Réseaux de Petri

3.2.1 Réseaux de Petri classiques

La fouille de processus trouve son origine principale dans un article fondateur de Will van der Aalst de 2001 [AWM04]. Celui-ci propose un algorithme capable de fouiller un journal d'événements et d'en extraire un modèle censé correspondre au processus ayant généré ce journal d'événements. Cet algorithme, appelé « *α-algorithm* ». Celui-ci vise en réalité à fouiller les flux présents dans la donnée afin d'obtenir un réseau de Petri [RE98], modèle mathématique servant à représenter des systèmes composés de variables discrètes. Un réseau de Petri peut être défini par un quadruplet $R = \langle P, T, Pre, Post \rangle$ (notation de [RE98]) composé de :

— P un ensemble fini de places. Celles-ci sont représentées par des cercles sur un réseau de Petri.

— T un ensemble fini de transitions, avec $P \cap T = \varnothing$. Celles-ci sont représentées par des rectangles sur un réseau de Petri.

— $Pre \in \mathbb{N}^m \times \mathbb{N}^n$ l'incidence avant.

— $Post \in \mathbb{N}^m \times \mathbb{N}^n$ l'incidence arrière.

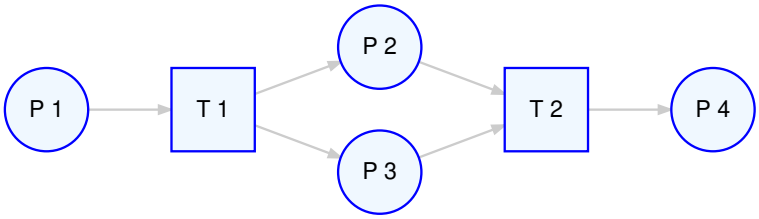

FIGURE 3.3 – Exemple de réseau de Petri

Un réseau de Petri est donc composé de places et de transitions connectées entre elles. La figure 3.3 montre un réseau de Petri constitué de quatre places et trois transitions : les places sont représentées par des cercles, les transitions par des carrés. Des jetons sont envoyés dans le réseau, et les places stockent ces jetons. Une transition consume un nombre fixe de jetons dans les places connectées à elle en amont, et crée des jetons dans les places connectées à elle en aval. Ceci correspond respectivement aux matrices Pre et $Post$: la première contient le nombre de jetons consumés par les transitions, l'autre les jetons créés par les transitions. Ainsi, une transition ne s'active que lorsque les places situées avant contiennent toutes assez de jetons, et lorsque les places situées après peuvent recevoir les jetons que la transition crée lors de son activation. Un 0 dans une de ces matrices indique l'inexistence d'un arc dans le sens correspondant. On peut également définir la matrice $C = Post - Pre$, nommée « matrice d'incidence ».

Un réseau de Petri est par ailleurs un graphe biparti orienté. Ainsi, un arc ne peut pas connecter deux places ni deux transitions. Si l'on appelle F l'ensemble des arcs du réseau, alors $F \subseteq (P \times T) \cup (T \times P)$. Il s'agit bien ici d'une inclusion non stricte puisque les arcs ne couvrent pas nécessairement toute la combinatoire des paires places-transitions et transitions-places.

La répartition des jetons dans les places d'un réseau de Petri est généralement définie par $M \in \mathbb{N}^P$, un vecteur contenant $Card(P)$ composantes. Chaque composante $M(p)$ représentant le nombre de jetons dans la place $p \in P$. Un réseau de Petri peut avoir un marquage initial, noté M_0.

Un réseau de Petri classique n'est pas déterministe par définition tant que le moment d'activation des transitions pouvant être activées n'as pas été défini. Ceci est d'autant plus vrai lorsque deux transitions peuvent être activées à la même étape, et qu'un ordre d'activation doit ainsi être choisi. Il est possible d'attribuer un ordre de priorité aux transitions [DA10].

La fouille de processus permet donc, à partir d'un journal d'événements, de créer un tel modèle exerçant un compromis entre les quatre notions suivantes :

— adéquation : le modèle tend à reproduire les comportements exhibés dans le journal d'événements.

— Généralisation : le modèle tend à permettre des comportements potentiellement observables dans le futur.

— Précision : le modèle ne tend à pas produire de comportements non observés.

— Simplicité : le principe de parcimonie, le modèle tend à être le moins complexe possible. Celle-ci peut par exemple être définie comme une mesure comprenant le nombre d'arcs, places et transitions [De +13].

De nombreux algorithmes ont pris le pas au α-*algorithm* de van der Aalst, tel *Flexible Heuristics Miner* [WR11], chacun proposant des méthodes permettant de minimiser le compromis entre adéquation, généralisation, précision et simplicité.

Nous voyons donc, par la nature du couplage entre fouille de processus et réseaux de Petri, que la prédiction relative aux processus devient possible. Le marquage initial M_0 correspond à l'état du processus au moment de lancer une prédiction, des jetons peuvent être rajoutés afin de simuler la présence ou l'arrivée de nouvelles unités, et une séquence d'activations de transitions s'enclenche grâce aux matrices d'incidences avant et arrière. Cette méthode comporte cependant plusieurs problèmes.

Tout d'abord, le temps nécessaire pour la fouille de processus est généralement considérable et au moins quadratique en fonction du nombre d'événements contenu dans un journal d'événements. Quelques dizaines de milliers de lignes suffisent à rendre ces algorithmes prohibitivement longs.

De même, problème principalement applicatif, mais de taille : un tel modèle ne permet de prédire que les états futurs du processus concerné. Cela n'implique aucunement d'avoir une quelconque information sur les unités elles-mêmes. Il s'agit d'une vision à l'échelle du processus et non de l'unité : on peut une série d'événements ordonnés dans le temps, sans être capable de les relier à des unités spécifiques. Les jetons ne sont effectivement pas des unités, il n'y a donc pas de suivi des unités possible. Les réseaux de Petri classiques ne proposent d'ailleurs pas de méthode pour placer les événements ainsi générés dans le temps, on possède leur ordre sans avoir leur éloignement temporel.

3.2.2 Réseaux de Petri temporels

Les réseaux de Petri temporels [BR07] pallient ce problème de modélisation des durées. Par exemple, les réseaux de Petri stochastiques [FN85 ; FFN91 ; Mar+98] ajoutent aux réseaux de Petri classiques un vecteur Λ de taux d'activations qui correspond au taux d'activation de chaque transition. Typiquement, ce taux correspond au paramètre d'une loi exponentielle $\mathcal{E}(\lambda)$, justifiant le choix du nom du vecteur de ces taux d'activations. D'autres modèles incluant les durées des transitions existent et sont présentés dans [WGK20], ainsi que leurs liens avec les chaînes de Markov [FH11]. Par ailleurs, [vSS11] montre que la fouille de processus peut être utilisée à des fins de prédictions temporelles en établissant des « systèmes de transitions », retraçant les transitions entre activités et leur durée dans chaque parcours, permettant ensuite d'estimer le temps restant jusqu'à la fin du parcours incomplet étant donné son état actuel.

Un *benchmark* effectué en 2018 dans [TTZ18] fait état des performances de modèles « classiques » tels que les réseaux de Petri, les chaînes de Markov, et les approches typiques de la fouille textuelle [Jo19].

Les problèmes de temps de calculs restent présents, les prédictions d'activités sont malgré tout impossibles à retracer jusqu'à une unité précise, et les prédictions temporelles ne concernent que le temps total restant et non la durée de chaque activité restante.

3.3 Apprentissage profond

3.3.1 Approches hybrides

Certains travaux mettent en lumière des approches hybrides de la tâche de prédiction. Les auteurs de [PS19] proposent un réseau de neurones bayésien pour la prédiction de l'allocation de ressources aux vues de prédictions d'activités et de durées, dans une optique d'optimisation de processus. D'autres, comme les auteurs de [TD19], évoquent une méthode alliant fouille de processus et réseaux de neurones afin de prédire l'activité suivante. Enfin, des algorithmes évolutionnistes de fouille de règles de décisions ont été développés dans [Már+17] afin de prédire des indicateurs de processus. Ces derniers sont des métriques

d'évaluations qui peuvent mesurer des caractères propres aux unités, ou des caractères à l'échelle du journal d'événements entier.

Ces approches traitent des problématiques connexes à cette thèse et constituent une base de travail pour l'étape suivant la prédiction d'événements. En effet, la prédiction d'événements d'une unité ne donne pas explicitement la marche à suivre afin d'éviter sa réalisation si celle-ci s'avérait préjudiciable. De même, les indicateurs de processus, bien que potentiellement calculables à partir d'un ensemble exhaustif d'unités prédites, sont certainement mieux prédits par un modèle spécialisé, ne serait-ce que parce que la prédiction d'événements d'unités ne tient pas compte de l'arrivée future de nouvelles unités, pour ne citer que ces exemples.

3.3.2 Réseaux de neurones récurrents

Long-Short Term Memory (LSTM)

L'approche de modélisation classique étant peu avantageuse dans un contexte industriel, entre données massives et besoins spécifiques des clients sur la prédiction d'unités et non de processus au sens global, l'apprentissage machine entre en jeu. Les auteurs de [Kra+21] établissent une comparaison entre les méthodes classiques d'apprentissage machine, telles que les machines à vecteurs de support [Vap98] et les forêts aléatoires [Ho95], comparées aux réseaux de neurones tels que les perceptrons multicouches [Van86], basés sur la rétropropagation de l'erreur [RM87] ou les réseaux récurrents types *Long-Short Term Memory* (LSTM) [HS97]. Cet article montre la supériorité de l'apprentissage *via* les réseaux de neurones, en particulier récurrents, dans le cas de la prédiction d'issues.

Les LSTM sont utilisés de façon assez extensive dans la prédiction de séquences, puisque leur architecture est par définition taillée pour traiter la donnée séquentielle. Décrivons leur architecture.
Soit $x = \langle x_1, x_2, \ldots, x_n \rangle$ une séquence donnée. Deux états caractérisent une cellule LSTM : un état caché, noté h, et un état de la cellule, noté c, qui correspondent respectivement à la mémoire à court terme et la mémoire à long terme du réseau récurrent. Différentes « portes » sont calculées, chacune ayant un rôle particulier dans l'apprentissage de la structure des séquences fournies. Les états caché et de la cellule initiaux sont généralement des vecteurs nuls, ou alors des vecteurs aléatoires typiquement tirés dans des lois uniformes continues. Classiquement, $\odot$ correspond au produit d'Hadamard, $\sigma(\cdot)$ correspond à la fonction sigmoïde et $\tanh(\cdot)$ à la tangente hyperbolique :

$$\sigma : \begin{array}{l} \mathbb{R} \longrightarrow]0;1[\\ x \longmapsto \dfrac{1}{1+e^{-x}} \end{array} \qquad \tanh : \begin{array}{l} \mathbb{R} \longrightarrow]-1;1[\\ x \longmapsto \dfrac{1-e^{-2x}}{1+e^{-2x}}. \end{array}$$

Ces fonctions sont appliquées élément par élément sur les vecteurs calculés dans une cellule LSTM. Ainsi, pour chaque élément x_t de x, une cellule LSTM effectue les calculs suivants, schématisés dans la figure 3.4 :

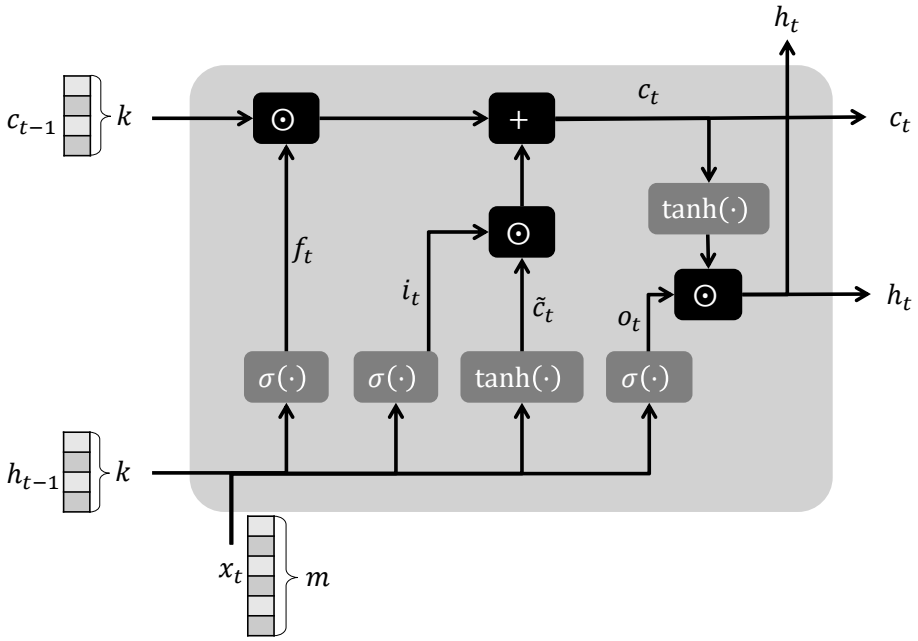

FIGURE 3.4 – Schéma d'une cellule LSTM

$$
\begin{array}{ll}
f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) & \textit{(forget gate)} \\
i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) & \textit{(input gate)} \\
o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) & \textit{(output gate)} \\
\widetilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) & \text{(mémoire candidate)} \\
c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t & \text{(état de la cellule)} \\
h_t = o_t \odot \tanh(c_t) & \text{(état caché)}
\end{array}
$$

Ici, f_t, i_t et o_t correspondent respectivement aux *forget gate*, *input gate* et *output gate*. Dans chacune, des matrices de poids W_f, W_i, W_o et W_c de dimensions $k \times m$ respectivement donnent des vecteurs qui sont des combinaisons linéaires des m valeurs de x_t. De même des matrices U_f, U_i, U_o et U_c de dimensions $k \times m$ respectivement donnent des vecteurs qui sont des combinaisons linéaires de l'état caché de la cellule précédente, h_{t-1}. Des vecteurs de biais b_f, b_i, b_o et b_c de dimension k respectivement sont ajoutés. Les sorties h_t et c_t sont des vecteurs de dimension k, et ce k est choisi par l'utilisateur en tant qu'hyperparamètre du modèle. Les matrices de poids et les vecteurs de biais peuvent être initialisés aléatoirement ou par leur élément nul.
La *forget gate*, dont les sorties sont entre 0 et 1, vise à diminuer l'impact des valeurs jugées

non pertinentes pendant l'apprentissage, et maintenir celles jugées pertinentes. Le produit d'Hadamard de la *forget gate* avec l'état de la cellule précédente c_{t-1} permet de diminuer l'impact de la donnée jugée non pertinente dans l'état de la cellule au fur et à mesure des étapes, sélectionnant ce qui se maintiendra dans cet état de la cellule qui est comparable à une mémoire à long terme. Ceci donne la première partie de l'expression de c_t : $f_t \odot c_{t-1}$.
L'*input gate* est constituée exactement des mêmes entrées x_t et h_{t-1}, et effectue des opérations parfaitement analogues à la *forget gate*, *idem* pour l'*output gate*.
La différence de l'*input gate* est que celle-ci est multiplée par produit d'Hadamard à $\widetilde{c}_t$, qui est d'une forme elle aussi similaire aux différentes portes, mais au lieu d'une sigmoïde, on a une tangente hyperbolique. Il s'agit ici de pondérer positivement ou négativement les valeurs de l'état caché précédent h_{t-1} et de la donnée actuelle x_t afin de mettre à jour leur influence sur c_t. Le produit d'Hadamard de $\widetilde{c}_t$ avec l'*input gate* permet de pondérer cette nouvelle mémoire, et c_t correspond à la somme de l'état de la cellule précédente pondérée par la *forget gate*, et de la nouvelle mémoire pondérée par l'*input gate*, menant à $c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t$.
Enfin le nouvel état caché h_t correspond à l'état de la cellule c_t passé par la tangente hyperbolique, pondéré par l'*output gate*.

Cette architecture vise à propager dans les différentes cellules LSTM la donnée apprise comme étant pertinente à long terme à l'aide des états des cellules. Les états cachés alimentent cette propagation en s'ajoutant à la donnée, et servent également d'éléments de sortie des cellules LSTM. Typiquement, dans la prédiction d'issues de parcours, le dernier état caché calculé sert de sortie du réseau de neurones. Cet état passe par une fonction $Softmax(\cdot)$ permettant de le transformer en vecteurs de probabilités, et l'élément contenant la probabilité la plus haute doit correspondre à l'issue observée dans la réalité. Un raisonnement parfaitement analogue peut être appliqué pour prédire l'activité suivante d'un parcours. De plus, concaténer l'activité prédite au parcours incomplet permet d'effectuer des prédictions en chaîne et de prédire ainsi les activités suivantes en chaîne.

C'est ce que proposent les auteurs de [ERF17], à un changement près : au lieu de sélectionner l'élément de sortie possédant la probabilité la plus haute, ils échantillonnent un élément de sortie selon les probabilités associées. Il y a donc un aléatoire dans la prédiction, appelé « hallucination ». Ceci permet la prédiction de séquences d'activités plus diverses et potentiellement nouvelles que de prendre l'élément le plus probable systématiquement.

Une autre approche, plus tardive, peut être trouvée dans [Tax+17], où les auteurs choisissent plutôt un LSTM multi-tâches :

— D'abord, les événements d'une unité sont pré-traités par un premier LSTM.
— Ensuite, ce pré-traitement est dupliqué et envoyé dans deux LSTM séparés :
  — Le premier LSTM prédit l'activité suivante, comme décrit ci-avant.
  — Le deuxième LSTM prédit la durée de l'activité prédite.

L'hallucination n'y est pas utilisée, la prédiction la plus probable est systématiquement choisie pour l'activité et sa durée. Cette approche est plus performante que [ERF17] et permet la prédiction conjointe d'activités et de temps. La prédiction des séquences d'événements se fait également par prédictions itératives. Cependant, toutes ces méthodes de prédictions successives possèdent le même désavantage de taille : la propagation de l'erreur. En admettant que le réseau de neurones ait une précision dans les prédictions d'activités de 95% dans tous les cas de figure, une prédiction de 10 activités de long n'aurait que 59% de chances de correspondre à la réalité. Avec une précision de 80%, nous tombons à 11% de chances de prédire la séquence entière correctement.
Les auteurs de [LWW19] proposent une architecture d'encodeur-décodeur permettant des générations plus précises de suffixes sur les activités et les temps.

Problème supplémentaire, un LSTM ne considère pas, par défaut, les variations temporelles entre les étapes des séquences.

Time-Aware Long-Short Term Memory (T-LSTM)

Les *Time-Aware LSTM*, ou T-LSTM [Bay+17], prennent en compte ces irrégularités dans les durées. Le raisonnement est simple : l'impact de l'étape $t-1$ doit être ajusté selon la durée écoulée jusqu'à l'étape t. En effet, plus cette durée est longue, moins la mémoire à court terme de la cellule $t-1$ doit avoir d'impact sur la prédiction à l'étape t.

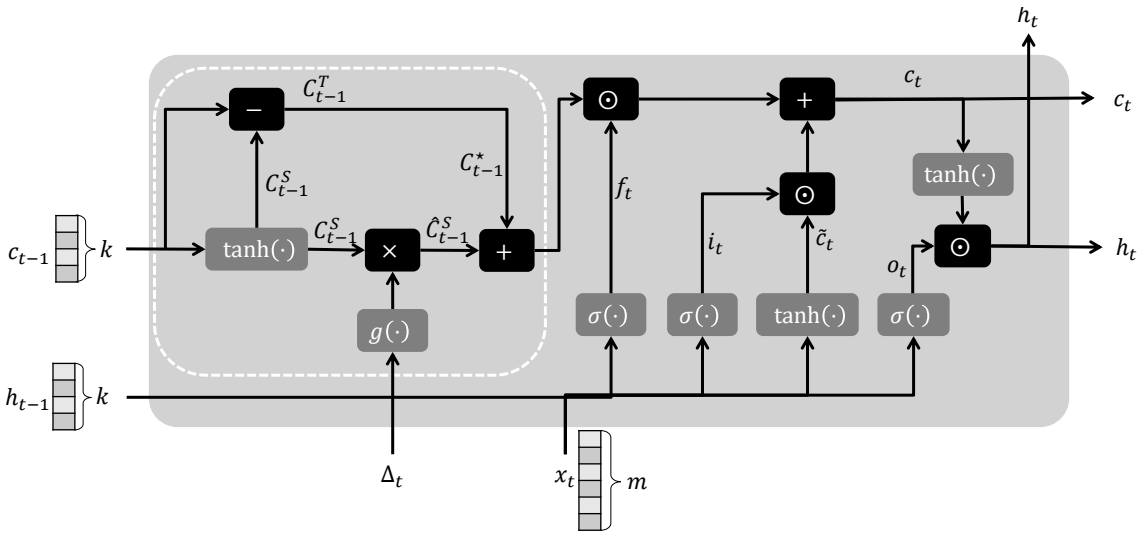

FIGURE 3.5 – Schéma d'une cellule LSTM

La figure 3.5 schématise les calculs effectués dans une cellule T-LSTM, détaillés dans les équations ci-après. Comme dans une cellule LSTM classique, x_t est de dimension m, h_{t-1} et c_{t-1} sont de dimension k hyperparamètre du modèle. On voit sur cette figure que les modifications apportées à l'architecture des LSTM, ici dans la zone en pointillés, ne concerne que c_{t-1} en amont des calculs des *forget*, *input* et *output gates*.

$$
\begin{aligned}
C^S_{t-1} &= \tanh(W_d c_{t-1} + b_d) && \text{(mémoire à court-terme)}\\
\widehat{C}^S_{t-1} &= g(\Delta_t) C^S_{t-1} && \text{(mémoire à court terme réduite)}\\
C^T_{t-1} &= c_{t-1} - C^S_{t-1} && \text{(mémoire à long terme)}\\
C^\star_{t-1} &= C^T_{t-1} + \widehat{C}^S_{t-1} && \text{(mémoire à long terme ajustée)}\\
f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) && \textit{(forget gate)}\\
i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) && \textit{(input gate)}\\
o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) && \textit{(output gate)}\\
\widetilde{c}_t &= \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) && \text{(mémoire candidate)}\\
c_t &= f_t \odot C^\star_{t-1} + i_t \odot \widetilde{c}_t && \text{(état de la cellule)}\\
h_t &= o_t \odot \tanh(c_t) && \text{(état caché)}
\end{aligned}
$$

Dans ces équations, le terme $g(\Delta_t)$ est la partie centrale. Δ_t correspond à $\pi_{\mathcal{T}}(e_t) - \pi_{\mathcal{T}}(e_{t-1})$. La fonction g doit être monotone et décroissante. Les auteurs suggèrent d'utiliser simplement $g(\Delta_t) = \frac{1}{\Delta_t}$ sur les jeux de données où Δ_t a tendance à être petit. Dans le cas contraire, les auteurs proposent plutôt $g(\Delta_t) = \frac{1}{\ln(e+\Delta_t)} \cdot\ g(\Delta_t)$ est alors exprimé dans l'état de la cellule :

$$
\begin{aligned}
c_t &= f_t \odot C^\star_{t-1} + i_t \odot \widetilde{c}_t\\
&= f_t \odot \left(C^T_{t-1} + \widehat{C}^S_{t-1}\right) + i_t \odot \widetilde{c}_t\\
&= f_t \odot \left(c_{t-1} - C^S_{t-1} + g(\Delta_t) C^S_{t-1}\right) + i_t \odot \widetilde{c}_t\\
&= f_t \odot \left(c_{t-1} - (1 - g(\Delta_t)) \tanh(W_d c_{t-1} + b_d)\right) + i_t \odot \widetilde{c}_t.
\end{aligned}
$$

Ainsi, la différence temporelle entre deux étapes d'une séquence est bien prise en compte dans la *forget gate* de façon inversement proportionnelle.

Des articles tels que [Ngu+20] utilisent l'architecture T-LSTM afin de modéliser cette dégradation de l'influence des événements trop éloignés dans le temps malgré leur succession dans un parcours. Cependant, les LSTM ne semblent pas posséder de mémoire à long terme à proprement parler, comme le démontrent les auteurs de [Zha+20]. La propagation de l'erreur est toujours à l'œuvre dans ce modèle pour la génération de séquences. Par ailleurs, des architectures plus à mêmes de retenir un contexte sur de longues séquences sont donc envisageables, telles que les *Transformers*.

3.3.3 *Transformers*

Les *Transformers* [Vas+17] prennent le contre-pied des LSTM et ne font aucunement référence à une mémoire à long ou court terme. Au lieu de cela, la notion d'*Attention* est au centre de ce modèle. Les entrées du calcul d'Attention sont des requêtes Q (de l'anglais *queries*), des clefs K (de l'anglais *keys*), Q et K de dimension $d_k > 0$, et des valeurs V (de l'anglais *values*) de dimension $d_v > 0$. L'Attention est calculée comme suit :

$$
\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q \cdot K^\top}{\sqrt{d_k}}\right) \cdot V
$$

Dans le contexte du *PBPM*, requêtes, clefs et valeurs peuvent toutes êtres calculées à partir des activités, donc $d_v = d_k$. On nomme $\mathbb{x}$ le vecteur d'activités d'une unité. Pour obtenir les requêtes, clefs et valeurs, un simple produit avec une matrice est effectué : on projette les activités dans un espace de requêtes Q, un espace de clefs K et un espace de valeurs V. Ainsi, les requêtes seront de la forme $\mathbb{x}W_Q$, les clefs de la forme $\mathbb{x}W_K$ et les valeurs de la forme $\mathbb{x}W_V$, avec W_Q, W_K et $W_V \in \mathcal{M}_{d_{\text{model}},d_k}$, d_{model} étant choisi par l'utilisateur.

Si l'on décompose le calcul de l'Attention, le vecteur $\mathbb{x}W_Q$ est multiplié au vecteur $\mathbb{x}W_K$, de sorte à former un score relatant les activités représentées dans $\mathbb{x}W_Q$ avec celles représentées dans $\mathbb{x}W_K$. Ainsi, une activité dans le parcours d'une unité génère un score pour toutes les autres activités dans ce parcours, qui score leur proximité contextuelle à cette activité. Ensuite, ce score de proximité contextuel est multiplié à $\mathbb{x}W_V$ afin de déterminer quelle valeur (*i.e.* quelle activité) est la plus à même de suivre l'activité observée. Puisque les requêtes, clefs et valeurs proviennent du même ensemble, il s'agit de *self-Attention*. Les valeurs présentes dans ces matrices sont l'objectif de l'apprentissage des *Transformers*.

Afin d'apprendre plusieurs contextes à plus ou moins longue distance (dans une phrase, un même mot peut être un verbe, un adjectif, ...), l'Attention multi-têtes est la concaténation de plusieurs (h, définis par l'utilisateur) systèmes d'Attention sur les mêmes entrées. On a donc :

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\dots,\text{head}_h)W_O$$

$$\text{où head}_i = \text{Attention}\left(\mathbb{x}W_Q^{(i)}, \mathbb{x}W_K^{(i)}, \mathbb{x}W_V^{(i)}\right) = \text{Softmax}\left(\frac{\mathbb{x}W_Q^{(i)} \cdot \mathbb{x}W_K^{(i)^\top}}{\sqrt{d_k}}\right) \cdot \mathbb{x}W_V^{(i)}.$$

Cette concaténation de systèmes d'Attention entre les différentes activités permet de relier ces activités selon les *patterns* présents dans les parcours, peu importe leur distance. Il n'y a aucune notion de mémoire, simplement de contextes. Cependant, afin de conserver un ordre temporel aux éléments des séquences et de l'utiliser dans le modèle, un encodage positionnel doit être effectué. Soit *pos* la position dans une séquence et $1 \leqslant n \leqslant d_{\text{model}}$ la dimension :

$$PE_{(pos,n)} = \begin{cases} \sin\left(\dfrac{pos}{10\,000^{\frac{n}{d_{\text{model}}}}}\right) & \text{si} \quad n = 2k,\ k \in \mathbb{N}^\star, \\ \cos\left(\dfrac{pos}{10\,000^{\frac{n}{d_{\text{model}}}}}\right) & \text{si} \quad n = 2k+1,\ k \in \mathbb{N}. \end{cases}$$

Cet encodage positionnel donne une valeur unique à chaque position dans une séquence, est déterministe et se généralise à des séquences plus longues de façon immédiate. Il n'y a donc pas de limite théorique à la longueur des séquences pouvant être traitées ou générées par un tel modèle, contrairement aux LSTM qui sont limités par le nombre de cellules

LSTM définies à l'avance par le modélisateur : en général, il y a autant de cellules LSTM que d'étapes dans la séquence la plus grande dans les données d'entraînement.

Des travaux ont réussi à passer à des réseaux de neurones basés sur l'Attention et aux *Transformers* tels que [Phi+20] et [BSD21], démontrant leur supériorité sur les LSTM pour la prédiction d'activités.

Cependant, ceux-ci ne tiennent jamais réellement compte du temps. Ils peuvent prédire, de façon détournée, la durée totale d'un parcours, mais il n'existe pas de modèle permettant de prédire conjointement les activités et leurs durées respectives se basant sur l'achitecture des *Transformers*.

3.3.4 Réseaux antagonistes génératifs (GAN)

Les réseaux antagonistes génératifs (GAN) [Goo+14] diffèrent des autres réseaux de neurones par le fait qu'il s'agit en réalité de deux réseaux de neurones cherchant à atteindre un équilibre de Nash [Nas50] lors de leur entraînement. Plus précisément, un GAN est composé :

— D'un générateur G, dont le but est de générer des données synthétiques.

— D'un discriminant D, dont le but est de distinguer avec succès la donnée réelle et la donnée synthétique. C'est de la classification binaire.

Le générateur a pour objectif d'entraînement de voir sa donnée classifiée comme étant réelle par le discriminant. Dans l'article d'origine, les auteurs proposent G et D sous la forme de perceptrons multicouches. La fonction de coût de l'apprentissage d'un tel modèle, proposée par les auteurs de [Goo+14], est de la forme :

$$\min_G \max_D L(D,G) = \mathbb{E}_{X\sim\mathbb{P}_r}\left[\ln D(X)\right] + \mathbb{E}_{Y\sim\mathbb{P}_G}\left[\ln\left(1-D(Y)\right)\right],$$

où $\mathbb{P}_r$ est la loi de probabilité de la donnée réelle, et $\mathbb{P}_G$ est la loi de probabilité de la donnée générée, elle-même implicitement définie par $Y = G(Z)$, $Z \sim \mathbb{P}_Z$. Pour de la génération de donnée, $\mathbb{P}_Z$ est généralement une loi uniforme continue multivariée. Ce coût d'apprentissage est dérivé de la divergence de Jensen-Shannon [Lin91], elle-même étant une symétrisation de la divergence de Kullback-Leibler [KL51]. Ces divergences quantifient la distance séparant deux lois de probabilités, d'où leur intérêt dans le cadre des GANs : on cherche à minimiser cette distance entre donnée générée et donnée réelle au travers de G et D, chacun alimentant l'autre lors de l'apprentissage.

Originellement conçus pour la génération d'images, les travaux de [Tay+20] et [TR20]ont prouvé que les GANs pouvaient être particulièrement efficaces dans la prédiction de séquences d'activités, y compris quand celles-ci sont longues, et dans la prédiction des séquences de durées.

L'avantage des GANs dans ce cas est que la qualité des données générées est évaluée au travers du discriminant, une séquence d'activités générée sera donc évaluée en tant que séquence d'un bloc, et non comme une succession de prédictions dont la qualité est jugée individuellement en supposant les prédictions précédentes comme vraies. On s'affranchit donc de la propagation de l'erreur. La contrepartie est que la première activité d'une prédiction peut être fausse par exemple, mais l'allure globale de la séquence prédite sera proche de la réalité. C'est pourquoi il s'agit du modèle qui sert de base aux recherches dans cette thèse : l'accent est mis sur les prédictions de séquences, et sur la qualité globale de ces prédictions.

Ces réseaux ont cependant, encore une fois, des problèmes de taille : premièrement, leur apprentissage est réputé laborieux. On ne cherche pas une minimisation de l'erreur, mais une convergence des deux réseaux de neurones vers un équilibre de Nash, bien plus compliqué à atteindre en pratique. Il faut pour cela s'assurer que l'apprentissage du générateur et du discriminant soient synchronisés, sans que l'un ne dépasse l'autre de façon irréversible. Ensuite, bien que la convergence soit atteinte, il est possible de subir un phénomène de *mode collapse* : le générateur génère la même donnée peu importe Z, et le discriminant est incapable de différencier cette donnée générée de la donnée réelle. Dans le cadre du *PBPM*, cela revient à systématiquement prédire la trace majoritaire, avec des durées fixes pour chaque étape.
Enfin, la convergence peut être longue. Même sans *mode collapse* et avec la certitude d'une convergence de qualité, le nombre d'époques d'apprentissage peut être considérable avant d'atteindre un réel équilibre entre le générateur et le discriminant.

Ces difficultés d'apprentissage compromettent leur utilisation dans un contexte industriel où les tentatives d'apprentissage ne peuvent pas être nombreuses par manque de temps, ou de ressources informatiques qui permettraient d'en lancer suffisamment pour qu'au moins une soit fructueuse.

3.3.5 Autres modèles

Réseaux de convolution (CNN)

Les réseaux de convolution (Convolutional Neural Networks (CNN)) [LeC+98], bien qu'antérieurs aux *Transformers*, sont plus atypiques en *PBPM* étant donné qu'ils se prêtent plutôt à le reconnaissance ou la segmentation d'images. Un CNN est composé d'au moins une couche de convolution, contenant un noyau correspondant à une matrice (ou tenseur), qui parcourt l'image en effectuant la convolution des pixels couverts par le noyau avec lui-même afin d'obtenir un filtre dont le but est de détecter un certain type de *patterns*.
Plus formellement, comme décrit par les auteurs de [DAB19], soit $x \in \mathcal{M}_{nm}$. Étant donné une l-ième couche de convolution, le k-ième filtre sur x_{ij} est déterminé par la matrice de poids W_k^l et le vecteur de biais b_k^l avec une fonction d'activation telle que la sigmoïde, comme suit :

$$h_{ijk}^l = \sigma\left(W_k^l \odot x_{ij}^l + b_k^l\right)$$

Une couche de *pooling* suivant une couche de convolution permet la réduction des dimensions de l'image en maintenant les *patterns* globaux mis en avant par le filtre. Souvent, l'opération retient simplement le maximum de chaque convolution. L'apprentissage consiste à trouver les valeurs optimales pour ces filtres selon les images utilisées et la tâche à exécuter.

Les réseaux de neurones convolutifs ont également été employés pour la prédiction de l'activité suivante, comme dans [DAB19], la prédiction de l'activité suivante et du temps dans [Pas+19a], et la prédiction d'issue dans [Pas+20]. Ceci qui demande une conversion astucieuse des unités en images.

Bien que cette méthode montre que la conversion de données horodatées en images soit efficace dans une optique prédictive, elle ne permet pas la prédiction de séquences, seulement de l'activité et / ou du temps suivant, le nombre de paramètres est globalement plus élevé que dans un LSTM, et la prédiction itérative, si elle est possible, est toujours victime de la propagation de l'erreur avec une précision par activité encore trop faible.

Réseaux de neurones en graphes (GNN)

De nouvelles architectures de réseaux de neurones, notamment les réseaux de neurones en graphes (*Graph Neural Networks* (GNN)) [ZCZ20], ont été testées avec succès dans des articles tels que [Har+20]. Cette approche semble d'ailleurs intuitive puisque les processus peuvent très facilement être représentés graphiquement par des cartes, des graphes et des réseaux.

L'utilisation d'un *GNN* dans cet article vise à prédire l'issue d'une unité en cours, et non l'événement suivant (ou la séquence d'événements suivants). L'architecture choisie a l'intérêt d'être explicable, dans le sens que les activités sont scorées selon leur impact sur la prédiction de l'issue d'une unité.

Cependant, la précision sur la prédiction d'issues ne dépasse pas les modèles plus classiques cités plus haut, et ce modèle ne s'applique pas à la prédiction d'activités ou de durées.

3.4 Commentaires

Des *benchmarks* ont été réalisés dans [RVL20] pour comparer les approches d'apprentissage profond, ainsi que dans [Ver+19] pour les prédictions temporelles spécifiquement. Ces *benchmarks* n'incluent pas ni les GANs, ni les Transformers, ces derniers étant postérieurs à la publication de ces articles de *benchmarking*. Par ailleurs, les modèles testés dans ces articles génèrent des prédictions de séquences bien souvent médiocres : les modèles testés sont efficaces pour la prédiction de l'événement suivant seulement.

Dernier élément, entièrement absent de la recherche en *PBPM*, à part dans les réseaux de Petri par leur construction : aucun modèle cité ci-avant ne tient compte de la présence

simultanée de plusieurs unités, pouvant s'influencer entre elles. Nous pensons pourtant qu'il s'agit d'une donnée cruciale, nécessaire pour l'obtention de prédictions aussi précises que possible. Nous pouvons donc passer aux différentes contributions de cette thèse, à commencer par sa pierre angulaire : la création du peuplement de processus.

Chapitre 4

Peuplement de processus

4.1 Introduction

La modélisation de parcours individuels demande de transformer la donnée brute tirée des journaux d'événements en matrices utilisables par les modèles prédictifs choisis. Le problème de ces approches est double :

— Les modèles sont entraînés à prédire les parcours en tant qu'entités isolées, or un parcours représente le trajet d'une unité au sein d'un processus, qui est elle-même entourée d'autres unités qui peuvent s'influencer entre elles. En effet, un surplus d'unités dans un processus peut causer un embouteillage qui ralentirait potentiellement les unités moins loin dans ce processus. Au contraire, une absence d'unités pourrait tout à fait fluidifier le processus de façon notable. Nous pouvons également noter, par exemple, les effets « *bucket* », où les unités s'accumulent à un endroit du processus jusqu'à-ce qu'elles soient en assez grand nombre pour passer à l'activité suivante.

— Afin de prédire les temps correspondant aux activités prédites, les horodatages sont transformés en durées. L'intérêt de prédire des durées au lieu de dates réside dans le fait que rien n'empêche un modèle prédictif, en pratique, de prédire un temps $t + 1$ antérieur au temps t. Prédire une durée règle ce problème en permettant la prédiction d'une quantité positive ou nulle qui ne fera que s'ajouter au temps t pour obtenir le temps $t + 1$. Cette transformation retire toute information de l'existence dans le temps des unités. Pourtant les processus exhibent généralement des saisonnalités dans leur traitement des unités : un centre d'appel a généralement deux pics d'activité, un le matin et un l'après-midi, avec une baisse des appels le midi et un arrêt complet la nuit, or le volume d'appels peut tout à fait influencer la rapidité d'exécution des affaires relatives au centre d'appel. Ces phénomènes sont également à considérer pour les fins de semaines et jours fériés. De plus, transformer les horodatages en durées retire toute

notion de simultanéité dans le processus : impossible de savoir quelles unités sont en même temps dans le processus.

La création de variables encodant l'activité globale d'un processus donné semble donc cruciale dans la prédiction de parcours individuels, or cette approche n'a jamais été considérée dans l'état de l'art. Pour cette raison, nous avons créé la notion de *peuplement de processus*, qui se trouve être une pierre angulaire de cette thèse et de ses avancées. Nous explorons dans ce chapitre notre définition du peuplement de processus, son illustration sur de la donnée réelle, puis une étude des corrélations entre peuplements.

4.2 Définition

Le peuplement tel que nous l'avons défini est fait pour être simple à calculer et à interpréter : il s'agit simplement du nombre d'unités dans chaque activité d'un processus à un instant donné. Réussir à calculer le peuplement d'un processus permet d'expliciter directement le nombre d'unités présentes simultanément dans le processus, ainsi que leur position dans ce dernier. Hypothétiquement, il deviendrait alors plus aisé de pallier les problèmes exposés en introduction de ce chapitre. Afin de fixer les idées, il convient de donner au peuplement une définition formelle.

Soit un journal d'événements L enregistrant les événements d'un processus donné. Soit $\mathcal{C}$ l'espace des identifiants, $\mathcal{E}$ l'espace des événements et $\mathcal{A}$ l'espace de toutes les activités enregistrées dans le processus, de cardinal $|\mathcal{A}|$. Soit $t \in \mathcal{T}$ un horodatage. Alors, en utilisant les fonctions que nous avons introduites dans la section 2.1 des préliminaires théoriques, nous pouvons définir une nouvelle fonction qui retourne le vecteur des peuplements pour chaque activité à l'instant t donné, que nous appelons $Cd(\cdot)$ (venant de l'anglais « *crowding* »). On suppose L classé par ordre chronologique des unités. Alors :

$$Cd(t) = \left(\sum_{i=1}^{|L|_e - 1} \mathbb{1}(\{e_i;\ (\pi_{\mathcal{A}}(e_i) = a_1) \wedge (t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1})[)\}), \ldots, \right.$$
$$\left. \sum_{i=1}^{|L|_e - 1} \mathbb{1}(\{e_i;\ (\pi_{\mathcal{A}}(e_i) = a_{|\mathcal{A}|}) \wedge (t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1})[)\}) \right) \tag{4.1}$$

L'idée derrière cette formule est la suivante : au temps t, si une unité a commencé une activité $a \in \mathcal{A}$ à un temps $t_1 \leqslant t$ et passe à l'activité suivante au temps $t_2 > t$, alors cette unité se trouve dans l'activité a au temps t. La dernière activité d'un parcours, signant sa fin, n'a pas d'événement suivant. Ainsi, on suppose que l'activité de fin se termine instantanément, créant un intervalle vide. Dans le cas de deux horodatages, l'horodatage de fin permet de créer un intervalle non-vide.

Le peuplement constitue un vecteur de $|\mathcal{A}|$ composantes, qui est finalement un simple vecteur de comptages décrivant le nombre d'unités dans chaque activité au temps t. Dans un journal d'événements L, $Cd(\cdot)$ peut être appliquée à chaque horodatage enregistré, ce qui donne une matrice $|L|_e \times |\mathcal{A}|$ d'entiers, notée $C = (Cd(\pi_{\mathcal{T}}(e_j)), j = 1, \ldots, |L|_e)$, dont chaque colonne est une série temporelle à valeurs dans $\mathbb{N}$ décrivant le peuplement d'une activité au cours du temps.

Une considération d'ordre computationnelle à ne pas négliger concerne les temps de calculs. La formule 4.1 est, une fois implémentée en Python 3.9, lente : celle-ci avoisine 500 itérations par seconde sur *Helpdesk* sur un processeur Skylake 6132 à cadence de 2,6GHz et 2x14 cœurs, pour l'exemple le plus rapide. On descend à 100 itérations par secondes sur *BPI2017* sur le même équipement pour l'exemple le plus lent. Les peuplements de *Helpdesk* prennent donc $13\,710 \div 500 = 27,42$ secondes à être calculés, et ceux de *BPI2017* mettent 2h54 à être calculés. Or la formule 4.1 peut être modifiée.

Propriété 1 (Yoann Valero (2023))**.** *$\forall t \in \mathcal{T}$, toute composante $k \in \{1, \ldots, |\mathcal{A}|\}$ du vecteur $Cd(t)$ est égale à :*

$$Cd(t)_k = \sum_{i=1}^{|L|_e-1} \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_i)\}) - \sum_{i=1}^{|L|_e-1} \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_{i+1})\}).$$

En effet, prenons un terme $i \in \{1, \ldots, |L|_e - 1\}$ de la somme d'un composante $Cd(t)_k$. On a :

$$\begin{aligned}\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1})[\} &= \{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\setminus [\pi_{\mathcal{T}}(e_{i+1}); +\infty[\} \\ &= \{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\cap [0; \pi_{\mathcal{T}}(e_{i+1})[\} \\ &= \{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\} \cap \{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}.\end{aligned}$$

Or $\{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}$ est le complémentaire de $\{e_i;\ t \in [\pi_{\mathcal{T}}(e_{i+1}); +\infty[\}$ dans $\mathcal{E}$. Donc :

$$\mathbb{1}\left(\{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}\right) = 1 - \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_{i+1}); +\infty[\}\right).$$

Par ailleurs, pour A et B deux sous-ensembles d'un ensemble E, une propriété des fonctions indicatrices est que $\mathbb{1}(A \cap B) = \mathbb{1}(A) \cdot \mathbb{1}(B)$. Ainsi :

$$\begin{aligned}
&\mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\} \cap \{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}\right) \\
&= \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\}\right) \cdot \mathbb{1}\left(\{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}\right) \\
&= \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\}\right) \cdot (1 - \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_{i+1}); +\infty[\}\right) \\
&= \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\}\right) - \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\}\right) \cdot \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_{i+1}); +\infty[\}\right) \\
&= \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\}\right) - \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\} \cap \{e_i;\ t \in [\pi_{\mathcal{T}}(e_{i+1}); +\infty[\}\right) \\
&= \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\}\right) - \mathbb{1}\left(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_{i+1}); +\infty[\}\right) \\
&= \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_i)\}) - \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_{i+1})\}).
\end{aligned}$$

Revenons à la formule 4.1. Pour une composante $k \in \{1, \ldots, |\mathcal{A}|\}$ du vecteur $Cd(t)$, l'ensemble $\{e_i;\ (\pi_{\mathcal{A}}(e_i) = a_k) \wedge (t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1})[)\}$ peut être récrit sous la forme :

$$\begin{aligned}
&\{e_i;\ (\pi_{\mathcal{A}}(e_i) = a_k) \wedge (t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1})[)\} \\
&= \{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\} \cap \{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1})[\} \\
&= \{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\} \cap \{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\} \cap \{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}.
\end{aligned}$$

En prenant l'indicatrice de cet ensemble, on a donc :

$$\begin{aligned}
&\mathbb{1}(\{e_i;\ (\pi_{\mathcal{A}}(e_i) = a_k) \wedge (t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1}[)\}) \\
&= \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\} \cap \{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\} \cap \{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}) \\
&= \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot \mathbb{1}(\{e_i;\ t \in [\pi_{\mathcal{T}}(e_i); +\infty[\}) \cdot \mathbb{1}\{e_i;\ t \in [0; \pi_{\mathcal{T}}(e_{i+1})[\}).
\end{aligned}$$

Donc, grâce au résultat démontré plus haut :

$$\begin{aligned}
&\mathbb{1}(\{e_i;\ (\pi_{\mathcal{A}}(e_i) = a_k) \wedge (t \in [\pi_{\mathcal{T}}(e_i); \pi_{\mathcal{T}}(e_{i+1}[)\}) \\
&= \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot (\mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_i)\}) - \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_{+1})\})) \\
&= \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_i)\}) - \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_{+1})\}).
\end{aligned}$$

De ce fait, une composante $k \in \{1, \ldots, |\mathcal{A}|\}$ du vecteur $Cd(t)$ peut s'écrire sous la forme :

$$\begin{aligned}
Cd(t)_k = &\sum_{i=1}^{|L|_e - 1} \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_i)\}) - \\
&\sum_{i=1}^{|L|_e - 1} \mathbb{1}(\{e_i;\ \pi_{\mathcal{A}}(e_i) = a_k\}) \cdot \mathbb{1}(\{e_i;\ t \geqslant \pi_{\mathcal{T}}(e_{i+1})\}). \qquad (4.2)
\end{aligned}$$

La formule 4.2 indique que l'on peut effectuer le calcul en deux temps : d'abord, pour le peuplement d'une activité à un temps donné t, on compte tous les événements dont l'activité

adéquate a commencé avant t. Mais cela compte les événements ayant aussi terminé avant t. On soustrait donc les événements ayant la bonne activité, mais dont la fin précède t. Dans la pratique, le peuplement au dernier temps enregistré est simplement constitué du premier membre de la soustraction dans la formule 4.2, puisqu'il n'y a pas de temps suivant indiquant une fin de ces activités. Cette formule alternative s'avère en moyenne 22 700 fois plus rapide en termes de temps de calcul que la formule 4.1 : on passe à un millième de seconde pour le calcul de tous les peuplements de *Helpdesk*, et une demi-seconde pour les peuplements de *BPI2017* sur le même équipement informatique.

Notons que la différence n'est pas faite avec un ou deux horodatages. En effet, dans le cas de figure de deux horodatages, nous pourrions chercher à calculer le peuplement dans les transitions entre activités. Le problème de cette approche réside dans la combinatoire observée dans les journaux d'événements pour les passages d'une activité à une autre : nous comptons *a minima* des dizaines, sinon des centaines de transitions observées entre activités comme exposé dans le tableau 2.6 ; ajouter autant de colonnes de comptages est prohibitif en termes de temps de calcul et de mémoire utilisée pour contenir ces données. De plus, dans le cadre d'un modèle prédictif, l'apparition d'une seule nouvelle transition entre activités demanderait un ré-apprentissage entier du modèle afin d'inclure cette dimension supplémentaire dans les données. Il y a la possibilité de définir une variable globale "transition" qui indique si une unité est dans une activité ou en transition, mais nous perdrions l'information de sa position dans le processus. Il fut donc décidé de considérer que dans le cas de deux horodatages, une activité englobe également sa transition vers l'activité suivante dans les calculs de peuplements.

Illustrons à présent le peuplement sur les jeux de données exposés en section 2.2.

4.3 Illustration

4.3.1 Peuplements calculés sur données réelles

Dans cette section, par souci de lisibilité, seules les six activités les plus peuplées en moyenne dans les différents journaux d'événements sont montrées dans les différentes figures. Il est important de noter que les peuplements affichés dans cette section sont classés dans l'ordre chronologique croissant, et non par unité et ordre chronologique à la fois : l'intérêt est de montrer ces peuplements au fur et à mesure du temps.

Nous avons sur la figure 4.1 des peuplements du jeu de données *Helpdesk*. L'activité 2 contient le plus de peuplements en moyenne, suivie par l'activité 1 puis l'activité 5, ainsi que la 3. Les activités 6 et 7 ne sont que peu peuplées, en particulier l'activité 7 qui a un peuplement de 0 sur toute la durée observée, excepté fin 2 012. Le fait que la trace majoritaire de *Helpdesk* soit $1, 2, 3$ mais que l'activité 3 soit aussi peu peuplée en moyenne que les activités 1 et 2 laisse effectivement entendre, comme l'a montré le tableau 2.3, que des répétitions gonflent les peuplements d'activités de milieu de parcours.

La figure 4.2 montre cette fois-ci les peuplements des activités de *Traffic fines*. Les activités 1, 3 et 2 ont toutes un pic de peuplement autour de 2 008, nous observons en revanche dans les activités 4 et 5 un effet « *bucket* » fin 2 009 : l'activité 5, à 0 jusqu'à 2 009, affiche soudainement un peuplement à plus de 3 000 avant de retomber immédiatement à 0. Au même moment, l'activité 4, dont le peuplement croissait de façon notable au-delà de 3 500, baisse soudainement à la même date à moins de 500. Il semble donc que le processus attende une accumulation à une ou plusieurs activités données avant de déclencher l'activité 5 pour 3 000 unités simultanément. Le fait que le peuplement de processus explicite cette dynamique semble crucial pour une tâche prédictive, nous en verrons l'impact dans le chapitre suivant.

BPI2012 (W) et *BPI2012* exhibent des peuplements similaires dans les figures 4.3 et 4.4. On y observe dans les activités 1 et 2 deux plateaux séparés par une légère décroissance. Ces plateaux semblent également présents dans l'activité 3. L'activité 5, bien que modérément peuplée, ne possède pas ces plateaux et semble plutôt en très légère croissance sur la période observée.

Sur la figure 4.4, des peuplements parfaitement analogues à la figure précédente peuvent être observés, seuls les numéros d'activités ont changé.

Enfin, la figure 4.5, qui montre les peuplements de *BPI2017*, met en lumière des saisonnalités intéressantes. L'activité 10, la plus peuplée en moyenne, possède un peuplement similaire à ceux observés dans *BPI2012 (W)* et *BPI2012*. Cependant, l'activité 15 montre une croissance de son peuplement suivie d'une baisse soudaine environ tous les mois et demi, avec régularité. Nous observons d'ailleurs que les pics et vallées du peuplement de l'activité 15 montrent un changement de régime qui semble suivre celui montré par l'activité 3. L'activité 8 est également intéressante par la présence de deux pics soudains, avoisinant un peuplement de 250 pendant environ un mois chacun, avant de décroître soudainement.

Il semble ainsi que le peuplement de processus contienne une quantité d'information insoupçonnée sur les saisonnalités du processus étudié, des seuils autrement indisponibles tels que les conditions d'un effet *bucket*, ainsi que sur les répétitions.

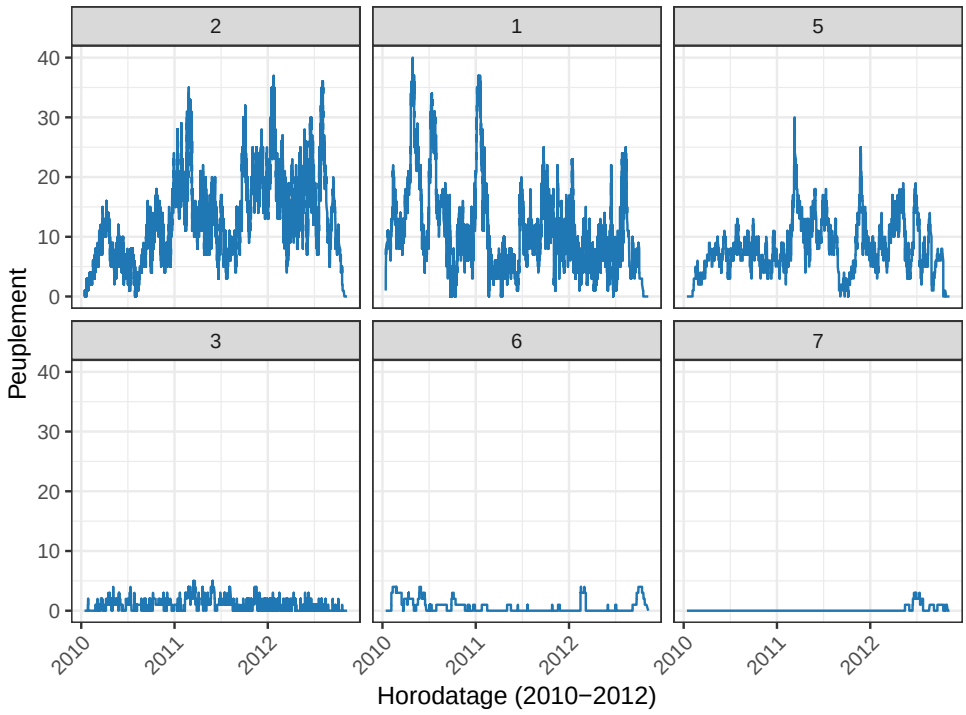

FIGURE 4.1 – Peuplements des 6 activités les plus peuplées du processus *Helpdesk*

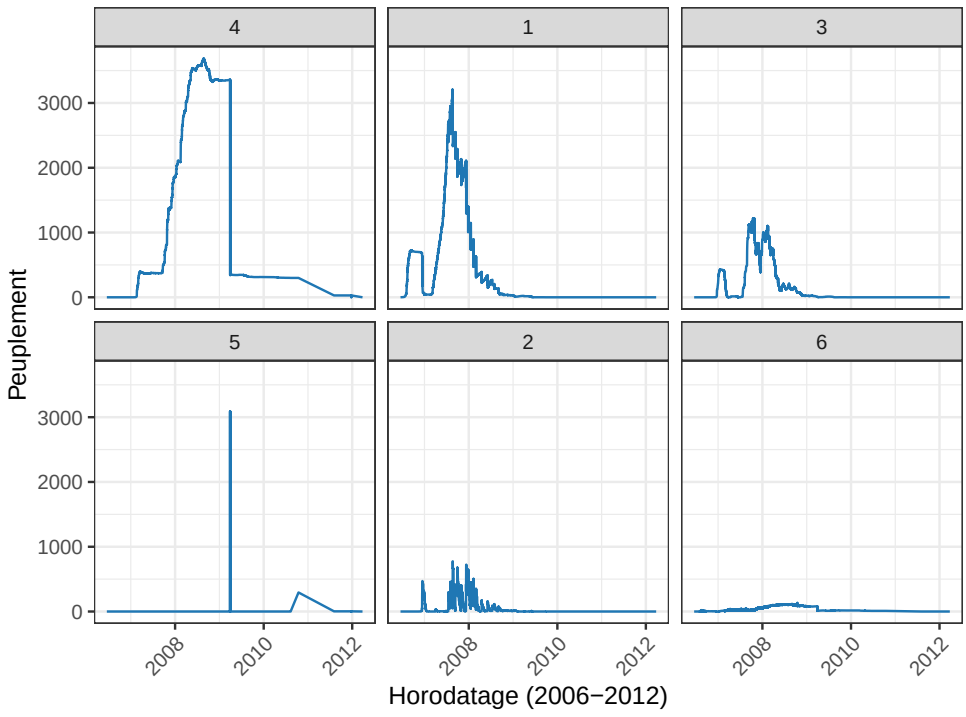

FIGURE 4.2 – Peuplements des 6 activités les plus peuplées du processus *Traffic fines*

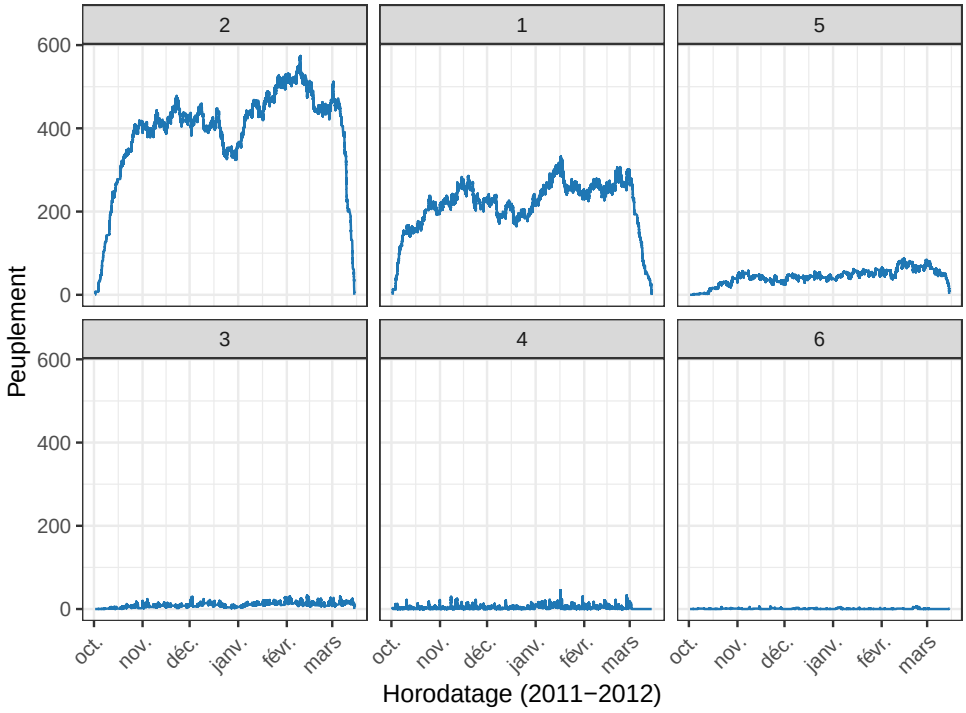

FIGURE 4.3 – Peuplements des activités du processus *BPI 2012 (W)*

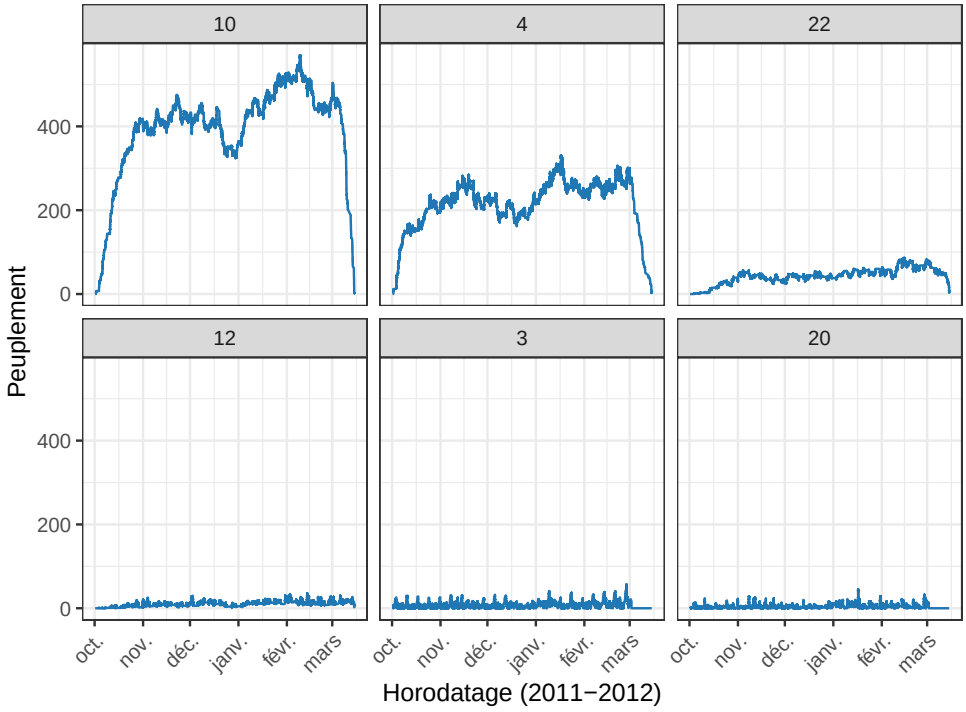

FIGURE 4.4 – Peuplements des 6 activités les plus peuplées du processus *BPI2012*

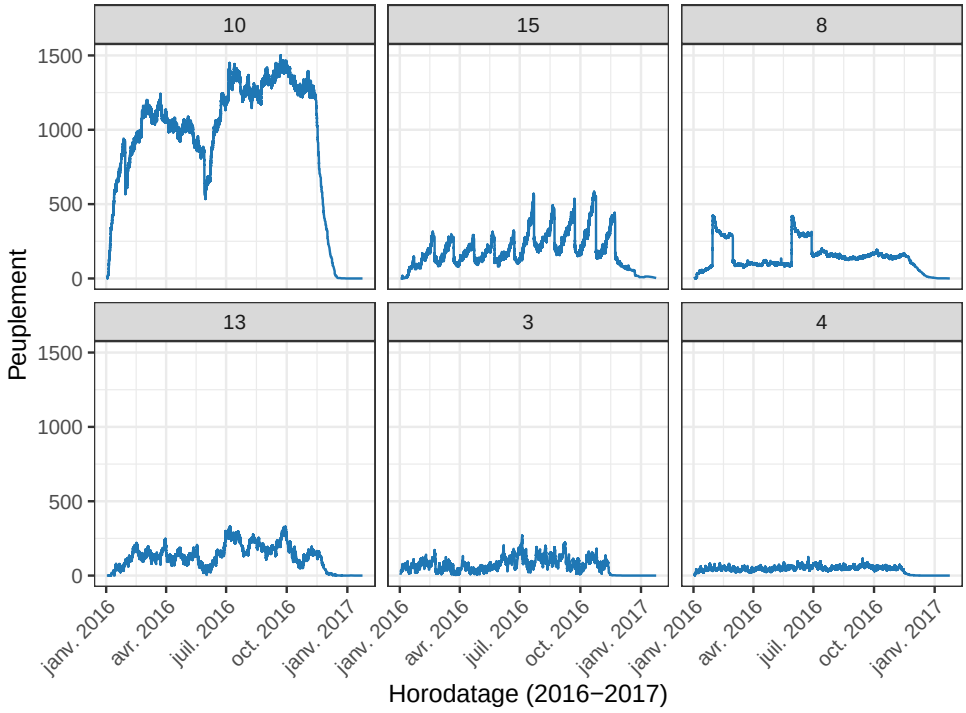

FIGURE 4.5 – Peuplements des 6 activités les plus peuplées du processus *BPI2017*

4.3.2 Corrélations croisées entre peuplements d'activités

Sachant que le peuplement de processus est composé de peuplements d'activités au cours du temps, et que ces mêmes activités agissent telles des vases communicants, il convient d'étudier les corrélations entre ces séries temporelles. Il faut cependant tenir compte du fait que justement, une activité menant à une autre donnera des peuplements aux variations similaires avec un décalage dans le temps. Il est donc judicieux d'étudier la corrélation croisée [VR02] entre peuplements.

Cette corrélation croisée correspond à un calcul de la corrélation entre deux séries temporelles, en tenant compte d'un « *lag* » : on calcule une corrélation entre les deux séries temporelles lorsque l'une des deux a été glissée en avant ou en arrière dans le temps. Soit X et Y deux séries temporelles. La corrélation croisée de X et Y avec un *lag* de t, notée $c_{XY}(t)$, est calculée :

$$c_{XY}(t) = \sum_{s=\max(1,-t)}^{\min(n-t,n)} \left(X(s+t) - \overline{X}\right)\left(Y(s) - \overline{Y}\right),$$

où $\overline{X}$ et $\overline{Y}$ sont les moyennes de X et Y. Ainsi, soit L un journal d'événements comportant $|L|_e$ événements, avec un ensemble d'activités $\mathcal{A}$ de cardinal $|\mathcal{A}|$. La matrice de

peuplements $C = (Cd(\pi_{\mathcal{T}}(e_j)), j = 1, \ldots, |L|_e)$, de dimensions $|L|_e \times |\mathcal{A}|$, est composée de $|A|$ vecteurs de peuplements de taille $|L|_e$ notés $C_{\bullet,a}$, $a \in \mathcal{A}$. Soit deux peuplements $C_{\bullet,a}$ et $C_{\bullet,b}$, $a, b \in \mathcal{A}$ et $a \neq b$. Le coefficient final, dans notre cas, vaut :

$$r_{C_{\bullet,a}C_{\bullet,b}}(t) = \frac{c_{C_{\bullet,a}C_{\bullet,b}}(t)}{\sigma_{C_{\bullet,a}}\sigma_{C_{\bullet,b}}} = \frac{\sum_{s=\max(1,-t)}^{\min(|L|_e-t,|L|_e)} \left(C_{s+t,a} - \overline{C_{\bullet,a}}\right)\left(C_{s,a} - \overline{C_{\bullet,b}}\right)}{\sqrt{\frac{1}{|L|_e}\sum_{s=1}^{|L|_e}\left(C_{s,a} - \overline{C_{\bullet,a}}\right)^2}\sqrt{\frac{1}{|L|_e}\sum_{s=1}^{|L|_e}\left(C_{s,b} - \overline{C_{\bullet,b}}\right)^2}}.$$

Comme la corrélation classique (de Pearson, Spearman, Tau de Kendall...), $r_{XY}(\cdot)$ donne un résultat entre -1 et 1. Dans notre cas, l'idée est de tester tous les *lags* compris dans $\left\{-\left\lceil\frac{|L|_e}{2}\right\rceil, -\left\lceil\frac{|L|_e}{2}\right\rceil + 1, \ldots, \left\lceil\frac{|L|_e}{2}\right\rceil\right\}$. On suppose par là que le décalage entre l'occurrence de deux activités dans un parcours ne durera pas, généralement, plus longtemps que la moitié de la période observée. Cela réduit le temps de calcul au prix d'une hypothèse généralement vérifiée. On prend ensuite la corrélation croisée maximale en valeur absolue. On cherche donc :

$$\max_{t\in\left\{-\left\lceil\frac{|L|_e}{2}\right\rceil,\ldots,\left\lceil\frac{|L|_e}{2}\right\rceil\right\}} \left(|r_{C_{\bullet,a}C_{\bullet,b}}(t)|\right).$$

On effectue ce calcul pour toutes les paires de peuplements dans la matrice C. *In fine*, on obtient une matrice de corrélations croisées maximales en valeur absolue (MVA) :

$$\left(\max_{t\in\{-|L|_e+1,\ldots,|L|_e-1\}} \left(|r_{C_{\bullet,a}C_{\bullet,b}}(t)|\right)\right)_{a,b\in\mathcal{A}}.$$

Les figures suivantes montrent les matrices résultant de ces calculs pour chaque journal d'événements décrit dans le chapitre 2. Ces matrices étant symétriques, seule la matrice triangulaire supérieure, sans la diagonale, est conservée.

La figure 4.6 montre les corrélations croisées maximales en valeur absolue pour les peuplements dans le journal d'événements *Helpdesk*. Aucune ne dépasse 0,4, ces corrélations ne semblent pas élevées.

Traffic fines exhibe des corrélations croisées avoisinant 1 en valeur absolue dans la figure 4.7 : les activités 4 et 6, 4 et 10, 3 et 7, 6 et 10 par exemple.

Dans *BPI2012 (W)* sur la figure 4.8, les corrélations croisées ne dépassent pas 0,8. La plus élevée concerne la paire de peuplements des activités 1 et 2.

BPI2012 contient plus de peuplements corrélés, comme le montre la figure 4.9. On a par exemple la paire 8 et 9, ou les paires d'activités formées par le quadruplet d'activités 13, 14, 15 et 16.

Malgré le grand nombre d'activités, *BPI2017* ne présente pas autant de peuplements fortement corrélés. On peut compter par exemple la paire de peuplements 10 et 23, ou encore la paire 18 et 19.

4.4 Conclusion

Le peuplement de processus constitue un ensemble de variables aisées à calculer, calculables dans n'importe quel journal d'événements. Il ne s'agit que de comptages, mais qui semblent contenir une information conséquente concernant les mécanismes des processus desquels ils sont tirés. Dans une optique de modélisation, il faut malgré tout faire attention : les corrélations entre les peuplements, possiblement avec un *lag*, sont présentes et sont à considérer.

L'utilité de la création de ces peuplements, et les conclusions qui en découlent, sera explorée dans les chapitres 5 et 6 : ceux-ci s'avèrent cruciaux pour la modélisation prédictive, la simulation, et l'analyse de sensibilité du modèle prédictif.

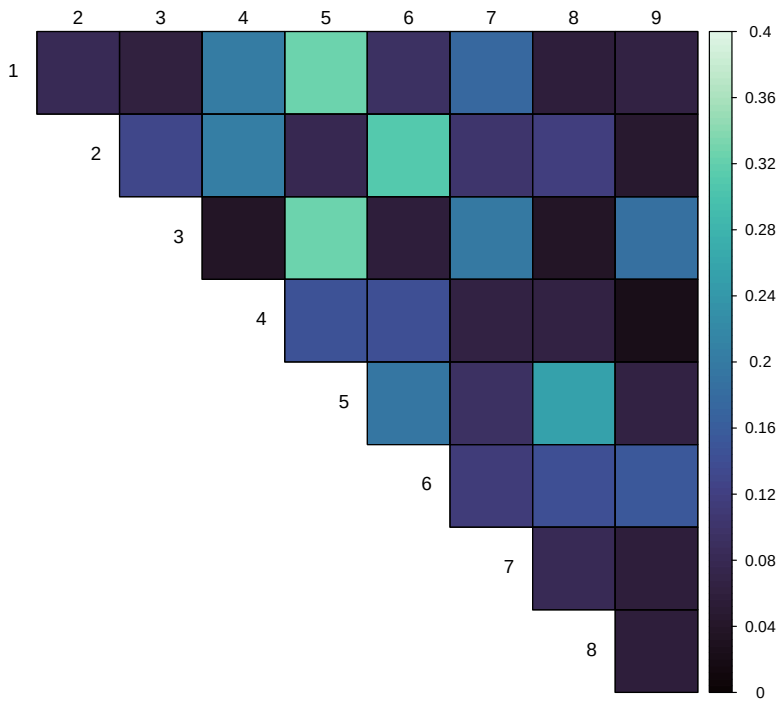

FIGURE 4.6 – Corrélations croisées MVA entre peuplements du processus *Helpdesk*

FIGURE 4.7 – Corrélations croisées MVA entre peuplements du processus *Traffic fines*

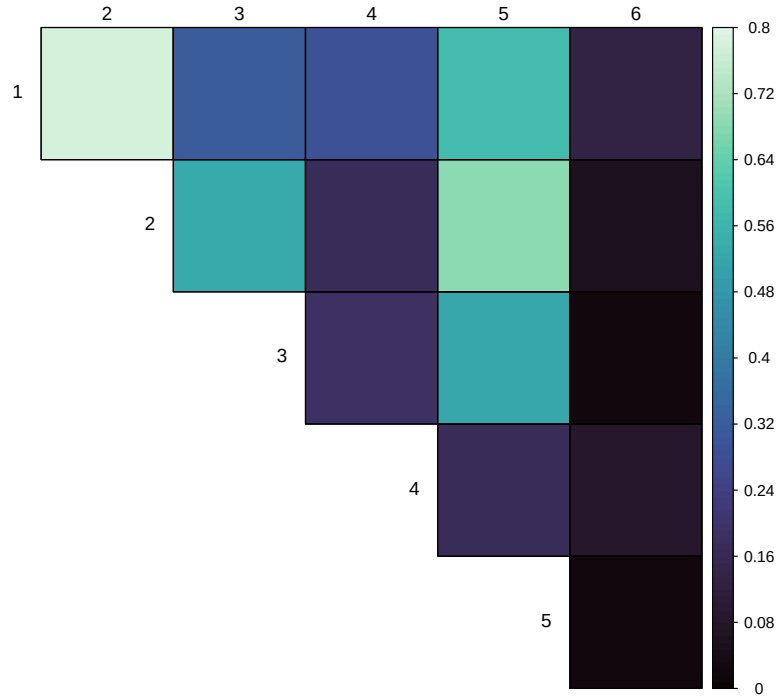

FIGURE 4.8 – Corrélations croisées MVA entre peuplements du processus *BPI2012 (W)*

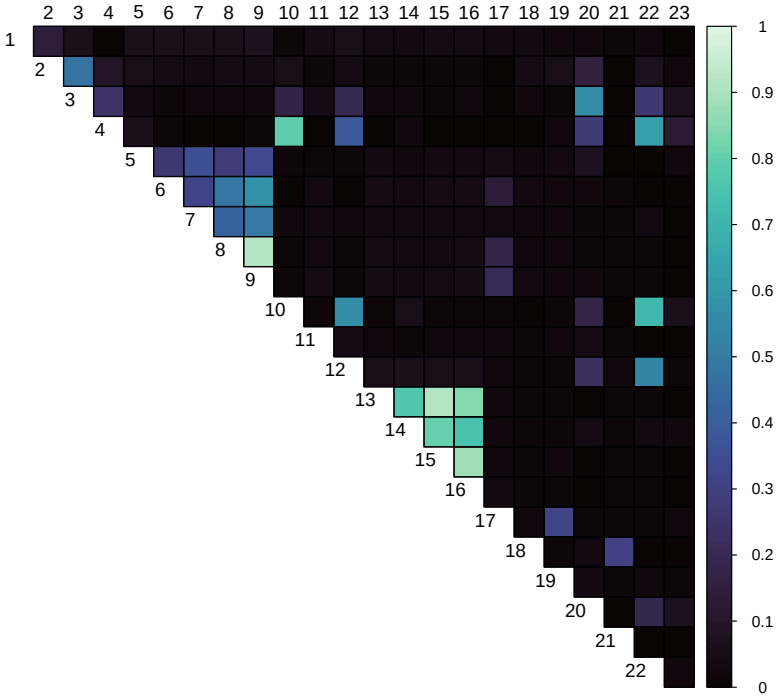

FIGURE 4.9 – Corrélations croisées MVA entre peuplements du processus *BPI2012*

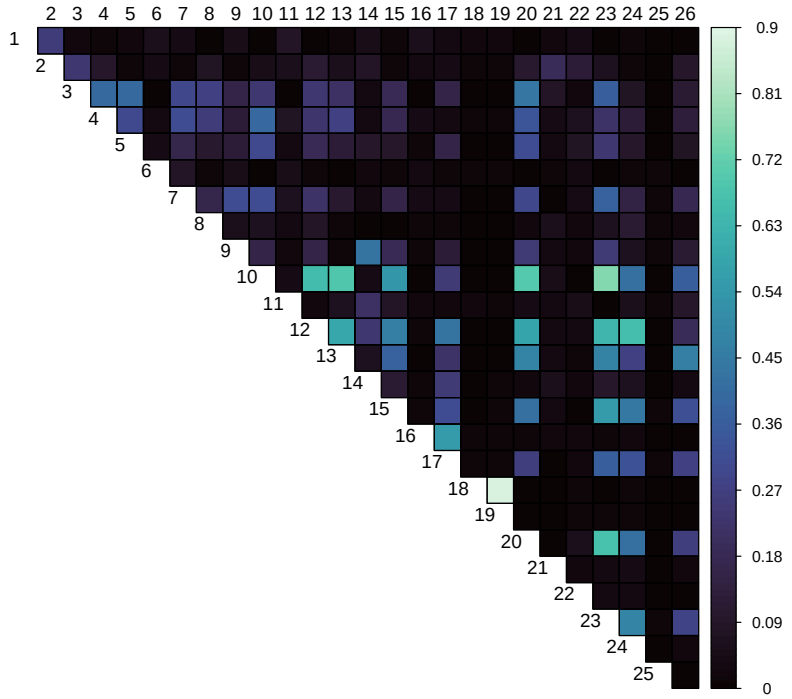

FIGURE 4.10 – Corrélations croisées MVA entre peuplements du processus *BPI2017*

Chapitre 5

Modélisation prédictive

5.1 Introduction

Possédant à présent une méthode de *feature engineering* effective sur n'importe quel journal d'événements, ce chapitre a pour objectif de décrire le modèle prédictif choisi ainsi que la démarche de traitement des données permettant d'aboutir aux résultats qui closent ce chapitre. Il s'agit ici de l'outil qui, couplé aux peuplements, permet les avancées décrites dans les chapitres suivants.

Après une description formelle de la tâche de prédiction sur les parcours incomplets d'un journal d'événements, le choix d'une architecture antagoniste générative est motivé. S'ensuit le détail du traitement des données qui inclut le traitement des horodatages dans le cas d'une ou deux colonnes dédiées, le traitement des activités, ainsi qu'un traitement simple des covariables pouvant être utilisées en plus du peuplement dans le cas où celles-ci seraient disponibles. Ensuite, l'utilisation du peuplement de processus est détaillée, en particulier dans le cas de peuplements manquants non observés puis leur sélection à partir des corrélations croisées. Les ajustements décrivant la forme globale des tenseurs utilisés lors de l'apprentissage sont énoncés, avant de décrire l'architecture du modèle utilisé dans le détail. Enfin, les résultats du modèle prédictif résultant sont exposés.

Commençons par formaliser la tâche de prédiction.

5.1.1 Tâche de prédiction

Soit un parcours σ de longueur n, et $k \in [\![1; n-1]\!]$. On a un k-préfixe $\sigma_{\leqslant k} = \langle e_1, e_2, \ldots, e_k \rangle$. Le réseau de neurones exposé plus loin dans ce chapitre apprend une fonction h telle que $h(\sigma_{\leqslant k}) = \widehat{\sigma_{>k}}$, où $\widehat{\sigma_{>k}}$ est le suffixe généré prédisant le vrai suffixe $\sigma_{>k}$. Puisqu'un suffixe est une sous-séquence liée à un parcours spécifique, il n'est pas nécessaire de prédire un identifiant. La prédiction se limitera donc à la séquence d'activités et aux temps correspondants.

Les définitions de base ainsi que la tâche de prédiction étant à présent établies, nous passons à présent à l'établissement d'un modèle prédictif qui doit être utilisable en production dans l'environnement Livejourney™. Il est à noter que son utilisation dans l'environnement logiciel a demandé des ajouts, entre autres d'un traitement des covariables et du cas de deux horodatages, dont les performances ont été évaluées sur des jeux de données confidentiels. De ce fait, le gain prédictif du traitement des covariables et les performances en cas de deux horodatages seront traités à part dans la section 5.5, les données utilisées de base dans cette thèse ne présentant ni covariables ni double horodatage.

5.1.2 Choix du modèle

La revue de l'état de l'art indiquait que les simples modèles récurrents, bien qu'efficaces pour la prédiction de l'événement suivant d'un parcours incomplet, présentaient de très basses performances prédictives tant sur les prédictions de durées que sur les prédictions d'activités lorsque la séquence entière d'événements restants devait être prédite. Ce manque de précision, dû à la propagation de l'erreur au fur et à mesure des prédictions successives, nous a poussés à pencher pour un réseau de neurones antagoniste génératif (GAN) [Goo+14].
En effet, un GAN est composé de deux réseaux de neurones qui s'affrontent lors de l'entraînement :

— un générateur, dont le but est de générer de la donnée synthétique

— un discriminant, dont le but est de distinguer entre la donnée produite par le générateur et la donnée réelle.

Ainsi, lors de la phase d'entraînement, le générateur a pour but de générer de la donnée capable d'induire le discriminant en erreur, c'est à dire que le discriminant classifie la donnée générée comme de la donnée réelle. D'un autre côté, le discriminant a pour but d'affiner sa capacité de différenciation entre la donnée générée et la donnée réelle.
Nous voyons que l'entraînement d'un tel réseau ne correspond donc pas à une simple minimisation d'une fonction de coût d'apprentissage, mais à un jeu où les participants, ici les modèles, doivent atteindre un équilibre de Nash [Nas50], de telle sorte que chaque modèle possède une stratégie optimale sans possibilités d'améliorations. Une fois l'entraînement terminé, le générateur seul est utilisé en production afin de générer des prédictions.
L'intérêt de ce type de modèle réside dans le fait que le générateur est entraîné à générer de la donnée, y compris séquentielle, sans la contrainte de prédictions successives : il est entraîné à générer de telles séquences de sorte que celles-ci soient aussi proches de la réalité que possible dans leur entièreté.

Cette méthode comporte cependant plusieurs désavantages. En effet, l'entraînement d'un GAN est instable et ne converge pas aisément : de nombreux essais doivent être menés afin d'atteindre un équilibre de Nash. Par ailleurs, un phénomène appelé le *mode collapse* est très présent lors de l'entraînement de tels modèles : un type particulier de

génération parvient à berner le discriminant mieux que les autres, entraînant le générateur à ne donner plus que ce type de génération, qu'importe la donnée en entrée. Ce phénomène est d'autant plus omniprésent dans notre cas, dans la mesure où les traces ont une représentation fortement déséquilibrée ; le risque que le générateur génère exclusivement le parcours le plus commun est ainsi décuplé. Ceci motiva un choix d'architecture du GAN décrite plus loin dans ce chapitre.

5.2 Pré-traitement des données

Nous supposons que nous avons à notre disposition un journal d'événements L composé de parcours complets. Ainsi, chaque fois qu'un parcours σ est mentionné dans cette section, sa longueur n correspond à la quantité totale d'événements du début à la fin de σ.

5.2.1 Traitement des horodatages

Un horodatage

Afin de prédire la séquence des horodatages dans un suffixe, nous transformons les horodatages en durées en calculant la durée entre l'horodatage de chaque événement et l'horodatage de l'événement suivant, dans chaque parcours. Formellement, soit $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ un parcours dans un journal d'événements L. Nous définissons la fonction :

$$\theta(\sigma, i) = \begin{cases} 0 & \text{si } i = 1, \\ \pi_{\mathcal{T}}(e_i) - \pi_{\mathcal{T}}(e_{i-1}) & \text{sinon.} \end{cases}$$

En appliquant cette fonction à σ pour $i = 1, \ldots, n$, nous obtenons la durée de chaque activité dans le parcours en question. Pour récupérer les dates au lieu des durées, il suffit de calculer leur somme cumulée.

Deux horodatages

Dans le cas de deux horodatages, appliquer la méthode précédente sur chaque colonne n'est pas une bonne idée : rien n'empêcherait le modèle de prédire qu'une fin d'activité surviendrait avant son début, or ceci ne doit jamais arriver. Pour contrecarrer cela, nous avons développé une méthode similaire utilisant conjointement les deux horodatages pour arriver au calcul de trois durées différentes pour une même activité. Définissons d'abord la fonction $\pi'_{\mathcal{T}}(\cdot)$. Pour un parcours $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ où $e_i = (c_i, a_i, t_i, t'_i)$, on a $\pi_{\mathcal{T}}(e_i) = t_i$ et $\pi'_{\mathcal{T}}(e_i) = t'_i$. On définit ensuite trois fonctions analogues à θ :

$$\theta_{\text{previous}}(\sigma, i) = \begin{cases} 0 & \text{si } i = 1, \\ \pi'_{\mathcal{T}}(e_{i-1}) - \pi_{\mathcal{T}}(e_{i-1}) & \text{sinon.} \end{cases}$$

$$\theta_{\text{next}}(\sigma, i) = \begin{cases} 0 & \text{si } i = 1, \\ \pi_{\mathcal{T}}(e_i) - \pi'_{\mathcal{T}}(e_{i-1}) & \text{sinon.} \end{cases}$$

$$\theta_{\text{end}}(\sigma, i) = \pi'_{\mathcal{T}}(e_i) - \pi_{\mathcal{T}}(e_i).$$

TABLEAU 5.1 – Illustration des calculs effectués par les fonctions θ_{previous}, θ_{next} et θ_{end}

Identifiant	**Activité**	**Date de début**	**Date de fin**
c_i	a_{i-1}	t_{i-1}	t'_{i-1}
c_i	a_i	t_i	t'_i

Le tableau 5.1 illustre les trois fonctions précédentes : la flèche correspondant à la durée de $t_{i-1} \longrightarrow t'_{i-1}$ renvoie à $\theta_{\text{previous}}(\sigma, i)$, la flèche de $t'_{i-1} \longrightarrow t_i$ correspond à $\theta_{\text{next}}(\sigma, i)$, et la flèche de $t_i \longrightarrow t'_i$ correspond à $\theta_{\text{end}}(\sigma, i)$. L'existence de $\theta_{\text{previous}}(\sigma, i)$ est justifiée par le cas où un parcours à prédire aurait pour dernier événement e_{i-1}, avec uniquement t_{i-1} observé et non t'_{i-1}, l'événement étant donc en cours. Dans ce cas, il faut d'abord pouvoir prédire t'_{i-1} afin de compléter e_{i-1}, pour ensuite prédire l'événement e_i.

5.2.2 Traitement des activités

Le prétraitement des activités est le même que celui exposé dans [TR20], que nous décrivons à nouveau dans cette section. La première étape consiste à ajouter une activité *End Of State* pour marquer la fin d'un parcours, que nous notons $\langle EoS \rangle$. Par conséquent, un parcours σ de longueur initiale n a maintenant une longueur $n+1$ avec $\pi_{\mathcal{A}}(e_{n+1}) = \langle EoS \rangle$. De plus, ce *End of State* étant artificiel, aucun délais n'est à considérer entre la dernière activité (ou la fin de la dernière activité) et l'occurrence du *End of State*. Ainsi, l'horodatage de cette activité artificielle est le même que celui de l'activité de fin : $\pi_{\mathcal{T}}(e_{n+1}) = \pi_{\mathcal{T}}(e_n)$. Dans le cas de deux horodatages, l'horodatage de début et de fin du *End of State* sont les mêmes, et correspondent à l'horodatage de fin de l'activité réelle de fin de parcours : $\pi_{\mathcal{T}}(e_{n+1}) = \pi'_{\mathcal{T}}(e_{n+1}) = \pi'_{\mathcal{T}}(e_n)$.

Nous codons ensuite les activités en vecteurs *one-hot*. Ceci code chaque activité en un vecteur binaire composé de zéros, avec seulement un 1 à une coordonnée dénotant l'activité : dans un journal d'événements L tel que $|\mathcal{A}| = 5$, l'activité 1 serait représentée par le vecteur $(1, 0, 0, 0, 0)$, l'activité 2 par $(0, 1, 0, 0, 0)$, et ainsi de suite.

5.2.3 Traitement simple des covariables

Les covariables potentiellement présentes dans un journal d'événements peuvent être de tous types : qualitatives, quantitatives, ordinales...

Afin de les traiter d'emblée de façon simple et efficace, une simple analyse factorielle des données mixtes (*AFDM*) [Pag04], qui permet de projeter des variables mixtes sur des axes maximisant la variance des projections. Cela permet de convertir toutes les variables, y compris qualitatives, en variables quantitatives, et d'avoir toutes les observations de ces variables projetées dans le même espace.
chaque axe ne capte qu'une partie de la variance totale de la donnée projetée. Ceux-ci sont classés dans l'ordre décroissant de la proportion de la variance totale que ceux-ci représentent. Le nombre d'axes retenus est laissé au modélisateur.

Les journaux d'événements présents dans cette thèse ne contiennent pas de covariables. Cette sous-section est destinée à l'application directe dans le logiciel Livejourney™, dans le cas où des journaux d'événements contiendraient des covariables exploitables.

5.2.4 Utilisation des peuplements de processus

Troncature

Une chose que nous remarquons est le fait que tous les peuplements commencent et finissent à 0. Cela est en fait dû au fait que les journaux d'événements utilisés ont été élagués des parcours tronqués par les dates de récupération de la donnée. Il y a en effet un certain nombre de parcours non observés ayant commencé avant les dates respectives de départ des journaux d'événements, d'autres se terminant après leur date de fin. Dans ce cas, sans information d'experts, il devient difficile de déterminer quels parcours sont effectivement incomplets, ceux-ci ont donc été retirées par les organismes émetteurs de ces journaux d'événements. Il y a donc une problématique de valeurs manquantes non-observées comme étant manquantes.

Une solution est de retirer du journal d'événements tous les parcours en début et fin de période présentant un peuplement ne correspondant pas à une tendance donnée dans les peuplements de la zone centrale du journal d'événements en termes de temporalité. Ainsi, l'apprentissage ne s'effectuerait que sur de la donnée où les peuplements représentent la réalité du processus.

Plusieurs approches de complexité variable ont été tentées sans succès, étant donné la multiplicité des formes que peuvent prendre les courbes de peuplements. Il fut également tenté de retirer les parcours commençant avant que la durée du parcours le plus long observé se soit écoulée depuis le début de la période observée, et ceux commençant au minimum autant de temps avant la fin de la période observée. Par exemple, dans *Helpdesk*, le parcours le plus long dure 59,850 jours. Ainsi, nous retenons la période commençant 59,850 jours après le début de période observée, et finissant 89,850 jours avant la fin. Le problème est que cette méthode résulte presque systématiquement en un intervalle de temps de longueur nulle. Recourir à des quantiles des durées au lieu du maximum demande une étude au cas par cas, les distributions des durées de parcours étant généralement très asymétriques et à queue de distribution lourde.

Finalement, une approche des plus simples a été choisie.
D'abord, pour un peuplement donné $Cd(t)$, $t \in \mathcal{T}$ dans un journal d'événements L, on peut calculer un peuplement total, qui n'est autre que la somme des composantes du vecteur de peuplements au temps t :

$$\sum_{a=1}^{|\mathcal{A}|} Cd(t)_a$$

Ce peuplement total est calculé sur tous les horodatages dans L, classés par ordre chronologique croissant, pour obtenir un peuplement total au cours du temps :

$$Cd_{\text{total}} = \begin{pmatrix} \sum_{a=1}^{|\mathcal{A}|} Cd(\pi_{\mathcal{T}}(e_1))_a \\ \sum_{a=1}^{|\mathcal{A}|} Cd(\pi_{\mathcal{T}}(e_2))_a \\ \vdots \\ \sum_{a=1}^{|\mathcal{A}|} Cd(\pi_{\mathcal{T}}(e_{|L|_e}))_a \end{pmatrix}, \quad \pi_{\mathcal{T}}(e_i) \leqslant \pi_{\mathcal{T}}(e_i + 1) \forall i \in \{1, \ldots, |L|_e - 1\}.$$

On note $\overline{Cd_{\text{total}}}$ la moyenne des valeurs de Cd_{total}. Il suffit ensuite de trouver deux indices d (pour « début ») et f (pour « fin ») dans $\{1, \ldots, |L|_e\}$ tels que :

$$\begin{cases} d = \max\left\{k \in \{1, \ldots, |L|_e\};\ \forall i \leqslant k,\ Cd_{\text{total}_i} \leqslant \overline{Cd_{\text{total}}}\right\} \\ f = \min\{k \in \{1, \ldots, |L|_e\};\ \forall i \geqslant k,\ Cd_{\text{total}_i} \leqslant \overline{Cd_{\text{total}}}\}. \end{cases}$$

De cette manière, on obtient l'indice du premier événement tel que le peuplement total atteint la moyenne en commençant de 0, et l'indice du dernier événement avec un peuplement total au-dessus ou égal à la moyenne avant que celui-ci ne redescende définitivement à 0. Tout événement $e \in \mathcal{E}$ tel que $\pi_{\mathcal{T}}(e_d) \leqslant \pi_{\mathcal{T}}(e) \leqslant \pi_{\mathcal{T}}(e_f)$ est alors conservé.

Une dernière phase de sélection est cependant à appliquer : certains parcours se retrouvent tronqués à leur tour par cette sélection. Seulement, sachant cette-fois quels parcours sont concernés, tous les événements restants contenus dans ces parcours sont eux-aussi retirés du journal d'événements final. Formellement, nous ne retenons que l'ensemble de parcours $\{\sigma \in L; \forall e \in \sigma, \pi_{\mathcal{T}}(e_d) \leqslant \pi_{\mathcal{T}}(e) \leqslant \pi_{\mathcal{T}}(e_f)\}$ inclus dans L.

Les figures 5.1, 5.2, 5.3, 5.4 et 5.5 ci-dessous illustrent les zones de rejet des événements calculées à partir de la méthode énoncée ci-avant. Les zones colorées en rouge correspondent aux événements dont l'horodatage est inférieur à $\pi_{\mathcal{T}}(e_d)$ ou supérieur à $\pi_{\mathcal{T}}(e_f)$, et la droite verte horizontale correspond à la moyenne des peuplements totaux.

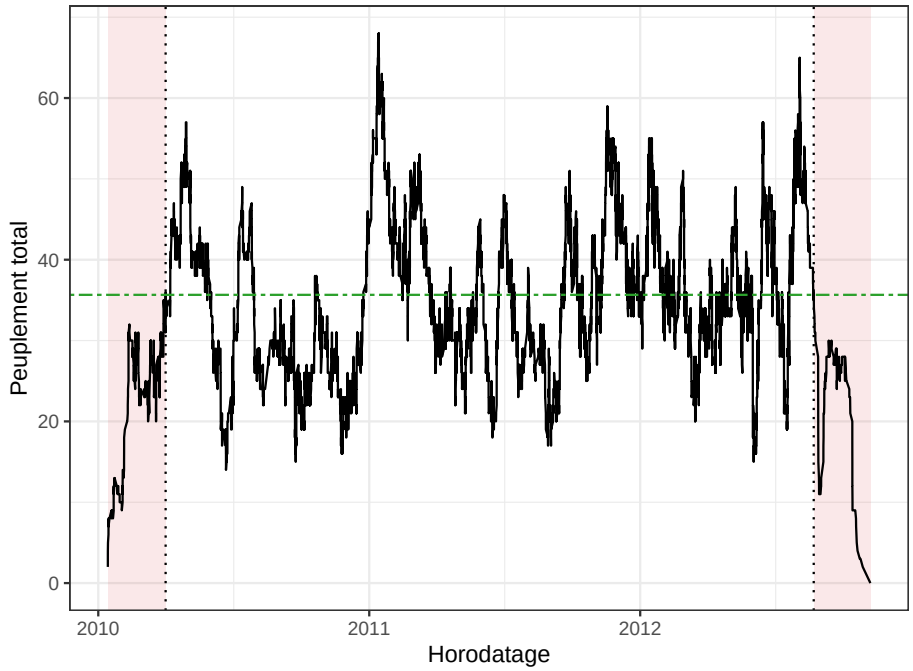

FIGURE 5.1 – Horodatages exclus de *Helpdesk* pour l'apprentissage du GAN

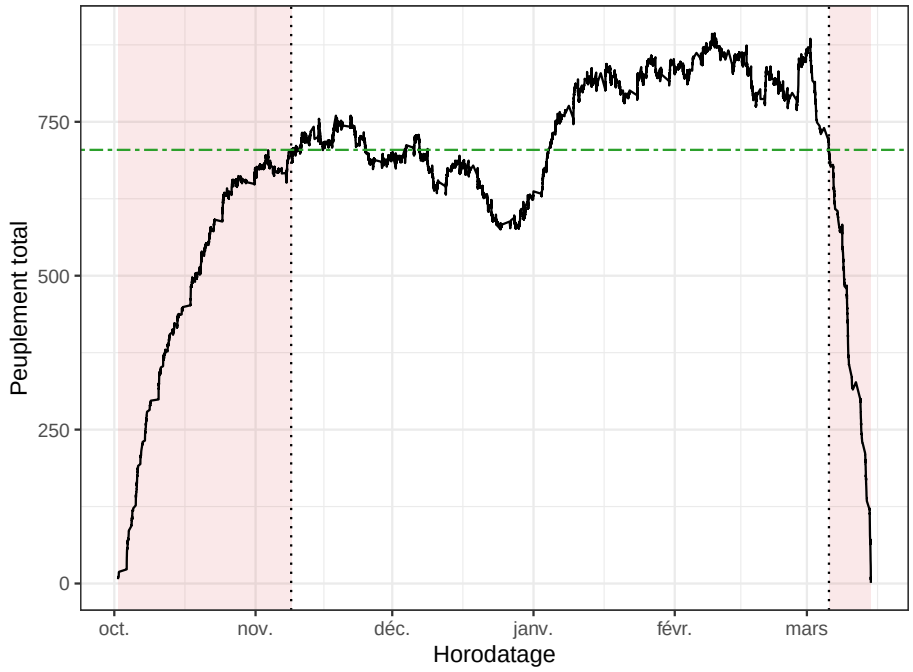

FIGURE 5.2 – Horodatages exclus de *BPI2012 (W)* pour l'apprentissage du GAN

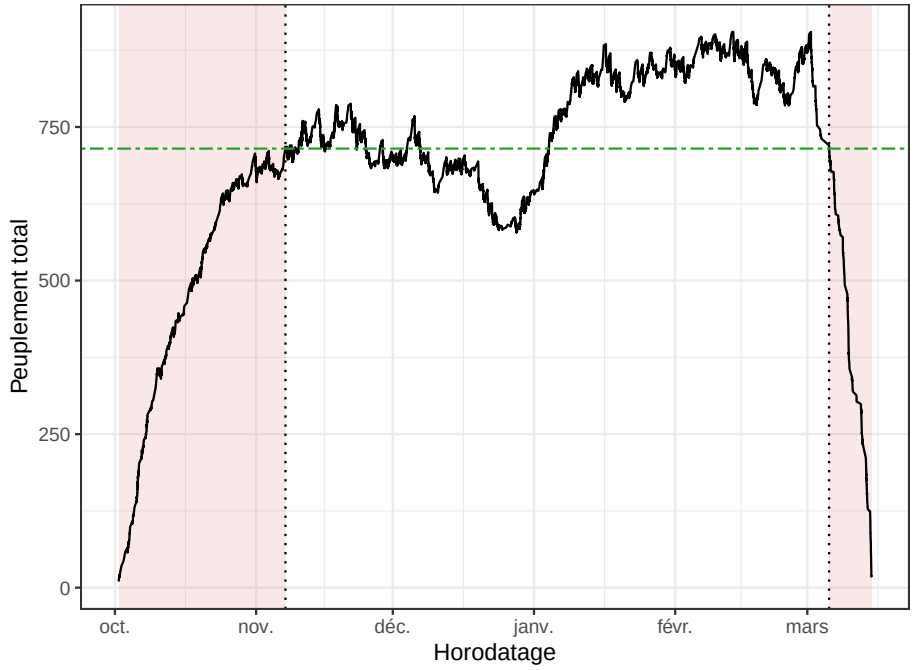

FIGURE 5.3 – Horodatages exclus de *BPI2012* pour l'apprentissage du GAN

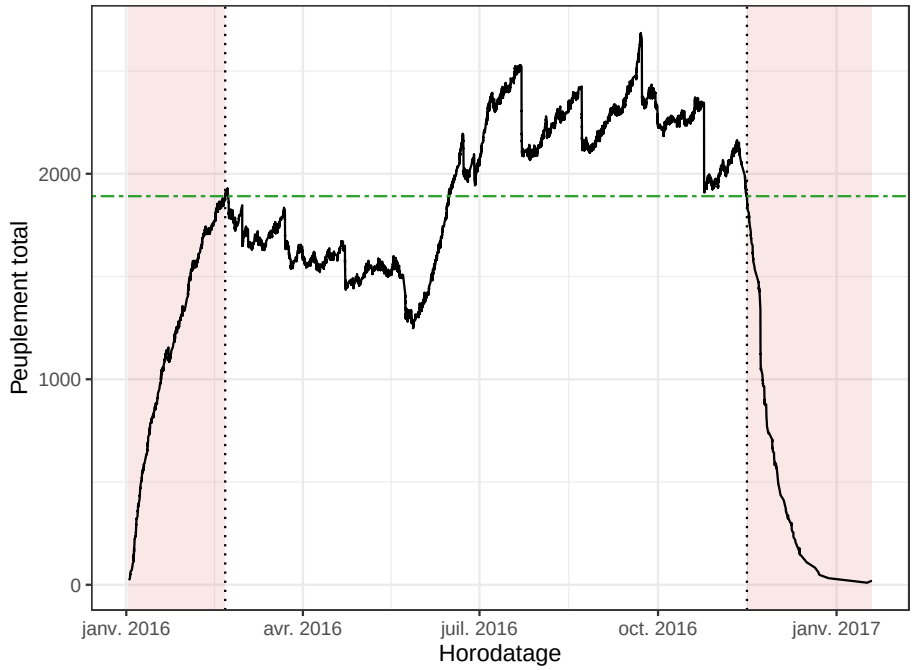

FIGURE 5.4 – Horodatages exclus de *BPI2017* pour l'apprentissage du GAN

La figure 5.1 montre le peuplement total de *Helpdesk* au cours du temps. Nous avons un peuplement assez erratique, pour lesquels il semble relativement difficile de déterminer à quels moments le régime réel du processus est disponible. Le peuplement total moyen est à 35,78. En retirant tous les événements relatifs aux parcours correspondant à cet élagage, 220 parcours sont ainsi retirés contre 3 538 parcours retenus, pour une rétention de 94,15%.

Pour *BPI2012 (W)* en figure 5.2, nous observons un peuplement total en augmentation assez flagrante en début de période, pour une descente à 0 tout aussi flagrante en fin de période. Le peuplement total exhibe d'ailleurs un changement de régime soudain entre fin décembre et début janvier. Le régime réel de peuplement de ce jeu de données semble plus aisé à déceler étant donné la clarté de ses variations en début et fin de période. La moyenne se trouve à 705,56, malheureusement tirée vers une valeur plus élevée par ce changement de régime de fin décembre. Cela cause la majorité des événements élagués à se trouver en début de période, là où le régime réel est plus bas. L'élagage total des parcours en exclut 2 975, contre 6 689 retenus, pour une rétention de 69,19%.

BPI2017, en figure 5.4 exhibe un comportement similaire à *BPI2012 (W)* et *BPI2012* (figure 5.3) en termes de peuplement total, en particulier concernant la croissance soudaine en milieu de période, tirant la moyenne vers des valeurs élevées. Cette fois-ci cependant, cette croissance s'effectue en juin, au lieu de fin décembre. La période sélectionnée exclut presque le premier pic de peuplement total entre janvier et avril, mais semble correctement exclure la décroissance raide avant janvier 2 017. Il semble donc encore une fois que le régime réel de peuplement total soit conservé de façon plus exhaustive en fin de période, sans pour autant en avoir la certitude par la nature même de ces valeurs manquantes.

Si nous regardons l'application de cette méthode sur le jeu de données *Traffic fines*, en figure 5.5, nous observons que la moyenne est bien trop élevée pour inclure suffisamment de parcours. Dans ce cas, seuls 3 954 seraient retenus, contre 6 046 supprimés, pour un taux de rétention de seulement 39,54%. De plus, il semble peu vraisemblable que toutes les amendes routières ne soient pas présentes entre fin 2 009 et 2 012. D'ailleurs, la forme du peuplement total rend la détection même à l'œil nu de la zone de régime réel du peuplement total compliquée. Il a donc été choisi, pour ce cas particulier, de garder l'intégralité du jeu de données pour l'apprentissage, malgré l'erreur (inconnue) induite par les faux peuplements en début et fin de période.

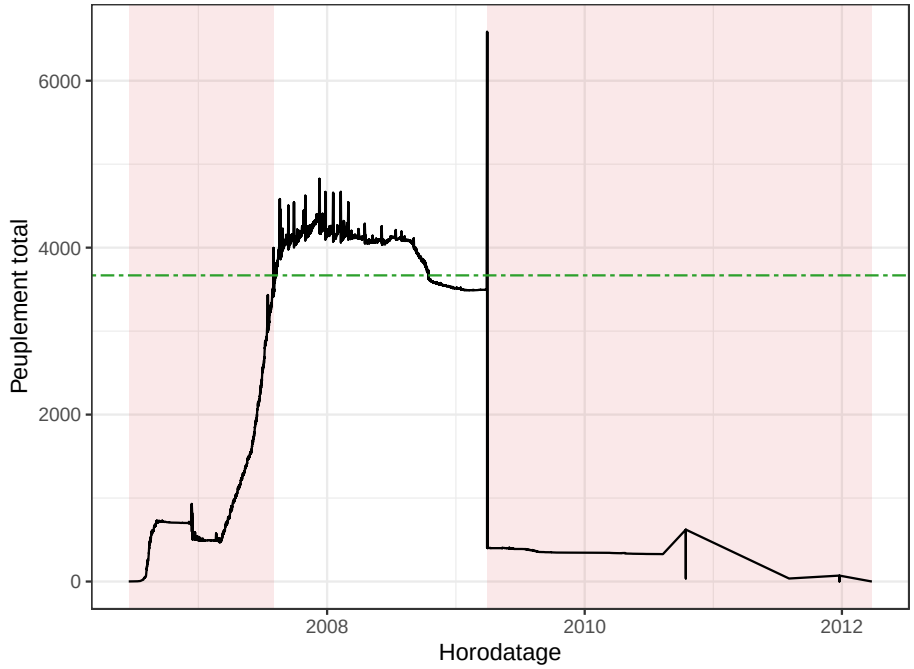

FIGURE 5.5 – Horodatages susceptibles d'être exclus de *Traffic fines* par la méthode de sélection du régime réel de peuplements

Sélection selon la corrélation croisée

Afin de réduire la complexité du modèle prédictif, il convient de ne pas conserver de variables fortement corrélées. Ce procédé doit être rapide et simple, et laisser à l'utilisateur la liberté de retirer les variables corrélées selon son choix. Ici, les seules covariables disponibles sont les peuplements calculés en section 4.3.1. Ainsi, grâce aux matrices 4.6, 4.7, 4.8, 4.9 et 4.10 qui affichent leurs corrélations croisées en valeur absolue, il devint possible d'isoler les peuplements corrélés au-delà d'un seuil choisi par l'utilisateur. Dans cette thèse, le seuil de $0,7$ est choisi par convention. Il est globalement accepté qu'une corrélation élevée en valeur absolue sera supérieure à 0,7, modérée de 0,3 à 0,7, et faible sous 0,3. Ce seuil n'a cependant pas vocation à être immuable et ne sert ici que de valeur illustrative.
Nous voyons par ailleurs sur la figure 5.6 les corrélations croisées entre paires de peuplements dans les différents journaux d'événements, en valeur absolue, et classées dans l'ordre croissant pour chaque journal d'événements. La très vaste majorité des corrélations croisées restent sous 0,5, tendance d'autant plus visibles pour BPI2012 et BPI2017 qui malgré les paires possibles de peuplements ne voient que peu de points au-dessus de 0,7, barre verte en pointillés sur la figure.

Le tableau 5.2 permet de voir les paires de peuplements dont la corrélation croisée dépasse 0,7. Helpdesk n'en possède pas, tous les peuplements sont donc conservés. Ce n'est pas le cas des autres journaux d'événements. Nous comptons 5 paires corrélées pour *Traffic fines*, 1 pour *BPI2012 (W)*, 9 pour *BPI2012* et 2 pour *BPI2017*.

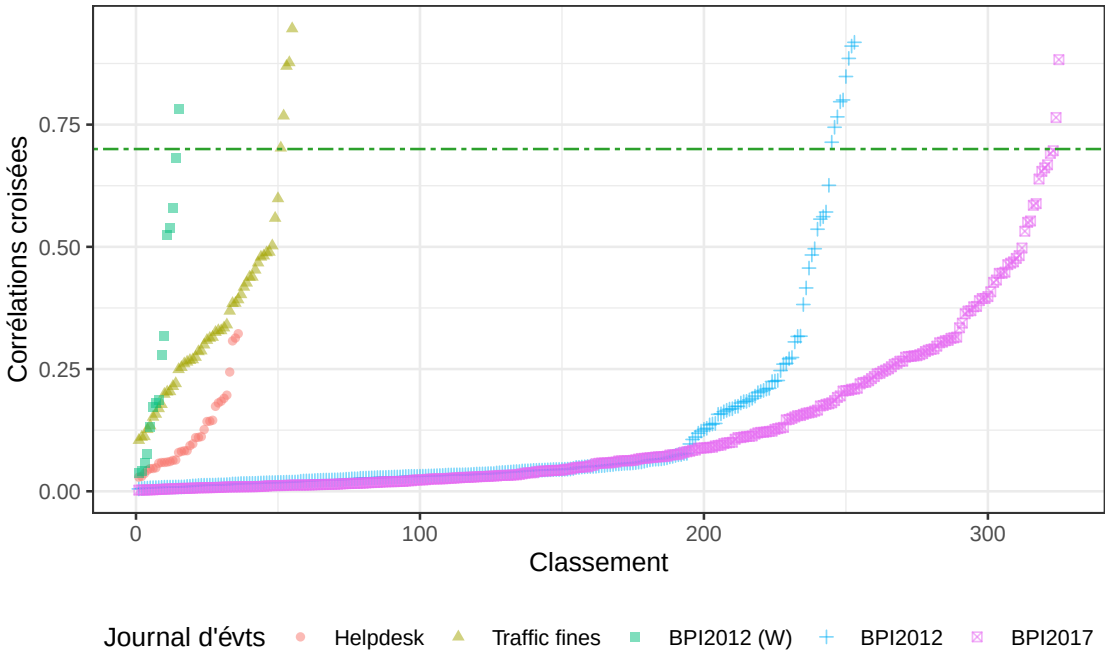

FIGURE 5.6 – Corrélations croisées en valeur absolue des paires de peuplements dans les journaux d'événements utilisés, classées dans l'ordre croissant

TABLEAU 5.2 – Corrélations entre peuplements dans les journaux d'événements supérieures en valeur absolue à 0,7

Journal d'événements	**Activité 1**	**Activité 2**	**Corrélation**
Helpdesk	—	—	—
Traffic fines	4	6	0,946
	3	7	0,87
	4	10	0,88
	5	10	0,702
	6	10	0,768
BPI2012 (W)	1	2	0,783
BPI2012	8	9	0,918
	4	10	0,797
	13	14	0,766
	13	15	0,911
	14	15	0,8
	13	16	0,848
	14	16	0,745
	15	16	0,885
	10	22	0,714
BPI2017	18	19	0,883
	10	23	0,764

Il faut ensuite décider, dans une paire corrélée de la sorte, quel peuplement garder. Il a été choisi de garder le peuplement de l'activité intervenant généralement en premier dans un parcours. L'idée tient au fait qu'il y a une temporalité dans les séquences d'activités, le fait qu'elles soient fortement corrélées indique un effet de vases communicants : une activité se remplit au fur et à mesure qu'une autre se vide, avec un certain décalage dans le temps. Ainsi, connaître le peuplement de l'activité en amont dans le processus donne peu ou prou le peuplement de l'activité en aval. Sans supposer présomptueusement de causalité, il convient donc malgré tout de conserver l'activité en amont. De ce fait, les peuplements présents dans la colonne « Activité 2 » du tableau 5.2 sont supprimées. Cela revient à supprimer les peuplements explicités dans le tableau 5.3.

Tableau 5.3 – Peuplements retirés grâce aux corrélations croisées

Journal d'événements	**Peuplements retirés**
Helpdesk	—
Traffic fines	$\{6, 7, 10\}$
BPI2012 (W)	$\{2\}$
BPI2012	$\{9, 10, 14, 15, 16, 22\}$
BPI2017	$\{19, 23\}$

Normalisation

Les peuplements n'étant en théorie pas bornés, appartenant à $\mathbb{N}$, il convient pour éviter l'explosion des gradients dans la phase d'apprentissage du modèle de les normaliser. La normalisation choisie est la normalisation min-max.
Il faut cependant faire une hypothèse avant d'effectuer cette normalisation, étant donné que deux cas s'offrent à nous :

1. normaliser chaque peuplement indépendamment,
2. normaliser selon le minimum et le maximum parmi tous les peuplements.

L'hypothèse que nous faisons dans ce cas est la suivante : les peuplements d'une activité, ainsi que sa capacité à accueillir des unités, n'a pas d'influence sur les peuplements et capacités d'une autre activité.

Par exemple, dans un processus de livraison, un centre de tri n'a aucune obligation de posséder la même capacité de peuplement qu'une camionnette de livraison, bien au contraire. Cette hypothèse n'est évidemment pas toujours vérifiée, mais elle semble moins forte que l'hypothèse contraire.

Ainsi, dans un journal d'événements L composé d'événements $\{e_1, \ldots, e_{|L|_e}\}$, pour un événement e_i horodaté par $\pi_{\mathcal{T}}(e_i)$, le peuplement d'une activité $a \in \mathcal{A}$ aura la forme :

$$Cd(\pi_{\mathcal{T}}(e_i))_a = \frac{Cd(\pi_{\mathcal{T}}(e_i))_a - \min\limits_{t \in \{\pi_{\mathcal{T}}(e_1), \ldots, \pi_{\mathcal{T}}(e_{|L|_e})\}} (Cd(t)_a)}{\max\limits_{t \in \{\pi_{\mathcal{T}}(e_1), \ldots, \pi_{\mathcal{T}}(e_{|L|_e})\}} (Cd(t)_a) - \min\limits_{t \in \{\pi_{\mathcal{T}}(e_1), \ldots, \pi_{\mathcal{T}}(e_{|L|_e})\}} (Cd(t)_a)}.$$

Notons que les extrema des peuplements sont pris dans l'ensemble d'entraînement lors de la phase d'apprentissage, et non la donnée entière.

5.2.5 Ajustements finaux

Afin de prédire la composante temporelle, les vecteurs codés en *one-hot* sont *augmentés* en y ajoutant les durées calculées dans la section 5.2.1 : pour un parcours σ, si nous avons initialement $\pi_{\mathcal{A}}(e_i) = (0, \ldots, 0, 1, 0, \ldots, 0)$, on y concatène la composante temporelle pour obtenir $(0, \ldots, 0, 1, 0, \ldots, 0, \theta(\sigma, i))$. Dans le cas de deux horodatages, on obtient $(0, \ldots, 0, 1, 0, \ldots, 0, \theta_{\text{previous}}(\sigma, i), \theta_{\text{next}}(\sigma, i), \theta_{\text{end}}(\sigma, i))$. Un journal d'événements complet est donc transformé en une matrice de design dont chaque ligne contient un identifiant, un vecteur *one-hot* et une durée (ou trois) d'activité. À cela, nous ajoutons les peuplements qui sont simplement concaténés en colonne à la matrice de design.

L'étape finale consiste à diviser les données en paires de préfixes et suffixes. Pour ce faire, chaque parcours σ, de longueur $n = |\sigma|$, est dupliqué en paires $(\sigma_{\leqslant k}, \sigma_{>k})$ $\forall k \in 1, \ldots, n$. De cette façon, le réseau de neurones est capable d'apprendre à prédire un suffixe à partir d'un préfixe de n'importe quelle longueur non nulle.

5.3 Architecture neuronale

5.3.1 GAN de Wasserstein conditionnel

La figure 5.7, inspirée de l'article [TR20], décrit le *pipeline* de prédiction utilisé dans ce chapitre : pour prédire un suffixe, après le prétraitement des données décrit dans la section 5.2, l'architecture GAN encodeur-décodeur choisie dans [TR20] utilise un générateur de séquence à séquence G composé d'un LSTM dans son encodeur, puis d'un LSTM et d'une couche linéaire dans son décodeur. En recevant un k-préfixe, le générateur le projette sur un espace latent à travers l'encodeur. La représentation du préfixe dans l'espace latent est ensuite décodée par le décodeur sous forme de suffixe, de façon analogue à une tâche de traduction. Ainsi, les suffixes prédits proviennent de la sortie du générateur, dans notre cas conditionnellement à la matrice de peuplements calculée, telle que :

$$\widehat{\sigma_{>k}} = G\left(\sigma_{\leqslant k} | C_{\sigma_{\leqslant k}}\right).$$

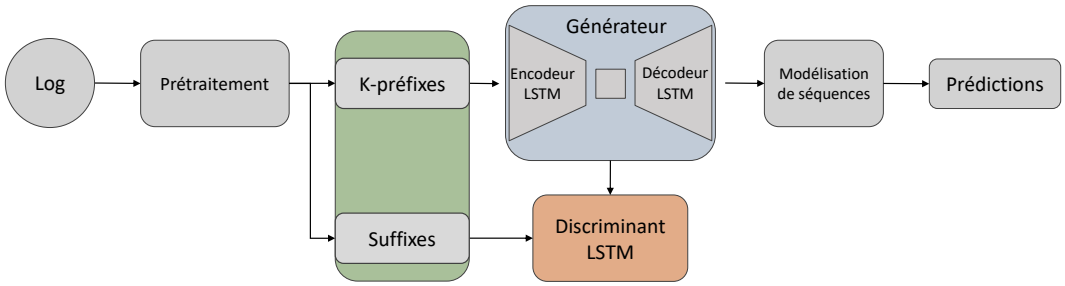

FIGURE 5.7 – *Pipeline* de prédiction présenté dans ce chapitre

Le discriminant D utilise ensuite un LSTM suivi d'une couche linéaire. Dans un contexte typique de GAN, le discriminant recevrait un vrai suffixe et sa prédiction, son rôle étant de classer les deux comme dans une tâche de classification binaire.

Les limites inhérentes au GAN classique exposées en section 5.1.2 ont mené à l'utilisation d'une fonction de coût de Wasserstein [ACB17]. Dans ce cas, au lieu d'une tâche de classification, le discriminant score les suffixes réels et générés. Cela conduit à ce que D soit appelé un *critique*, et non un discriminant. La tâche du critique est maintenant d'estimer un score pour les suffixes réels et générés, en maximisant leur distance de Wasserstein. Reprenons les notations de cet article : soit $(\mathcal{X}, \Sigma)$ un espace mesurable, dont Σ est la tribu borélienne. Soit $\text{Prob}(\mathcal{X})$ l'espace des mesures de probabilités définies sur $\mathcal{X}$. Les auteurs définissent la distance de Wasserstein entre deux distributions $\mathbb{P}_r, \mathbb{P}_g \in \text{Prob}(\mathcal{X})$:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(X,Y)\sim\gamma} \left[\|X - Y\| \right],$$

où $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ représente l'ensemble des probabilités jointes de $\mathbb{P}_r$ et $\mathbb{P}_g$. Les auteurs citent également [Vil09] pour utiliser la dualité de Kantorovich-Rubinstein afin d'obtenir :

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leqslant 1} \mathbb{E}_{X\sim\mathbb{P}_r}[f(X)] - \mathbb{E}_{X\sim\mathbb{P}_g}[f(X)],$$

où le supremum est sur les fonctions au plus 1-lipschitziennes $f : \mathcal{X} \longrightarrow \mathbb{R}$. Dans ce coût, $f(\cdot)$ est remplacée, et donc approximée, par le critique. Il approxime donc une fonction qui doit être 1-lipschitzienne. Pour assurer cela, on peut utiliser une pénalité de gradient dans la fonction de coût finale, comme le propose [Gul+17]. Dans ce cas, la fonction de coût du modèle peut s'exprimer comme suit :

$$\min_G \max_D \mathbb{E}_{X\sim\mathbb{P}_r}[D(X)] - \mathbb{E}_{\hat{X}\sim\mathbb{P}_G}\left[D\left(\hat{X}\right)\right] - \lambda \mathbb{E}_{\tilde{X}\sim\mathbb{P}_{\tilde{X}}}\left[\left\|\left(\nabla_{\tilde{X}} D\left(\tilde{X}\right)\right\|_2 - 1\right)^2\right], \quad (5.1)$$

où $\tilde{X}$ est un point échantillonné sur le segments reliant X à son homologue généré $\hat{X}$. Cela signifie que, lors de l'entraînement, pour chaque observation, la pénalité de gradient

calcule le gradient d'un point uniformément échantillonné sur le segment entre un suffixe réel et son suffixe généré correspondant. La pénalité est la distance entre le gradient à ce point, et 1. En effet, comme les auteurs l'ont indiqué dans [Gul+17], le critique devrait être 1-lipschitzien en tout point. Or cette contrainte est insoluble, ils se contentent donc de faire tendre le critique vers la 1-lipschitzianité le long de ces segments, en partant du principe qu'assurer cette propriété sur cette « cage » de points l'assurera partout. Les auteurs ont montré expérimentalement que l'application de cette norme de gradient unitaire uniquement le long de ces segments semble effectivement suffisante à cette fin.

Les fonctions de coût de notre modèle sont $L(D;G)$, coût du critique lorsque le générateur est fixe (c'est-à-dire qu'il ne rétropropage pas l'erreur calculée), et $L(G;D)$, coût du générateur lorsque le critique est fixe. A partir des articles [ACB17] et [Gul+17] ainsi que des définitions précédentes, on déduit les fonctions de coûts suivantes pour un *batch* de données de taille m :

$$L(D;G) = \frac{1}{m}\sum_{i=1}^{m} D\left(\widehat{\sigma_{>k}^{(i)}}\right) - \frac{1}{m}\sum_{i=1}^{m} D\left(\sigma_{>k}^{(i)}\right) + \lambda\frac{1}{m}\sum_{i=1}^{m}\left(\left\|\nabla_{\widetilde{\sigma_{>k}^{(i)}}} D\left(\widetilde{\sigma_{>k}^{(i)}}\right)\right\|_2 - 1\right)^2,$$

$$L(G;D) = -\frac{1}{m}\sum_{i=1}^{m} D\left(\widehat{\sigma_{>k}^{(i)}}\right),$$

où $\widetilde{\sigma_{>k}^{(i)}}$ est un suffixe échantillonné entre les suffixes générés et les suffixes réels selon la formule $\widetilde{\sigma_{>k}^{(i)}} = \psi\widehat{\sigma_{>k}^{(i)}} + (1-\psi)\sigma_{>k}^{(i)}$, avec ψ une observation d'une variable aléatoire uniformément distribuée $\Psi \sim \mathcal{U}(0,1)$.

On remarque que les coûts d'apprentissage en pratique prennent l'opposé de la fonction de coût du modèle en équation 5.1. On voit donc que le critique a pour but d'octroyer un score faible aux suffixes générés, et un score élevé aux suffixes réels. Au contraire, le générateur a pour but de rapprocher les scores des suffixes générés des scores des suffixes réels, il cherche donc à ce que le critique maximise les scores des suffixes générés, ce qui revient à minimiser leur opposé. Nous évitons ainsi une tâche de maximisation et nous nous ramenons à une tâche de minimisation des coûts d'apprentissage.

Notons également que, contrairement à l'approche typique des GANs visant à produire des sorties uniques à partir de bruit aléatoire, aucun bruit n'est fourni à notre GAN afin de n'utiliser que de la donnée réelle et éviter des comportements non répétables de la part du modèle.

5.3.2 Reparamétrisation Gumbel-softmax

Par défaut, nous pouvons constater que nos suffixes générés et réels sont composés de vecteurs augmentés codés en *one-hot*, ce qui signifie qu'ils ne sont pas différentiables.

Par conséquent, nous appliquons la reparamétrisation Gumbel-softmax, que nous décrivons telle qu'exposée dans [Gum54] et [JGP17].

Tout d'abord, soit X une variable aléatoire discrète avec $k \in \mathbb{N}^\star$ classes, avec des probabilités de classes $\mathbb{P}(X = i) = p_i$, $i = 1, \ldots, k$. Chaque observation de cette variable aléatoire est représentée par un vecteur *one-hot* à k dimensions. Une façon d'échantillonner une observation *one-hot* $z = (z_1, \ldots, z_k)$ à partir des probabilités de classes $p_1, \ldots, p_k$ est d'utiliser la formule suivante :

$$z = \text{one-hot}\left(\max_{1 \leqslant l \leqslant k}\left\{\sum_{i=1}^{l} p_i \leqslant u\right\}\right), \tag{5.2}$$

où u est une observation d'une variable aléatoire $U \sim \mathcal{U}(0, 1)$. Cependant, l'équation (5.2) ne peut pas être utilisée telle quelle car le $\max(\cdot)$ la rend non différentiable. Une autre façon d'échantillonner z est d'utiliser la méthode *Gumbel-Max* décrite dans [Gum54] :

$$z = \text{one-hot}\left(\underset{i \in \{1,\ldots,k\}}{\text{argmax}}\left\{g_i + \log(p_i)\right\}\right), \tag{5.3}$$

où les g_i sont des observations de variables aléatoires $G_i \overset{\text{iid}}{\sim} \text{Gumbel}(0, 1)$. Cela base l'échantillonnage de z sur une fonction déterministe des p_i, avec un bruit indépendant généré à partir d'une distribution de Gumbel. L'équation (5.3) est cependant toujours non différentiable, à cause de la fonction $\text{argmax}(\cdot)$. Mais il suffit de la remplacer par une approximation différentiable pour pouvoir la différencier, et donc permettre la rétropropagation. Une approximation différentiable de la fonction $\text{argmax}(\cdot)$ est la fonction $\text{softmax}(\cdot)$ comme indiqué dans [JGP17], ce qui donne lieu à la méthode *Gumbel-softmax*. Cela nous donne un vecteur $y = (y_1, \ldots, y_k)$, où :

$$y_i = \frac{\exp\left(\dfrac{(\log(p_i) + g_i)}{\tau}\right)}{\displaystyle\sum_{j=1}^{k} \exp\left(\frac{(\log(p_j) + g_j)}{\tau}\right)}, \quad i = 1, \ldots, k. \tag{5.4}$$

Dans l'équation (5.4), $\tau \in \mathbb{R}_+^\star$ est appelé le paramètre de *température*. Il contrôle le degré d'approximation de y du vecteur *one-hot* correspondant : lorsque $\tau \to \infty$, y s'approche d'une loi uniforme, tandis que lorsque $\tau \to 0$, y s'approche d'un vecteur *one-hot*.

À titre d'illustration, supposons que X soit une variable aléatoire discrète comportant 10 classes $\{A, B, C, \ldots, J\}$, avec un vecteur de probabilités correspondant $\mathbb{p} = (p_1, \ldots, p_{10})$, visible sur la figure 5.8. Un échantillon de taille 1000 tiré selon une loi uniforme discrète, en utilisant l'équation (5.2), donne un vecteur one-hot $z = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0)$, comme le montre la figure 5.9. Cependant, l'échantillonnage à l'aide de l'équation (5.4) produit un

vecteur continu y qui se rapproche d'une loi uniforme lorsque τ devient grand, comme le montre la figure 5.9.

Dans notre cas, les classes de la variable aléatoire discrète X correspondraient aux noms des activités, et leurs probabilités pourraient être la sortie d'un réseau de neurones prédisant l'activité la plus probable lors d'une tâche de prédiction. Nous appliquons la méthode Gumbel-softmax aux activités codées en *one-hot* avec $\tau = 0.9^{\text{époque}}$, qui fait décroître τ au cours des époques d'entraînement, en rapprochant les vecteurs échantillonnés de leur forme originelle *one-hot* au fur et à mesure de l'entraînement.

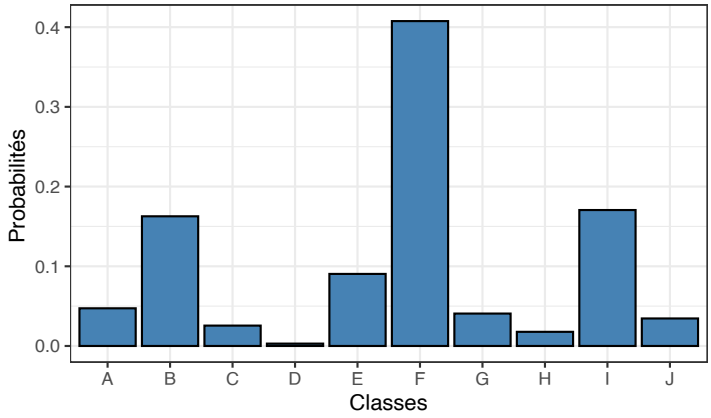

FIGURE 5.8 – Variable aléatoire discrète X et probabilités de classes correspondantes

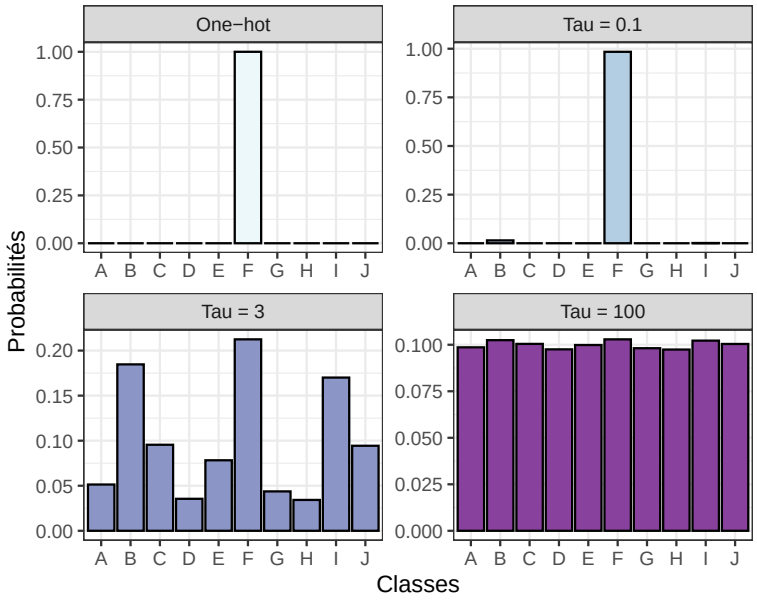

FIGURE 5.9 – Un échantillon de taille 1000 d'une loi Gumbel-softmax avec $\tau = 0, 1,\ 3,\ 100$, comparé à l'échantillon correspondant en *one-hot*

Notons également que la distance de Wasserstein se définit sur les espaces polonais, c'est à dire séparables et complets. Or la reparamétrisation *Gumbel-Softmax* transforme nos vecteurs *one-hot* en vecteurs continus à valeurs dans $[0;1]^{|\mathcal{A}|}$. L'intervalle unitaire est séparable car $\mathbb{Q} \cap [0;1]$ y est dense et $\mathbb{Q}$ est dénombrable. Par ailleurs, $[0;1]$ est complet car les suites de Cauchy y convergent. Puisque la distance de Wasserstein n'intervient que dans le coût du Critique, qui est le seul à recevoir les suffixes traités par la méthode *Gumbel-Softmax*, nous nous plaçons bien dans le cadre d'espaces polonais et la distance de Wasserstein peut bien être utilisée et estimée au travers du Critique.

5.4 Configuration expérimentale

5.4.1 Matériel

Le code mettant en exécution les principes évoqués dans ce chapitre a été implémenté en Python 3.9, avec PyTorch 1.12.0 et CUDA 11.6. Nous avons utilisé la puissance de calcul du supercalculateur ROMEO 2018 [1], qui dans sa totalité contient 115 serveurs équipés de processeurs Skylake 6132 à cadence de 2,6 GHz et 2×14 cœurs chacun, ainsi que 280 cartes graphiques NVIDIA Tesla P100 SXM2. Dans notre configuration, nous avons utilisé 1 processeur principalement pour le traitement des données, et 4 cartes graphiques pour les phases d'apprentissage et de test.

Nous avons testé notre approche avec et sans peuplement et l'avons comparée à 4 références ([LWW19], [Tax+17], [Tay+20], [TR20]), principalement pour garder les mêmes comparaisons que celles utilisées par les auteurs de [TR20], puisque nous utilisons leur algorithme comme notre propre référence.

Pour chaque journal d'événements, l'ensemble d'entraînement est constitué des premiers 70% des parcours dans l'ordre chronologique. Les ensembles de validation et de test représentent chacun 50% des 30% de parcours restants.

5.4.2 Métriques d'évaluation

Afin de maintenir possibles les comparaisons entre les articles, nous utilisons les mêmes métriques d'évaluation que dans [Tax+17] et [TR20] :

— Pour la précision de la prédiction des séquences d'activités, nous utilisons la similarité de Damerau-Levenshtein, tirée de la distance de Damerau-Levenshtein ([Dam64], [OL97]). Elle est basée sur la distance d'édition théorisée dans [Lev65], en ajoutant les transpositions aux insertions, suppressions et substitutions comme erreurs prises en compte. Plus précisément, étant donné un suffixe réel $\sigma_{>k}$ et sa prédiction $\widehat{\sigma_{>k}}$, en

1. `https://romeo.univ-reims.fr/pages`

notant la distance de Damerau-Levenshtein d_{DL} :

$$\text{Similarité}_{\mathrm{DL}}(\widehat{\sigma_{>k}}, \sigma_{>k}) = 1 - \frac{d_{\mathrm{DL}}(\widehat{\sigma_{>k}}, \sigma_{>k})}{\max\left(|\widehat{\sigma_{>k}}|, |\sigma_{>k}|\right)}.$$

La notation $d_{\mathrm{DL}}(\widehat{\sigma_{>k}}, \sigma_{>k})$ sous-entend que la distance de Damerau-Levenshtein est calculée entre leurs séquences d'activités respectives. Si cette quantité atteint 1, la séquence d'activités prédite est égale à la véritable séquence d'activités.

— Pour la précision de la prédiction temporelle, puisqu'un suffixe prédit est composé de plusieurs événements ayant chacun une durée donnée, nous utilisons l'erreur moyenne absolue (*Mean Absolute Error*, MAE).

5.4.3 Optimiseur

L'optimiseur choisi pour l'apprentissage du WGAN conditionnel est ADAM[KB17]. Réputé pour son efficacité dans le cas de données *sparse*, c'est à dire contenant beaucoup de zéros, il semble globalement plus performant et adapté à nos données que l'optimiseur RMSProp, utilisé dans [TR20].

Son taux d'apprentissage est fixé par défaut à 5×10^{-5}, de façon analogue à la valeur proposée pour RMSProp dans ce contexte. Un système de réduction du taux d'apprentissage est également mis en place : celui-ci est divisé par 2 lorsqu'une mesure des performances du modèle stagne, en utilisant l'outil et le coefficient de division proposés dans PyTorch [2].

La mesure des performances du modèle ne peut se baser sur les coûts d'apprentissage, ce qui serait en général l'approche typique puisque les coûts d'apprentissage sont généralement à minimiser. Or, bien que $L(D; G)$ et $L(G; D)$ soient facilement interprétables, il n'y a pas de garantie sur les valeurs que ceux-ci vont prendre, ni sur leur signe, malgré leur convergence. Il semble donc plus approprié de se référer à une métrique dont la définition même indique le sens d'évolution à mesure que le modèle converge. Afin d'évaluer cette convergence en termes de prédictions des temps et prédictions des activités à la fois, le rapport de la similarité de Damerau-Levenshtein et la MAE moyenne entre un suffixe prédit et le suffixe réel est choisi. La similarité de Damerau-Levenshtein entre les suffixes doit augmenter et se rapprocher de 1, tandis que la MAE moyenne entre les deux suffixes doit se rapprocher de 0. Ce rapport doit donc augmenter à mesure que le modèle converge vers une solution optimale.

Par ailleurs, dans ce cas, la stagnation est définie comme ce rapport n'augmentant pas pendant un certain nombre d'époques à un certain ϵ près. Ce nombre d'époques est appelé la *patience*, et constitue un hyperparamètre du modèle, ici mis à 10 époques, avec $\epsilon = 10^{-4}$.

Il est à noter qu'il est généralement conseillé d'entraîner plus fréquemment le critique que le générateur, comme mentionné dans [Gul+17]. Ainsi, le générateur n'est entraîné

2. `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html`

qu'un nombre n_{critic} d'époques, tandis que le critique est entraîné à chaque époque. Ici, $n_{\text{critic}} = 5$, paramétrage par défaut utilisé dans [Gul+17].

5.5 Résultats

Dans cette section, les noms de *BPI2012 (W)*, *BPI2012* et *BPI2017* sont abrégés en *BPI12 (W)*, *BPI12* et *BPI17* respectivement, par souci de lisibilité dans les tableaux de dépouillement des résultats.

Le tableau 5.4 indique les résultats de nos algorithmes prédictifs face aux références citées en section 5.4 en termes de similarité de Damerau Levenshtein, ce tableau concerne donc les performances sur les prédictions de séquences d'activités. Les deux premières lignes concernent notre GAN conditionnel de Wasserstein avec, puis sans peuplement, et les lignes suivantes concernent les références.

Tableau 5.4 – Similarité de Damerau-Levenshtein moyenne pour la prédiction de suffixes

Approche	**Helpdesk**	**Traffic fines**	**BPI12 (W)**	**BPI12**	**BPI17**
WGAN + Peuplement	**0.8914**	**0.7583**	**0.4612**	**0.4245**	**0.4322**
WGAN	0.8468	0.7316	0.4463	0.4093	0.3931
Taymouri & La Rosa [TR20]	0.8411	—	0.2662	0.3326	0.3361
Taymouri et al. [Tay+20]	0.8089	—	0.3520	0.2266	0.2958
Tax et al. [Tax+17]	0.7670	—	0.0632	0.1652	0.3152
Lin et al. [LWW19]	0.8740	—	—	0.2810	0.3010

Nous remarquons que le GAN de Wasserstein sans peuplement dépasse les performances de référence en termes de similarité de Damerau-Levenshtein, cette architecture semble donc par défaut supérieure en termes de prédiction de séries d'activités. L'ajout du peuplement augmente encore légèrement, et de façon systématique, les similarités de Damerau-Levenshtein, indiquant si les peuplements de processus contiennent effectivement de l'information relative aux séquences d'activités qui, bien qu'implicitement dans la donnée, n'était pas nécessairement capturée par les modèles.

Le tableau 5.5 présente la même disposition que le tableau précédent, cette fois-ci avec les performances en termes d'erreur moyenne absolue sur les temps totaux prédits, exprimée en jours. Ainsi, une MAE de 4 indique une erreur de 4 jours entre la durée totale restante réelle, et la durée restante totale prédite.
Concernant cette métrique d'évaluation, le GAN de Wasserstein sans peuplements ne surpasse que de peu les autres modèles, la progression n'est pas nécessairement notable. En revanche, l'ajout du peuplement de processus et la prédiction conditionnellement à ce dernier semble augmenter de façon bien plus notables la précision des prédictions temporelles.

Pour *Helpdesk*, les prédictions de durées passent de 6,09 jours de MAE à 4,42. Une augmentation notable de la précision concerne *Traffic fines*, qui voit son erreur moyenne divisée par 2,62, ainsi que *BPI2017*, pour qui l'erreur moyenne est divisée par 2,24.
Il semble donc que le peuplement de processus, conjointement à une architecture générative, améliorée *via* l'utilisation d'un coût d'entraînement de Wasserstein, permette un apprentissage des parcours plus précis sur les séquences d'activités, et surtout sur les durées. L'augmentation drastique de la précision sur les temps dans *Traffic fines* et *BPI2017*, dont le premier présente un effet *bucket* clair et le second des saisonnalités observables directement dans les peuplements de ses activités, confirme l'impact positif de l'emploi du peuplement dans les tâches prédictives de parcours individuels.

Tableau 5.5 – MAE moyenne pour le temps total restant prédit par la génération de suffixes

Approche	Helpdesk	Traffic fines	BPI12 (W)	BPI12	BPI17
WGAN + Peuplement	**4.62**	**98.27**	**8.84**	**9.22**	**5.73**
WGAN	6.09	257.33	11.19	11.89	12.87
Taymouri & La Rosa [TR20]	6.21	—	12.12	13.62	13.95
Taymouri et al. [Tay+20]	6.30	—	34.56	169.23	80.81
Tax et al. [Tax+17]	6.32	—	50.11	380.1	170
Lin et al. [LWW19]	—	—	—	—	—

5.6 Conclusion

L'utilisation du coût de Wasserstein dans le GAN a déjà apporté une amélioration des performances du modèle, pour un entraînement grandement facilité. Cependant, le réel saut en termes de performances, en particulier pour les prédictions de durées, est dû à l'ajout des peuplements de processus. Les capacités prédictives du modèle sont notablement améliorées dans tous les jeux de données testés, malgré leur variabilité en termes de nombre de parcours, nombre de traces, nombre d'événements et durées.
Ceci semble indiquer que l'inclusion d'informations concernant l'état global des processus au cours du temps est crucial pour des prédictions précises à l'échelle des parcours. Les modèles d'apprentissage profond ne semblent en effet pas intégrer cette information lorsqu'elle est implicite, en particulier à cause des transformations des temps en durées.

De plus, bien que cette méthode ait été testée dans un GAN de Wasserstein conditionnel, il est possible qu'elle fonctionne dans d'autres architectures de modèles d'apprentissage profond.

Le peuplement de processus offre également plusieurs ouvertures sur le *clustering* de parcours. Les méthodes de *clustering* les plus efficaces, telles qu'ActiTraC (*Active Trace Clustering*) [De +13], se basent sur des modèles de fouille de processus tels que [WR11]

afin de déterminer si un parcours a sa place dans un *cluster* : si l'ajoute du parcours résulte en un modèle dont l'adéquation aux données est trop diminuée, ce parcours n'appartient pas à ce *cluster*. Ces techniques, bien que reconnues pour leur efficacité, sont extrêmement longues en termes de temps de calcul. D'autres approches, comme [Di +19], cherchent plutôt à grouper des parcours selon leurs préfixes, dans un but de distinguer par la suite les parcours dont l'issue compte plus que le trajet (retard / non retard, *et caetera*). Des méthodes de *clustering* se basent sur l'alignement de séquences, y compris aidé d'apprentissage machine comme dans [BCH20], tandis que les auteurs de [JA09] utilisent des métriques telles que la distance d'édition, autrement dit la distance de Levenshtein, avec une automatisation de la pondération des opérations d'éditions.
Nous pouvons imaginer qu'utiliser le peuplement de processus dans une optique de groupement est triplement intéressant :

— le *clustering* de parcours complets selon des peuplements, dans une optique de modélisation plus classique.

— Le *clustering* de parcours incomplets selon des peuplements, en particulier dans une optique prédictive.

— Lors d'un *clustering* de parcours, ceux-ci sont segmentés mais pas indépendants : un parcours peut en influencer d'autres. Dans ce cas, un *clustering* consisterait plutôt en un zoom sur un sous-processus donné, mais les *clusters* doivent malgré tout être utilisés conjointement à de l'information relative au processus global, ce que permet le peuplement.

De plus, moyennant quelques changements au modèle prédictif, le peuplement de processus ouvre la porte de la génération de données à partir de conditions initiales, et donc de la simulation et de la tomographie. Il s'agit de l'objet du chapitre suivant.

Chapitre 6

Simulation et tomographie

6.1 Introduction

Possédant à présent un modèle prédictif fonctionnel et surpassant l'état de l'art ainsi qu'une méthode de *feature engineering* utilisable sur n'importe quel journal d'événements (dans la mesure où celui-ci ne contient pas de valeurs manquantes), il est à présent possible de prédire les événements restants de n'importe quel parcours incomplet avec une précision accrue. Mais le peuplement offre d'autres possibilités.

En effet, notre GAN n'utilisant pas, comme un GAN typique pour la génération d'images par exemple, de bruit afin de générer ses prédictions, lui fournir un début de parcours donné plusieurs fois ne résultera pas en des prédictions différentes, il n'y a pas d'aléatoire dans la génération. On pourrait essayer de générer des parcours à partir de bruit malgré tout, mais une problématique appliquée se rajoute : nous pensons qu'un client trouvera plus de satisfaction à créer des scénarii et à en observer les conséquences prédites, qu'à simplement générer des parcours à partir de bruit. De plus, la génération aléatoire ne serait que peu satisfaisante dans un cadre industriel. Il faut donc trouver une façon de permettre au client de créer ces scénarii, puis au modèle de générer des parcours en fonction.

C'est ainsi que le peuplement de processus intervient. Il semble en effet utile de permettre au client de modifier les peuplements des activités selon son bon vouloir, et de lui donner en retour le parcours typique que ce peuplement impliquerait pour une nouvelle unité entrant dans le processus selon ces peuplements.
De plus, la simulation permet une forme simple de tomographie. Le modèle, par la nature de son entraînement avec des réseaux antagonistes, ainsi que de la sur-couche d'encodage-décodage dans le générateur, rend difficile d'expliquer quelles entrées dans la donnée donnent lieu aux différentes prédictions. Or l'alliance de la simulation, permettant de tester des combinaisons de peuplements à notre guise et de la simple analyse factorielle, peuvent donner un premier aperçu instructif de la façon qu'a le modèle de générer des prédictions.

Ainsi, ce chapitre se concentre sur la simulation de parcours à partir de conditions initiales de peuplements, puis sur une façon de créer des jeux de données simulés permettant ensuite d'expliquer partiellement les sorties du modèle prédictif. L'analyse des résultats de simulation à visée explicative sert à la fois d'illustration de la capacité de simulation du modèle, et de sa tomographie.

6.2 Méthode de simulation

6.2.1 Marquer le début des parcours

Il arrive relativement fréquemment que les processus métiers ne possèdent pas de départ unique. Prenons les journaux d'événements utilisés dans cette thèse. Le tableau 6.1 fait l'inventaire des activités présentes en début de parcours dans ces journaux d'événements. Nous voyons que *Helpdesk* comporte 6 activités possibles pour marquer un début de parcours, et *BPI2012 (W)* en possède 3.

Tableau 6.1 – Activités observées au départ des parcours des journaux d'événements utilisés

Journal d'événements	**Activités de départ**
Helpdesk	$\{1, 2, 3, 4, 5, 6\}$
Traffic fines	$\{1\}$
BPI2012 (W)	$\{1, 4, 6\}$
BPI2012	$\{1\}$
BPI2017	$\{1\}$

Il serait donc intéressant de pouvoir simuler un parcours dans sa totalité à l'aide de conditions de départ, y compris sa première activité.

Il fut donc jugé judicieux d'ajouter, à l'instar du *End of State* en fin de parcours pour la modélisation prédictive, un *Start of State* en début de parcours. Ceci permettrait de débuter la séquence d'événements caractérisant un parcours par ce *Start of State*, qui ne serait qu'une activité artificielle non présente dans la donnée d'origine, et donc de prédire la suite d'événements à partir de ce départ artificiel – et ainsi prédire le premier événement.

Formellement, notons $\langle SoS \rangle$ ce *Start of State*. Nous partons du principe que ceci représentant un début artificielle, il n'a pas d'antériorité réelle à la première activité du processus. De ce fait, les variables observées dans le premier événement sont considérées de valeurs identiques lors du *Start of State*. Ceci implique, en termes de simulation, que l'on s'attend potentiellement à ce qu'un peuplement donné (ainsi que de potentielles covariables) puisse conditionner la première activité, qui n'en serait pas qu'une observation

conjointe mais bien potentiellement résultante. Le cas échéant, on s'attend à ce qu'un modèle bien entraîné puisse prédire avec succès la première activité uniquement à partir des peuplements (et covariables) concomitants.
Ainsi, dans un journal d'événements L, un parcours $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ de longueur n serait augmenté jusqu'à avoir une longueur $n+2$, avec $\pi_{|\mathcal{A}|}(e_1) = \langle SoS \rangle$ et $\pi_{|\mathcal{A}|}(e_{n+2}) = \langle \text{EoS} \rangle$.
L'horodatage du *Start of State* est le même que celui de la première activité du parcours : $\pi_{\mathcal{T}}(e_1) = \pi_{\mathcal{T}}(e_2)$. Dans le cas de deux horodatages, celui-ci commence et se termine en même temps que le début de la première activité réelle : $\pi_{\mathcal{T}}(e_1) = \pi'_{\mathcal{T}}(e_1) = \pi_{\mathcal{T}}(e_2)$.

6.2.2 Sur-échantillonnage

Nous avons vu en section 2.2.4 que les effectifs des traces étaient fortement déséquilibrées, et que nombre d'entre elles n'apparaissaient qu'une fois dans leur journal d'événements respectifs. Il n'est donc pas étonnant de voir une majorité de ces traces disparaître des prédictions offertes par le WGAN conditionnel. Bien que les traces les plus fréquentes apparaissent avec succès dans le cadre de la simple modélisation prédictive, l'intérêt de la simulation réside surtout dans la capacité du modèle à générer les traces existantes avec une certaine diversité. Trois options s'offrent à nous :

— le sous-échantillonnage des traces majoritaires.

— Le sur-échantillonnage des traces minoritaires.

— La création d'individus synthétiques parmi les traces minoritaires.

Le sous-échantillonnage des traces majoritaires retirerait une partie de l'information concernant les peuplements et covariables menant à leur apparition. Cette option n'a donc pas été considérée.
L'approche par création d'individus synthétiques semble optimale puisqu'elle permet d'éviter la simple répétition de parcours et de leurs variables explicatives : nous aurions des traces aux proportions plus homogènes, avec une variation dans les variables explicatives (les peuplements en particulier) qui suivrait les *patterns* des journaux d'événements. Or la création de tels individus synthétiques demanderait que leurs constituants soient conformes à un ensemble de règles inaccessibles et souvent inconnues du processus. Ainsi, des algorithmes de type SMOTE [Cha+02], qui crée des individus synthétiques par interpolation et k-plus proches voisins, ne garantirait pas la conformité des individus synthétiques aux processus. En effet, les approches de type SMOTE suivent une logique proche de celle utilisée pour calculer la pénalité du gradient dans le chapitre 5 :

— un parcours est sélectionné aléatoirement parmi les parcours dont la trace est minoritaire.

— Ses k plus proches voisins sont identifiés, puis un de ces voisins est sélectionné aléatoirement.

— Un parcours est échantillonné sur le segment reliant l'individu initial à son voisin. Ainsi, si σ_1 est l'individu initial et σ_2 le voisin sélectionné, une observation ψ d'une variable aléatoire $\Psi \sim \mathcal{U}(0,1)$ est utilisée pour créer $\sigma_{SMOTE} = \psi\sigma_1 + (1-\psi)\sigma_2$.

Or le fait d'échantillonner sur un segment liant deux parcours sélectionnés aléatoirement implique que le segment entier soit inclus dans le sous-espace des parcours pouvant exister au travers du processus. Ceci implique que ce sous-espace soit convexe, chose qui ne semble ni évidente ni garantie, et encore moins vérifiable.

Le sur-échantillonnage des traces minoritaires semble donc être la méthode impliquant le moins de perte d'information. Le problème avec cette méthode est que le sur-échantillonnage ne fait que répéter des individus, risquant donc de mener le modèle à ne considérer que les conditions exactes de leur apparition comme valables pour les prédire par la suite.

Le sur-échantillonnage choisi est proportionnel à la fréquence des traces : plus une trace est rare, plus son sur-échantillonnage sera fort. Soit L un journal d'événements et $|L|_T$ le nombre de traces dans le journal d'événements, et soit $\mathbb{p} = (p_1, p_2, \ldots, p_{|L|_T})$ le vecteurs des fréquences des traces. On peut définir les probabilités pour un échantillonnage simple avec remise des traces selon la formule :

$$p_i^\star = \frac{1/p_i}{\sum_{k=1}^{|L|_T} 1/p_k}, \quad i = 1, \ldots, |L|_T.$$

Un $p_i\star$ correspond à probabilité d'effectuer un échantillonnage simple avec remise parmi les parcours représentés par la i$^{\text{eme}}$ trace, on a alors le vecteur $\mathbb{p}^\star = (p_1^\star, \ldots, p_{|L|_T}^\star)$. Ainsi, si le modélisateur souhaite augmenter son échantillon d'environ N parcours, on échantillonne $\lfloor p_i^\star N \rfloor$ fois avec remise les parcours représentés par la i$^{\text{ème}}$ trace, ce avec $i = 1, \ldots, |L|_T$. Le choix de N est laissé au modélisateur et constitue un hyperparamètre.

6.2.3 Apprentissage

L'apprentissage du WGAN conditionnel se déroule de façon parfaitement analogue au cas de la seule modélisation prédictive :

— les horodatages sont transformés en durées au moyen de la fonction $\theta(\cdot)$ dans le cas d'un horodatage, $\theta_{\text{previous}}(\cdot)$, $\theta_{\text{next}}(\cdot)$, $\theta_{\text{end}}(\cdot)$ dans le cas de deux horodatages.

— Les activités sont converties en vecteurs *one-hot*.

— Chaque parcours est scindé en préfixes et suffixes successifs.

6.3 Tomographie

6.3.1 Plans d'expériences

La tomographie est, dans le principe, l'analyse d'un object opaque en y envoyant un signal déterministe, puis en analysant le signal sortant de cet objet. Ici, l'objet opaque est notre modèle prédictif, et la tomographie se base sur la simulation. L'idée est de créer des jeux de données à faire prédire par le WGAN conditionnel, puis à interpréter ces prédictions au regard du jeu de données créé. La simulation nous permet de faire prédire des parcours complets uniquement à partir de peuplements et d'un *Start of State*. Nous pouvons donc nous concentrer sur la création de combinaisons de peuplements, auxquelles un *Start of State* et un horodatage viendraient s'ajouter.

Dans la section 5.2.4, nous avons vu que les peuplements de chaque activité étaient normalisés entre 0 et 1 par la normalisation min-max. Une première approche serait donc de considérer des combinaisons de peuplements minimaux et maximaux. Nous réduisons finalement un peuplement à une variable binaire valant 0 ou 1, qui correspondra à son minium ou à son maximum observé. Ainsi, il devient immédiat d'utiliser les plans d'expériences factoriels afin de générer des combinaisons de peuplements de la sorte.

Pour un journal d'événements donné L avec un ensemble d'activités $\mathcal{A}$, dans le cas où $|\mathcal{A}|$ est suffisamment petit, un plan factoriel complet est utilisable. Ainsi, on crée $2^{|\mathcal{A}|}$ combinaisons de peuplements minimaux et maximaux, chaque combinaison étant représentée sous la forme d'un vecteur binaire.

Tableau 6.2 – Plan factoriel complet pour 3 peuplements d'activité

A1	**A2**	**A3**
0	0	0
1	0	0
0	1	0
0	0	1
1	1	0
1	0	1
0	1	1
1	1	1

Le tableau 6.2 liste les différentes combinaisons de peuplements minimaux et maximaux dans le cas de 3 activités. Nous avons bien les $2^3 = 8$ combinaisons, et chaque ligne correspond aux peuplements d'un événement donné.

La notion de « grand » nombre d'activités est relative et dépend du matériel utilisé et des conditions de cette génération. Par exemple, dans le cas d'un modèle générant une

prédiction à chaque centième de seconde, un plan factoriel complet aura ses prédictions pour 15 activités en 0,09 heures, contre 93 heures pour 25 activités. Garder les 26 peuplements de *BPI2017* résulterait en une génération prenant 186 heures, devant générer 67 108 864 parcours à partir d'autant de combinaisons. L'intérêt de la sélection de peuplements à l'aide de la corrélation croisée donnée en section 5.2.4 est donc d'autant plus importante dans ce cas.

En revanche, dans le cas où le nombre de peuplements résulterait en un temps de prédiction prohibitivement long, il faut considérer des plans factoriels $2^{|\mathcal{A}|-k}$.

6.3.2 Méthode du cube

Outre les plans d'expériences, il peut être intéressant de chercher quels peuplements réels permettent la prédiction de différentes traces. Il convient donc d'échantillonner des peuplements parmi le premier événement des parcours disponibles, afin de créer des individus à simuler à partir de ces peuplements réels et d'un *Start of State*. Pour effectuer cette tâche, la méthode du cube est utilisée [Til06].

La méthode du cube est une méthode d'échantillonnage permettant aux totaux des variables d'un échantillon d'approcher autant que possible les totaux de ces variables dans la population.
Plus formellement, soit une population $U = \{U_1, \ldots, U_N\}$ d'effectif $N \in \mathbb{N}^\star$. Pour simplifier, on écrit $U = \{1, \ldots, N\}$. Soit une variable aléatoire X, qui est observée pour chaque individu $i \in U$ et vaut x_i. Le total de ces observations est $T_x = \sum_{i \in U} x_i$.
À présent, supposons que l'on dispose d'un échantillon S d'effectif $1 \leqslant n < N$, effectué selon un plan de sondage à probabilités inégales effectué sur U. La probabilité qu'un individu i soit dans l'échantillon S est la probabilité d'inclusion notée $\pi_i = \mathbb{P}(i \in S)$. L'estimateur d'Horvitz-Thompson du total est :

$$\widehat{T_{\mathrm{HT}}} = \sum_{i \in S} \frac{x_i}{\pi_i}.$$

La méthode du cube vise à créer un échantillon de taille n contenant des individus de sorte que, sur un ensemble de variables aléatoires, la différence entre l'estimateur de Horvitz-Thompson du total sur l'échantillon et leur total sur la population soit minimisée.

6.3.3 Analyses factorielles

Une fois les peuplements issus des deux méthodes exposées ci-avant utilisés au travers du modèle génératif, nous disposons de parcours générés selon ces peuplements initiaux. Puisque ces peuplements sont simplement des variables aléatoires quantitatives, une possibilité simple et visuelle d'en analyser les sorties réside dans l'analyse factorielle.
Afin de procéder, les parcours générés sont classées par trace. ainsi, toutes les générations

résultant en la même trace sont regroupées, et une analyse factorielle des peuplements pour ces générations peut ensuite être effectuée.

Le tableau 6.3 montre un exemple fictif d'un tel groupement. Le groupe 1 indique que la trace $\langle 1, 2, 3, 4\rangle$ serait générée, en partant d'un parcours ne comportant qu'un *Start of State*, par les peuplements tous minimaux ou par un seul peuplement maximal à la fois. Un groupement parfaitement analogue est effectué dans le cas d'une simulation à partir de peuplements échantillonnés dans le jeu de données d'origine à l'aide de la méthode du cube.

TABLEAU 6.3 – Exemple fictif de groupement de prédictions par traces dans le cadre de la tomographie avec un plan factoriel 2^3

Identifiant	**Trace**	**Peuplement**	**Groupe**
U_1	$\langle 1, 2, 3, 4\rangle$	000	Groupe 1
U_2	$\langle 1, 2, 3, 4\rangle$	001	
U_3	$\langle 1, 2, 3, 4\rangle$	010	
U_4	$\langle 1, 2, 3, 4\rangle$	100	
U_5	$\langle 1, 4\rangle$	011	Groupe 2
U_6	$\langle 1, 4\rangle$	101	
U_7	$\langle 1, 3, 4\rangle$	110	Groupe 3
U_8	$\langle 1, 3, 4\rangle$	111	

Cas des plans d'expériences

Dans le cas des plans d'expériences, les valeurs des peuplements sont restreintes à 0 ou 1. Ne s'agissant que de deux valeurs, elles sont finalement analogues à deux classes : la classes « minimum » et la classe « maximum ». Nous pouvons donc immédiatement effectuer une Analyse des Composantes Principales (ACM) [Ben76] puisque les peuplements peuvent être ici traités comme des variables qualitatives.

La figure 6.1, créée grâce aux librairies R *FactoMineR*[LJH08] et *factoextra*[KM20] montre une telle ACM sur le groupe 1 du tableau 6.3. Nous pouvons voir que la trace $\langle 1, 2, 3, 4\rangle$ semble bien générée lorsque le peuplement de l'activité 1 est maximal, ou bien l'activité 2, ou bien l'activité 3 : nous observons en effet que les peuplements maximaux, notés A1_1, A2_1 et A3_1 sur le graphique, sont éloignés au maximum les uns des autres et forment presque un triangle équilatéral. Nous pouvons en déduire, par les règles d'interprétation de l'ACM, que la trace $\langle 1, 2, 3, 4\rangle$ du tableau 6.3 semble générée lorsque n'importe quel peuplement est maximal, avec les deux autres minimaux. Cette analyse peut être répétée sur les autres groupes, donnant *in fine* une interprétation des peuplements donnant lieu aux différentes traces générées par le modèle à partir d'un *Start of State*.

Cas de la méthode du cube

Dans le cas de la méthode du cube, les peuplements utilisés ne peuvent plus aisément être discrétisés. Une Analyse des Composantes Principales (ACP) [Pea01 ; Hot33] normée est donc effectuée, au lieu d'une ACM.

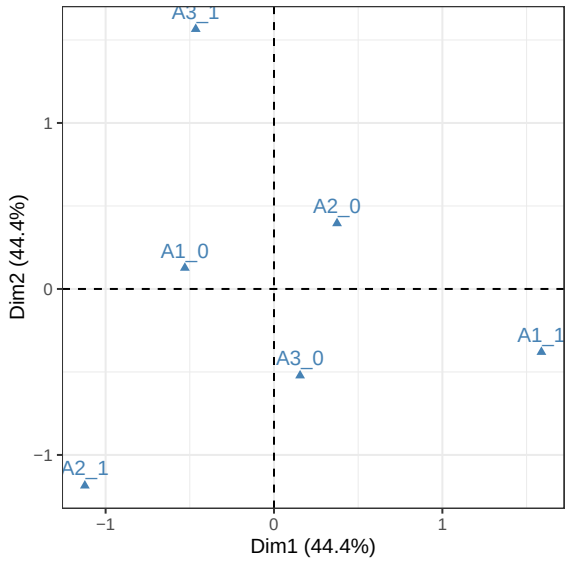

FIGURE 6.1 – Exemple d'ACM sur les peuplements fictifs issus d'un plan d'expériences 2^3

6.4 Résultats

Les résultats se démultipliant étant donné la méthode employée, des résultats illustratifs sont ici présentés au lieu de l'exhaustivité. En effet, l'analyse est effectuée par journal d'événements, par trace. Des exemples nécessaires et suffisants sont présentés afin d'illustrer le propos ainsi que son intérêt et ses limites.

6.4.1 Plans d'expériences et ACM

Commençons par un résultat relatif aux plans d'expériences. le WGAN conditionnel a été entraîné sur *Traffic fines*, dont les parcours ont été augmentés d'un *Strat of State*. ensuite, un plan d'expériences 2^8 a été créé, afin de générer des peuplements à 0 ou 1 pour les activités 1, 2, 3, 4, 5, 8, 9 et 11, sachant que les activités 6, 7 et 10 ont été retirées préalablement en conséquence des corrélations croisées. Les préfixes ainsi créés ont été envoyés pour prédiction dans le modèle. Les peuplements donnant des parcours ayant la même trace peuvent ensuite être examinés à l'aide d'une ACM.

La figure 6.2 permet de voir un résultat sur *Traffic fines*.
Ici, les unités créées à partir d'un peuplement et d'un *Start of State* sont représentées, ainsi

que les peuplements qui leur ont été affectés. Ces préfixes ont tous donné des prédictions dont la trace est $\langle 1, 2, 7, 8 \rangle$. Nous voyons sur cette figure que la trace $\langle 1, 2, 7, 8 \rangle$ est générée lorsque les peuplements des activités 9 et 1 sont maximaux, tandis que le peuplement de l'activité 5 est minimal. De même, cette trace a tendance a être générée quand les activités 2 et 4 sont maximales, ou quand l'activité 3 a un peuplement maximal. L'inertie de 33,3% représentée sur ce plan demande cependant de faire preuve de nuance face à ces résultats, qui ne sont donc pas à prendre comme une conclusion absolue.

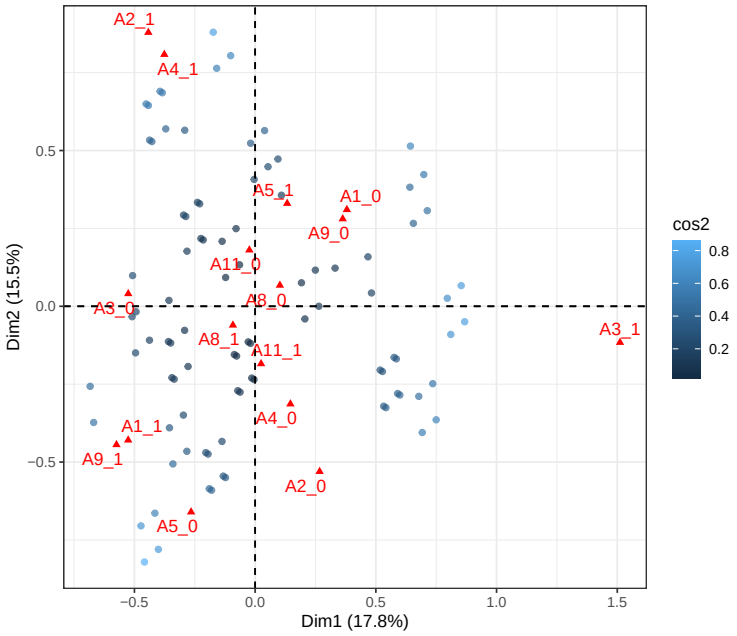

FIGURE 6.2 – ACM sur les peuplements ayant généré la trace $\langle 1, 2, 7, 8 \rangle$ à partir d'un plan d'expériences 2^8 dans *Traffic fines*

6.4.2 Méthode du cube et ACP

Penchons-nous sur *Helpdesk*. Un échantillon d'environ 1000 peuplements de débuts de parcours a été requis, résultant en un échantillon final de 1250 débuts de parcours. De manière analogue au cas des plans d'expériences, ces peuplements se voient concaténés un *Start of State*, permettant leur prédiction par le modèle. Ces prédictions peuvent ensuite être regroupées par traces, et analysées à l'aide d'une ACP. La figure 6.3 montre une telle ACP sur les peuplements ayant permis la prédiction de parcours dont la trace est $\langle 1, 2, 2, 3 \rangle$. Nous observons que cette trace est typiquement générée lorsque les peuplements 4 et 8 sont similaires, de même que pour les peuplements 9, 5 et 6. D'une certaine façon, nous pouvons conclure des peuplements 2 et 7 similaires permettent la génération de cette trace, mais la longueur de ces flèches indique que leur représentation n'est pas optimale sur le graphique : cette dernière interprétation n'est donc qu'à considérer avec précaution.

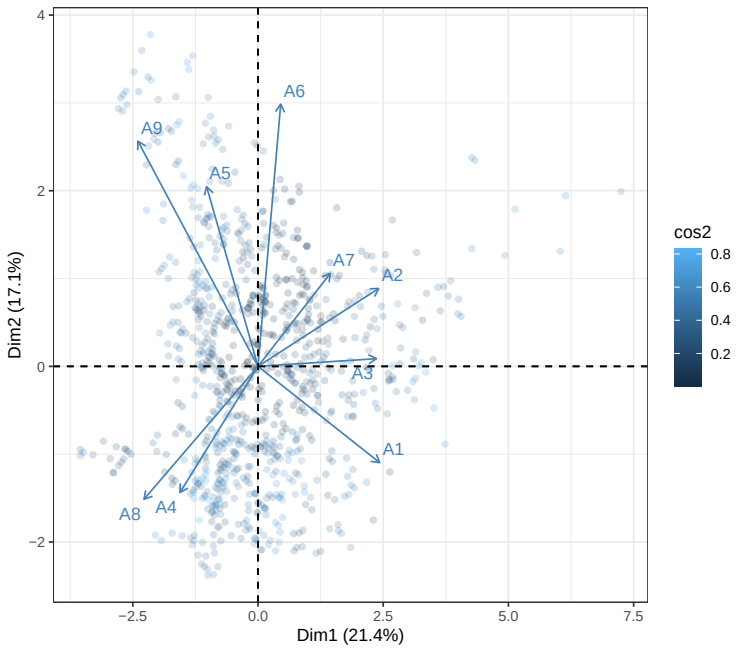

FIGURE 6.3 – ACP sur les peuplements ayant généré la trace $\langle 1, 2, 2, 3 \rangle$ à partir de la méthode du cube pour un échantillon de 1250 peuplements dans *Helpdesk*

Il est intéressant de voir si des conclusions similaires aux plans d'expériences peuvent être déductibles de la méthode du cube et de l'ACP subséquente. Nous avons sur la figure 6.4 une ACP des peuplements échantillonnés par méthode du cube, sur un échantillon de taille 1000, ayant donné la trace $\langle 1, 2, 7, 8 \rangle$, comme dans le cas du plan d'expériences. Nous voyons sur cette figure que lorsque les peuplements 2 et 4 ont des valeurs similaires, comme dans la figure 6.2, cette trace semble être générée. En revanche, la paire de peuplements 1 et 9 ne semble plus de mise dans l'interprétation de cette génération : au lieu de cela, la paire de peuplements 1 et 8 semble permettre la prédiction de cette trace lorsque leurs valeurs sont similaires, et lorsque le peuplement de l'activité 3 suit la tendance inverse.

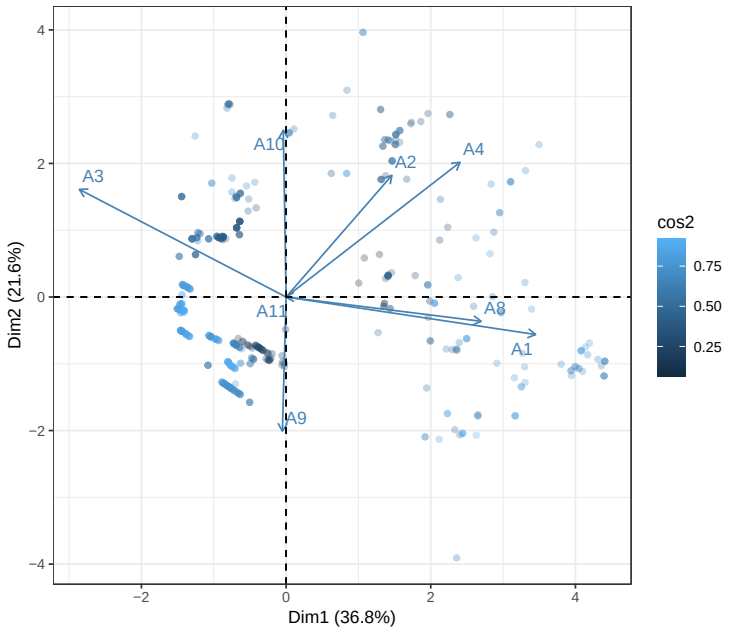

FIGURE 6.4 – ACP sur les peuplements ayant généré la trace $\langle 1, 2, 7, 8 \rangle$ à partir de la méthode du cube pour un échantillon de 1000 peuplements dans *Traffic fines*

6.5 Conclusion

La méthode développée ici permet la simulation de parcours à partir de peuplements de processus définis par l'utilisateur, et son illustration est faite *via* une tomographie du modèle sur ces peuplements. En soi cette tomographie permet d'établir un lien entre peuplements et prédictions, mais ce lien est pour l'instant restreint au cas de préfixes ne contenant qu'un *Start of State*. La tomographie pourrait également s'effectuer sur les parcours incomplets, mais la quantité de préfixes possibles correspond finalement au nombre d'événements dans un journal d'événements. Nous pourrons nous restreindre aux traces, étant donné que les dates et heures sont transformées en durées, retirant donc la position dans le temps des préfixes. Mais la combinatoire issue du couplage entre préfixes et peuplements devient rapidement prohibitivement grande. Des méthodes de plans d'expériences fractionnaires deviendraient alors nécessaires.

En soi, ce chapitre est également une manière de montrer, outre les capacités prédictives améliorées du modèle, qu'un seul vecteur de peuplements contient en lui-même une quantité d'informations sur le processus qui est effectivement exploitable à travers le modèle, permettant de générer des parcours différents uniquement selon la combinaison de peuplements renseignée. L'entraînement du modèle en incluant le *Start of State* est sensiblement identique en termes des métriques d'évaluations présentes dans le chapitre 5 : les séquences prédites à partir d'une seule activité sont en fait très similaires, voire identiques, à celles prédites à partir du *Start of State*. Il n'y a donc aucune perte dans les performances prédic-

tives à rajouter un *Start of State* systématique afin de posséder un modèle aussi polyvalent que possible.

Bien sûr, cette méthode ne transforme pas le modèle en jumeau numérique. En réalité, la conformité au processus réel (observé) des parcours générés par le modèle ne tient qu'à la capacité du critique du modèle à l'imposer au générateur, donc à sa capacité à l'apprendre à partir de la donnée. Or l'application réelle des processus est souvent régie par des règles tacites, des décisions des opérateurs du processus qui ne sont pas documentées, ou pas transmises, et qui peuvent être flexibles, se mouvoir au cours du temps. Il ne s'agit pas d'un réseau de Petri [RE98] ou équivalent, il n'y a donc pas de certitude que des liens interdits entre activités ne seront pas permis par le modèle prédictif dans cette thèse.

Par ailleurs, l'aspect simulatoire mène invariablement vers la problématique posée par le fait de retirer ou rajouter des activités dans le processus. Ceci n'est évidemment pas possible avec cette méthode, et demande une recherche à part entière.

On peut malgré tout tirer un enseignement potentiellement très utile de ce chapitre : le modèle parvient à générer plus d'un parcours grâce aux premiers peuplements enregistrés dans les parcours. On peut donc imaginer que ces peuplements peuvent être utiles à des fins de *clustering* de parcours. Il serait intéressant d'explorer le fait de grouper des parcours uniquement à l'aide de leur premier vecteur de peuplement, à des fins par la suite prédictives.

Chapitre 7

Dérive conceptuelle et cartes de contrôle

7.1 Introduction

Ce chapitre constitue une dernière extension, axée cette fois-ci sur le principe de dérive conceptuelle, également appelée « *concept drift* » en anglais. Au sens large, la dérive conceptuelle se définit par un changement des propriétés statistiques de la variable cible d'un modèle prédictif.
Plusieurs recherches ont déjà été effectuées à ce sujet, dont un inventaire a été entrepris par les auteurs de [Sat+21]. Une multitude d'approches et de définitions y sont exposées. Certaines utilisent le *clustering* et l'évolution des *clusters* afin de détecter une dérive, d'autres utilisent des tests statistiques, d'autres la conformité de modèles de processus fouillés à partir de méthodes classiques de fouille de processus, la détection de tendances, de points de rupture, *et caetera*. Ces approches sont de complexité variées, tant dans leur implémentation que dans leur utilisation dans un contexte d'entreprise.

Ce chapitre apporte donc une méthodologie permettant de détecter la dérive conceptuelle à la fois dans le but d'alerter un client sur les dérives de ses processus, en plus de nous donner la capacité de déterminer à quel moment nos modèles prédictifs perdraient en capacité prédictive sur les données de façon formelle, d'une façon simple à implémenter et interpréter, en particulier dans un contexte industriel de production et de chaîne d'approvisionnement.

Ces considérations pratiques ont naturellement mené à la maîtrise statistique des procédés (MSP) et les cartes de contrôle, mentionnées entre autres dans [EP04], [Mon09] et [Pil05]. En effet, celles-ci permettent le suivi et le diagnostic de la dérive dans la qualité de production de machines dans le cadre de processus industriels. Elles permettent, à partir d'échantillons issus de la production, de contrôler par la mesure ou selon des critères de non-conformité la qualité de production aux étapes choisies de cette dernière. Elles apportent un formalisme et une approche statistique de la dérive de qualité de production, permettant d'estimer le risque de première espèce, le risque de deuxième espèce, et de les

exprimer en termes de nombre d'échantillons nécessaires en moyenne pour qu'elles se produisent. Il semble donc assez naturel d'utiliser de tels outils pour mesurer non pas la dérive d'une production, mais la dérive conceptuelle de données de processus.
Cette approche a déjà été utilisée pour le contrôle de la dérive conceptuelle dans la donnée de *streaming* dans [Kun09], et un type de cartes de contrôle pour les systèmes dynamiques variant dans le temps dans [MLW21]. Une autre méthode basée sur le gradient de la log-vraisemblance d'un modèle d'apprentissage supervisé est décrite dans [ZBA22].
L'idée étant une utilisation en production et une compréhension rapide et simple par n'importe quel opérateur, l'utilisation de cartes de contrôle simples est privilégiée.
De plus, un modèle pouvant potentiellement apprendre une dérive et donc prédire en conséquence, il est important de distinguer les cas : une approche pour la donnée, et une approche relative aux performances du modèle prédictif.

La méthode développée ici demande dans un premier temps l'établissement d'une période de référence, permettant de définir la donnée typique générée par un processus en régime dit « normal ». Puisque nous optons pour l'utilisation de cartes de contrôle, nous nous focalisons ensuite sur plusieurs façons de définir des non-conformités dans la donnée de processus, ainsi qu'un moyen de les mesurer ou les compter. Ces définitions permettent ensuite leur mesure ainsi que leur représentation dans des cartes de contrôle aux mesures, des cartes de contrôle aux attributs, ainsi que des cartes de contrôle pondérant les observations passées afin de dégager les tendances plus longues et lentes de dérive conceptuelle. Chaque carte de contrôle fait partie de normes nationales et internationales, en particulier les normes AFNOR, qui seront citées au cours de ce chapitre. Nous commençons cela dit par rappeler les bases du contrôle de qualité par cartes de contrôle

7.2 Préliminaires sur les cartes de contrôle

7.2.1 Principes généraux

Une carte de contrôle permet l'étude d'un ou plusieurs caractères définissant la qualité d'une production. Cela peut être à partir de mesures comme l'épaisseur d'une pièce ou la résistance à la torsion d'un matériau, cela peut être un comptage de non-conformités comme le nombre d'éraflures sur de la peinture, ou encore un caractère binaire rendant une pièce conforme ou non, comme son étanchéité. Il existe des cartes de contrôle pour chaque cas de figure.
Dans une production, l'échantillonnage doit être effectué régulièrement et se justifie face au contrôle exhaustif par l'erreur des opérateurs, l'erreur des outils de mesure, la perte de temps occasionnée par l'échantillonnage, voire l'aspect destructif des contrôles. Par exemple, un test de résistance d'une pièce demande de la briser, il devient donc impératif de prendre un échantillon aussi petit que possible afin d'effectuer les contrôles dans ce cas.

Les caractères mesurés sur chaque échantillon sont ensuite reportés sur la carte associée, pourvue de deux axes :

— en abscisse, on a le rang de chaque échantillon, qui correspond à leur numéro dans l'ordre chronologique.

— En ordonnée, le caractère étudié (par exemple une moyenne par échantillon).

Chaque échantillon est donc représenté par un point sur une carte de contrôle. En plus des axes et des points, au moins trois lignes y sont ajoutées :

— une ligne centrale, notée **LC**, qui représente le plus souvent la valeur nominale du caractère étudié, c'est à dire sa valeur estimée lorsque le la production est jugée bien réglée.

— Une limite de contrôle inférieure, notée **LCI**, qui correspond à la limite inférieure pouvant être calculée sur un échantillon avant de donner une alerte quant à une dérive de la production.

— Une limite de contrôle supérieure, notée **LCS**, qui correspond à la limite supérieure pouvant être calculée sur un échantillon avant de donner une alerte quant à une dérive de la production.

Il est également possible d'y rajouter deux lignes intermédiaires, constituant des limites de surveillance inférieure et supérieure, notées **LSI** et **LSS** respectivement. Des points situés entre une limite de surveillance et la limite de contrôle correspondante indiquent qu'il faut porter une attention accrue à la production et son réglage.

7.2.2 Types de cartes de contrôle

Comme dit précédemment, il existe plusieurs types de cartes de contrôle, chacune propice à un type de contrôle. Ainsi, on commence par distinguer les cartes suivant notre approche vis-à-vis des échantillons : prise de décision à la vue du dernier échantillon prélevé, ou d'un ensemble d'échantillons successifs. On distingue ensuite les cartes en fonction du caractère contrôlé, noté X. X peut indiquer un caractère mesurable, ou bien la présence / non présence d'une propriété. On a donc, au cas par cas, pour les plus classiques :

À la vue du dernier échantillon prélevé :

1. Si X est un caractère mesurable, alors les cartes de contrôle dites « par mesures » seront utilisées, essentiellement de type Shewhart, détaillées dans la norme NF X 06-031-1 [AFN96a]. Ces cartes demandent de suivre deux paramètres de X :
   — un paramètre de centrage, par exemple la moyenne $\bar{X}$ de l'échantillon (on peut aussi prendre la médiane par exemple),
   — un paramètre de dispersion, par exemple l'écart-type S de l'échantillon. Précédemment à l'avènement de l'informatique, l'étendue W était utilisée. On peut aussi, conjointement à la médiane, utiliser le MAD (*Median Absolute Deviation*)

On obtient dans ce cas de cartes de contrôle : une carte de la moyenne de l'échantillon, notée « *xbar* », et une carte de l'écart-type de l'échantillon, notée « S ».

2. Si X est la présence ou non-présence d'une propriété, on adopte les cartes de contrôle par attributs, décrites dans la norme NF X 06-031-2 [AFN02]. En particulier, sont utilisées dans cette thèse le cartes :

   — « *np* », comptant le nombre d'unités considérées comme non conformes dans un échantillon,

   — « *u* », comptant le nombre de non-conformités sur une même unité.

À la vue d'un ensemble d'échantillons successifs :

1. Les cartes à moyenne mobile avec pondération exponentielle, dites EWMA (« *Exponentially Weighted Moving Average* »), qui sont décrites dans la norme NF X 06-031-3 [AFN96b].
2. Les cartes à somme cumulée, notées CUSUM (« *CUmulative SUM* »)), décrites dans la norme NF X 06-031-4[AFN96c].

7.2.3 Efficacité d'une carte de contrôle

Soit X et $X_1, X_2, \ldots, X_n$ des variables aléatoires indépendantes et identiquement distribuées (iid). On suppose que le point reporté sur la carte de contrôle est la moyenne $\bar{X} = \sum_{i=1}^{n} X_i$. Le déréglage de la production, appelé δ, est défini par :

$$\delta = \frac{\mathbb{E}_1[X] - \mathbb{E}_0[X]}{\sqrt{\mathbb{V}\mathrm{ar}_0(X)}},$$

avec $\mathbb{E}_1[X]$ l'espérance de X dans la situation de déréglage, et $\mathbb{E}_0[X]$ et $\mathbb{V}\mathrm{ar}_0(X)$ l'espérance et la variance de X en production bien réglée. L'efficacité d'une carte de contrôle réside dans sa capacité à détecter un déréglage, tout en évitant les fausses alertes. C'est ainsi que les risques de première et deuxième espèce interviennent :

— le risque de première espèce, α, d'annoncer à tort un déréglage.

— Le risque de deuxième espèce, β, de ne pas détecter un déréglage.

Partant de cela, il existe deux méthodes permettant d'évaluer l'efficacité des cartes de contrôle. La première est la courbe d'efficacité, qui donne la probabilité P_a d'acceptation d'un échantillon en fonction du déréglage, et les périodes opérationnelles moyennes, notées POM, définies comme le « nombre moyen d'échantillons successifs conduisant au premier point hors limites » d'après la norme NF X 06-031-3 [AFN96b].

Les cartes CUSUM et EWMA tiennent compte de plusieurs échantillons successifs, donc de plusieurs déréglages successifs. elles ne peuvent donc faire l'objet de courbes d'efficacité. Ainsi pour pouvoir comparer les cartes qui seront utilisées dans cette thèse, le calcul des POM est obligatoire. Quelques notations supplémentaires sont à introduire :

— la POM_0 est la période opérationnelle moyenne quand le déréglage vaut 0, c'est à dire en cas de processus bien réglé. En d'autres termes, il s'agit du nombre d'échantillons moyens avant de détecter un déréglage à tort en cas de production bien réglée.

— La POM_1 est la période opérationnelle moyenne quand le processus est déréglé avec un déréglage δ_1.

De façon assez pragmatique, on souhaite qu'une carte de contrôle possède une POM_0 aussi grande que possible afin d'éviter les fausses alarmes, et une POM_1 aussi petite que possible afin de détecter rapidement les déréglages. Les liens entre risques, courbes d'efficacité et périodes opérationnelles moyennes s'expriment ainsi pour une carte de Shewhart :

$$\alpha = 1 - P_a(\delta = 0), \quad POM_0 = \frac{1}{\alpha} = \frac{1}{1 - P_a(\delta = 0)},$$

$$\beta = P_a(\delta = \delta_1), \quad POM_1 = \frac{1}{1 - \beta} = \frac{1}{1 - P_a(\delta = \delta_1)}.$$

Ces liens permettent de comparer des cartes dont on possède les courbes d'efficacité à des cartes dont on possède les périodes opérationnelles moyennes.

Les préliminaires étant établis, la section suivante détaille notre méthode afin d'utiliser ces outils dans le contexte de la dérive conceptuelle dans la donnée de processus en distinguant les cas : celui de la dérive conceptuelle cantonnée aux données, et celui de la dérive conceptuelle relativement à notre modèle prédictif.

7.3 Dérive conceptuelle sans modèle prédictif

7.3.1 Période de comparaison et régime normal

Travaillant avec de la donnée, les notions d'échantillonnages destructifs ou de difficulté de l'échantillonnage ne se posent pas. La donnée devient périodiquement disponible et s'ajoute à l'historique, permettant la création d'un nouvel échantillon. De plus, l'enregistrement des données étant effectué informatiquement et les mesures étant en réalité des calculs effectués par des algorithmes directement sur la donnée, la fiabilité de l'équipement de mesure et l'erreur de l'opérateur peuvent être considérées comme négligeables – sous réserve d'un algorithme correctement codé.

Il faut cela dit scinder le journal d'événements en « *sous-logs* », qui constituent les échantillons. Soit r le nombre d'échantillons. Pour les cartes de Shewhart, il est recommandé

de posséder suffisamment d'observations réparties dans les échantillons de sorte que $\sum_{i=1}^{r} n_i \geqslant 100$, n_i la taille d'un échantillon i. Pour les cartes de contrôle d'attributs, il est plutôt conseillé d'avoir $\sum_{i=1}^{r} n_i \geqslant 300$. Cette limite inférieure étant largement inférieure à la taille des données disponibles, la question est plutôt de savoir quelle taille d'échantillons considérer, qui conditionnera le nombre de sous-logs en lesquels le journal d'événements sera subdivisé. Cette taille d'échantillon est à envisager en fonction des POM_0 et POM_1 préférées par les parties utilisant ces cartes de contrôle. *In fine*, cela revient également à définir la durée d'attente entre deux sous-logs, les journaux d'événements étant typiquement mis à jour de façon périodique dans Livejourney™.

De façon générale, chaque sous-log apporte de nouvelles traces, impliquant de nouvelles séquences d'activités. Nous pouvons alors compter combien de nouvelles traces chaque sous-log successif apporte aux sous-logs précédents.
Le premier sous-log ne comportera que des nouvelles traces, et chaque nouveau sous-log devrait en apporter de moins en moins, jusqu'à arriver à une stagnation du nombre de nouvelles traces par sous-log. une fois cette stagnation atteinte, les sous-logs antérieurs à cette stagnation constituent la période de comparaison, dans laquelle les traces considérées comme normales sont stockées. Les sous-logs dans la phase de stagnation constituent le régime normal du processus, apportant un nombre plus ou moins constant de nouvelles traces dans le journal d'événements.

La figure 7.1 a été créée en scindant chaque journal d'événements selon la règle suivante : la taille d'un sous-log en termes de parcours vaut $\max\left\{\left\lfloor\frac{|L|_\sigma}{100}\right\rceil, 50\right\}$. Ainsi, chaque journal d'événements est subdivisié en 100 sous-log, à moins que les sous-log ne possèdent des effectifs inférieurs à 50 parcours, auquel cas la taille des sous-logs est fixée à 50 parcours. Cela mène à des sous-logs comportant 50 parcours pour *Helpdesk* en moyenne, 100 pour *Traffic fines*, 85 pour *BPI2012 (w)*, 132 pour *BPI2012*, et enfin 278 pour *BPI2017*.
Nous voyons que chaque journal d'événements voit son nombre de nouvelles traces diminuer à l'apparition de nouveaux sous-logs, sans pour autant atteindre 0 de façon constante. Nous pouvons cependant se demander si cela n'est pas une indication que le processus génère simplement moins de traces, convergeant vers un parcours unique, à l'exception de quelques nouvelles traces. La figure 7.2 montre le nombre de traces contenues dans chaque sous-log, sans considération pour leur nouveauté.

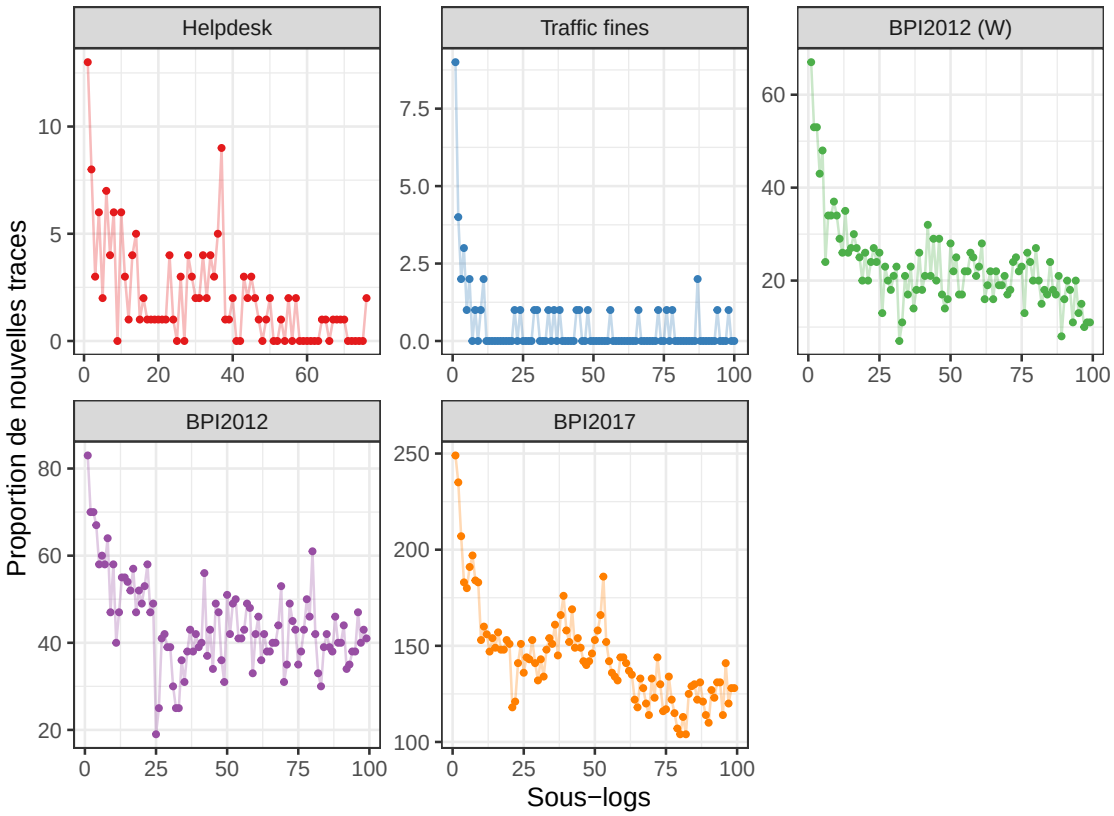

FIGURE 7.1 – Nombre de nouvelles traces apportées par chaque sous-log successif dans les jeux de données utilisés

TABLEAU 7.1 – Statistiques descriptives positionnelles et et de dispersion du nombre de traces par sous-log dans les journaux d'événements utilisés

Journal d'événements	**Moyenne**	**Écart-type**	**Médiane**	**MAD**
Helpdesk	10.78	3	10	2.97
Traffic fines	7.74	1.38	8	1.48
BPI2012 (W)	64.1	6.76	65	5.93
BPI2012	72.22	8.92	72	7.41
BPI2017	228.98	11.09	229	10.38

Nous sur la figure 7.2 que le nombre de traces présentes dans chaque sous-log est assez stable d'un sous-log à l'autre. Ceci indique que les processus ne semblent pas tendre vers la production d'une seule trace. Le tableau 7.1 nous donne par ailleurs des statistiques sur la position et les variations du nombre de traces dans les sous-logs de chaque journal d'événements.

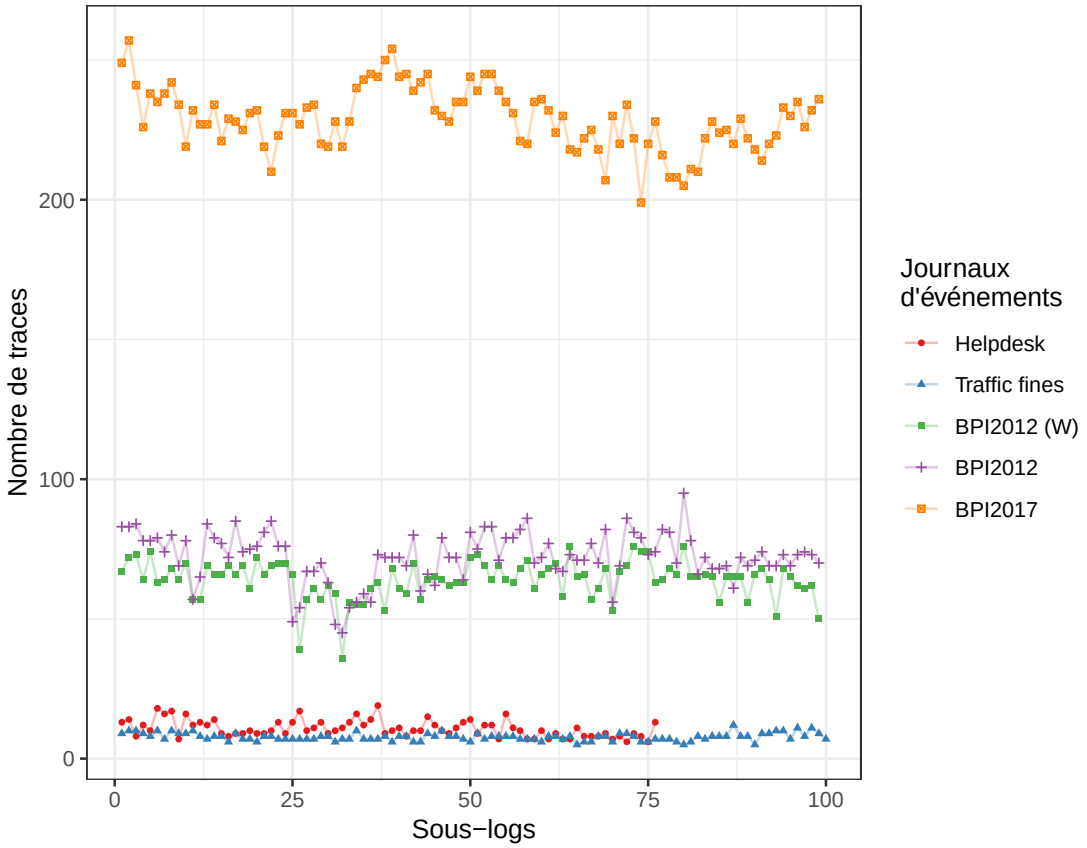

FIGURE 7.2 – Nombre de traces dans chaque sous-log successif dans les jeux de données utilisés

On y trouve la moyenne et l'écart-type, ainsi que la médiane et la médiane de la déviation en valeur absolue (*Median Absolute Deviation*, ou MAD) du nombre de traces dans leurs sous-logs.
Ainsi, nous observons que les sous-logs de *Helpdesk* possèdent en moyenne 10,78 traces pour un écart-type de 3, une médiane de 10 traces pour un MAD de 2,97. Il s'agit d'une distribution qui semble assez symétrique et concentrée autour de la moyenne. Ce constat peut être fait pour chacun des journaux d'événements.

Plus intéressant encore : si de nouvelles traces ne font que s'ajouter, et que le nombre de traces de chaque sous-log gravite autour de la moyenne, cela peut impliquer que les anciennes traces disparaissent au fil du temps, pour être remplacées par les nouvelles. On a ici une première preuve de la dérive conceptuelle à l'œuvre dans les journaux d'événements. Reste à pouvoir la quantifier, la suivre, et la qualifier.

Reprenons la figure 7.1 afin de déterminer les périodes de comparaison ainsi que les régimes normaux. Les coudes dans ces comptages peuvent être considérés comme suit :

Helpdesk : la tendance n'est pas claire mais nous pouvons considérer que le régime stable est atteint à partir du sous-log 40, la période de comparaison est donc constituée des sous-logs de 1 à 40, et le rédime normal des sous-logs 40 à 76.

Traffic fines : Le régime stable de *Traffic fines* semble être atteint dès le 10ème sous-log. La période de comparaison concerne donc les 10 premiers sous-logs, le régime normal les 90 suivants.

BPI2012 (W) : On a une décroissance nette jusqu'au sous-log 30 est observée, suivie d'une croissance soudaine qui se stabilise entre les sous-logs 40 et 80, pour finir avec une décroissance dans les sous-logs 80 à 100. Ainsi, la période de comparaison est constituée des sous-logs 1 à 40, le régime normal des sous-logs 40 à 80, et les points restants peuvent constituer de nouvelles observations.

BPI2012 : Ce journal d'événements est plus net. La période de référence est typiquement à considérer du sous-log 1 jusqu'au 35^{e}, et le régime normal est constitué des 65 sous-logs suivants.

BPI2017 : Celui-ci est moins direct. On a une décroissance suivie d'un plateau à partir du sous-log 13, jusqu'au 50^{e} environ. Une autre décroissance est ensuite visible du 53^{e} jusqu'au 65^{e} environ, pour finir par stagner jusqu'à la fin. Il y aurait donc 2 coudes dans ces comptages. Nous prendrons le dernier, le premier coude semble d'ailleurs correspondre au premier régime de peuplement total dans la figure 5.4. Nous prendrons donc le dernier régime stable observé.

Avec ces périodes définies, nous supposons, finalement, que la période de comparaison correspond à l'état initial du processus. Ensuite, la période stable correspond à la phase où les anciennes traces se font peu à peu remplacer par les nouvelles.
Les cartes de contrôle permettent de déterminer à partir de quel instant l'écart entre le passé et le présent d'un processus se creuse de façon statistiquement significative.

7.3.2 Méthode pour les activités

Considérations importantes

Considérons un processus enregistré dans un journal d'événements L, comportant $|L|_e$ événements, $|L|_\sigma$ parcours et $|L|_T$ traces. Une première piste de réflexion concerne ce que nous pouvons considérer comme une nouvelle trace dans la donnée. En effet, imaginons un parcours dont la trace serait la séquence d'activités $\langle 1, 2, 3\rangle$. Imaginons ensuite un nouveau parcours ayant la séquence d'activités $\langle 1, 2, 1, 2, 1, 2, 1, 2, 3\rangle$. Ces deux séquences sont différentes. Malgré tout, la deuxième séquence ne montre pas de déviation flagrante par rapport au comportement de la première, excepté la répétition de la sous-séquence $1, 2$ (et donc la transition de l'activité 2 vers l'activité 1).
Considérons également des jeux de données comme *BPI2012 (W)*. Celui-ci ne comporte que 6 activités, mais les répétitions donnent lieu à un grand nombre de traces par rapport

au nombre de parcours. Dans ce cas de figure, sachant que la trace la plus longue comporte 74 événements, un nouveau parcours comportant une seule répétition supplémentaire déjà observée serait considérée comme une nouvelle trace, tout comme $\langle 1, 2, 1, 2, 1, 2, 3\rangle$ ou $\langle 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 3\rangle$ pour reprendre l'exemple précédent. Il semble donc judicieux de comparer les traces principalement une fois élaguées des répétitions trouvées dans la période de comparaison. Conserver les répétitions non observées dans cette période revient à comptabiliser toute occurrence d'une nouvelle séquence d'activités, même répétée.

Il est par ailleurs possible de considérer ces répétitions élaguées comme un caractère à part à étudier : il peut être intéressant de suivre l'évolution de ces répétitions.

Tout comme dans la section 2.2.3, les répétitions maximales sont considérées pour cette tâche.

Une autre considération sur les répétitions concerne l'ordre dans lequel elles sont retirées. Imaginons que l'on retire les répétitions par taille croissante : il est possible qu'une répétition de grande taille en contienne plusieurs de petites tailles, toutes considérées comme maximales. Or retirer les petites répétitions laisserait une sous-séquence criblée de « trous », donnant une sous-séquence qui ne constituerait plus une répétition en soi. Il resterait alors des artefacts de ces grands répétitions dans la donnée. Il semble donc plus judicieux de retirer les répétitions dans l'ordre décroissant de leur taille, afin de retirer les plus grandes sous-séquences répétées d'un bloc et d'affiner l'élagage au fur et à mesure.

Par ailleurs, une activité seule peut constituer une répétition maximale. Il suffit qu'une seule trace comporte une seule répétition maximale de longueur unitaire pour que l'activité correspondante soit retirée de toutes les traces. Dans le cas où chaque activité constituerait au moins une fois une répétition maximale, on finirait avec un journal d'événements vide une fois élagué. Il est donc important de ne considérer que les répétitions de taille $\geqslant 2$.

En reprenant l'exemple ci-dessus, la trace $\langle 1, 2, 1, 2, 1, 2, 1, 2, 3\rangle$ peut être élaguée de la répétition maximale $\langle 1, 2, 1, 2\rangle$, de $\langle 1, 2, 1\rangle$ et de $\langle 1, 2\rangle$. En les enlevant dans l'ordre décroissant des tailles, il ne reste plus que $\langle 3\rangle$. De même, la trace $\langle 1, 2, 3\rangle$ se fait élaguer de la répétition $\langle 1, 2\rangle$ puisque celle-ci est présente dans $\langle 1, 2, 1, 2, 1, 2, 1, 2, 3\rangle$. Il reste donc également $\langle 3\rangle$, et l'objectif est rempli : les répétitions maximales retirées, ces traces sont identiques.

Mesure de la dérive des activités

De toutes les traces dans la période de comparaison, on fouille et stocke les répétitions maximales. Ces répétitions sont ensuite retirées de toutes les traces, toutes périodes confondues, y compris des traces futures. Ainsi, les traces sont réduites à leur « squelette ». Les seules répétitions restantes sont celles n'étant pas observées dans la période de comparaison.

Ensuite, nous pouvons imaginer que chaque activité apparaissant à un endroit atypique dans ces traces élaguées pourrait constituer une non-conformité.
Nous possédons déjà les traces élaguées de la période de comparaison et de la période stable.

Une bonne façon de les comparer est de calculer leurs distances de Damerau-Levenshtein par paires, et pour chaque trace de la période stable, retenir la distance minimum. Ainsi, soit T_C l'ensemble des traces de la période de comparaison, et soit k l'indice du premier parcours hors de la période de comparaison. Pour pour un parcours σ_i avec $k \leqslant i \leqslant |L|_\sigma$, nous calculons :

$$\min_{\zeta \in T_C} d_{\mathrm{DL}}(\sigma_i, \zeta).$$

Puisque cette distance compte les éditions (suppressions, insertions, échanges) pour passer d'une séquence de symboles à une autre, retenir la distance minimum pour un parcours dans la période stable revient à trouver son homologue le plus proche parmi les traces de la période de comparaison, et compter le nombre de points de divergences entre les deux. On peut considérer ces points de divergences comme des non-conformités par rapport aux traces de la période de comparaison. Moyenné sur chaque sous-log, ce nombre de non-conformités par unité sert ensuite à calculer les valeurs de référence de la future carte de contrôle, à savoir sa LC, sa LCI et sa LCS.

Dans notre cas, il s'agit d'un comptage de non-conformités par unités. Il convient donc d'utiliser une carte u.

Le choix de la carte u, par rapport à la carte c qui comptabilise le nombre de non-conformités par échantillon, se justifie par le fait que la carte u autorise les tailles d'échantillons à varier. Présentons maintenant les principes fondamentaux des cartes u.

Soit $c_0(n)$ (venant des cartes c) le nombre de non-conformités moyen par échantillon de taille n, établi dans une période de référence. L'indice 0 indique la période de référence. Le nombre de non-conformités par échantillon de taille n_j, suit une loi de poisson paramétrée par $c_0(n_j)$. On note $X_j \sim \mathcal{P}(c_0(n_j))$. On regarde le nombre de non-conformités par unité, noté $U_j = \dfrac{X_j}{n_j}$. Pour cette variable aléatoire, on a pour l'espérance :

$$\mathbb{E}_0[U_j] = \mathbb{E}_0\left[\frac{X_j}{n_j}\right] = \frac{\mathbb{E}_0[X_j]}{n_j} = \frac{c_0(n_j)}{n_j} = u_0.$$

De même, pour la variance :

$$\mathbb{V}\mathrm{ar}(U_j) = \mathbb{V}\mathrm{ar}\left(\frac{X_j}{n_j}\right) = \frac{\mathbb{V}\mathrm{ar}(X_j)}{n_j^2} = \frac{c_0(n_j)}{n_j^2} \iff \sigma_{U_j,0} = \sqrt{\frac{c_0(n_j)}{n_j^2}} = \sqrt{\frac{u_0}{n_j}}.$$

La ligne centrale est $LC = u_0$, et on a pour limites de contrôle :

$$LCI = u_0 - 3\sqrt{\frac{u_0}{n_j}} \quad \text{et} \quad LCS = u_0 + 3\sqrt{\frac{u_0}{n_j}},$$

$$LSI = u_0 - 2\sqrt{\frac{u_0}{n_j}} \quad \text{et} \quad LSS = u_0 + 2\sqrt{\frac{u_0}{n_j}}.$$

Dans la pratique, l'estimation de u_0 se fait en comptant le nombre de non-conformités dans les échantillons de la période de référence (ici notre période stable). Soit $r \in \mathbb{N}^\star$ le nombre d'échantillons (ici sous-logs), et $i \in \{1, \ldots, r\}$ un échantillon de taille n_i. On note c_i le nombre de non-conformités comptées dans l'échantillon. Alors :

$$u_0 = \frac{\sum_{i=1}^{r} c_i}{\sum_{i=1}^{r} n_i}.$$

Ces définitions et propriétés, tirées de la norme NF X 06-031-2 [AFN02], permettent l'établissement de cartes u pour nos sous-logs. Il est à noter que malgré la possibilité d'avoir des tailles d'échantillons variables, il est recommandé de contraindre cette variabilité entre tailles d'échantillons à un maximum de 25%.
Par ailleurs, bien que les cartes de contrôle d'attributs possèdent des limites de surveillance et de contrôle inférieures, celles-ci ne constituent pas une alerte en soi. En effet, un franchissement des limites inférieures indique en fait une amélioration notable de la production (ici de la dérive conceptuelle), puisque le nombre de non-conformités est bien en-dessous de la moyenne de référence. Cela dit, il peut être intéressant, lorsque c'est possible, d'explorer les raisons de telles améliorations.

En appliquant ces principes à nos distances minimales de Damerau-Levenshtein entre parcours de la période stable, et les traces de la période de comparaison, toutes élaguées des répétitions observées pendant la période de comparaison, nous obtenons la figure 7.3. Celle-ci représente les cartes u correspondant à chaque journal d'événements, composée des points issus de la période de référence. *Helpdesk* montre une majorité de points à 0, indiquant que les nouvelles traces observées en figure 7.2 ne sont pas, en général, consituées de nouvelles connexions entre activités ou nouvelles répétitions. On observe une montée soudaine en fin de période. Une observation assez similaire peut être faite pour *Traffic fines*, et *BPI2012 (W)* ne montre aucune violation. *BPI2012* et *BPI2017*, cependant, montrent un bon exemple de dérive conceptuelle graduelle telle qu'exposée dans la section précédente : le nombre de traces par sous-log est stable, mais de nouvelles traces remplacent les anciennes. On observe effectivement dans les deux cas une augmentation graduelle du nombre de non-conformités par unité, jusqu'à atteindre des violations.

Au niveau de l'efficacité de la carte u, nous pouvons calculer les courbes d'efficacité, qui donnent la probabilité d'acceptation de la production en fonction de la proportion réelle de non-conformités par unité. Dans une carte u, le déréglage δ vaut $\frac{u - u_0}{\sqrt{u_0}}$. Avec cette carte de contrôle, la limite acceptable pour les non-conformités est la LCS. On considère donc le déréglage $\delta_1 = \frac{LCS - u_0}{\sqrt{u_0}}$. Dans un échantillon de taille n, le nombre de non-conformités par échantillon nécessaire pour dépasser la LCS est :

$$k_1 = E(nLCS) + 1,$$

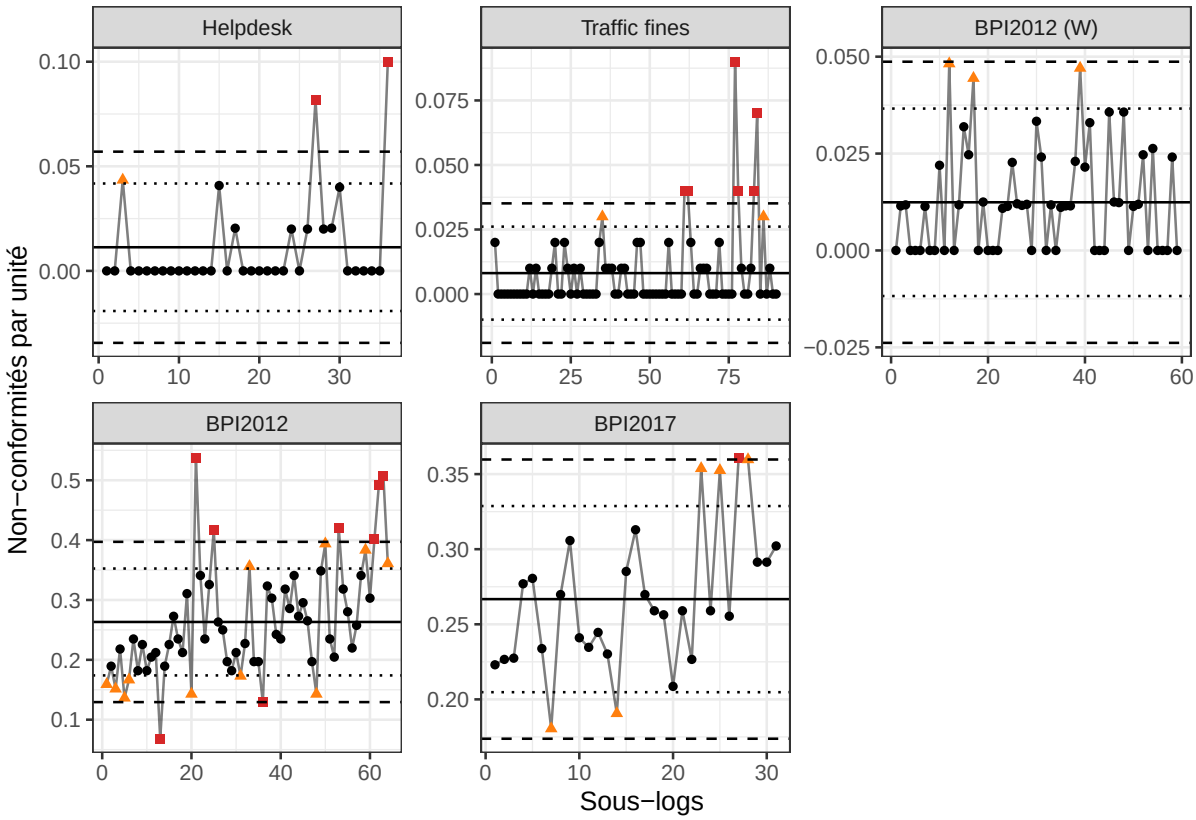

FIGURE 7.3 – Cartes de contrôle u pour le contrôle du nombre de non-conformités dans les sous-logs des périodes stables dans les journaux d'événements utilisés

où $E(\cdot)$ est la fonction partie entière. Bien que les échantillons puissent être de taille variable, limiter leur variabilité à un maximum de 25% permet entre autres de remplacer n par leur moyenne $\frac{1}{n}\sum_{i=1}^{r} n_i$.

De k_1, on déduit que le nombre maximal de non-conformités acceptable dans un échantillon est $k_1 - 1$. Ainsi, la courbe d'efficacité, définie par la probabilité d'acceptation P_a, vaut :

$$P_a(n,u) = \sum_{k=0}^{k_1-1} e^{-nu}\frac{(nu)^k}{k!},$$

où u correspond au nombre de non-conformités par unité. Nous voyons que la courbe d'efficacité dépend à la fois de n et de k_1. On peut donc calculer P_a lorsque le processus étudié n'a pas dérivé, et lorsqu'un déréglage est présent. On peut ensuite calculer la POM_0 et la POM_1 correspondantes.

TABLEAU 7.2 – Efficacité des cartes u pour la dérive conceptuelle sans modèle prédictif

Journal d'évts	$\mathbf{k_1}$	δ_1	$\mathbf{P_a}(\delta = \delta_0)$	$\mathbf{P_a}(\delta = \delta_1)$	α	β	$\mathbf{POM_0}$	$\mathbf{POM_1}$
Helpdesk	3	4.05	0.981	0.47	0.019	0.47	53.24	1.89
Traffic fines	4	3.33	0.99	0.53	0.01	0.53	105.13	2.14
BPI2012 (W)	5	2.92	0.995	0.60	0.005	0.60	217.59	2.51
BPI2012	53	0.51	0.998	0.51	0.002	0.51	422.23	2.06
BPI2017	101	0.35	0.998	0.53	0.002	0.53	569.01	2.11

Le tableau 7.2 fait l'inventaire de l'efficacité des cartes de contrôle visibles en figure 7.3. On y observe k_1 le nombre de non-conformités nécessaires dans un sous-log pour dépasser la LCS, qui dépend de la taille des sous-logs. On a le déréglage δ_1 associé à la LCS, suivi de la probabilité d'acceptation d'un sous-log sans dérive et avec dérive correspondant à la LCS. Les risque α et β associés sont calculés, et la POM_0 et POM_1 sont données. Typiquement, ce que nous déduisons de ce tableau, est que les cartes de contrôle créées avec la méthode par défaut décrite ci-avant de tailles de sous-logs sont conservatives en termes d'alertes sur la dérive. On a une probabilité d'acceptation très élevée en cas d'absence de dérive, très proche de 1, mais une probabilité d'acceptation autour de 0,5 en cas de dérive correspondant à la LCS. Cela donne des POM_0 élevées, indiquant qu'il faut un nombre conséquent de sous-logs successifs sans dérive avant d'arriver à une fausse alerte. Au contraire, il faut en moyenne 2 à 3 échantillons déréglés pour détecter un déréglage correspondant à la LCS. Cela peut ne pas être problématique, mais dépend assez largement du temps que prendront les futurs sous-logs à être disponibles : rater un déréglage sur 2 sous-logs peut être moins grave si ceux-ci sont disponibles tous les jours, contrairement à une disponibilité hebdomadaire ou mensuelle.
Ces cartes sont malgré tout fonctionnelles, analysables, et surtout ajustables afin d'ajuster les risques α et β (et donc les POM_0 et POM_1) en fonction des attentes sur le terrain.

Les cartes de contrôle de la figure 7.3 ont été calculées, mais une librairie R permet d'en produire de toutes sortes avec aisance : la librairie `qcc` [Scr04]. Celle-ci est utilisée pour le restant de cette section, par commodité.
Entre autres, elle rend aisée la création de cartes EWMA et CUSUM. Comme exposé en section 7.2.2, ces cartes prennent en compte les observations passées et permettent un suivi sur des tendances de plus longue durée. Ceci peut être important afin de déterminer si une violation sur une carte u, par exemple, correspond à un « accident » ponctuel ou est plutôt symptomatique d'une dérive étalée dans le temps.

Dans la suite, les cartes EWMA seront favorisées aux cartes CUSUM. L'outil développé dans ce chapitre n'est pas nécessairement adressé à des opérateurs et utilisateurs familiers avec la maîtrise statistique des procédés, et la carte EWMA est plus facilement compréhensible que la carte CUSUM, cette dernière pouvant afficher deux points pour un même échantillon. De plus, les auteurs de l'article [HW14] montrent que les cartes EWMA

peuvent être plus efficaces que les cartes CUSUM dans le cas d'une dérive plus légère que prévu. Ceci est un avantage important étant donné le risque β observé dans les cartes u précédentes.

Les cartes EWMA font l'objet de la norme NF X 06-031-3 [AFN96b], les propriétés exposées ici en sont tirées. Une carte EWMA étudie un caractère supposé gaussien à variance constante. De façon plus précise, il est supposé que la moyenne est constante, avant de se dérégler et redevenir constante dans ce déréglage. Cette dernière hypothèse est peu réaliste dans un processus industriel, et l'est encore moins dans notre cas, mais l'outil reste commode pour son utilisation de « support » à la carte u.
On définit les variables aléatoires X, $X_1, \ldots, X_n \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma)$. Une carte EWMA évalue la variable aléatoire Z_i :

$$Z_i = \lambda \sum_{j=0}^{i-1} (1-\lambda)^j \bar{X}_{i-j} + (1-\lambda)^i m_0,$$

avec λ un paramètre de lissage, $Z_0 = m_0$ la moyenne en période de bon réglage, et $\bar{X}_{i-j}$ la moyenne du caractère dans l'échantillon $i-j$. Plus λ est proche de 0, plus les observations passées comptent dans le calcul du dernier échantillon. Les limites de contrôle valent :

$$LCI = m_0 - L\sqrt{\frac{\lambda}{2-\lambda}}\frac{\sigma}{\sqrt{n}}, \quad LCS = m_0 - L\sqrt{\frac{\lambda}{2-\lambda}}\frac{\sigma}{\sqrt{n}}.$$

Les paramètres λ et L peuvent être choisis selon la POM_0 et la POM_1 désirées, en fonction du déréglage maximal autorisé δ_1. La norme NFX 06-031-3 fournit des tables permettant un tel choix. Typiquement, avec une POM_0 de 370 et une POM_1 de $28,7$, pour un déréglage $\delta_1\sqrt{n} = 0,5$, on a $\lambda = 0,05$ et $L = 2,62$.

Il est également possible, grâce à ces tables, de déterminer la taille de l'échantillon nécessaire afin d'obtenir une POM_0 et une POM_1 particulières en fonction du déréglage maximal autorisé. Par exemple, si le déréglage maximal autorisé est $\delta_1 = 5$, que l'on souhaite une POM_0 de 370 et une POM_1 de 28,7, la taille d'échantillon nécessaire sera $n = \left(\frac{5}{0,5}\right)^2 = 100$.

Bien que le caractère étudié dans notre cas ne soit pas gaussien, mais un comptage, il est possible de l'approximer par une loi normale, la taille des échantillons et des comptages étant assez élevée pour le permettre.
Soit c le nombre moyen de non-conformités moyen dans un sous-log, c_0 quand il n'y a aucune dérive. Le nombre de non-conformités de l'échantillon suit une loi proche d'une $\mathcal{N}(c, \sqrt{c_0})$. On a donc les limites de contrôle :

$$LCI = c_0 - L\sqrt{\frac{\lambda}{2-\lambda}}\sqrt{c_0}, \quad LCS = c_0 - L\sqrt{\frac{\lambda}{2-\lambda}}\sqrt{c_0}.$$

Pour illustrer l'utilisation des cartes EWMA sur de grandes échelles de temps (mois, années), deux cartes générées avec la librairie `qcc` sur les non-conformités calculées dans *Traffic fines* et *BPI012 (W)* sont fournies ici, avec un paramètre $\lambda = 0,05$, adéquat pour la détection de tendances à long terme. On a sur la figure 7.4 la carte EWMA pour les non-conformités dans *Traffic fines* (graphique du haut) et *BPI2012 (W)* (graphique du bas). Sans surprise, *BPI2012 (W)* ne montre pas de tendance claire y compris à long terme, les cartes u et EWMA ne laissent pas penser à une dérive. En revanche, *Traffic fines* montre une augmentation du nombre de non-conformités en dents de scie jusqu'à dépasser la LCS de la carte EWMA, concomitamment aux dernières violations observées dans la carte u correspondante.

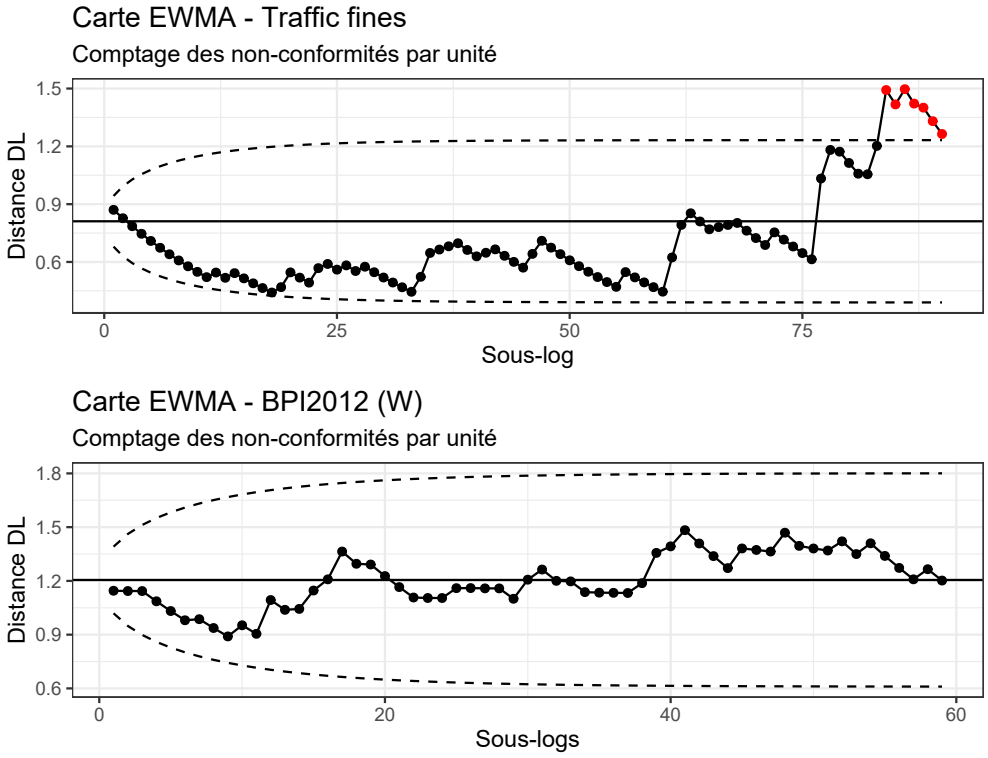

FIGURE 7.4 – Cartes EWMA pour les non-conformités dans *Traffic fines* et *BPI2012*(W) sans modèle prédictif

7.3.3 Mesure de la dérive des durées

La partie la plus complexe concernant l'établissement de la méthodologie étant établie, la dérive des durées peut à présent facilement être suivie. Une première approche réside dans la mesure des durées totales : nous pouvons, dans chaque sous-log, calculer la durée totale moyenne des parcours et suivre son évolution au cours du temps.
Ce caractère est quantitatif, il convient donc d'utiliser des cartes de Shewhart de la moyenne, décrites dans la norme NF X 06-031-1 [AFN96a]. Ces cartes permettent de suivre des

moyennes de mesures, et sont généralement accompagnées de cartes S afin de vérifier la stabilité de l'écart-type des valeurs mesurées. Par ailleurs, y ajouter une carte EWMA permet de contrôler les dérives de ces durées sur des tendances à plus long terme.

Une méthode bien plus simple que pour les activités s'offre à nous : la période de comparaison, correspondant aux sous-logs avant le coude observé dans la figure 7.1, peut maintenant servir de période de références pour le calcul des limites et de la ligne centrale d'une carte de contrôle. Les sous-logs présents dans cette période sont simplement utilisés pour calculer, chacun la moyenne des durées totales des parcours qu'il contient, formant ensuite les carte de contrôle. Les parcours dans la période de référence peuvent être utilisés comme nouvelle donnée à placer sur les cartes.

Ainsi, pour un parcours donné $\sigma = \langle e_1, \ldots, e_k \rangle$, nous calculons simplement sa durée totale $\Delta_\sigma = \pi_\mathcal{T}(e_k) - \pi_\mathcal{T}(e_1)$. Dans le cas de deux horodatages, nous calculons $\Delta_\sigma = \pi'_\mathcal{T}(e_k) - \pi_\mathcal{T}(e_1)$.

Le calcul de la ligne centrale, c'est à dire de la moyenne de référence, notée μ_0 et estimée par m_0, se fait simplement en calculant la moyenne des moyennes des sous-logs dans la période de comparaison. Donc, si cette période est constituée de r sous-logs, on a :

$$m_0 = \sum_{i=1}^{r} \overline{\Delta_i},$$

où $\overline{\Delta_i}$ est la moyenne des durées totales dans le sous-log i. Pour une estimation de l'écart-type de référence σ_0 estimé par s_0, avec les mêmes r sous-logs de taille n dans la période de comparaison, on a :

$$s_0^2 = \frac{1}{r} \sum_{i=1}^{r} s_i^2, \quad \text{avec } s_i^2 = \frac{1}{n-1} \sum_{j=1}^{J} (\Delta_{i,j} - \overline{\Delta_i})^2.$$

Les limites de contrôle valent, d'après la norme NF X 06-031-1 :

$$LCI = \mu_0 - \Phi^{-1}_{\mathcal{N}(0,1)} \left(1 - \frac{\alpha}{2}\right) \frac{\sigma_0}{\sqrt{n}} \quad \text{et} \quad LCI = \mu_0 + \Phi^{-1}_{\mathcal{N}(0,1)} \left(1 - \frac{\alpha}{2}\right) \frac{\sigma_0}{\sqrt{n}},$$

avec α le risque de première espèce choisi et $\Phi^{-1}_{\mathcal{N}(0,1)}(\cdot)$ la fonction quantile d'une loi $\mathcal{N}(0,1)$. Typiquement, pour les limites de contrôle, ce risque est choisi à $\alpha = 0,0027$. Ainsi, le quantile de loi normale centrée réduite à $1 - \frac{\alpha}{2}$ vaut ~ 3. Dans le cas de limites de surveillance, on a $\alpha = 0,0455$, le quantile vaut donc ~ 2.

La figure 7.5, graphique du haut montre une carte de contrôle de Shewhart de la moyenne calculée à partir de la période de comparaison de *BPI2012 (W)*. La ligne centrale est à 11 jours. Les points placés avant la ligne verticale en pointillés correspondent aux sous-logs de cette période de comparaison, tandis que les suivants sont dans la période

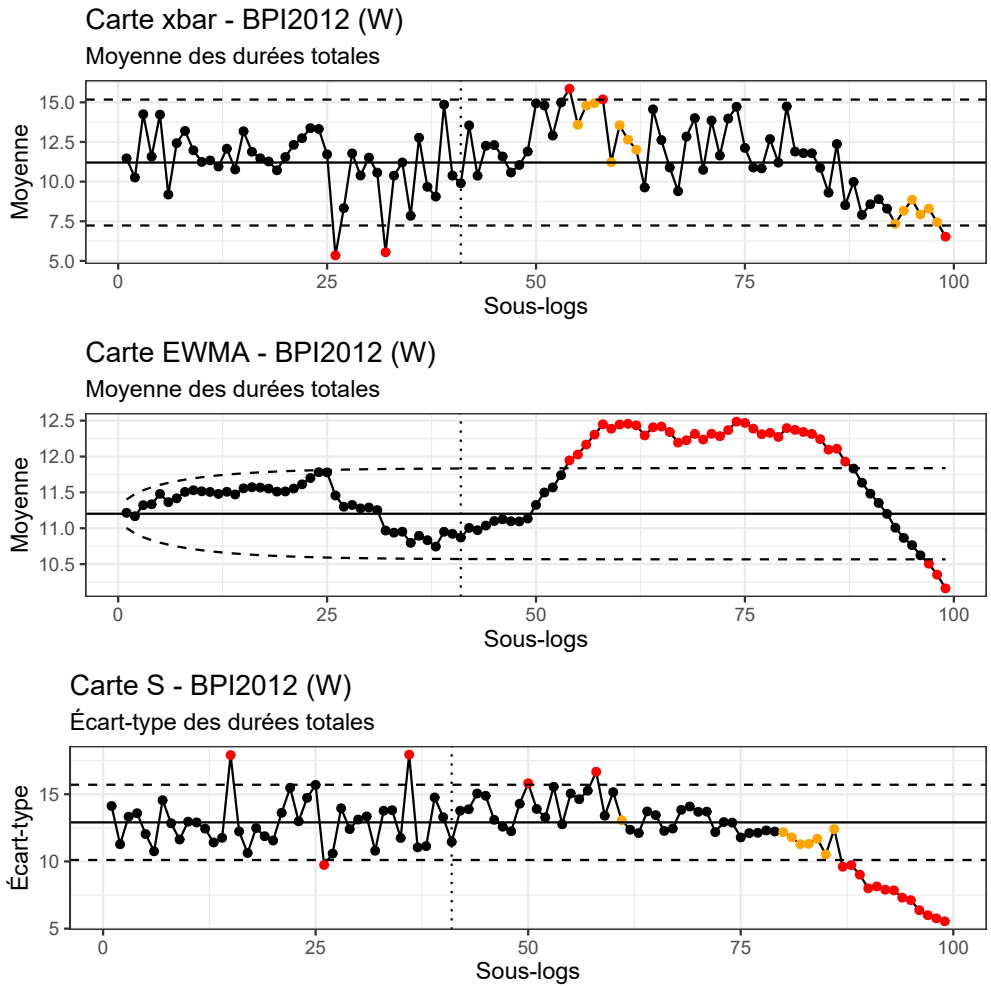

FIGURE 7.5 – Cartes *xbar*, EWMA et *S* pour les durées totales dans *BPI2012 (W)* sans modèle prédictif

de référence. Sur les cartes de contrôle générées par `qcc`, les points deviennent deviennent orange si plus de 7 points successifs sont du même côté de la limite centrale. Une décroissance nette s'observe à partir du sous-log 80. On observe sur les cartes *xbar* et *S* des points en orange. Il s'agit de points violant une règle de séquence : si plus de 7 points consécutifs se trouvent du même côté de la ligne centrale, ils demandent surveillance. Cette règle est appliquée dans toute la suite du manuscrit.

Une carte EWMA de ces durées moyennes peut être intéressante. C'est l'objet de la figure 7.5, graphique du milieu, avec $\lambda = 0,05$. Ici, nous observons une période entre les échantillons 50 et 89 entièrement hors limite, dépassant la LCS. Sur la carte *xbar*, seuls 2 points sont hors limites dans cette période. Il semble donc que la tendance haussière sur les durées totales soit en réalité une tendance à long terme. La même diminution drastique des temps totaux s'observe sur les derniers sous-logs.

Puisque les cartes EWMA nécessitent l'hypothèse d'un écart-type constant, l'établissement d'une carte de l'écart-type s'impose. Dans ce cas, la norme NFX 06-031-1 propose les limites de contrôle suivantes :

$$LCI = \frac{\sigma_0}{\sqrt{n-1}}\sqrt{F^{-1}_{\chi^2_{n-1}}\left(\frac{\alpha}{2}\right)} \quad \text{et} \quad LCS = \frac{\sigma_0}{\sqrt{n-1}}\sqrt{F^{-1}_{\chi^2_{n-1}}\left(1-\frac{\alpha}{2}\right)},$$

avec $F^{-1}_{\chi^2_{n-1}}(\cdot)$ la fonction quantile d'une loi χ^2_{n-1}. En général, $\alpha = 0,0027$ dans le cas des limites de contrôle et $0,0455$ dans le cas des limites de surveillance.

Ainsi, la figure 7.5, graphique du bas, nous permet de voir la carte S pour les durées totales dans *BPI2012 (W)*. On observe une certaine stabilité malgré quelques violations, suivie d'une baisse drastique sur les derniers sous-logs. L'écart-type ne pouvant être jugé constant sur cette période, la carte EWMA ne peut pas être correctement interprétée sur ces derniers sous-logs. Il serait intéressant, de plus, de chercher la raison de cette baisse de l'écart-type, chose considérée comme positive puisque les durées totales varient de moins en moins.

Une approche intéressante pour compléter cette analyse consiste en l'évaluation du rapport de la durée totale sur le nombre d'événements dans un parcours. cela revient à contrôler le temps moyen par événement dans les parcours.

La figure 7.6, graphique du haut, montre une carte *xbar* aux variations similaires à la carte *xbar* présente en figure 7.5. Cela dit, les derniers sous-logs ne présentent pas de baisse et semblent au contraire se stabiliser autour de la ligne centrale.

La figure 7.6, graphique du bas, montre la carte S des ratios. Un certain nombre de violations sont visibles, mais ils semblent également se stabiliser autour de la ligne centrale sur les derniers sous-logs. Cela semble indiquer la dérive suivante : les événements ne durent pas moins longtemps, il y a simplement moins d'événements dans les parcours observés dans les derniers sous-logs, expliquant la baisse des durées totales mais la stabilité du ratio durée totale / nombre d'événements.

Cet exemple, choisi sciemment, montre l'importance de différents caractères à suivre conjointement afin d'expliquer avec exhaustivité les dérives conceptuelles possibles dans la donnée. Si le ratio seul avait été considéré pour le suivi, la stabilité affichée par les cartes de contrôle n'aurait mené à aucune conclusion autre que cette stabilité.

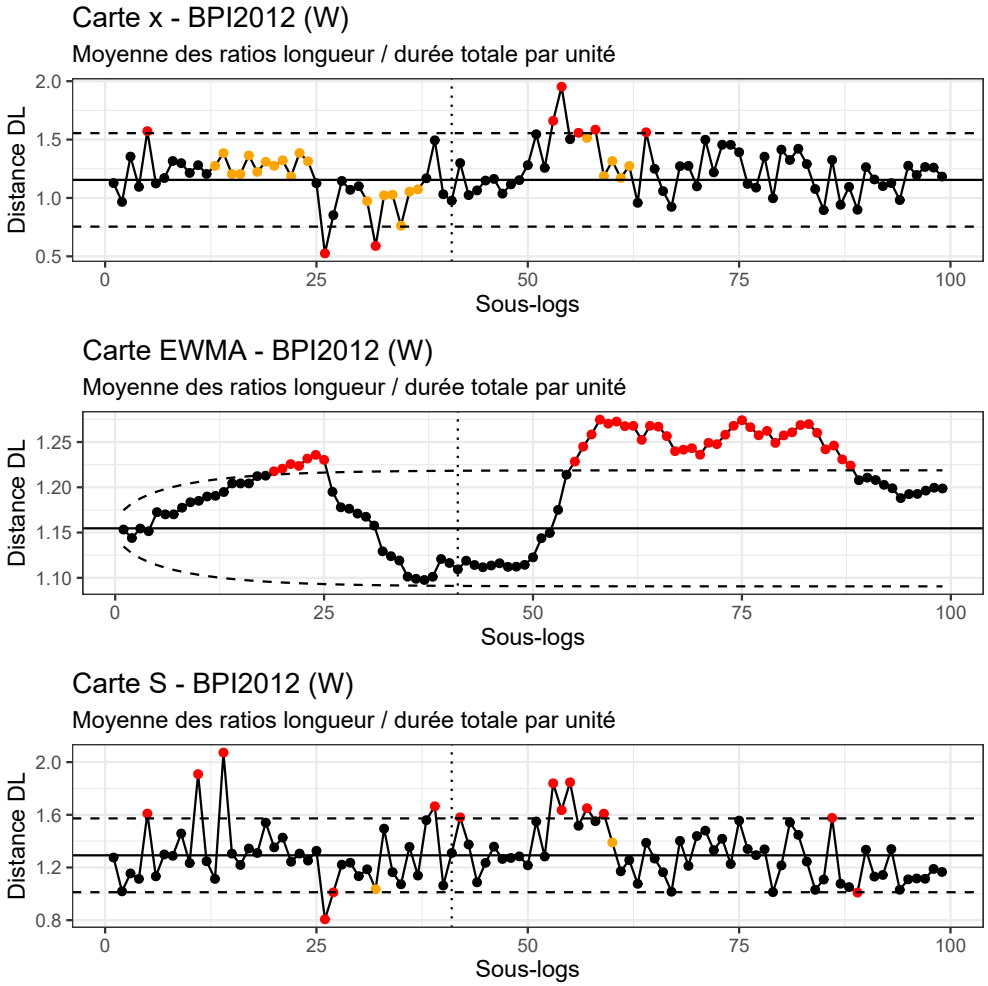

FIGURE 7.6 – Cartes *xbar*, EWMA et S pour le ratio durées totales / nombre d'événements dans *BPI2012 (W)* sans modèle prédictif

7.3.4 Remarques

Une première remarque porte sur la différence de traitement entre les activités et les durées. Le contrôle de la dérive conceptuelle dans les activités demande une base de comparaison, ne permettant donc pas les premiers sous-logs d'être utilisés pour construire les limites et ligne centrale des cartes de contrôle. Par ailleurs, un détail est à prendre en compte : la période qualifiée « de stabilité » peut ne pas être stable dans le sens des transitions entre activités : si de nouvelles traces apparaissent et remplacent les anciennes, une dérive constante est en action dans le processus contrôlé. La période de référence, utilisée pour construire les cartes de contrôle, est donc considérée comme la période de référence malgré la dérive déjà à l'œuvre dans les données.
Les situations rencontrées ne sont, cela dit, jamais aussi tranchées : d'anciennes traces peuvent apparaître à nouveau plus loin dans le temps, et une proportion souvent non-

négligeable de nouvelles traces ne sont que ponctuelles, constituant des apparitions uniques, atypiques. La méthode est donc, en soi, un compromis entre la dérive constante et la réalité du terrain.

7.4 Dérive conceptuelle relative au modèle prédictif

7.4.1 Principe

Il s'agit maintenant d'évaluer la dérive conceptuelle en présence d'un modèle prédictif, en l'occurrence le WGAN conditionnel développé dans cette thèse. Il est effectivement possible que le modèle ait appris une certaine tendance présente dans la donnée indiquant une certaine dérive, et que sa généralisation inclue cette dérive-ci. Il y aura malgré tout des dérives, déjà présentes et futures, qui échapperont au modèle. Cette section développe une méthode simple et intuitive permettant la détection et le suivi d'une telle dérive relative à un modèle prédictif dans le cas de journaux d'événements.

7.4.2 Utilisation de la recherche par faisceaux

Le GAN de [TR20] utilise une recherche par faisceaux [Low90]. Celle-ci permet la génération de plusieurs prédictions candidates pour un même préfixe en créant un arbre de recherche. La recherche s'effectue comme suit :

— la racine de l'arbre de recherche est un suffixe vide.

— Les feuilles de l'arbre sont des suffixes ayant atteint le *End of State*.

— Un suffixe peut être étendu en y concaténant une activité.

— Dans le cas d'une recherche par faisceaux de « largeur » n, la première étape propose n concaténations possibles d'une activité, proposant n suffixes de longueur 1.

— Ensuite, chaque suffixe de longueur 1 est augmenté d'une activité, n fois ; on possède alors n^2 suffixes.

— Lors de la concaténation d'une activité, la probabilité du suffixe incluant les étapes précédentes et la concaténation actuelle est estimée par le modèle.

— les n suffixes les plus probables selon le modèle sont conservés pour l'étape de concaténation suivante.

— Lorsque tous les suffixes considérés comme les plus probables par le modèle ont atteint le *End of State*, la recherche s'arrête et les n candidats complets sont proposés.

Utiliser la recherche par faisceaux dans ce contexte permet de générer un certain nombre de candidats, plus ou moins probables, pour un même préfixe à prédire. Ainsi, nous pouvons choisir le candidat qui s'avère être le plus proche de la réalité. Dans une optique de dérive

conceptuelle, si aucun candidat ne possède une distance de Damerau-Levenshtein nulle avec son suffixe réel, cela signifie que le *pattern* exhibé par le préfixe et ses variables descriptives n'ont pas permis au modèle de prédire correctement son suffixe. Si de plus en plus de préfixes à prédire se retrouvent sans prédiction candidate satisfaisante, alors des *patterns* indiscernables par le modèle prennent petit à petit le pas sur les anciens *patterns* dans la donnée. Il est par ailleurs possible que le nombre de préfixes prédits correctement par un candidat soit constant, mais que ce candidat corresponde de moins en moins au candidat jugé le plus probable par le modèle.

La méthode de détection de la dérive conceptuelle est donc basée sur ce principe, en présence d'un modèle prédictif.

7.4.3 Méthode

Période de référence

Contrairement à la méthode sans modèle prédictif, il n'y a pas besoin de chercher de périodes de comparaison et de comparaison et de stabilité. L'objectif de cette méthode est de pouvoir comparer les performances du modèle à l'arrivée de nouvelles données avec ses performances initiales : la période de référence correspond à l'ensemble d'entraînement créé lors de l'entraînement du modèle sur un journal d'événements donné.

Il suffit ensuite de déterminer la taille des échantillons, par exemple en fonction des *POM* que ces tailles d'échantillons permettent. Dans notre cas, la même méthodologie est appliquée pour la tailles des sous-logs : avec un journal d'événements L de taille $|L|_\sigma$, la taille des échantillons vaut $n = \max\left\{\left\lfloor \frac{|L|_\sigma}{100} \right\rceil, 50\right\}$, simplement pour illustrer la méthodologie développée dans ce chapitre.

Mesure de la dérive des activités

Soit $\sigma = \langle e_1, \ldots, e_n \rangle$ un parcours appartenant à un journal d'événements L. Soit $k \in \{1, \ldots, n-1\}$ un entier, $\sigma_{\leqslant k}$ un préfixe de σ, et $\sigma_{>k}$ le suffixe correspondant. Le générateur du modèle peut générer un nombre arbitraire de prédictions pour un même préfixe, mais dans le cas où $k = 1$ ou $k = n-1$, il est inutile en soi de proposer plus d'alternatives qu'il y a d'activités disponibles, et l'application réelle demande de limiter le temps de prédiction, or celui-ci augmente invariablement avec le nombre de candidats proposés. On fixe ainsi le nombre de candidats à $|\mathcal{A}|$. Le modèle génère donc $|\mathcal{A}|$ prédictions $\widehat{\sigma_{>k}}^{(1)}, \ldots, \widehat{\sigma_{>k}}^{(|\mathcal{A}|)}$.

En termes d'activités, il suffit de prendre les candidats générés par le modèle et de calculer leur similarité de Damerau-Levenshtein au suffixe réel, puis d'en récupérer le maximum :

$$\max_{j=1,\ldots,|\mathcal{A}|} \left\{ \text{Similarité}_{\text{DL}} \left(\widehat{\sigma_{>k}}^{(j)}, \sigma_{>k} \right) \right\}.$$

Si ce maximum est différent de 1, nous sommes dans le cas où aucun candidat généré ne correspond exactement à la réalité en termes de séquence d'activités. On contrôle l'évolution

de cette quantité moyenne par rapport aux valeurs observées dans l'ensemble de test lors de l'apprentissage.

Mesure de la dérive des durées

Dans le cas de la dérive des durées, une approche parfaitement analogue à la dérive des activités peut être employée, portant cette fois-ci sur la différence entre durée totale réelle et durée totale prédite. Pour un préfixe $\sigma_{\leqslant k}$ d'un parcours σ, nous générons $|\mathcal{A}|$ prédictions $\widehat{\sigma_{>k}}^{(1)}, \ldots, \widehat{\sigma_{>k}}^{(|\mathcal{A}|)}$. Ensuite, nous calculons la MAE entre chaque candidat et le suffixe réel en termes de durées totales, puis nous récupérons le minimum :

$$\min_{j=1,\ldots,|\mathcal{A}|} \left\{ \text{MAE}\left(\widehat{\sigma_{>k}}^{(j)}, \sigma_{>k}\right) \right\},$$

où $\text{MAE}\left(\widehat{\sigma_{>k}}^{(j)}, \sigma_{>k}\right)$ est la valeur absolue de la différence de durée totale des deux suffixes. Plus cette quantité est élevée (elle appartient à $[0; +\infty[$ et n'a donc pas de borne supérieure finie), moins le modèle est capable de prédire la durée totale du suffixe en question avec précision, malgré $|\mathcal{A}|$ essais. On contrôle également l'évolution de cette quantité moyenne par rapport aux valeurs observées dans l'ensemble de test lors de l'apprentissage.

7.4.4 Résultats

Commençons par exposer quelques résultats relatifs à la dérive des séquences d'activités. Pour illustrer cette méthode par l'exemple ici, des échantillons de l'ensemble de test lors de l'entraînement du modèle prédictif ont été créés de la façon suivante :

— un échantillon de parcours d'effectif $\min\left\{\left\lfloor \frac{|L|_e}{4} \right\rceil, 1000\right\}$ est tiré, avec probabilités égales : chaque parcours a une probabilité $\frac{1}{|L|_e}$ d'être tiré.

— Un parcours σ composé de n événements peut être scindé en préfixes de tailles 1 à $n-1$. Tout parcours σ de l'échantillon est scindé en une paire de préfixe et suffixe $(\sigma_{\leqslant k}, \sigma_{>k})$, k une observation d'une variable aléatoire uniforme discrète $K \sim \mathcal{U}(\{1, \ldots, n-1\})$.

Cette procédure est utile afin de créer une carte de contrôle grâce à peu de ressources, et d'y placer les points de la période de référence.
Les sous-échantillons sont ensuite envoyés au modèle prédictif afin que celui-ci génère $|\mathcal{A}|$ candidats pour chaque prédiction. Le tableau 7.3 montre 3 candidats proposés par le modèle pour 3 préfixes échantillonnés comme décrit ci-dessus dans *Helpdesk*. Il y a en réalité 9 candidats par préfixe avec ce jeu de données.

Dans le tableau 7.3, un candidat parmi les trois affichés est sélectionné comme étant le meilleur en termes de similarité de Damerau-Levenshtein. Les candidats pour le premier suffixe parviennent à atteindre l'exactitude dans le troisième candidat. Pour les autres

TABLEAU 7.3 – 3 prédictions générées par le WGAN conditionnel pour 3 préfixes avec une recherche par faisceaux de largeur 11, depuis un échantillon de 2 500 parcours de *Helpdesk*

Id.	Préfixe	Vérité	Pred. 1	Pred. 2	Pred. 3
2387	$\langle SoS \rangle$	$\langle 2, 3, 3, EoS \rangle$	$\langle 2, 2, 3, EoS \rangle$	$\langle 1, 2, EoS \rangle$	$\langle \mathbf{2}, \mathbf{3}, \mathbf{3}, \mathbf{EoS} \rangle$
4378	$\langle SoS, 1, 1 \rangle$	$\langle 2, 5, 2, 3, EoS \rangle$	$\langle \mathbf{2}, \mathbf{5}, \mathbf{3}, EoS \rangle$	$\langle 3, 3, EoS \rangle$	$\langle 2, 3, EoS \rangle$
2993	$\langle SoS, 1, 2 \rangle$	$\langle 5, 2, 3, EoS \rangle$	$\langle \mathbf{2}, \mathbf{3}, \mathbf{EoS} \rangle$	$\langle 3, EoS \rangle$	$\langle 2, 3, 3, EoS \rangle$

suffixes du tableau, le candidat 1 est le meilleur sans pour autant avoir une similarité de Damerau-Levenshtein de 1.

Étudions les résultats sur de la donnée réelle.

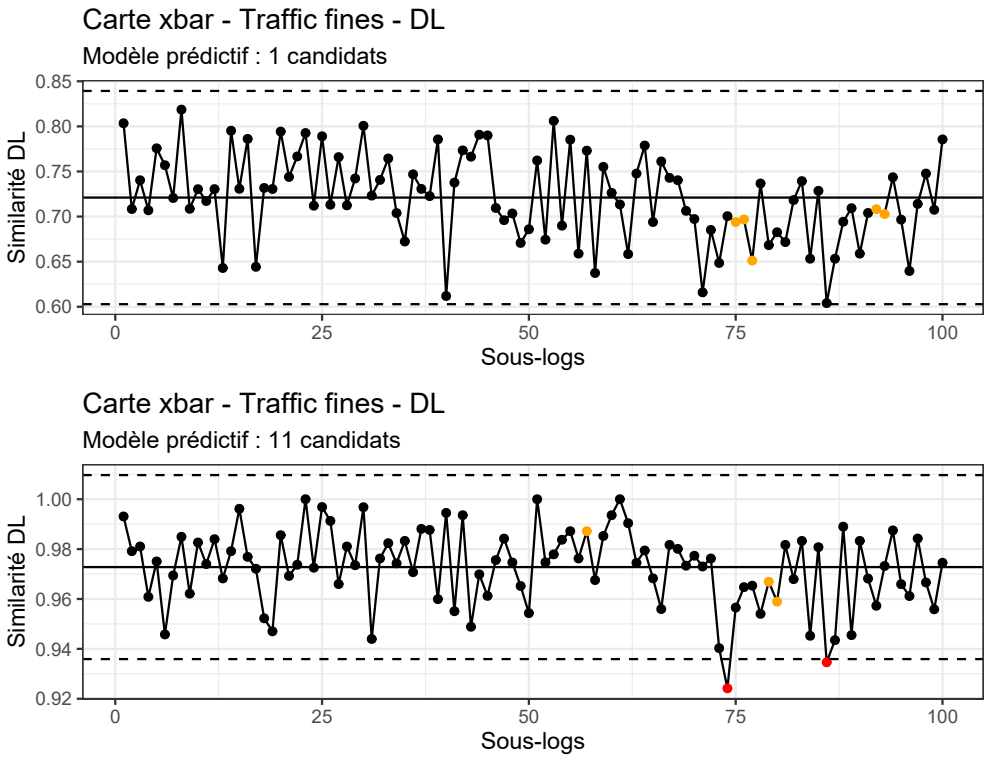

FIGURE 7.7 – Carte *xbar* pour la similarité de Damerau-Levenshtein des suffixes prédits par le modèle et la réalité dans l'ensemble de test de *Traffic fines*

Sur la figure 7.7, nous avons un exemple de la différence qu'offre l'utilisation de la recherche par faisceaux illustrée par *Traffic fines*. De prime abord, la figure du dessus, ne générant qu'un candidat sans concaténations successives d'activités mais seulement par génération de tout le suffixe par le modèle prédictif, il ne semble y avoir aucune violation,

contrairement au cas de la recherche par faisceaux avec $|\mathcal{A}| = 11$ candidats qui montre deux violations. Cependant, il faut regarder la valeur de la ligne centrale et l'écart entre les limites de contrôle : dans le cas d'une seule prédiction, celle-ci est entre 0,72 et 0,73 avec un écart de 0,13 entre la ligne centrale et chaque limite de contrôle. Alors qu'avec 11 candidats, la ligne centrale atteint presque 0,98, avec un écart à peine supérieur à 0,2 entre la ligne centrale et chaque limite de contrôle. Le premier graphique suit les performances de production du modèle, tandis que le deuxième évalue un apprentissage plus approfondi et complexe des *patterns* de la donnée.

La figure 7.8 permet de voir les tendances à plus long terme dans *Traffic fines* à travers une carte EWMA dans le graphique du haut. On a une augmentation lente de la similarité de Damerau-Levenshtein, jusqu'à dépasser la limite supérieure de contrôle, ce qui est en soi une bonne chose : un ré-entraînement aurait été contre-productif à la vue de cette alerte. En revanche, sans pour autant qu'il y ait de violations, à partir de l'échantillon 73, la similarité de Damerau-Levenshtein descend soudainement sous 0,97.
Si nous regardons le graphique du bas, sur lequel figure la carte S de l'écart-type, nous voyons qu'un grand nombre de violations a lieu. Les résultats de la carte EWMA sont donc à prendre avec précaution étant donné la volatilité de l'écart-type – bien que celui-ci évolue seulement entre 0 et 0,2.

Sur les deux premiers graphiques de la figure 7.9, nous observons une MAE moyenne aux alentours de 150 jours, alors que la moyenne donnée en chapitre 5 est d'environ de 98 jours. Ceci est frappant dans le sens que rajouter des candidats et retenir ceux qui maximisent la similarité de Damerau-Levenshtein a fait croître l'erreur sur le temps d'un facteur 1,5. Ceci indique que l'entraînement favorise un compromis entre précision des activités et précision sur les temps, de telle sorte que forcer une augmentation en précision sur les activités diminue la précision sur les durées. Ceci dit, comme le montre la figure 4.2 dans le chapitre 4, il semble que *Traffic fines* ne couvre pas une durée suffisamment grande pour qu'une tendance dans son exécution puisse être tirée de la donnée : les peuplements ne montrent pas de saisonnalités mais laissent supposer une occurrence unique dans la façon de traiter près de 3000 amendes routières, en vertu du peuplement de l'activité 5. Il semble donc possible que le modèle ait appris ce compromis en termes de précision spécifiquement à cause de ce manque de *patterns* dans la donnée.

La figure 7.10 montre, elle, les cartes *xbar*, EWMA et S contrôlant quelle prédiction candidate est la meilleure en moyenne en termes de similarité de Damerau-Levenshtein. Dans le graphique du haut, la carte *xbar* permet de voir que le meilleur candidat est en général celui évalué par le modèle comme étant le 3ème plus probable (la numérotation commençant à 0). Aucune violation n'est à noter et la carte *xbar* ne dénote aucune tendance particulière.
La carte EWMA, sur le graphique du milieu, permet de dégager une certaine tendance baissière à l'échantillon 50, puis haussière des échantillons 75 à 100.
La carte S, sur le graphique du bas, ne montre qu'une violation. L'écart-type est stable autour de la ligne centrale et ne montre aucune tendance particulière, outre la constance.

Dans le cas de ce critère, une baisse, même dépassant la limite inférieure de contrôle, constitue une dérive de bonne augure : le suffixe considéré comme le plus probable par le modèle est également plus proche de la réalité. Un ré-entraînement ne serait donc pas bienvenu. Cependant, une recherche de la cause de cette amélioration peut être intéressante, tant au niveau du modèle qu'à l'exécution du processus sur le terrain.
Cette méthode de contrôle du meilleur candidat pourrait permettre de choisir, au fil du temps, quel candidat proposer aux clients afin de maximiser les performances prédictives du modèle en production, au lieu de systématiquement proposer le premier candidat.

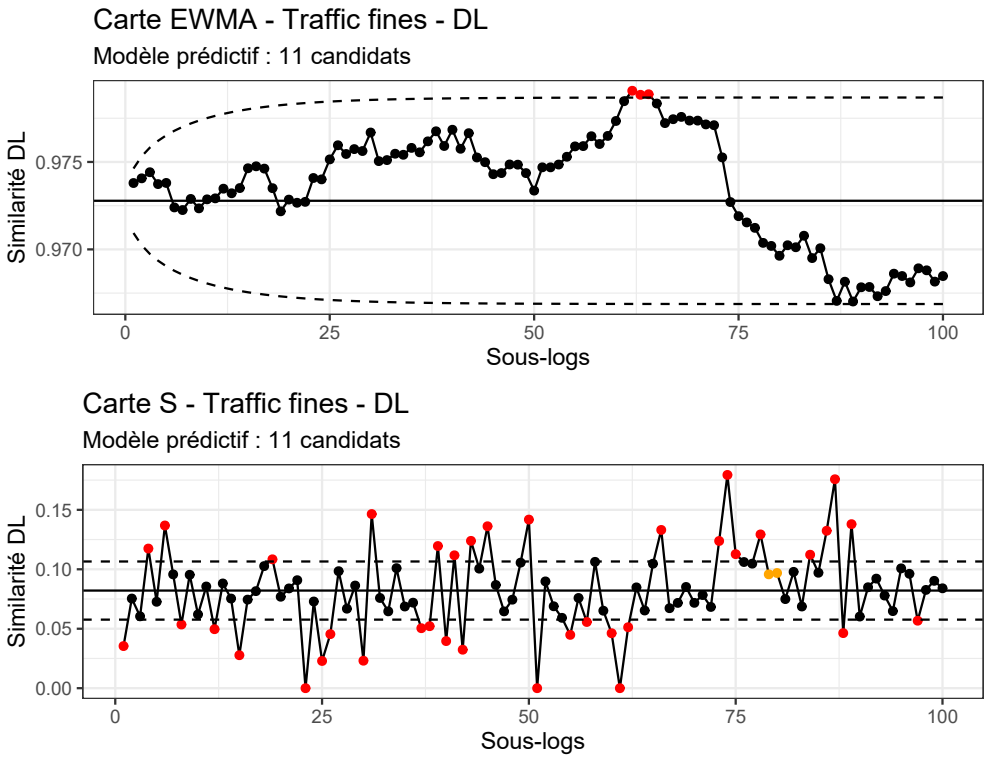

FIGURE 7.8 – Cartes EWMA et S pour la similarité de Damerau-Levenshtein des suffixes prédits par le modèle et la réalité dans l'ensemble de test de *Traffic fines*

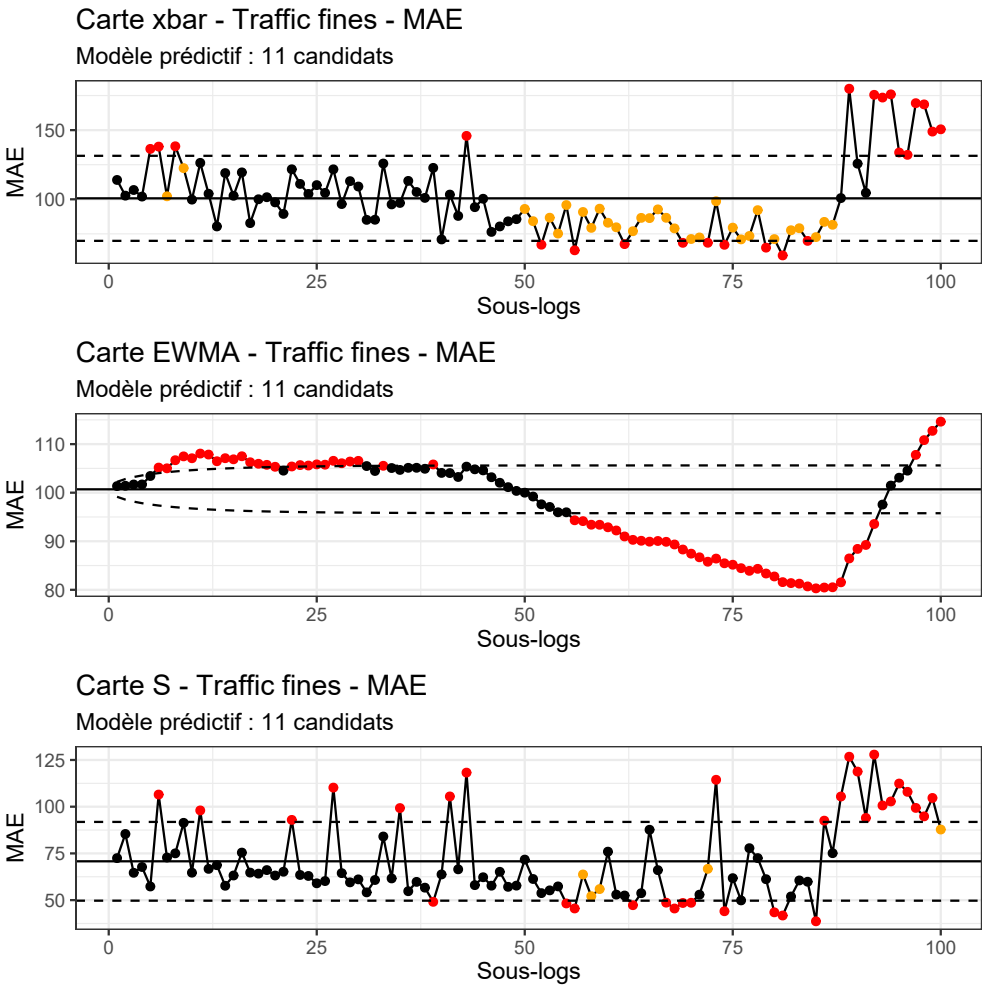

FIGURE 7.9 – Cartes *xbar*, EWMA et *S* pour la MAE des durées totales des suffixes prédits par le modèle et la réalité dans l'ensemble de test de *Traffic fines*

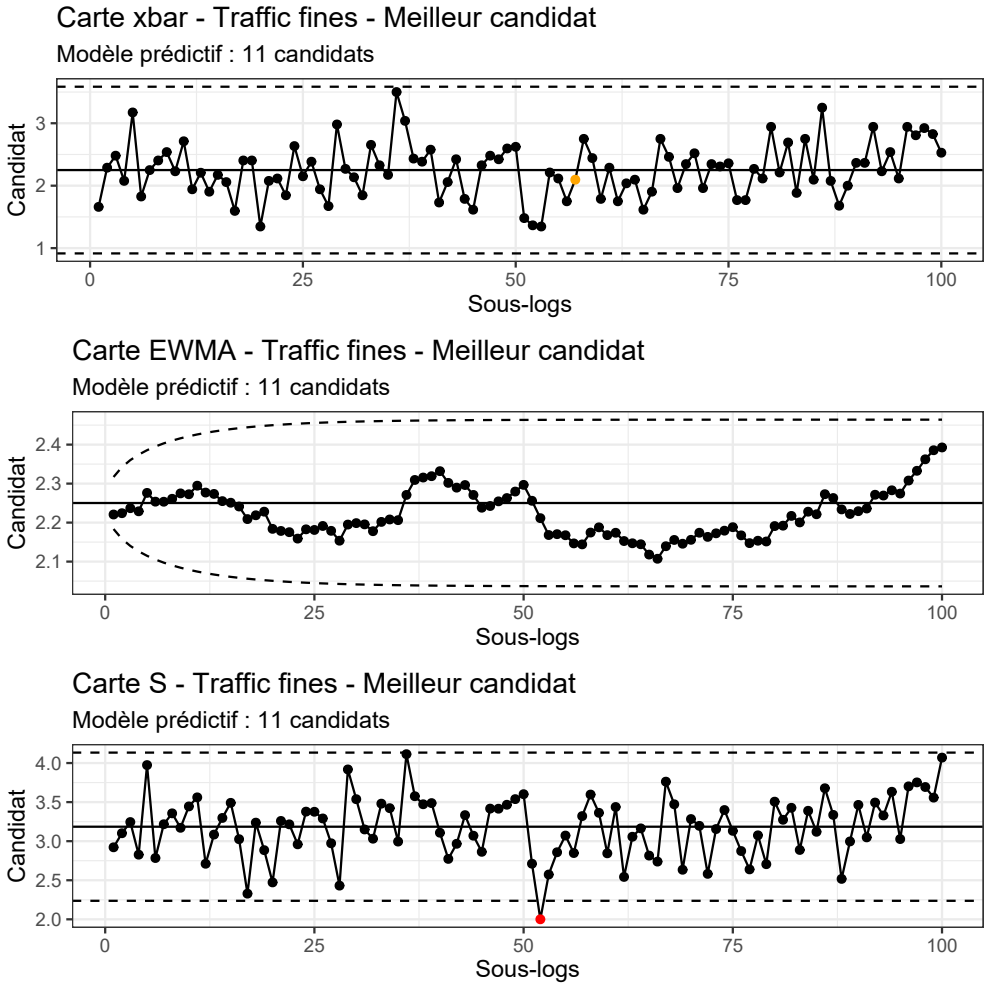

FIGURE 7.10 – Cartes *xbar*, EWMA et S de la moyenne du meilleur candidat maximisant la similarité de Damerau-Levenshtein, généré par le modèle sur l'ensemble de test de *Traffic fines*

7.5 Conclusion

La méthodologie développée dans ce chapitre permet le suivi de la dérive conceptuelle dans la donnée de processus, directement sur la donnée brute ou relativement à un modèle prédictif. La méthode sans modèle prédictif, qui inclut un élagage des répétitions maximales observées dans la période de comparaison, pourrait en soi permettre également le suivi de ces répétitions maximales afin de contrôler l'évolution des *reworks* au cours du temps. Quoi qu'il en soit, la méthodologie développée ici est utilisable sur n'importe quel journal d'événements, et avec n'importe quel modèle prédictif, y compris sans recherche par faisceaux – celle-ci ne fait qu'offrir un contrôle d'autant plus approfondi des *patterns* appris par le modèle.

Une des précautions à prendre réside dans la nature gaussienne des données lorsque le contrôle s'effectue avec des cartes *xbar* ou *EWMA*. La normalité de ces données est en effet une des conditions d'utilisation de ces cartes, bien que ceci ne soit pas nécessairement vérifié sur le terrain. La commodité de la méthode permet son utilisation malgré l'erreur encourue par la non-normalité des données. De plus, en vertu du théorème central limite en Statistique, les moyennes des échantillons ont tendance à suivre une loi normale malgré la non-normalité de la donnée de base. En ce qui concerne la méthode de suivi des performances du modèle en termes de similarité de Damerau-Levenshtein cependant, ceci peut ne pas être le cas : typiquement, lorsque celle-ci est suffisamment élevée, en particulier grâce à la recherche par faisceaux et à la multiplicité des candidats, la similarité peut avoir une moyenne trop proche de 1. Si proche de 1 en réalité, que la distribution de ces similarité deviendrait très asymétrique, et donc non gaussienne. Dans ce cas, un passage à la distance au lieu de la similarité permet de repasser dans un contexte de comptage, et donc de loi de Poisson. La présence conséquente de 0 pourrait cependant malgré tout mener à une sous-estimation du nombre de 0 par le paramètre d'intensité estimé pour la loi de Poisson correspondante.

La méthode permettant le suivi du candidat optimal en termes de similarité de Damerau-Levenshtein, pourrait éventuellement permettre l'utilisation dynamique de la recherche par faisceaux : en effet, une fois la ligne centrale déterminée, nous pouvons proposer systématiquement le candidat correspondant, à l'arrondi, à la valeur de la ligne centrale. Une dérive, croissante comme décroissante, permettrait de proposer un autre candidat lorsque la carte EWMA semble évoluer vers une nouvelle valeur. On a donc au moins deux utilisations possibles : la première orientée exclusivement vers le contrôle, permet d'évaluer quel candidat parmi ceux générés semblent correspondre de plus en plus à la réalité, nous aurions donc un suivi de l'évolution des *patterns* de la donnée et leur passage de probable à improbable, et vice-versa. La deuxième concerne la mise en production du modèle et permet la proposition de candidats mieux adaptés, plutôt que ne proposer que celui considéré comme le plus probable sans considération de l'optimalité des autres candidats au fur et à mesure du temps.

Cette méthode peut évidemment être affinée. Pour l'instant, tous les caractères étudiés ici sont univariés. Or il serait intéressant de pouvoir contrôler des ensembles de variables. Par exemple, bien que la durée totale soit en elle-même informative, étudier la durée moyenne de chaque activité serait encore bien plus intéressante.
Un autre aspect à approfondir, permettant un contrôle d'autant plus sur-mesure en fonction des clients et des processus étudiés, serait l'utilisation de cartes aux démérites. Ces cartes hiérarchisent les non-conformités et les pondèrent selon cette hiérarchisation, permettant d'éviter des alarmes pour un surplus de non-conformités peu importantes, ou d'en déclencher plus rapidement en présence de non-conformités jugées importantes.

Chapitre 8

Conclusion et ouvertures

Cette thèse propose une méthode de modélisation des parcours individuels à l'aide d'apprentissage profond, dans une optique prédictive. À l'origine, les recherches devaient s'orienter sur les modèles les plus adaptés afin de maximiser la performance prédictive. Une piste consistait en une approche hiérarchique, à travers des *clusterings* des traces, puis un modèle tenant compte de ces *clusters*. Nous nous sommes immédiatement rendu compte que les *clusters* n'étaient pas des groupes isolés mais bel et bien des « zooms » sur des sous-processus, qui ne sont pas isolés les uns des autres. Une forme de communication entre les *clusters* devait avoir lieu. C'est ainsi qu'est née l'idée du peuplement, qui contient de l'information relative au processus global. *In fine*, cette création et utilisation du peuplement a ouvert les possibilités explorées dans cette thèse, qui a finalement montré que les journaux d'événements contenaient de l'information implicite en grande quantité qui n'était simplement pas utilisée, ni apprise automatiquement par les modèles.
En réalité, la quasi-totalité des modèles d'apprentissage profond de la littérature traite la prédiction de parcours de la même façon qu'un modèle de traitement du langage naturel prédit la fin d'une phrase, mais un parcours dans un processus contient une complexité temporelle et une existence simultanée avec d'autres unités qui n'existe pas dans le langage naturel, malgré la place proéminente du contexte dans la prédiction des mots suivants dans une phrase.
Nous avons, en somme, montré que la recherche immédiate d'un meilleur modèle revenait donc en réalité à sauter une étape cruciale, puisque la donnée était largement sous-exploitée. La création du peuplement de processus ainsi que son utilisation ont d'ores et déjà permis de dépasser les performances des autres modèles dans l'état de l'art tant en prédiction des activités restantes que des durées restantes, permettent la simulation de parcours, et leur nature rend facilement interprétables les prédictions du modèle selon les configurations de peuplements choisies.

Par ailleurs, cette thèse apporte une méthodologie permettant de définir, quantifier et contrôler la dérive conceptuelle dans la donnée de processus à l'aide de cartes de contrôle,

d'une manière simple à mettre en place et à interpréter. Cette méthodologie permet tant le suivi de la dérive de la donnée brute que la dérive de la donnée par rapport à un modèle prédictif, donnant un suivi exhaustif de la donnée et des performances prédictives du modèle.
Livejourney™ est ainsi muni d'un modèle prédictif performant, requérant peu de tentatives d'entraînements grâce à son coût de Wasserstein, simulatoire, partiellement explicable, avec un contrôle des performances permettant de décider de façon éclairée quand un ré-entraînement semble opportun, évitant les ré-entraînements périodiques potentiellement superflus et coûteux.

De façon évidente, ces avancées apportent leur lot de perspectives et ouvertures. Tout d'abord, la modélisation elle-même : un modèle génératif est utilisé, mais d'autres architectures telles que les réseaux de neurones en graphes, intuitivement proches de la représentation des processus, semblent également adéquates et à explorer conjointement à l'utilisation du peuplement. Par ailleurs, des modèles de langages de grande taille tels que GPT-3 [Bro+20] apprennent une quantité de langages et de contextes à la fois. Ainsi, il serait intéressant de tenter de créer des modèles de processus de grande taille, apprenant des processus au moins à l'échelle d'entreprises entières.

Concernant la simulation, la méthode présentée dans cette thèse n'est qu'un premier pas et demande des raffinements. En effet, l'objectif *business* réel derrière cette fonctionnalité réside dans la création de jumeaux numériques des processus, conformes aux règles tacites et implicites régissant ces processus et permettant toutes sortes de manipulations dans une optique de scénarisation et d'étude des issues de ces scénarii. Ainsi, rien n'indique que la meilleure piste réside dans les modèles d'apprentissage profond, ni même d'apprentissage machine en général. Il s'agit là d'un sujet à part entière bien plus vaste que la bribe de savoir apportée par cette thèse.

La méthode décrite dans cette thèse pour le contrôle de la dérive conceptuelle avec et sans modèle prédictif a été créée dans une optique d'application simple, mais exhaustive. La philosophie des cartes de contrôle est, avant tout, leur simplicité, permettant à n'importe quel technicien de comprendre leur utilisation rapidement, et rendant leur interprétation intuitive. Cette approche est motivée avant tout par le contexte appliqué de ces recherches : la dérive conceptuelle doit pouvoir être suivie de façon simple par les employés de Livejourney™ mais aussi par leurs clients, qui ne sont pas nécessairement versés dans la statistique. Ceci dit, cette méthodologie trouve sa limite dans les approximations nécessaires à son exécution (supposition de la normalité des données mesurées, utilisation assumée du théorème central limite de la statistique, *et caetera*). Par ailleurs, les dépendances entre virtuellement chaque variable décrivant un processus rend ardue le contrôle multivarié, sans démultiplier les cartes de contrôle univariées. Par ailleurs, ces méthodes permettent la détection de changements graduels ou brutaux tant dans les activités, les séquences d'activités et les durées, mais ce n'est pas nécessairement le cas pour les changements cycliques, pour peu qu'un cycle ait lieu dans la période de comparaison. L'approche par élagage des répétitions et cartes de contrôle est donc à raffiner pour ces cas.

Il est également possible de prendre une direction tout autre et de choisir des approches d'apprentissage machine, en particulier à partir de générations de journaux d'événements : si la donnée apprise pour la génération a dérivé, les journaux d'événements issus du modèle génératif seraient sensiblement différents de la nouvelle donnée, par rapport à leur écart vis-à-vis de la donnée d'apprentissage. Une approche par auto-encodeurs est même envisageable, dans la mesure où un auto-encodeur avec une faible erreur de reconstruction sur la donnée d'entraînement pourrait voir cette erreur de reconstruction augmenter sur de la donnée ayant dérivé.

Une question plus englobante se pose quant à la trajectoire que devrait prendre un modèle prédictif au cours du temps. En effet, l'un des objectifs de ces modèles en *Predictive Business Process Monitoring* est de permettre aux clients d'anticiper des parcours considérés comme indésirables, afin d'agir en amont dans le but d'empêcher ces caractères indésirables de se produire. Ceci a pour conséquence de rendre la prédiction fausse, pour peu qu'elle eût été correcte sans l'intervention d'un opérateur. Ainsi, les mesures réelles des performances prédictives du modèle sont difficiles étant donné cette optique d'intervention sachant une prédiction. Par ailleurs, si ces prédictions révèlent un dysfonctionnement inhérent au processus prédit, le client aurait tout intérêt à modifier son processus afin d'éliminer ce dysfonctionnement. Le processus étant ainsi changé, le modèle doit être ré-entraîné soit sur demande du client, soit par détection d'une dérive conceptuelle. L'historique du nouveau processus serait cependant trop nouveau, et donc trop peu fourni en termes de données. Il se pose donc la question d'un apprentissage à partir du modèle précédent, ou bien d'une forme de *transfer learning*.

Bibliographie

[AWM04] Wil van der Aalst, Ton Weijters et Laura Maruster. « Workflow mining : discovering process models from event logs ». Dans : *IEEE Transactions on Knowledge and Data Engineering* 16.9 (2004), p. 1128-1142. doi : 10.1109/TKDE.2004.47.

[AFN96a] AFNOR. *Application de la statistique - Cartes de contrôle - Partie 1 : cartes de contrôle de Shewhart aux mesures*. NF X 06-031-1. Normes nationales et documents normatifs nationaux. Association française de normalisation, fév. 1996.

[AFN02] AFNOR. *Application de la statistique - Cartes de contrôle - Partie 2 : cartes de contrôle aux attributs*. NF X 06-031-2. Normes nationales et documents normatifs nationaux. Association française de normalisation, avr. 2002.

[AFN96b] AFNOR. *Application de la statistique - Cartes de contrôle - Partie 3 : cartes de contrôle à moyennes mobiles avec pondération exponentielle (EWMA)*. NF X 06-031-3. Normes nationales et documents normatifs nationaux. Association française de normalisation, avr. 1996.

[AFN96c] AFNOR. *Application de la statistique - Cartes de contrôle - Partie 4 : cartes de contrôle des sommes cumulées (CUSUM)*. NF X 06-031-4. Normes nationales et documents normatifs nationaux. Association française de normalisation, fév. 1996.

[ACB17] Martin Arjovsky, Soumith Chintala et Léon Bottou. *Wasserstein GAN*. 2017. arXiv : 1701.07875 [stat.ML].

[Bay+17] Inci M. Baytas et al. « Patient Subtyping via Time-Aware LSTM Networks ». Dans : *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '17. Halifax, NS, Canada : Association for Computing Machinery, 2017, p. 65-74. isbn : 9781450348874. doi : 10.1145/3097983.3097997. url : https://doi.org/10.1145/3097983.3097997.

[BDH13] Veronica Becher, Alejandro Deymonnaz et Pablo Ariel Heiber. *Efficient repeat finding via suffix arrays*. 2013. arXiv : 1304.0528 [cs.DS].

[Ben76] Jean-Paul Benzécri. *L'Analyse des données : L'Analyse des correspondances*. vol. 2. Dunod, 1976. URL : https://books.google.fr/books?id=-A8SwQEACAAJ.

[BCH20] Mathilde Boltenhagen, Benjamin Chetioui et Laurine Huber. « An Alignment Cost-Based Classification of Log Traces Using Machine-Learning ». Dans : *ML4PM2020 - First International Workshop on Leveraging Machine Learning in Process Mining*. Padua/ Virtual, Italy, oct. 2020. URL : https://hal.inria.fr/hal-03134114.

[BR07] Marc Boyer et Olivier H. Roux. « Comparison of the Expressiveness of Arc, Place and Transition Time Petri Nets ». Dans : *Petri Nets and Other Models of Concurrency – ICATPN 2007*. Sous la dir. de Jetty Kleijn et Alex Yakovlev. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 63-82. ISBN : 978-3-540-73094-1.

[Bro+20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv : 2005.14165 [cs.CL].

[BSD21] Zaharah A. Bukhsh, Aaqib Saeed et Remco M. Dijkman. *ProcessTransformer : Predictive Business Process Monitoring with Transformer Network*. 2021. arXiv : 2104.00721 [cs.LG].

[Cha+02] Nitesh Chawla et al. « SMOTE : Synthetic Minority Over-sampling Technique ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 16 (juin 2002), p. 321-357. DOI : 10.1613/jair.953.

[Dam64] Frederick J. Damerau. « A Technique for Computer Detection and Correction of Spelling Errors ». Dans : *Commun. ACM* 7.3 (mars 1964), p. 171-176. ISSN : 0001-0782. DOI : 10.1145/363958.363994. URL : https://doi.org/10.1145/363958.363994.

[DA10] René David et Hassane Alla. *Discrete, continuous, and hybrid Petri nets*. en. 2e éd. Berlin, Germany : Springer, fév. 2010.

[De +13] Jochen De Weerdt et al. « Active Trace Clustering for Improved Process Discovery ». Dans : *IEEE Transactions on Knowledge and Data Engineering* 25.12 (2013), p. 2708-2720. DOI : 10.1109/TKDE.2013.64. URL : https://doi.org/10.1109/TKDE.2013.64.

[Di +19] Ciara Di Francescomarino et al. « Clustering-Based Predictive Process Monitoring ». Dans : *IEEE Transactions on Services Computing* 12.6 (2019), p. 896-909. DOI : 10.1109/TSC.2016.2645153.

[DAB19] Nicola Di Mauro, Annalisa Appice et Teresa Basile. « Activity Prediction of Business Process Instances with Inception CNN Models ». Dans : nov. 2019, p. 348-361. ISBN : 978-3-030-35165-6. DOI : 10.1007/978-3-030-35166-3_25.

[EP04] Hubert Egon et Pascal Porée. *Statistique et probabilités en production industrielle*. Sous la dir. d'Hermann. T. 2. 2004.

[ERF17] Joerg Evermann, Jana-Rebecca Rehse et Peter Fettke. « Predicting process behaviour using deep learning ». Dans : *Decision Support Systems* 100 (août 2017), p. 129-140. ISSN : 0167-9236. DOI : 10.1016/j.dss.2017.04.003. URL : http://dx.doi.org/10.1016/j.dss.2017.04.003.

[FFN91] Gérard Florin, Céline Fraize et Stéphane Natkin. « Stochastic Petri nets : Properties, applications and tools ». Dans : *Microelectronics Reliability* 31.4 (1991), p. 669-697. ISSN : 0026-2714. DOI : https://doi.org/10.1016/0026-2714(91)90009-V. URL : https://www.sciencedirect.com/science/article/pii/002627149190009V.

[FN85] Gérard Florin et Stéphane Natkin. *Les Réseaux de Petri Stochastiques*. T. 1. 4. TSI, 1985.

[FH11] Arnoldo Frigessi et Bernd Heidergott. « Markov Chains ». Dans : *International Encyclopedia of Statistical Science*. Sous la dir. de Miodrag Lovric. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 772-775. ISBN : 978-3-642-04898-2. DOI : 10.1007/978-3-642-04898-2_347. URL : https://doi.org/10.1007/978-3-642-04898-2_347.

[Gab+11] Alexis Gabadinho et al. « Analyzing and Visualizing State Sequences in R with TraMineR ». Dans : *Journal of Statistical Software* 40.4 (2011), p. 1-37. DOI : 10.18637/jss.v040.i04.

[Gar+21] Simon Garnier et al. *viridis - Colorblind-Friendly Color Maps for R*. R package version 0.6.2. 2021. DOI : 10.5281/zenodo.4679424. URL : https://sjmgarnier.github.io/viridis/.

[Goo+14] Ian Goodfellow et al. « Generative Adversarial Networks ». Dans : *Advances in Neural Information Processing Systems* 3 (juin 2014). DOI : 10.1145/3422622.

[Gul+17] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv : 1704.00028 [cs.LG].

[Gum54] Emil Julius Gumbel. *Statistical Theory of Extreme Values and Some Practical Applications : A Series of Lectures*. Applied mathematics series. U.S. Government Printing Office, 1954.

[Har+20] Maximilian Harl et al. « Explainable predictive business process monitoring using gated graph neural networks ». Dans : juin 2020.

[HW14] Douglas M. Hawkins et Qifan Wu. « The CUSUM and the EWMA Head-to-Head ». Dans : *Quality Engineering* 26.2 (2014), p. 215-222. DOI : 10.1080/08982112.2013.817014. eprint : https://doi.org/10.1080/08982112.2013.817014. URL : https://doi.org/10.1080/08982112.2013.817014.

[Ho95] Tin Kam Ho. « Random decision forests ». Dans : *Proceedings of 3rd International Conference on Document Analysis and Recognition*. T. 1. 1995, 278-282 vol.1. DOI : 10.1109/ICDAR.1995.598994.

[HS97] Sepp HOCHREITER et Jürgen SCHMIDHUBER. « Long Short-Term Memory ». Dans : *Neural Computation* 9.8 (nov. 1997), p. 1735-1780. ISSN : 0899-7667. DOI : 10.1162/neco.1997.9.8.1735. eprint : https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL : https://doi.org/10.1162/neco.1997.9.8.1735.

[Hot33] Harold HOTELLING. « Analysis of a complex of statistical variables into principal components. » Dans : *Journal of Educational Psychology* 24 (1933), p. 498-520.

[JA09] R.P. JAGADEESH CHANDRA BOSE et Wil van der AALST. « Context Aware Trace Clustering : Towards Improving Process Mining Results ». Dans : avr. 2009. DOI : 10.1137/1.9781611972795.35.

[JGP17] Eric JANG, Shixiang GU et Ben POOLE. *Categorical Reparameterization with Gumbel-Softmax*. 2017. arXiv : 1611.01144 [stat.ML].

[Jan22a] Gert JANSSENSWILLEN. *bupaR : Business Process Analysis in R*. R package version 0.5.2. 2022. URL : https://CRAN.R-project.org/package=bupaR.

[Jan22b] Gert JANSSENSWILLEN. *eventdataR : Event Data Repository*. R package version 0.3.0. 2022. URL : https://CRAN.R-project.org/package=eventdataR.

[Jan22c] Gert JANSSENSWILLEN. *processmapR : Construct Process Maps Using Event Data*. R package version 0.5.2. 2022. URL : https://CRAN.R-project.org/package=processmapR.

[Jo19] Taeho JO. *Text Mining : Concepts, Implementation, and Big Data Challenge*. Studies in big data. Springer, 2019. ISBN : 9783319918167. URL : https://books.google.ca/books?id=qPo3yQEACAAJ.

[KM20] Alboukadel KASSAMBARA et Fabian MUNDT. *factoextra : Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.7. 2020. URL : https://CRAN.R-project.org/package=factoextra.

[KB17] Diederik P. KINGMA et Jimmy BA. *Adam : A Method for Stochastic Optimization*. 2017. arXiv : 1412.6980 [cs.LG].

[Kra+21] Wolfgang KRATSCH et al. « Machine Learning in Business Process Monitoring : A Comparison of Deep Learning and Classical Approaches Used for Outcome Prediction ». Dans : *Business & Information Systems Engineering* 63 (juin 2021). DOI : 10.1007/s12599-020-00645-0.

[KL51] Solomon KULLBACK et Richard LEIBLER. « On Information and Sufficiency ». Dans : *Annals of Mathematical Statistics* 22.1 (1951), p. 79-86.

[Kun09] Ludmila I. KUNCHEVA. « Using Control Charts for Detecting Concept Change in Streaming Data ». Dans : 2009.

[LJH08] Sébastien LÊ, Julie JOSSE et François HUSSON. « FactoMineR : A Package for Multivariate Analysis ». Dans : *Journal of Statistical Software* 25.1 (2008), p. 1-18. DOI : 10.18637/jss.v025.i01.

[LBH15] Yann LeCun, Yoshua Bengio et Geoffrey Hinton. « Deep learning ». Dans : *Nature* 521.7553 (1er mai 2015), p. 436-444. DOI : 10.1038/nature14539. URL : https://doi.org/10.1038/nature14539.

[LeC+98] Yann LeCun et al. « Gradient-based learning applied to document recognition ». Dans : *Proceedings of the IEEE* 86.11 (1998), p. 2278-2324. DOI : 10.1109/5.726791.

[Lev65] Vladimir I. Levenshtein. « Binary codes capable of correcting deletions, insertions, and reversals ». Dans : *Soviet physics. Doklady* 10 (1965), p. 707-710.

[Lin91] Jianhua Lin. « Divergence measures based on the Shannon entropy ». Dans : *IEEE Transactions on Information Theory* 37.1 (1991), p. 145-151. DOI : 10.1109/18.61115.

[LWW19] Li Lin, Lijie Wen et Jianmin Wang. « MM-Pred : A Deep Predictive Model for Multi-attribute Event Sequence ». Dans : mai 2019, p. 118-126. ISBN : 978-1-61197-567-3. DOI : 10.1137/1.9781611975673.14.

[Low90] Bruce Lowerre. « The Harpy Speech Understanding System ». Dans : *Readings in Speech Recognition*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1990, p. 576-586. ISBN : 1558601244.

[Már+17] Alfonso E. Márquez-Chamorro et al. « Run-time prediction of business process indicators using evolutionary decision rules ». Dans : *Expert Systems with Applications* 87 (2017), p. 1-14. ISSN : 0957-4174. DOI : https://doi.org/10.1016/j.eswa.2017.05.069. URL : https://www.sciencedirect.com/science/article/pii/S0957417417303950.

[Mar+98] Marco Ajmone Marsan et al. « Modelling with Generalized Stochastic Petri Nets ». Dans : *SIGMETRICS Perform. Eval. Rev.* 26.2 (août 1998), p. 2. ISSN : 0163-5999. DOI : 10.1145/288197.581193. URL : https://doi.org/10.1145/288197.581193.

[MLW21] Dhouha Mejri, Mohamed Limam et Claus Weihs. « A new time adjusting control limits chart for concept drift detection ». Dans : *IFAC Journal of Systems and Control* 17 (2021), p. 100170. ISSN : 2468-6018. DOI : https://doi.org/10.1016/j.ifacsc.2021.100170. URL : https://www.sciencedirect.com/science/article/pii/S2468601821000195.

[Mon09] Douglas Montgomery. *Introduction to statistical quality control*. Hoboken, N.J : Wiley, 2009. ISBN : 978-0470169926.

[Nas50] John Forbes Nash. « Equilibrium Points in N-Person Games ». Dans : *Proceedings of the National Academy of Sciences of the United States of America* 36 (1950), p. 48-49.

[Neu22] Erich Neuwirth. *RColorBrewer : ColorBrewer Palettes*. R package version 1.1-3. 2022. URL : https://CRAN.R-project.org/package=RColorBrewer.

[Ngu+20] An Nguyen et al. *Time Matters : Time-Aware LSTMs for Predictive Business Process Monitoring*. 2020. arXiv : 2010.00889 [cs.LG].

[OL97] Basantkumar John Oommen et Richard K. S. Locke. « Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions ». Dans : *Pattern Recognition* 30.5 (1997), p. 789-800. issn : 0031-3203. doi : 10.1016/S0031-3203(96)00101-X.

[Pag04] Jérôme Pagès. « Analyse factorielle de données mixtes ». fre. Dans : *Revue de Statistique Appliquée* 52.4 (2004), p. 93-111. url : http://eudml.org/doc/106558.

[PS20] Gyunam Park et Minseok Song. « Predicting performances in business processes using deep neural networks ». Dans : *Decision Support Systems* 129 (2020), p. 113191. issn : 0167-9236. doi : https://doi.org/10.1016/j.dss.2019.113191. url : http://www.sciencedirect.com/science/article/pii/S0167923619302209.

[PS19] Gyunam Park et Minseok Song. *Prediction-based Resource Allocation using Bayesian Neural Networks and Minimum Cost and Maximum Flow Algorithm*. 2019. arXiv : 1910.05126 [cs.AI].

[Pas+20] Vincenzo Pasquadibisceglie et al. « ORANGE : Outcome-Oriented Predictive Process Monitoring Based on Image Encoding and CNNs ». Dans : *IEEE Access* Volume 8 (oct. 2020). doi : 10.1109/ACCESS.2020.3029323.

[Pas+19a] Vincenzo Pasquadibisceglie et al. « Using Convolutional Neural Networks for Predictive Process Analytics ». Dans : *2019 International Conference on Process Mining (ICPM)*. 2019, p. 129-136. doi : 10.1109/ICPM.2019.00028.

[Pas+19b] Adam Paszke et al. « PyTorch : An Imperative Style, High-Performance Deep Learning Library ». Dans : *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA : Curran Associates Inc., 2019.

[Pea01] Karl Pearson. « LIII. On lines and planes of closest fit to systems of points in space ». Dans : *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), p. 559-572. doi : 10.1080/14786440109462720.

[Phi+20] Patrick Philipp et al. « Predictive Analysis of Business Processes Using Neural Networks with Attention Mechanism ». Dans : fév. 2020. doi : 10.1109/ICAIIC48513.2020.9065057.

[Pil05] Maurice Pillet. *Appliquer la maîtrise statistique des processus (MSP/SPC)*. Paris : Editions d'Organisation, 2005. isbn : 978-2708133495.

[R C21] R Core Team. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2021. url : https://www.R-project.org/.

[RVL20] Efrén Rama-Maneiro, Juan C. Vidal et Manuel Lama. *Deep Learning for Predictive Business Process Monitoring : Review and Benchmark*. 2020. arXiv : 2009.13251 [cs.LG].

[RE98] Grzegorz Rozenberg et Joost Engelfriet. « Elementary net systems ». Dans : *Lectures on Petri Nets I : Basic Models : Advances in Petri Nets*. Sous la dir. de Wolfgang Reisig et Grzegorz Rozenberg. Berlin, Heidelberg : Springer Berlin Heidelberg, 1998, p. 12-121. isbn : 978-3-540-49442-3. doi : 10.1007/3-540-65306-6_14. url : https://doi.org/10.1007/3-540-65306-6_14.

[RM87] David E. Rumelhart et James L. McClelland. « Learning Internal Representations by Error Propagation ». Dans : *Parallel Distributed Processing : Explorations in the Microstructure of Cognition : Foundations*. 1987, p. 318-362.

[Sat+21] Denise Maria Vecino Sato et al. « A Survey on Concept Drift in Process Mining ». Dans : *ACM Comput. Surv.* 54.9 (oct. 2021). issn : 0360-0300. doi : 10.1145/3472752.

[Scr04] Luca Scrucca. « qcc : an R package for quality control charting and statistical process control ». Dans : *R News* 4/1 (2004), p. 11-17. url : https://cran.r-project.org/doc/Rnews/.

[Sha48] Claude Elwood Shannon. « A Mathematical Theory of Communication ». Dans : *The Bell System Technical Journal* 27 (1948), p. 379-423.

[TTZ18] Niek Tax, Irene Teinemaa et Sebastiaan J. van Zelst. *An Interdisciplinary Comparison of Sequence Modeling Methods for Next-Element Prediction*. 2018. arXiv : 1811.00062 [stat.ML].

[Tax+17] Niek Tax et al. « Predictive Business Process Monitoring with LSTM Neural Networks ». Dans : *Lecture Notes in Computer Science* (2017), p. 477-492. issn : 1611-3349. doi : 10.1007/978-3-319-59536-8_30. url : http://dx.doi.org/10.1007/978-3-319-59536-8_30.

[TR20] Farbod Taymouri et Marcello La Rosa. *Encoder-Decoder Generative Adversarial Nets for Suffix Generation and Remaining Time Prediction of Business Process Models*. 2020. arXiv : 2007.16030 [cs.LG].

[Tay+20] Farbod Taymouri et al. *Predictive Business Process Monitoring via Generative Adversarial Nets : The Case of Next Event Prediction*. 2020. doi : 10.48550/ARXIV.2003.11268. url : https://arxiv.org/abs/2003.11268.

[TD19] Julian Theis et Houshang Darabi. « Decay Replay Mining to Predict Next Process Events ». Dans : *IEEE Access* 7 (2019), p. 119787-119803. issn : 2169-3536. doi : 10.1109/access.2019.2937085. url : http://dx.doi.org/10.1109/ACCESS.2019.2937085.

[Til06] Yves TILLÉ. *Sampling Algorithms*. Springer Series in Statistics. Springer, 2006. ISBN : 9780387308142. URL : https://books.google.fr/books?id=2auW1rVAwGMC.

[VBM23] Yoann VALERO, Frédéric BERTRAND et Myriam MAUMY. « Concept Drift in Process Data : A Control Chart Approach and Methodology ». Dans : *Joint Statistical Meetings 2023 Proceedings*. 2023.

[VBM22] Yoann VALERO, Frédéric BERTRAND et Myriam MAUMY. « Use of Process Crowding in Conditional WGAN for Remaining Process Events Prediction ». Dans : *Joint Statistical Meetings 2022 Proceedings*. 2022.

[vSS11] Wil VAN DER AALST, Helen M. SCHONENBERG et Minseok SONG. « Time prediction based on process mining ». Dans : *Information Systems* 36.2 (2011). Special Issue : Semantic Integration of Data, Multimedia, and Services, p. 450-475. ISSN : 0306-4379. DOI : https://doi.org/10.1016/j.is.2010.09.001. URL : https://www.sciencedirect.com/science/article/pii/S0306437910000864.

[Van86] Christoph VAN DER MALSBURG. « Frank Rosenblatt : Principles of Neurodynamics : Perceptrons and the Theory of Brain Mechanisms ». Dans : *Brain Theory*. Sous la dir. de Günther PALM et Ad AERTSEN. Berlin, Heidelberg : Springer Berlin Heidelberg, 1986, p. 245-248. ISBN : 978-3-642-70911-1.

[Vap98] Vladimir N. VAPNIK. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[Vas+17] Ashish VASWANI et al. *Attention Is All You Need*. 2017. arXiv : 1706.03762 [cs.CL].

[VR02] Bill VENABLES et B RIPLEY. « Modern Applied Statistics With S ». Dans : jan. 2002. DOI : 10.1007/b97626.

[Ver+19] Ilya VERENICH et al. « Survey and Cross-Benchmark Comparison of Remaining Time Prediction Methods in Business Process Monitoring ». Dans : *ACM Trans. Intell. Syst. Technol.* 10.4 (juill. 2019). ISSN : 2157-6904. DOI : 10.1145/3331449. URL : https://doi.org/10.1145/3331449.

[Vil09] Cédric VILLANI. « Optimal transport : Old and new ». Dans : t. 338. Grundlehren der mathematischen Wissenschaften. Springer, 2009, p. XXII, 976. DOI : 10.1007/978-3-540-71050-9.

[WR11] Ton WEIJTERS et Joel RIBEIRO. « Flexible Heuristics Miner (FHM) ». Dans : avr. 2011, p. 310-317. DOI : 10.1109/CIDM.2011.5949453.

[Wic16] Hadley WICKHAM. *ggplot2 : Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN : 978-3-319-24277-4. URL : https://ggplot2.tidyverse.org.

[WGK20] Remigiusz WIŚNIEWSKI, Iwona GROBELNA et Andrei KARATKEVICH. « Determinism in Cyber-Physical Systems Specified by Interpreted Petri Nets ». Dans : *Sensors* 20 (sept. 2020), p. 5565. DOI : 10.3390/s20195565.

[ZBA22] Kungang Zhang, Anh T. Bui et Daniel W. Apley. « Concept Drift Monitoring and Diagnostics of Supervised Learning Models via Score Vectors ». Dans : *Technometrics* 0.0 (2022), p. 1-13. doi : 10.1080/00401706.2022.2124310. eprint : https://doi.org/10.1080/00401706.2022.2124310. url : https://doi.org/10.1080/00401706.2022.2124310.

[ZCZ20] Ziwei Zhang, Peng Cui et Wenwu Zhu. *Deep Learning on Graphs : A Survey.* 2020. arXiv : 1812.04202 [cs.LG].

[Zha+20] Jingyu Zhao et al. *Do RNN and LSTM have Long Memory ?* 2020. arXiv : 2006.03860 [stat.ML].

Yoann VALERO

Doctorat : Optimisation et Sûreté des Systèmes

Année 2023

Intelligence artificielle pour la modélisation de parcours individuels

Les processus métiers sont indissociables des structures, organisations et entreprises pour leur bon fonctionnement, afin d'apporter un cadre à l'exécution de leur production et de la rendre aussi efficace que possible. Cependant, lors de leur application sur le terrain, ces processus finissent inévitablement par se complexifier, dériver, se modifier, et permettre des comportements uniques dans les unités qui les traversent. Ces unités et leurs parcours sont enregistrés informatiquement dans des journaux d'événements, composés au minimum d'identifiants uniques pour ces unités, des activités par lesquelles ces unités sont passées et de la date à laquelle les unités ont effectué les activités de leurs parcours. À partir de cette donnée, la fouille de processus permet d'extraire le processus tel qu'il est effectué dans la réalité et d'évaluer la conformité du processus réel au processus tel qu'il a été conceptualisé en amont. La fouille de processus permet également de détecter les points de blocages d'un processus. L'étape suivante consiste en la capacité de prédire le parcours d'une unité encore en cours dans un processus. Cette thèse s'attèle à développer une méthode d'apprentissage profond permettant la prédiction de parcours d'unités peu importe leur niveau d'avancement dans un processus, ainsi que la simulation de leurs comportements selon des conditions de production au choix d'utilisateurs. Enfin, Cette thèse propose une méthode de mesure, d'évaluation et de suivi de la dérive des processus, à partir de la donnée brute et au regard du modèle prédictif développé.

Mots clés : réseaux neuronaux (informatique) – modélisation prédictive – logistique (gestion) – simulation, méthodes de.

Artificial Intelligence for Individual Journey Modelling

Business processes are essential to the smooth running of structures, organisations and companies, providing a framework for the execution of their production and making it as efficient as possible. However, during their real-life application, these processes inevitably end up becoming more complex, drifting, changing, and allowing unique behaviours in the units that pass through them. These units and their paths are recorded in computerised event logs, consisting at least of unique identifiers for these units, the activities through which these units pass, and the date on which the units pass through these activities. From this data, process mining can extract the process as it is performed in reality and assess the conformity of the actual process to the previously theorized process. Process mining can also be used to detect bottlenecks and other low performance points in a process. The next step is to be able to predict the path of a unit still in progress. This thesis sets on developing a deep learning method for predicting the remaining path to be taken by ongoing units at any stage in a process, as well as simulating their behaviour according to production conditions chosen by users. Finally, this thesis proposes a method for measuring, evaluating and monitoring process drift, both through raw data only as well as through the lens of the developed deep learning model.

Keywords: neural networks (computer science) – predictive analytics – logistics (management) – simulation methods.

Thèse réalisée en partenariat entre :

ANRT ASSOCIATION NATIONALE RECHERCHE TECHNOLOGIE — Cifre Conventions — QAD — utt UNIVERSITÉ DE TECHNOLOGIE TROYES

Ecole Doctorale "Sciences pour l'Ingénieur"