

الجمهورية الشعبية الديمقراطية الجزائرية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la
Recherche Scientifique



École supérieur en informatique
Sidi Bel Abbés 8 mai 1945

Laboratoire d'Informatique de
Paris Nord Université Sorbonne
Paris Nord

Mémoire de fin d'étude

En vue de l'obtention du diplôme : Ingénieur

Filière : Informatique

Spécialité : Système d'information et web (SIW)

Thème

Conception et réalisation d'un outil d'analyse et de
visualisation des séries temporelles
- VisIo -

Présenté par : Sidi Mohamed Hicham Zekri

Soutenu le : 20/09/2020

Devant le jury composé de :

Mr. Nabile Keskes
Mr. Mohamed Kechar
Mr. Mustapha Lebbah
Mme. Sakina Rim Benabi

Docteur	Président
Docteur	Encadreur
Professeur	Co-Encadreur
Docteur	Examinateuse

Année Universitaire : 2019/2020

RÉSUMÉ

Dans de nombreuses applications du monde réel, les données sont générées au fil du temps, constituant une série temporelle. Avec l'augmentation de ces données leur traitement est devenue une nécessité. Pour un analyste la visualisation des séries est une tache préliminaire pour n'importe quel traitement de ces données. cette dernière, nous permis de comprendre les données, comprendre les résultats, et de détecter des modèles visuels dans les données.

Il existe plusieurs outils pour la visualisation des séries temporelles, mais généralement, ces outils soit il ne traite pas les données massives, soit ils ne sont pas directement utilisables pour les séries temporelles.

A cet effet, dans notre projet de fin d'étude nous avons conçu et implémenté un outil de visualisation et d'analyse de séries temporelles. Ce dernier permet, dans un premier temps, de charger des données provenant de différentes sources SQL en insérant simplement leur type et leur emplacement. Il fournit aussi un espace pour générer des pipelines d'algorithmes d'analyse de données pour les séries temporelles, en tenant compte des algorithmes simples qui s'exécute dans un même serveur , et distribués en particulier les algorithmes qui fonctionnent sur le framework Apache SPARK. Enfin, il permet de visualiser les résultats à n'importe quel étape sous forme de graphiques 2D (graphique linéaire et nuage de points). Tous ces implémentations sont développées à l'aide d'une architecture web microservices à la fois extensible et simple à déployer.

Mots clés : Séries temporelles, visualisation, microservices, analyse de données, algorithme distribué

ABSTRACT

In many real-world applications, data is captured over time, forming a time series, and with its increase, processing has become a trend. For an analyst, the visualization of the series is a preliminary task for any processing of these data, it allows us to understand it, understand our results, and sometimes even detect visual patterns.

There are several tools for time series visualization but generally, these tools either do not process massive data or they are not directly usable for time series.

For our internship, we set up a time series visualization and analysis tool. The latter allows, in a first step, to load data from different SQL sources by simply inserting their type and location. It also provides a space to generate pipelines of data analysis algorithms for time series, taking into account simple algorithms that run on the same server or centralized and distributed ones, in particular, the algorithms that run on the SPARK engine. Finally, it allows visualizing the results at any stage in the form of 2D graphics (line graph, scatter plot ...). All these works are developed using a micro-services web architecture that is both extensible and easy to deploy.

Keys words : Time series, visualization, microservices, data analysis, ditrbuted algorithms

الملاخص

خلال تدريينا، قمنا بإعداد أداة لتحليل وتصوير المتسلسلات الزمنية. تسمح هذه الأداة بتحميل البيانات من مصادر مختلفة عن طريق إدخال نوعها ومكانها ببساطة. كما توفر هذه الأخيرة مساحة لإنشاء مخطط من الخوارزميات، مع مراعات الخوارزميات العادية (التي تعمل في نفس الجهاز او المركزية) والخوارزميات الموزعة (بالتحديد التي تعمل على محرك سبارك). وأخيراً يسمح نظامنا بتكبير النتائج في مختلف مراحل التحليل في شكل رسومات ثنائية الابعاد. تم تطوير هذا النظام باستخدام تقنية الخدمات الصغيرة التي يمكن توزيعها ونشرها بسهولة. خلال تدريينا، قمنا بإعداد أداة لتحليل وتصوير المتسلسلات الزمنية. تسمح هذه الأداة بتحميل البيانات من مصادر مختلفة عن طريق إدخال نوعها ومكانها ببساطة. كما توفر هذه الأخيرة مساحة لإنشاء مخطط من الخوارزميات، مع مراعات الخوارزميات العادية (التي تعمل في نفس الجهاز او المركزية) والخوارزميات الموزعة (بالتحديد التي تعمل على محرك سبارك). أخيراً يسمح نظامنا بتصوير النتائج في مختلف مراحل التحليل في شكل رسومات ثنائية الابعاد. تم تطوير هذا النظام باستخدام تقنية الخدمات المصغرة التي يمكن توزيعها ونشرها بسهولة.

كلمات مفتاحية: المتسلسلات الزمنية ، التصوير ، الخدمات المصغرة ، تحليل البيانات ، الخوارزميات الموزعة

REMERCIEMENT

Durant la réalisation de ce travail le monde à passer par une période très difficile face au virus **COVID-19**. A cet effet je commence par remercier le corps de santé qui combattent pour notre bien être et pour protéger nos proches.

Je voudrais aussi adresser toute ma gratitude et mon remerciement à mon encadreur, **M. KECHAR Mohamed**, et mon directeur du stage **M. LEBBAH Mustapha**, pour leur patience, leur disponibilité et surtout leurs judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je tiens à témoigner toute ma reconnaissance a **M. Mekri Zouaoui** , qui m'a beaucoup appris de côté mathématique du projet et **Anthony cautant, Florent Forest, Geal Beck, Walid Attaoui, Etienne Goffinet** , les membres de l'équipe **A3**, et tout le personnelles du laboratoire **LIPN**, pour leur participation dans ce projet.

Un grand merci à ma mère **Ahmane Baya** et mon père **Zekri Mustapha**, pour leur amour, leurs conseils ainsi que leur soutien inconditionnel, à la fois moral et économique, qui m'a permis de réaliser les études que je voulais et par conséquent ce mémoire.

Je voudrais exprimer ma reconnaissance envers mes chére amies pour leur aides technique et morale pendant tout mon cursus universitaire.

Enfin, nous tenons à saisir cette occasion et adresser nos profonds remerciements aux personnelles administratives et pédagogiques de l'**École Supérieure d'Informatique 8 Mai 1945**.

TABLE DES MATIÈRES

1	Introduction générale	15
1.1	Introduction	15
1.2	Université Sorbonne Paris Nord	16
1.2.1	Présentation	16
1.2.2	Institut Galilée	16
1.2.3	Laboratoire d'informatique de Paris Nord LIPN	17
1.3	Organisation du mémoire	18
2	Les séries temporelles	19
2.1	Introduction	19
2.2	Définition des séries temporelles	19
2.3	Types des séries temporelles	20
2.3.1	Séunrieis et multivariées	21
2.3.2	Séries stationnaires et non-stationnaires	22
2.3.3	Séries discrètes et continues	23
2.4	Décomposition des séries temporelles	24
2.4.1	Tendance T_t	25
2.4.2	Saisonnalité S_t	25
2.4.3	Cycles C_t	26

2.4.4	Bruit ϵ_t	26
2.4.5	Types de décomposition des séries temporelles	27
2.5	Conclusion	27
3	Visualisations séries temporelles	28
3.1	Introduction	28
3.2	Visualisation des séries temporelles multivariées	29
3.3	Préparation des données	30
3.4	Opérations pour la visualisations	31
3.4.1	Lissage des données	31
3.4.2	Weighted Moving Average (WMA)	32
3.4.3	Réduction de dimensionnalité	33
3.4.4	Relation entre les variables	36
3.5	Discrétisation	36
3.6	Tracer une série temporelle	37
3.6.1	graphique linéaire (Line Chart)	37
3.6.2	La carte thermique (Heatmap)	37
3.6.3	Le diagramme de barres (Bar Charts)	38
3.6.4	Graphe de points (Scatter)	39
3.6.5	Le graphique de flux (Stream Graph)	39
3.7	Conclusion	40
4	Étude de l'existant	42
4.1	Introduction	42
4.2	Tableau	42
4.3	Grafana	43
4.4	Dial-A-Cluster (DAC)	44
4.5	SDRVR	45
4.6	Bipeline	46
4.7	Synthèse	47

4.8	Conclusion	48
5	VisIo : outil de visualisation pour les séries temporelles	51
5.1	Introduction	51
5.2	Analyse des besoins	52
5.3	Architecture	53
5.3.1	Microservices	55
5.3.2	Communication entre les microservices	56
5.3.3	Base de données partagée	57
5.4	Conception	61
5.5	Implémentation	61
5.5.1	Configuration manuellement de la partie SPAKR	61
5.5.2	Gestion des sources de données	62
5.5.3	Gestion du pipeline	64
5.5.4	Visualisation graphique	65
5.6	Déploiement	67
5.7	Environnement expérimental	68
5.8	conclusion	70
6	Conclusion et perspective	73
6.0.1	Ajouter un algorithme SPARK	75
6.0.2	Ajouter un algorithme Simple	75
6.0.3	Ajouter l'algorithme a notre interface	76

TABLE DES FIGURES

1.1	Organigramme de l’Institut Galilée	17
2.1	Un exemple de deux fenêtres sur une série	21
2.2	série non-stationnaire	23
2.3	Série stationnaire	23
2.4	Un exemple d’une série discrète	24
2.5	Un exemple d’une série contenue	24
2.6	Les trois types de tendances dans les séries temporelles	25
2.7	Un exemple sur d’une série saisonnière qui présente le nombre de vestes vendu par un magasin chaque mois pendant trois ans	26
2.8	Un exemple d’une série cyclique	26
3.1	La carte figurative des pertes successives en hommes de l’Armée française dans la campagne de Russie en 1812-1813 est un exemple de diagramme de Sankey (même si Minard a utilisé cette technique avant l’ingénieur irlandais Sankey). [4]	29
3.2	Exemple de variété	34
3.3	Exemple line chart	38
3.4	Exemple Heatmap	39
3.5	Exemple barchart	40

3.6	Exemple scatter	41
3.7	Exemple Stream Graph	41
4.1	Exemple de l'interface graphique de Tableau	43
4.2	Exemple de l'interface graphique de Grafana	44
4.3	l'interface utilisateur. Nous montrons ici le DAC fonctionnant dans Firefox sur un PC Windows.	45
4.4	l'interface utilisateur SDRVR	46
4.5	Interface utilisateur Bipeline. (Figure de couleur en ligne)	47
5.1	Illustration pour l'enchaînement des bloc visuel	53
5.2	Diagramme de cas d'utilisations	53
5.3	Diagramme de séquences pour un scénario	54
5.4	Diagramme des routes et paramètres pour les microservices des algorithmes spark	56
5.5	Diagramme des routes et paramètres pour les microservices des algorithmes simple	56
5.6	Diagramme des routes et paramètres pour les microservices de gestion	57
5.7	Diagramme des routes et paramètres pour les microservices qui gère l'interface	58
5.8	Architecture Microservices de VisIo	59
5.9	Communication microservices et base de données partagé	60
5.10	Schéma base de données partagé	60
5.13	Capture sur la fonctionnalités : configuration du serveur	62
5.14	Capture sur la fonctionnalités : ajoute de session SPARK	62
5.15	Capture sur la fonctionnalités : ajoute de base de données	63
5.16	Capture sur la fonctionnalités : gestion des sources de données	63
5.17	pipeline pour les UTS	65
5.18	Pipeline pour les MTS	65
5.19	Résultat pipeline pour UTS	66

5.20 Résultat pipeline pour les MTS linechart	66
5.21 Résultat pipeline pour les MTS scatterchart	67
5.11 Diagramme de classes	71
5.12 L'implémentation de deux algorithmes en microservice	72
5.22 Diagramme de déploiement avec Kubernetes	72

LISTE DES TABLEAUX

2.1	Exemple d'une série temporelle à deux variables (température et humidité)	20
2.2	Une série univariée (température)	21
2.3	Une série multivariée (température, humidité et vent (km/h))	22
4.1	Comparaison entre les outils suivant les mesures : Disponibilité, open-source, déploiement/installation, Les graphes proposés	49
4.2	Comparaison entre les outils suivant les mesures : Interactivité, Bigdata, algorithmes de pré-traitement, source de données	50
5.1	Etude comparative entre Kubernetes et Docker Swarm	67
6.1	Framework for micro-services architecture	77

ACRONYMES

- TS** série temporelle
- MTS** série temporelle multi-variée
- UTS** série temporelle uni-variée
- SMA** Moyenne mobile SIMPLE
- WMA** Moyenne mobile pondérée
- SES** Lissage exponentiel simple
- DES** Lissage exponentiel double
- TES** Lissage exponentiel triple
- PCA** Analyse des composantes principales
- LDA** Analyse discriminante linéaire
- MDS** Positionnement multidimensionnel
- t-SNE** t-distributed stochastic neighbor embedding
- UMAP** Uniform manifold approximation and projection
- NMF** Non-negative matrix factorization
- DAC** Dial-A-Cluster
- SDRVR** smoothing, Dimensionality reduction and variable relationship tool
- MS** MicroService
- RPC** Remote Procedure Call

CHAPITRE 1

INTRODUCTION GÉNÉRALE

1.1 Introduction

La visualisation des données temporelles est l'une des techniques essentiel dans tout projet d'analyse de données pour extraire des informations des données. Elle aide les analyste à mieux comprendre les données et les résultats en les transformant en éléments visuels tels que des graphiques et des diagrammes. Ces éléments visuels révèlent le modèle, la forme et les statistiques sur les données, qu'il est difficile de les reconnaître uniquement en regardant les données sources. Notre outil permet la visualisation des séries temporelles. Cette solution rassembler la majorité des fonctions des autres solutions existantes, de plus, elle permet de traité en plus les données massives, et traitement distribué. Ce dernier est développer d'un manier extensible avec une architecture web (client léger) en utilisions les derniers technologies de visualisation interactive et d'analyse de données.

1.2 Université Sorbonne Paris Nord

1.2.1 Présentation

L'université Sorbonne Paris Nord est l'une des universités qui ont succédé à la Sorbonne après l'éclatement de l'Université de Paris en treize universités autonomes en 1968.

Elle réaffirme son identité qui met en lumière la symbolique de son histoire, la richesse de son activité, ainsi que son ancrage territorial revendiqué avec fierté et conviction.

L'université Sorbonne Paris Nord accueille plus de 24 000 étudiants en formation initiale ou continue, dans tous les domaines : Santé, Médecine et Biologie humaine -Lettres, Langues, Sciences Humaines et des Sociétés –Droit, Sciences politiques et sociales –Sciences de la communication –Sciences économiques et de gestion. Elle propose ainsi à ses 24 000 étudiants une offre de formation pluridisciplinaire, résolument tournée vers le monde professionnel.

1.2.2 Institut Galilée

L’Institut Galilée, composante en sciences dures de l’Université Sorbonne Paris Nord, regroupe des formations en licence et master, une école d’ingénieurs – Sup Galilée – et sept laboratoires de recherche. Cette structure originale permet d'accroître la synergie nécessaire entre recherche, formations généralistes et formations professionnelles et technologiques, permettant ainsi d'offrir à nos étudiants les meilleurs cursus.

Les laboratoires de l’Institut Galilée mènent des travaux de recherche innovants et de haut niveau dans un large spectre de champs scientifiques allant des mathématiques à la chimie, en passant par l’informatique, la physique, les sciences pour l’ingénieur, les matériaux et bio-matériaux.

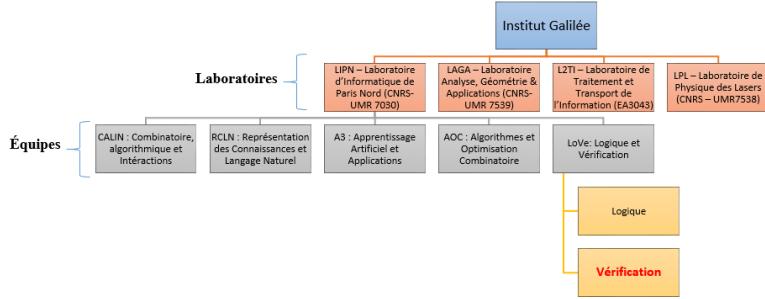


FIG. 1.1 : Organigramme de l’Institut Galilée

La quasi-totalité de ces laboratoires est associée aux grands organismes de recherche (CNRS et INSERM). Ils bénéficient d’un fort rayonnement aux niveaux national et international, comme l’attestent les résultats des dernières évaluations nationales et leurs participations à six LABEX et un EQUIPEX soutenus par le programme des investissements d’avenir.

L’Institut Galilée se consacre à l’enseignement et à la recherche des mathématiques, de la physique, de la chimie et de l’informatique, depuis 41 ans.

Une véritable politique de recherche qui s’appuie sur 7 laboratoires reconnus et le développement des formations professionnelles sont les atouts majeurs de l’Institut scientifique. La figure (Fig. 1.1) montre les différents laboratoires associés à l’Institut Galilée.

1.2.3 Laboratoire d’informatique de Paris Nord LIPN

Le Laboratoire d’Informatique de Paris-Nord (LIPN), créé en 1985, est lié au CNRS depuis janvier 1992. Les recherches effectuées au LIPN portent particulièrement sur l’algorithmique, le langage naturel, la logique, le génie logiciel et l’apprentissage¹.

Pendant mon stage, je faisais partie à l’équipe A3 (Apprentissage Artificiel et Applications) qui travaille sur ces axes de recherche : Apprentissage pour l’aide à la décision,

¹<https://lipn.univ-paris13.fr/accueil/presentation/le-laboratoire/>

apprentissage non supervisé de représentations pour le transfert et la collaboration, Apprentissage dans les graphes et apprentissage de modèles topologiques à partir de données massives. J'ai principalement travaillé dans le groupe de l'apprentissage non supervisé dont leurs activités sont concentrées essentiellement sur le traitement des séries temporelles.

1.3 Organisation du mémoire

Dans le chapitre 02, nous présentons les notions de base pour les séries temporelles. Leur définitions et leur typologies : Uni-variée / multi-variée, stationnaire / non stationnaire et discrète / conteneuse. Puis les différentes décompositions des séries telles que : Tendances, saisonnalité, cycles et le bruit.

Le chapitre 03, nous passons en revue l'histoire de la visualisation des TS et quelque définitions nécessaire pour les différentes techniques de visualisation des TS qui sont : le lissage, la réduction de dimension et la relation entre les variables. Enfin, nous présentons les différents graphes qui sont utilisés pour la visualisation des TS.

Le chapitre 04, on aura une étude comparative entre quelques solutions existant pour la visualisation des TS qui sont : Tableau, Grafana, CAD, SDRVR et Pipeline . Selon les critères suivants : disponibilité en ligne, la gratuité d'utilisation de code source, déploiement et l'installation, adaptation pour MTS, graphes proposés par l'outil, interactivité des graphes, traitement des données massifs, algorithmes de prétraitement, source de données.

Enfin, nous décrivons notre solution **VisIo** qui permis la visualisation des Ts et qui ajoute des nouvelles fonctionnalités telles que le traitement des données massive et la gestion des pipelines.

CHAPITRE 2

LES SÉRIES TEMPORELLES

2.1 Introduction

Dans notre vie on dit toujours qu'il faut apprendre des événements du passé pour éviter de refaire les mêmes erreurs dans le futur. Pour appliquer cette règle, il faut collecter les données générées dans le passé et voir leur évolution dans le temps. Ce qui par conséquence nous aidera à comprendre le passé. On appelle ces données qui dépendent du temps : les données temporelles ou les séries temporelles. On peut utiliser ces données dans différents domaines, tel que, les affaires commerciales, les tests des machines, la maintenance de matériels, et dans n'importe quel phénomène qui change en fonction du temps. Dans ce chapitre nous allons définir les séries temporelles et leur différents types : stationnaire et non stationnaire, univariées et multivariées, discrètes et continues. Ensuite nous présenterons les différentes caractéristiques d'une série, telles que sa tendance, sa saisonnalité, sa cyclicité et, son bruit.

2.2 Définition des séries temporelles

Dans cette section nous donnons quelques définitions sur les séries temporelles.

Définition 2.2.1. Une série temporelle (série chronologique), est une suite d'observations finies indexées par le temps. En fonction des cas d'utilisation, l'index temporel peut correspondre à différentes granularités (seconde, milliseconde, minute, jour, année, ...) ou être purement symbolique (t_1, t_2, \dots). Formellement, une série temporelle est définie comme une suite finie. $x_1, x_2, x_3, \dots, x_n$, telle que n est appelé la longueur de la série et chaque élément de la suite est indexé par un moment t qui indique le temps [3].

Le tableau 2.1 est un exemple d'une série temporelle.

série	température	humidité
x_1	10	14
x_2	25	12
x_3	40	20
x_4	-1	10
x_5	45	43
$x_{6\dots}$
x_n	14	50

TAB. 2.1 : Exemple d'une série temporelle à deux variables (température et humidité)

Définition 2.2.2. Fenêtre de données (Matrice WM), étant donné une série temporelle T de longueur fixe m (un sous-ensemble d'un flux de données de série temporelle) et x_t un élément de la série au moment t , une matrice WM de toutes les sous-séquences possibles de longueur k peut être construite en déplaçant une fenêtre glissante de taille k sur T et en plaçant la sous-séquence $x_p = x_p, x_{p+1}, \dots, x_{p+k}$ comme une ligne dans la matrice WM . La Figure 2.1¹ montre un exemple des fenêtres [23].

2.3 Types des séries temporelles

Selon certaines caractéristiques, plusieurs types de séries temporelles peuvent exister. On distingue trois caractéristiques : le nombre de variables (univariées ou multivariées),

¹<https://stats.stackexchange.com/questions/390674/using-trend-as-a-feature-in-time-series-sliding-window?rq=1>

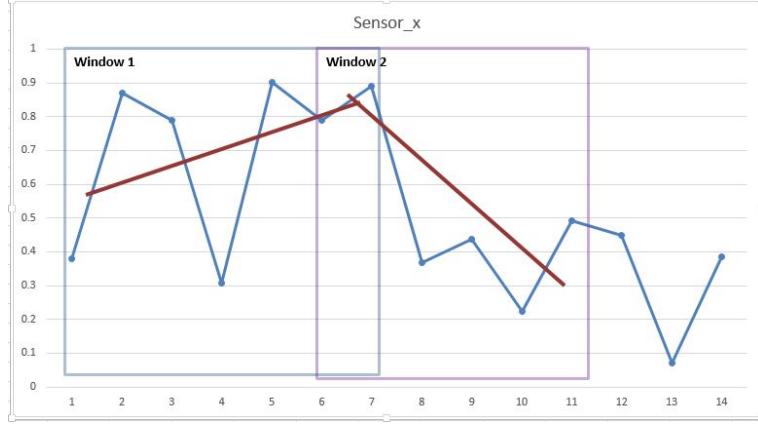


FIG. 2.1 : Un exemple de deux fenêtres sur une série

le caractère régulier ou stationnaire de la série, et le caractère discret ou continu de l’index temporel.

2.3.1 Séries univariées et multivariées

Dans le contexte de nombre de variables [1], si une série a une seule variable donc $x(t)$ est un scalaire on l’appelle série univariée ou bien une série unidimensionnelle. Le tableau 2.2, montre un exemple d’une série univariée. Dans le cas contraire, si une série a plusieurs variables, donc $x(t)$ est un vecteur, on l’appelle une série multivariée ou bien série multidimensionnelle. Le tableau 2.3 montre un exemple de cette dernière.

Dans le cas présenté dans le tableau 2.2, si on prend la série à l’instant $t = 1$ on aura

série	température
x_1	10
x_2	25
x_3	40
x_4	-1
x_5	45
$x_{6\dots}$...
x_n	14

TAB. 2.2 : Une série univariée (température)

une seule valeur 10 qui décrit une la température, mais dans les cas du tableau 2.3, si on prend l’instant $t = 1$ on aura un vecteur de trois valeurs 0, 14 et 10 qui concerne la

série	température	humidité	vent (km/h)
x_1	10	14	10
x_2	25	12	12
x_3	40	20	14
x_4	-1	10	6
x_5	45	43	5
$x_{6\dots}$
x_n	14	50	15

TAB. 2.3 : Une série multivariée (température, humidité et vent (km/h))

température, l'humidité et la vitesse de vent respectivement.

2.3.2 Séries stationnaires et non-stationnaires

La stationnarité est l'un des concepts les plus importants dans le domaine des séries temporelles. Nous pouvons définir une série temporelle stationnaire comme une série dont les propriétés - moyennes, variance et covariance - ne changent pas avec le temps [9].

Dans la Figure 2.2², les séries présentées sont non stationnaires puisque dans le graphe 01 la moyenne varie avec le temps. Dans le graphe 02, la variance change avec le temps et dans le graphe 03 la covariance est une fonction du temps. Par contre dans la Figure 2.3², nous avons une série stationnaire puisque les trois propriétés ne changent pas avec le temps.

Les trois paramètres, savoir, la moyenne, la variance et la covariance, sont calculés respectivement par les Équations 2.1, 2.2 et 2.3 respectivement.

$$\text{moyenne} = \frac{1}{n} \times \sum_{i=1}^n x_i \quad (2.1)$$

$$\text{variance} = \frac{1}{n} \times \sum_{i=1}^n (x_i - \text{moyenne})^2 \quad (2.2)$$

$$\text{covariance}(X, Y) = E[(X - E[X])(Y - E[Y])] \quad (2.3)$$

²<https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python/>

tel que $E[\cdot]$ est l'espérance mathématique.

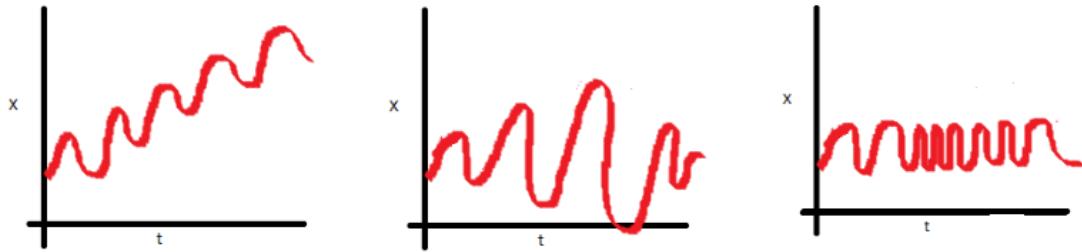


FIG. 2.2 : série non-stationnaire

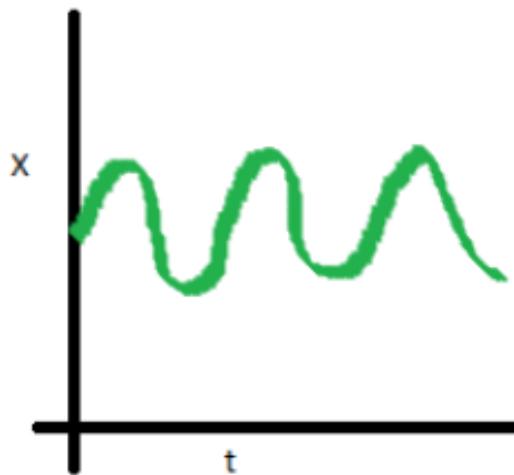


FIG. 2.3 : Série stationnaire

2.3.3 Séries discrètes et continues

Une série temporelle discrète est une série constituée de points (ou de vecteurs pour le cas multivarié) de données séparés par des intervalles de temps, ou mesurés à des moments distincts, séparés les uns des autres dans le temps. La Figure 2.4³ montre un exemple d'une série discrète.

Une série temporelle continue est définie sur un intervalle continu de temps et pour chaque élément de l'intervalle on a une valeur, sans discontinuité. La Figure 2.5

³<https://www.researchgate.net/publication>

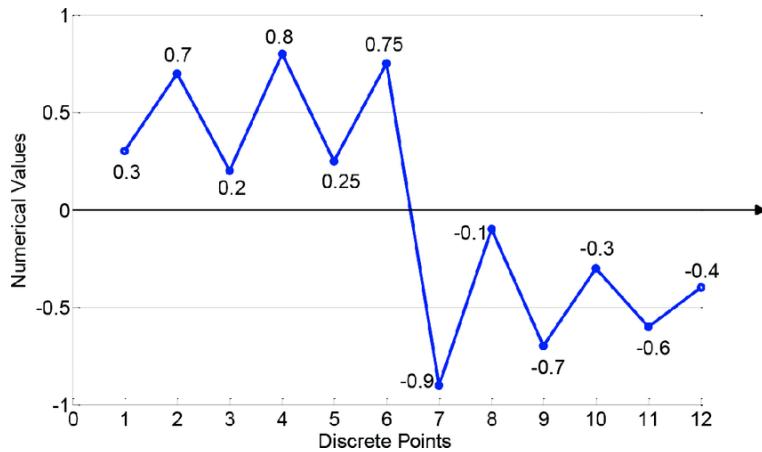


FIG. 2.4 : Un exemple d'une série discrète

⁴ montre une série continue. Il est possible de passer d'une série continue à une série discrète en appliquant une technique dite "discrétisation", impliquant généralement des pertes d'information. L'inverse est également possible via des techniques dites "interpolation" [1].

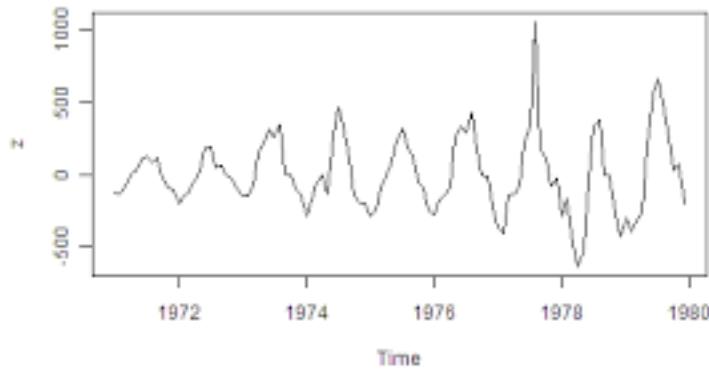


FIG. 2.5 : Un exemple d'une série contenue

2.4 Décomposition des séries temporelles

Les données de séries chronologiques peuvent présenter une variété de modèles, et il est souvent utile de diviser une série chronologique en plusieurs composantes, chacune

⁴<https://perso.univ-rennes1.fr/valerie.monbet/STM1/CoursST2017.pdf>

représentant une catégorie de modèle sous-jacente. Dans les sections suivantes, nous allons présenter les quatre types de composantes des séries temporelles : tendance, saisonnalité, cycles et le bruit (erreur).

2.4.1 Tendance T_t

Dans l'analyse des séries temporelles, une tendance est un déplacement ou un mouvement progressif vers une valeur relativement plus ou moins élevée sur une longue période de temps, on distingue trois types de tendances : tendance montante, tendance descendante, tendance horizontale. Les trois types de tendance sont illustrées dans la Figure 2.6⁵.

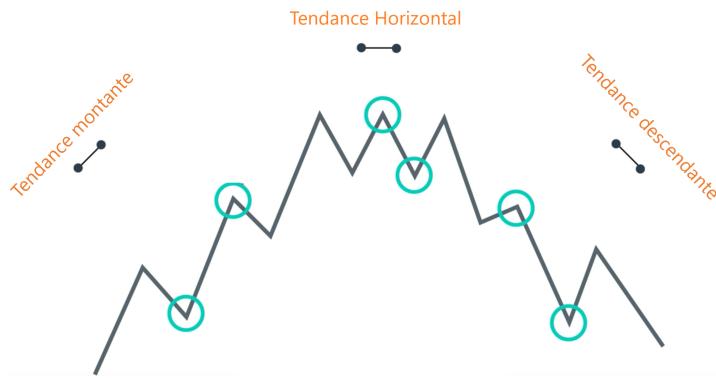


FIG. 2.6 : Les trois types de tendances dans les séries temporelles

2.4.2 Saisonnalité S_t

Une série temporelle qui présente un schéma répétitif à intervalles fixes au cours d'une période d'un an est dite saisonnière. L'exemple d'un magasin qui vend des vestes. Ces magasins ont un taux de vente faible pendant la période d'été et un taux plus au moins élevé dans les autres saisons, dans la Figure 2.7 on remarque une répétition dans le schéma chaque année [17].

⁵<https://classroom.udacity.com/courses/ud980/lessons/7bcd2c42-a582-4e07-967cf93a52de59f1/concepts/66f5b8de-a249-4605-8dd6-e0e87748bfcb>

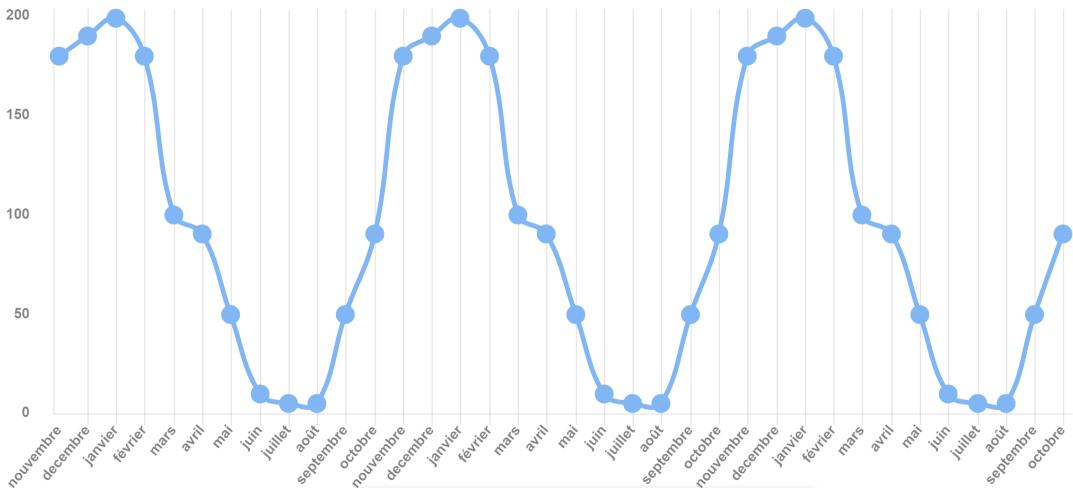


FIG. 2.7 : Un exemple sur d'une série saisonnière qui présente le nombre de vestes vendu par un magasin chaque mois pendant trois ans

2.4.3 Cycles C_t

Une série temporelle qui présente un schéma cyclique, est une série saisonnière telle que les saisons ne se produisent pas à fréquence fixe. La Figure 2.8 montre un schéma d'une série cyclique.



FIG. 2.8 : Un exemple d'une série cyclique

2.4.4 Bruit ϵ_t

Le bruit statistique correspond à des fluctuations de nature irrégulière, aléatoire et inexpliquée. Donc il représente toutes les autres valeurs qui ne sont pas représentatives. Il s'agit par exemple d'erreurs de mesure ou bien d'erreurs statistiques. Pour simplifier

l'analyse des séries temporelles en général, il est d'usage de s'abstraire du bruit, ce qui est une tâche difficile en pratique.

2.4.5 Types de décomposition des séries temporelles

Cette décomposition peut être additive (voir l'équation 2.4) ou multiplicative (voir l'équation 2.5). Lorsque nous décomposons une série chronologique en composantes, nous combinons généralement la tendance et le cycle en une seule composante tendance-cycle

$$Y_t = S_t + TC_t + \epsilon_t \quad (2.4)$$

$$Y_t = S_t \times TC_t \times \epsilon_t \quad (2.5)$$

Où : Y_t : La donnée ; S_t : Composante saisonnière ; TC_t : Composante tendance-cycle ; ϵ_t : Le bruit ou bien l'erreur.

La décomposition additive est la plus appropriée si l'ampleur des fluctuations saisonnières, ou la variation autour de la tendance-cycle, ne varie pas avec le niveau de la série chronologique. Lorsque la variation du schéma saisonnier, ou la variation autour de la tendance-cycle, semble être proportionnelle au niveau de la série temporelle, alors une décomposition multiplicative est plus appropriée.

2.5 Conclusion

Dans ce chapitre ,nous avons présenté les notions principales sur les données temporelles. Dans le chapitre suivant, nous détaillerons les différentes techniques de visualisations des séries temporelles multivariées.

CHAPITRE 3

VISUALISATIONS SÉRIES TEMPORELLES

3.1 Introduction

Dans le domaine d'analyse des séries temporelles, il existe plusieurs méthodes pour extraire l'information des données . L'une de ces méthodes est la visualisation de données par des graphes visuels. Mais parfois, nous nous retrouvons avec des grands problèmes pour visualiser ces données. Deux problèmes sont notamment importants : d'une part, le problème des valeurs aberrantes, et, d'autre part, celui de la grande dimensionnalité des données, c'est-à-dire si on a n variables donc on aura besoin de n -dimensions pour pouvoir visualiser ces données ce qui est impossible si on a plus de trois variables. C'est le cas des séries temporelles multi-variées. Dans cette partie, nous allons en premier lieu aborder la notion de visualisation des séries temporelles multivariées. Nous présenterons ensuite quelques opérations pour résoudre une partie des problèmes susmentionnés, telles que : le lissage de données, la réduction de dimensionnalité, comment faire la relation entre les variables d'une même série et la discrétisation de données, qui vont nous permettre de visualiser nos séries et faire face a ces problèmes. Enfin, nous aborderons les méthodes pour tracer ces séries.

3.2 Visualisation des séries temporelles multivariées

La visualisation de données de séries chronologiques n'est pas une science nouvelle. Les techniques de la visualisation de ces ensembles de données existent depuis de nombreuses années et remontent à jusqu'au XVIII^e siècle. La représentation classique de Minard sur l'armée de Napoléon lors de la campagne contre la Russie présentée en 1812 (voir la Figure 3.1) est considérée comme l'un des meilleurs exemples de visualisation de données chronologiques [4]. Souvent, quand on parle de visualisation d'une série temporelle, nous pensons à des graphes statiques que nous avons déjà vu tels que les graphes à Courbes, camemberts, en bâtons ou bien nuages de points etc. Mais comme il est défini dans la section 2.3.1, une série temporelle multi-variée (MTS) est un ensemble de points indexés par le temps, et pour chaque point x_t il existe un vecteur de valeurs de sortie. Pour visualiser une MTS, il faut nécessairement la tracer dans un plan 2D ou un espace 3D simulé pour pouvoir l'analyser. Dans notre travail, nous allons nous concentrer sur les représentations 2D .

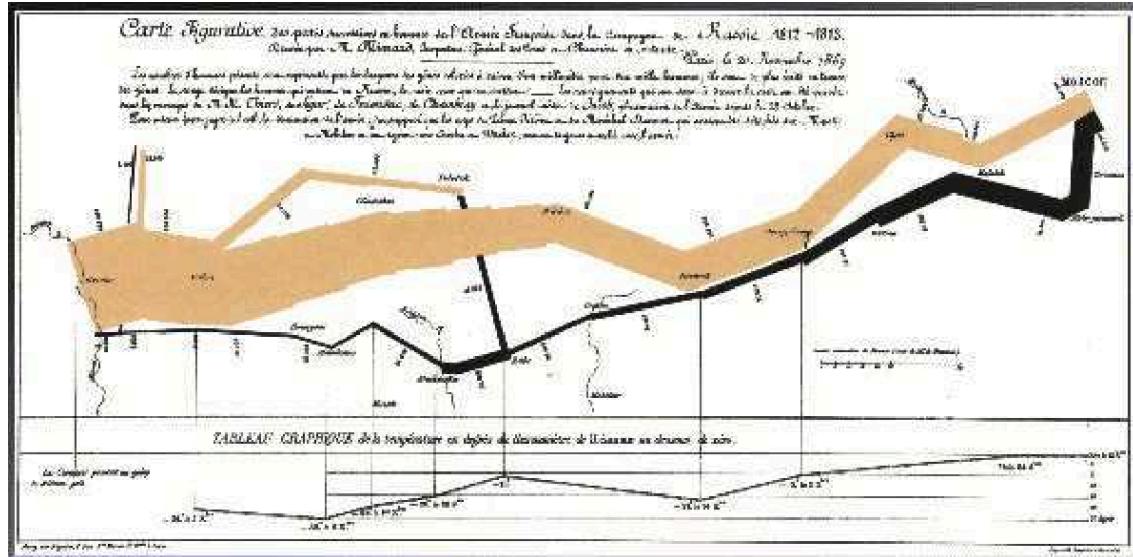


FIG. 3.1 : La carte figurative des pertes successives en hommes de l'Armée française dans la campagne de Russie en 1812-1813 est un exemple de diagramme de Sankey (même si Minard a utilisé cette technique avant l'ingénieur irlandais Sankey). [4]

3.3 Préparation des données

Souvent dans l'apprentissage sur les séries, nous cherchons à trouver des tendances ou bien un certain schéma dans les données. Ce qui reviens a comparer les caractéristiques des points de données. Cependant, il y a un problème lorsque les caractéristiques sont à des échelles radicalement différentes. Nous devrions alors appliquer une normalisation des données pour nous assurer que toutes les variables ont le même échelle, de sorte que la variable ayant des valeurs élevées ne dominera pas les séries temporelles aux valeurs plus faibles lors d'analyses quantitatives par exemple. Dans cette partie, nous allons définir deux algorithmes de normalisation connue : Normalisation min-max et normalisation du z-score.

Définition 3.3.1. NORMALISATION MIN-MAX :

Pour chaque variable, la valeur minimale de cette variable est transformée en 0, la valeur maximale est transformée en 1 et toutes les autres valeurs sont transformées en un réel entre 0 et 1 (voir l'équation 3.1)[18]

$$V_{norm} = \frac{V_{org} - min}{max - min} \quad (3.1)$$

Définition 3.3.2. NORMALISATION DU Z-SCORE :

Si une valeur est exactement égale à la moyenne de toutes les valeurs de l'élément, elle sera normalisée à 0. Si elle est inférieure à la moyenne, ce sera un nombre négatif, et si elle est supérieure à la moyenne, ce sera un nombre positif. La taille de ces nombres négatifs et positifs est déterminée par l'écart-type de la caractéristique originale. Si les données non normalisées présentent un écart type important, les valeurs normalisées seront plus proches de 0 (voir l'équation 3.3) [18].

$$V_{norm} = \frac{V_{org} - \mu}{\sigma} \quad (3.2)$$

tel que μ : la moyenne et σ : l'écart-type.

3.4 Opérations pour la visualisations

Pour commencer n'importe quel processus de visualisation, nous avons besoin d'effectuer quelques opérations sur les données pour les rendre prêtes à visualiser. Ces opérations sont généralement : le lissage de données, la réduction de dimension, trouver les relations entre les variables et la discréétisation. Chacune de ces opérations peut être appliquée seule ou bien en combinaison avec l'autres, en fonction du besoin et de la nature de données. Par la suite, nous présentons ces opérations et quelques définitions qui seront utiles dans la partie suivante.

3.4.1 Lissage des données

Le lissage de données est une technique qui permet d'identifier des schéma importants dans les données tout en réduisant le bruit dans celles-ci. On peut aussi utiliser le lissage dans le processus de visualisation en réduisant le bruit pour rendre nos séries claires et pour pouvoir détecter les tendances dans les données[3].

Simple Moving Average (SMA)

Dans cette méthode, nous utilisons la technique de fenêtre coulissante (voir section 2.2.2), qui fait glisser une fenêtre sur la série temporelle et remplace la dernière instance dans la fenêtre par la valeur moyenne de toutes les observations dans cette fenêtre [7, 22].
On peut aussi le définir formellement par :

$$s_t = \frac{1}{n} \sum_{i=0}^{n-1} x_{t-i} \quad (3.3)$$

où x_{t-i} est l'instance de la série temporelle de temps $t-i$ (si $i = 0$ donc x_{t-i} est la première instance de la fenêtre) et s_t est la valeur lissée jusqu'au t (donc c'est la valeur pour la fenêtre) et n c'est la largeur de la fenêtre coulissante.

3.4.2 Weighted Moving Average (WMA)

Par l'application de la méthode précédente (SMA), des fois on perdre des informations importantes telles que la maximum et minimum. Nous pouvons surmonter ce problème en attribuant des pondérations aux données plus récentes [7]. Les chercheurs ont proposé une nouvelle méthode qui se base sur la notion de poids qui s'appelle Weighted Moving Average (WMA). Pour celle la il faut définir une fonction de poids qui permet de attribué un poids plus grand au données plus récentes. Il existe plusieurs formules pour cette fonction qui sont déjà proposé (voir les équations 3.4 , 3.5 et 3.6) [15, 8] :

$$w_i = i^k, i = 1, 2, 3, \dots n \quad (3.4)$$

$$w_i = i, i = 1, 2, 3, \dots n \quad (3.5)$$

$$w_i = i/n, i = 1, 2, 3, \dots n \quad (3.6)$$

Où n est la longueur de la fenêtre, et i est l'index ($i=0$ est équivalent de la date la moins récentes). Dans l'équation 3.4 k est un entier telles que $k > 1$. w_i est le poids qui correspond à l'instante i .

Dans les trois équations on voit que si i est plus grand donc w_i est plus grand. En suit, pour calculer WMA on applique l'équation 3.7 et 3.8.

$$W = \sum_{i=1}^n w_i \quad (3.7)$$

$$l_t = \frac{1}{W} \sum_{i=1}^{n-1} w_i c_{t-i} \quad (3.8)$$

telles que W est la somme totale des poids, l_t est la valeur lissée à l'instant t et x_{t-i} est la donnée à l'instante $t - i$.

3.4.3 Réduction de dimensionnalité

Dans le domaine de visualisation des MTS, les données sont généralement de très grande dimension. Pour les visualiser, il faut qu'elles soient représentées dans un espace de dimension 2 ou 3 . En utilisant une technique de réduction de la dimensionnalité, nous pouvons transformer nos données en une représentation significative de la dimension réduite, qui est plus facile à visualiser et à analyser. La réduction de la dimensionnalité est le processus de réduction d'un ensemble de données à haute dimension en une représentation à plus faible dimension qui conserve la majeure partie de sa structure importante . Dans cette opération, on distingue deux grands types de méthodes : Les méthodes linéaires et les méthodes non-linéaires .

Définition 3.4.1. méthodes linières :

Étant donnés les points de données n-dimensionnels $X = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n}$ et un choix de dimensionnalité $r < d$, optimiser certaines fonctions objectives $f_X(\cdot)$ pour produire une transformation linéaire $P \in \mathbb{R}^{r \times d}$. $Y = PX \in \mathbb{R}^{r \times n}$ sont les données transformées en faible dimension [5].

Définition 3.4.2. méthodes non-linéaires :

Une approche de la simplification consiste à supposer que les données d'intérêt se trouvent sur une variété non linéaire intégrée dans l'espace de dimension supérieure.

Définition 3.4.3. variété mathématique (Manifold)

Une variété mathématique est un espace topologique localement euclidien M tel que tout point $m \in M$ possède un voisinage V_m tel qu'il existe une application bijective : $\Phi : V_m \rightarrow B$ (*boule de \mathbb{R}^n*) , Φ est une application homéomorphe.C'est à dire que Φ : est continue et Φ^{-1} : continue. On appelle Φ une Carte (Chart , Map).

Exemple : l'exemple le plus intuitif est l'exemple de la terre. Si on veut déterminer les coordonnées d'un point m sur notre terre (qui peut se modéliser grossièrement par une sphère),Donc il faut le présenter dans une carte de dimension 2 puis déterminer leurs coordonnées (voir la Figure 3.2 ¹).

¹

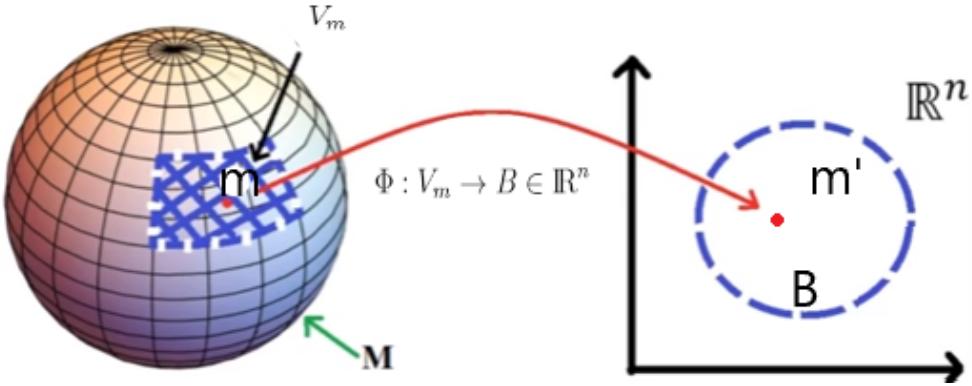


FIG. 3.2 : Exemple de variété

Définition 3.4.4. Manifold learning :

L'apprentissage de variété est une approche de la réduction de la dimension non linéaire. Les algorithmes pour cette tâche sont basés sur l'idée que la dimensionnalité de nombreux ensembles de données n'est que artificiellement élevé [10]. Si nous représentons une variété paramétrée sous forme de $Map f : \Omega \rightarrow \mathbb{R}^m$ d'un sous-ensemble Ω de \mathbb{R}^d en \mathbb{R}^m avec $d < m$, de sorte que $x_i = f(y_i)$, où les x_i sont nos points de données dans l'espace à haute dimension, et les y_i sont les points correspondants dans l'espace à basse dimension des "caractéristiques", l'apprentissage de variété signifie développer une reconstruction de la carte f qui peut être utilisée pour associer des points dans l'espace de coordonnées réduit \mathbb{R}^d avec des points dans l'espace de données d'origine [20].

Définition 3.4.5. Variété riemannienne :

D'après [6], il s'agit d'une variété, c'est-à-dire un espace courbe généralisant les courbes (de dimension 1) ou les surfaces (de dimension 2) à une dimension n quelconque, et sur laquelle il est possible d'effectuer des calculs de longueur.

Analyse des composants principales (PCA)

L'idée de l'analyse en composantes principales (PCA) est de réduire la dimensionnalité d'un ensemble de données constitué d'un grand nombre de variables, tout en conservant autant que possible la variation présente dans l'ensemble des données. On peut y parvenir en transformant en un nouvel ensemble de variables, les composantes principales (CP), qui ne sont pas corrélées, et qui sont ordonnées de manière à ce que les premières conservent la majeure partie de la variation présente dans toutes les variables d'origine [16]. Mathématiquement il faut d'abord calculer la matrice de covariance de X qui peut être calculer par la formule 3.9 :

$$\sum = \frac{1}{n} XX^T \quad (3.9)$$

Ensuite on va diagonaliser la matrice \sum en calculant les vecteurs propres de cette matrice pour avoir une nouvelle matrice diagonale \sum^d . La matrice \sum^d peut être définie par l'équation 3.10 :

$$\sum^d = P \sum^d P^T \quad (3.10)$$

On appelle P la matrice de projection telles que les lignes de P sont les composants principales de X .

uniform manifold approximation and projection (UMAP)

Est une méthode de réduction de dimensionnalité qui a été créée en 2018 par L.McInnes , J.Healy et J.Melville [14]. C'est une technique un peu similaire t-SNE mais elle est basée sur la théorie de variété mathématique et l'analyse de données typologique (voir la Définition 2.4.3). L'algorithme est fondé sur trois hypothèses concernant les données :

- Les données sont uniformément réparties sur le variété riemannien (voir la Définition 2.4.5) ;
- La métrique riemannienne est localement constante (ou peut être approchée comme telle) ;

- La variété est localement connecté.

Pour plus de détails théorique il faut consulter [11, 19, 13] ou bien l'article originale [14].

3.4.4 Relation entre les variables

Dans la majorité des cas des séries multivariées il existe une certaine relation entre les variables de cette série. Il est important de découvrir et de quantifier le degré de cette relation. Cette connaissance peut nous aider à mieux préparer nos données pour répondre aux attentes des algorithmes de visualisation. Parmi ces techniques, nous avons (pour plus de détails voir [3]) :

- La différence
- Le rapport
- La corrélation
- Le rapport de pente
- La différence de pente
- Le rapport de variance
- La différence de variance

3.5 Discrétisation

La discrétisation est une fonction qui transforme une série numérique (temporelle) en une séquence de symboles. Par exemple l'utilisation des couleur pour représenter le temps. Dans certain référence on trouve que la discrétisation est un type de réduction de dimensionnalité [21].

3.6 Tracer une série temporelle

Dans la littérature, il existe plusieurs types de visualisation pour visualiser nos données dans un format qui convient le mieux à notre cas d'utilisation. Dans les sections suivantes, nous allons présenter quelques graphes qu'il est possible d'utiliser pour représenter des séries temporelles. Chacun des graphes suivants a ses caractéristiques et ses paramètres. Il est possible de composer ces graphes pour avoir de nouveaux graphes qui répondent à nos besoins.

3.6.1 graphique linéaire (Line Chart)

Un graphique linéaire est le moyen le plus simple et le plus utilisé pour représenter des données de séries chronologiques. Il permet à l'utilisateur d'avoir une idée rapide de l'évolution d'un élément au fil du temps. Un graphique linéaire utilise des points reliés par des lignes :

- Une variable indépendante, comme son nom l'indique, n'est pas affectée par d'autres paramètres.
- La variable dépendante dépend de la façon dont la variable indépendante change.

Pour les visualisations temporelles, le temps est toujours la variable indépendante, qui est tracée sur l'axe horizontal. Ensuite, la variable dépendante est tracée sur l'axe vertical. On peut aussi tracer plusieurs variables dans le même graphe avec des couleurs différentes (voir la Figure 3.3²).

3.6.2 La carte thermique (Heatmap)

Une carte thermique affiche la distribution des données sur les axes x et y, où la couleur représente les différentes concentrations de points de données. Les cartes thermiques divisent les points de données en "bacs" - des segments de la visualisation avec des limites supérieures et inférieures pour les axes X et Y. L'option "longueur du bac" détermine les limites de chaque bac. Le nombre total de points qui se trouvent

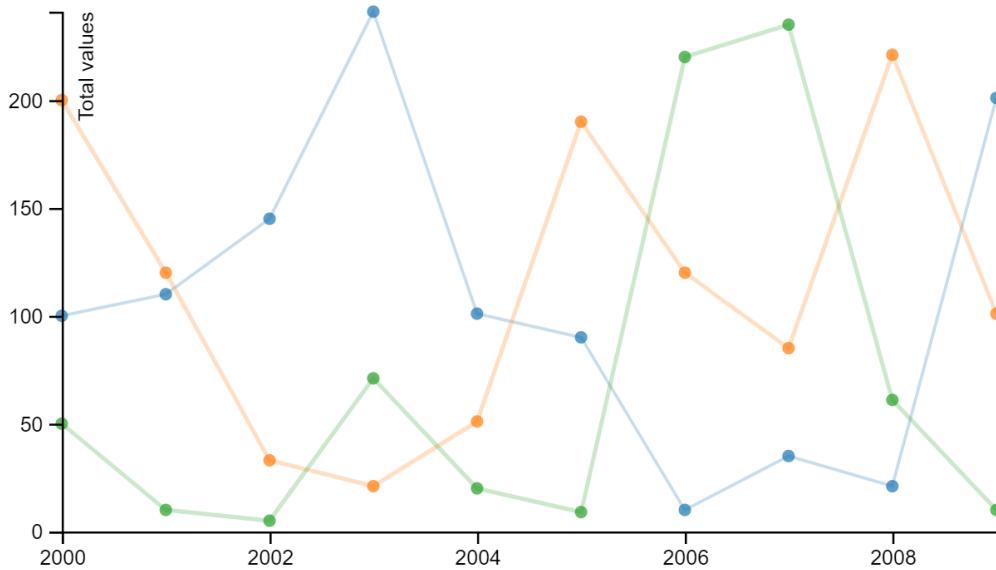


FIG. 3.3 : Exemple line chart

dans un casier détermine sa valeur et sa couleur. Des couleurs plus chaudes ou plus vives représentent des valeurs de bac plus élevées ou une densité de points dans le bac. Dans la Figure 3.4², nous présentons un exemple d'une carte thermique qui affiche les dates de naissance des bébés nés aux États-Unis entre 1973 et 1999. L'axe vertical représente les 31 jours d'un mois tandis que l'axe horizontal représente les 12 mois d'une année³.

3.6.3 Le diagramme de barres (Bar Charts)

Les diagrammes à barres représentent les données sous forme de barres horizontales ou verticales. La longueur de chaque barre est proportionnelle à la valeur de la variable à ce moment précis. Ce graphe est une solution idéale quand il existe un grande nombre de variables à un intervalle de temps discret. Dans la Figure 3.5², nous présentons un exemple de nombre de décès par type de maladie (diarrhée, malaria et maladies respiratoires aiguës) en Inde dans le temps.

²<https://humansofdata.atlan.com/2016/11/visualizing-time-series-data/>

³<https://www.influxdata.com/how-to-visualize-time-series-data>

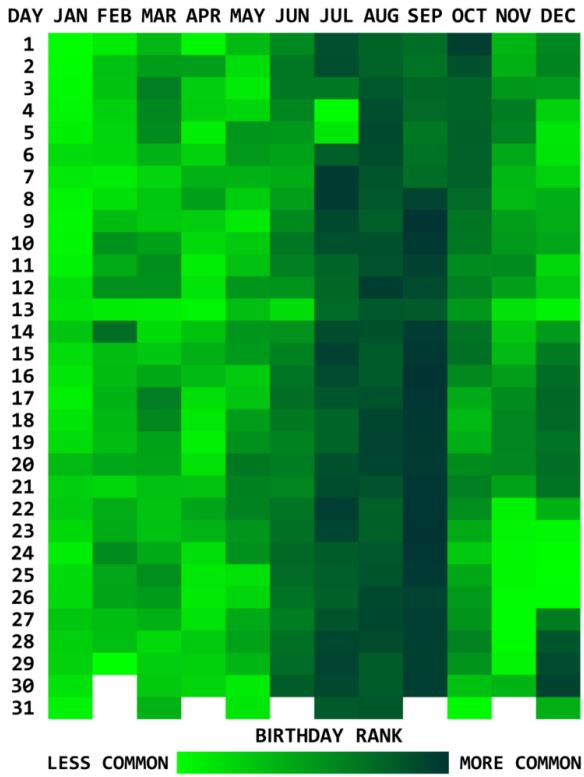


FIG. 3.4 : Exemple Heatmap

3.6.4 Graphe de points (Scatter)

La vue Scatter utilise un diagramme de dispersion pour afficher des données de séries chronologiques. Un nuage de points peut avoir n’importe quel point sur l’axe horizontal, dans n’importe quelle transformation, et les points ne sont pas connectés ou ordonnés (voir l’exemple dans la Figure 3.6²).

3.6.5 Le graphique de flux (Stream Graph)

Un graphique de flux est essentiellement un graphique de zone empilé, mais centré autour d’un axe horizontal central. Le graphique de flux ressemble à un liquide qui s’écoule, d’où son nom. Dans la Figure 3.7² nous présentons un graphique de flux montrant les habitudes d’écoute de la dernière musique .FM d’un auditeur choisi au hasard au fil du temps.

Number of deaths by type of diseases in India (2006-2011)

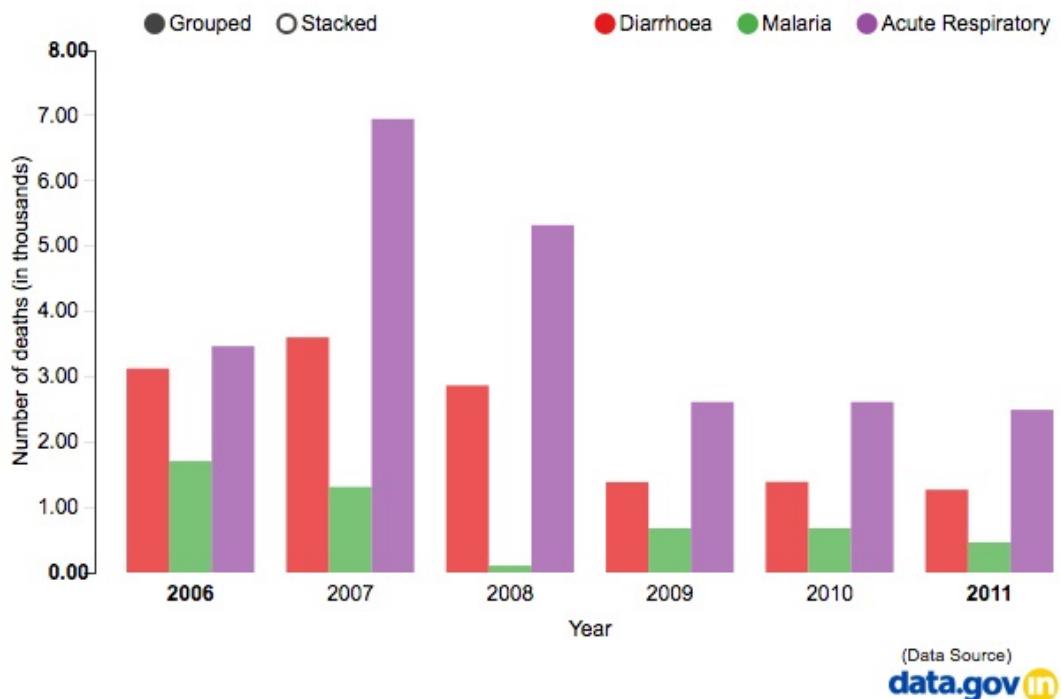


FIG. 3.5 : Exemple barchart

3.7 Conclusion

Dans ce chapitre, nous avons vu ce qu'est la visualisation des séries multivariées et leur histoire et les différentes opérations que l'on peut les appliquer sur ces séries . Nous avons ensuite présenté les principaux types de graphe pour visualiser les séries temporelles.

Ideal look on PowerBI

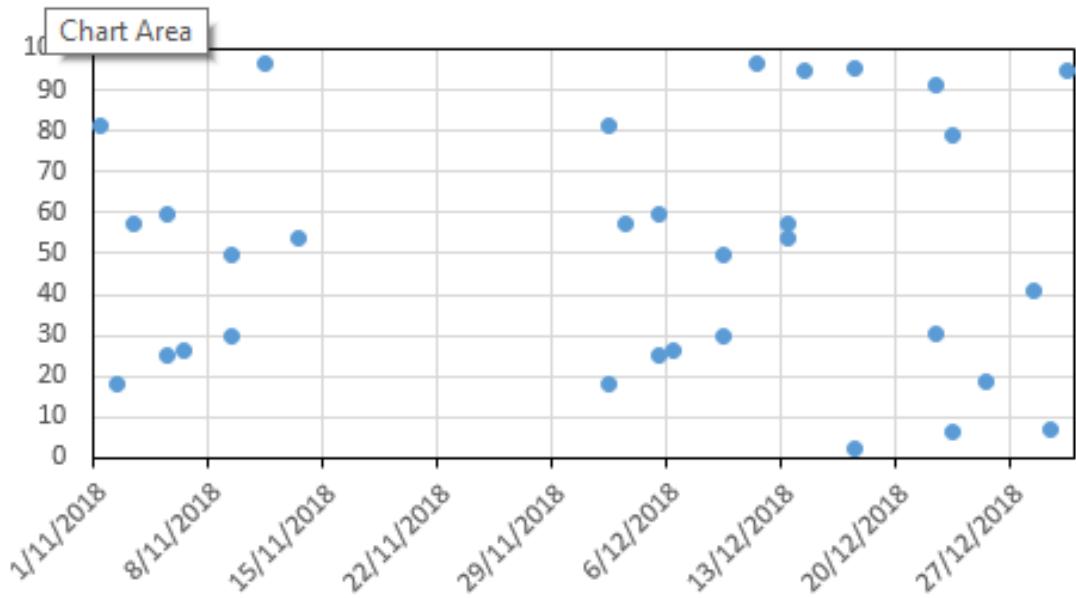


FIG. 3.6 : Exemple scatter

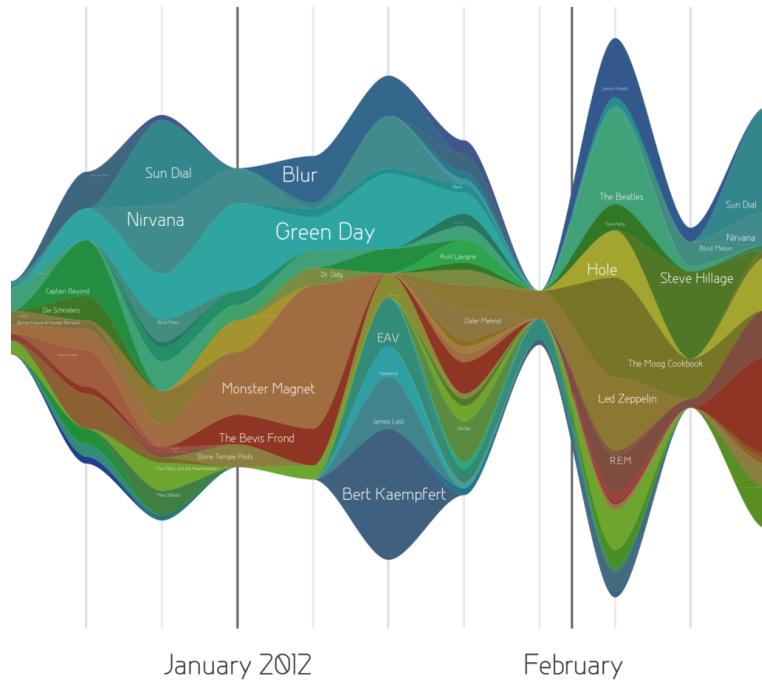


FIG. 3.7 : Exemple Stream Graph

CHAPITRE 4

ÉTUDE DE L'EXISTANT

4.1 Introduction

Dans le domaine de la visualisation de données, il existe plusieurs outils informatiques qui servent à représenter les données dans des graphes. Il existe permis ces outils qui sont spécifiques aux séries temporelles et d'autres non, mais ils peuvent être adaptés pour un cas de série. Dans ce chapitre, nous dressons un panorama et une étude comparative des outils de visualisation modernes et leur adéquation à nos besoins.

4.2 Tableau

Tableau est un outil de visualisation de données puissant et à croissance rapide utilisée dans l'industrie de l'intelligence économique. Il permet de simplifier les données brutes dans un format très facilement compréhensible. Il permet le chargement des données de différentes sources (CSV, SQL, Mongo Db, google drive, spark Sql, Hadoop, postgress, oracle ...), la préparation de données avec des filtrages, la visualisation avec différents graphes (camemberts, diagrammes à barres et à bulles, carte, carte thermique, diagramme de dispersion ...) et l'application de quelques fonctions d'agrégations (la somme, la moyenne, le max, le min, l'écart type ...). On peut aussi lancer des routines

de calcul Python et traiter les données en temps réel. Il a été crée en janvier 2003 il est classé parmi les outils les plus utilisés dans le domaine du traitement et de la visualisation de données. Il est disponible en trois versions : Tableau Desktop, Tableau serveur, Tableau online. Les trois solutions ne sont pas gratuites (voir l'exemple dans la Figure 4.1) .

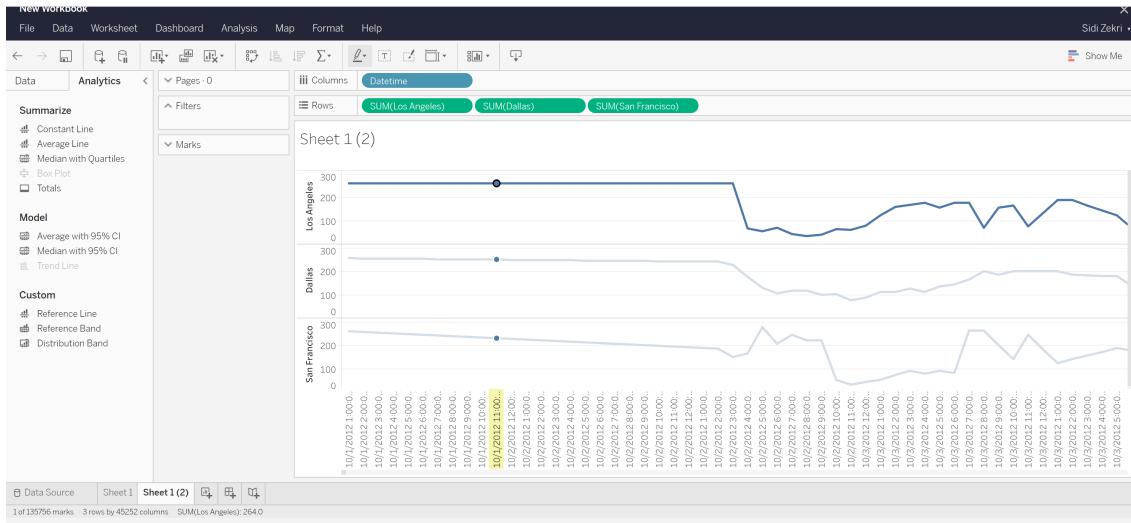


FIG. 4.1 : Exemple de l'interface graphique de Tableau

4.3 Grafana

Grafana est un outil open source qui permet de réaliser des tableaux de bord graphiques pour les données temporelles (mais pas spécifiquement les multivariées). Il permet d'interroger, de visualiser, d'alerter et de comprendre vos mesures, quel que soit l'endroit où elles sont stockées. Créez, explorez et partagez des tableaux de bord avec votre équipe et favorisez une culture axée sur les données. Il propose plusieurs types de graphe tel que : diagramme barre, graphique linéaire, pies etc. Et il dispose aussi une API HTTP complète. Il travail bien avec les bases de données temporelle tel que Graphite, InfluxDB et OpenTSDB. Il a été crée en 2013 mais la dernière version est publiée en 2020 (voir l'exemple dans la Figure 4.2).

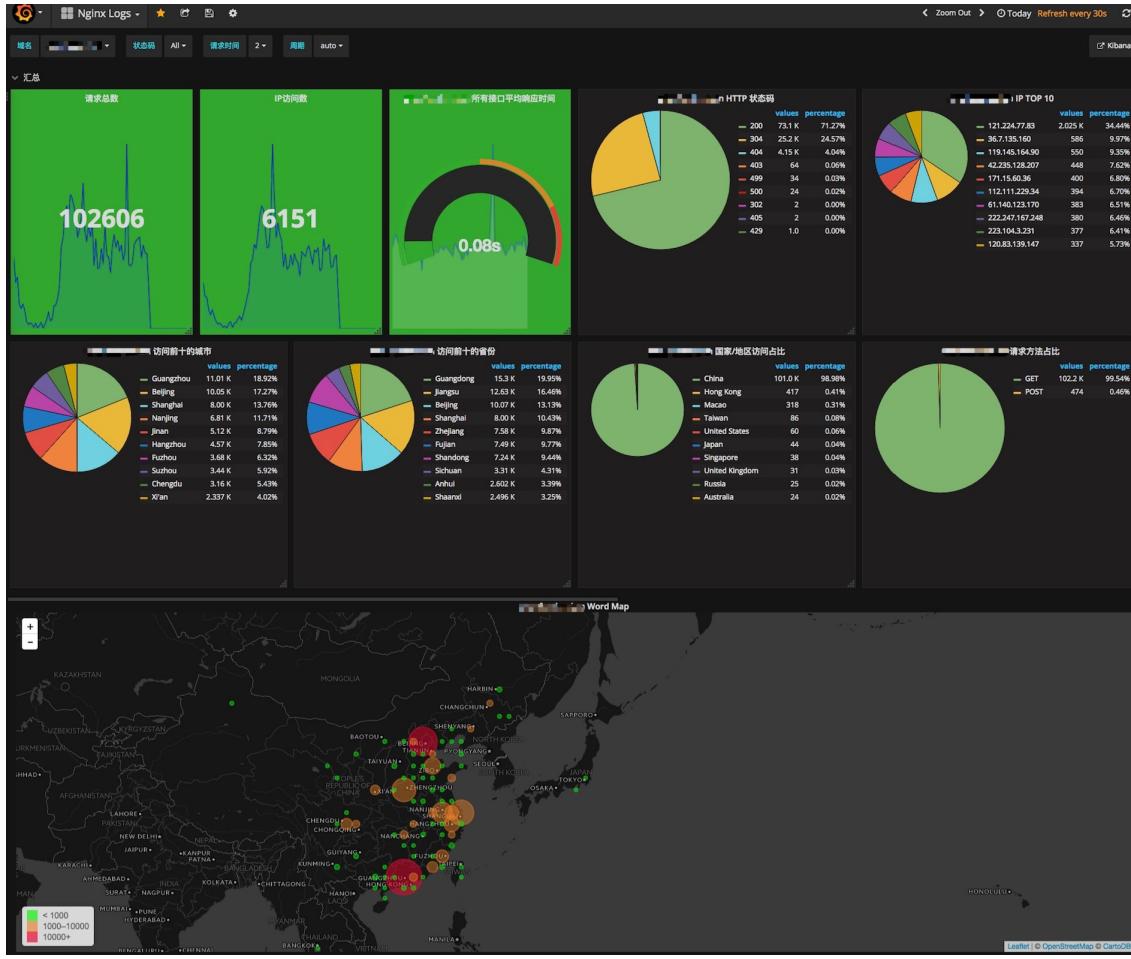


FIG. 4.2 : Exemple de l'interface graphique de Grafana

4.4 Dial-A-Cluster (DAC)

DAC est outils de visualisation pour les séries temporelles multi-variées qui a été créé par Shawn Martin et Tu-Toan Quach en 2016 [12]. D'après le créateur de cette solution, le premier objectif est de fournir une interface utilisateur sous la forme d'un client léger (web friendly) et interactif pour la visualisation des MTS en fonction des intérêts de l'analyste. Elle permet d'exécuter trois algorithmes : la visualisation à l'aide d'une mise à l'échelle multidimensionnelle MDS, identifiant les MTS qui sont les plus responsables des différences entre les groupes sélectionnés par les analystes, et l'optimisation de la visualisation selon les métadonnées spécifiées par l'analyste. Elle fournit deux types de graphe : graphe des points et graphique linéaire

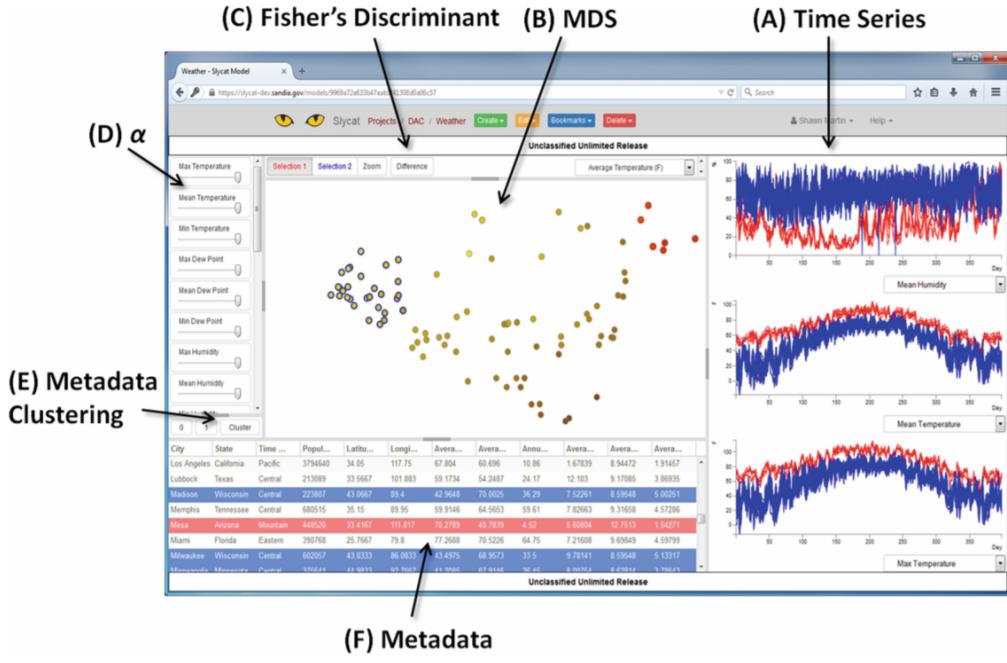


FIG. 4.3 : l'interface utilisateur. Nous montrons ici le DAC fonctionnant dans Firefox sur un PC Windows.

4.5 SDRVR

SDRVR (smoothing , Dimensionality reduction and variable relationships tool) est un outil de visualisation des MTS qui a été développé en python par CAO Anh Quan en 2019 [3] qui permet d'appliquer les différents algorithmes de lissage, des algorithmes de réduction de dimension, ainsi qu'une recherche de relations entre les variables sur les MTS. Il donne l'accès à une panoplie d'algorithmes tels que (SMA, WMA, PCA, t-SNE, différences ...). Nous pouvons naviguer dans l'ensemble des données et sélectionner les opérations que nous souhaitons effectuer puis visualiser les résultats dans des graphiques. Il propose deux types de graphes (scatter, graphique linéaire) à l'aide de l'outil. Il a développé une nouvelle technique de visualisation pour étendre le nuage de points de la carte de chaleur (heatmap scatter plot) dans la mesure où elle peut visualiser la relation dynamique entre deux variables (comment elles changer ensemble) nommé diagramme de dispersion temporel (temporal scatter plot)

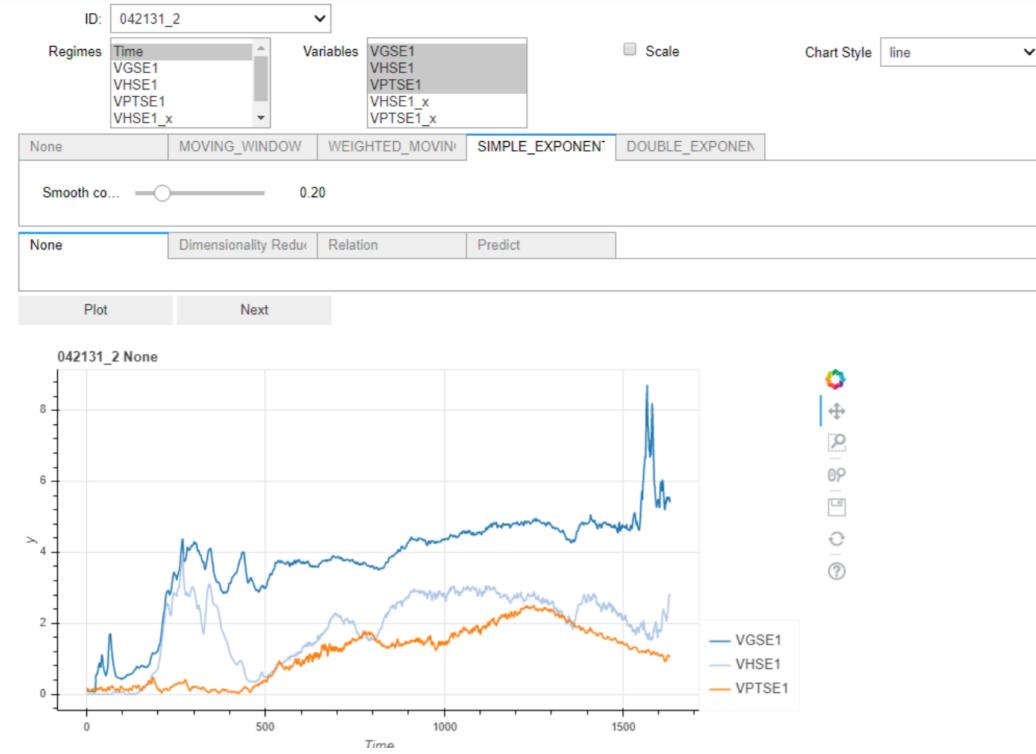


FIG. 4.4 : l'interface utilisateur SDRVR

4.6 Bipeline

Bipeline est un outil qui a été créé par Ricardo Cachucho, Kaihua Liu, Siegfried Nijssen et Arno Knobbe en 2016 [2]. Disponible en format web, il permet d'exécuter un pipeline de biclustering des MTS. Avec Bipeline, il est facile de sauvegarder les expériences et d'essayer différents algorithmes de biclustering, permettant de passer intuitivement du pré-traitement à l'analyse visuelle de biclustering. Le pipeline est décrit par les étapes suivantes : Importation les données, traçage interactif en utilisant la bibliothèque dygraphs de langage R, Pré-traitement (l'exclusion des variables, la normalisation, la suppression conditionnelle et le remplacement des données et l'élimination des valeurs aberrantes), Segmentation et Biclustering. Le traçage peut être fait à n'importe quel moment du pipeline.

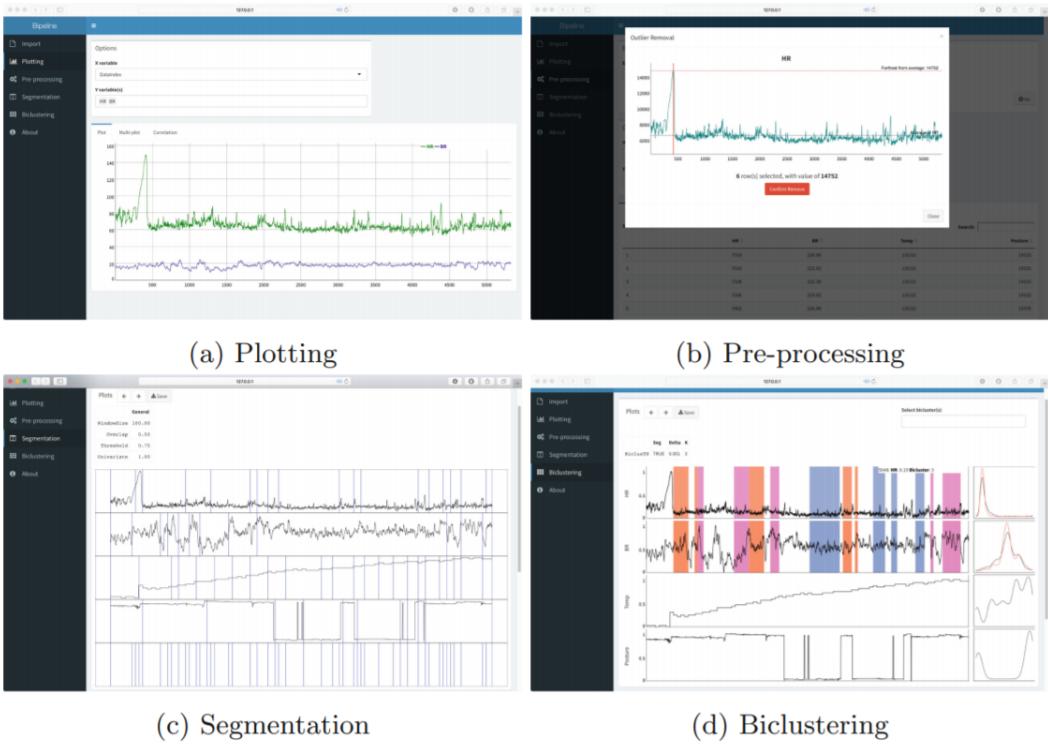


FIG. 4.5 : Interface utilisateur Bipeline. (Figure de couleur en ligne)

4.7 Synthèse

Dans ce chapitre, nous avons présenté les outils les plus récents pour la visualisation des séries temporelles multi-variées. Nous avons vu que l’adéquation de ces outils à notre cas de visualisation de séries multivariées varie. Les tableaux 4.1 et 4.2 présentent une étude comparative entre ces outils. En ligne on a les différents outils et en colonne on a les mesures de comparaison : Disponibilité, opensource, déploiement/installation, adaptation pour MTS, graphes proposés, interactivité, gros données, algorithmes de pré-traiteme nt, source de données. Nous remarquons que généralement les solutions ne peuvent traiter les gros volumes donnés sauf Tableau et Grafana qui peuvent lire des formats de données distribuées (Hadoop / Spark), mais ils ne proposent pas une programmation visuelle. Pour gestion de pipeline sauf CAD et Bipeline qui propose un seul pipeline (enchaînement de plusieurs traitements).

4.8 Conclusion

Dans la littérature il existe plusieurs outils de visualisation des données temporelles mais la majorité de ces outils ne sont pas adaptés pour les MTS (généralement on trouve des outils pour les UTS). Dans ce chapitre, nous avons réalisé une étude comparative selon certaine critères des différents outils de visualisation pour les MTS qui se base sur les caractéristiques suivantes : Disponibilité en ligne, la gratuité d'utilisation de code source, déploiement et l'installation, adaptation pour MTS, graphes proposés par l'outil, interactivité des graphes, traitement des données massive, algorithmes de pré-traitement, source de données.

Outils/ mésures	Disponibilité	Opensource	Déploiement /installation	Les graphes proposés
Tableau	✓	X	serveur, bureau et en ligne	pies, diagrammes à barres et à bulles, carte, carte thermique, diagramme de dispersion ..etc
Grafana	✓	✓	bureau en ligne	diagramme barre, graphique linéaire, pies ..etc
SDRVR	X	X	bureau	graphe des points et graphique linéaire
CAD	✓	✓	Web (pas d'installation)	graphe de points, graphique linéaire
Bipeline	X	X	Web (pas d'installation)	dygraphes de R

TAB. 4.1 : Comparaison entre les outils suivant les mesures : Disponibilité, opensource, déploiement/installation, Les graphes proposés

Outils/ mésures	Interactivité	données massives	Algorithmes de pré-traitement	Source de données
Tableau	✓	✓	Functions d'agregation	Fichier CSV, SQL , Mongo Db, google drive spark SQL, hadoop, postgress, oracle
Grafana	✓	✓	Functions d'agregation	Graphite, InfluxDB , OpenTSDB ...
SDRVR	✓ certain paramètres	X	SMA, WMA, SES, DES PCA, NMF, MDS, ISOMAP, LLE, t-SNE Diffrence, ratio, corelation ... + Nouvelle méthode de visualisation	Fichier CSV
CAD	✓	X	MDS, Différences entre les TS Regroupement par métadonnées	Fichier CSV
Bipeline	X	X	l'exclusion des variables , la normalisation, la suppression conditionnelle et le remplacement des données, et la suppression des valeurs aberrantes + la segmentation + le biclustering	Fichier de données

TAB. 4.2 : Comparaison entre les outils suivant les mesures : Interactivité, Bigdata, algorithmes de pré-traitement, source de données

CHAPITRE 5

VISIO : OUTIL DE VISUALISATION POUR LES SÉRIES TEMPORELLES

5.1 Introduction

Dans le chapitre précédent, nous avons vu les différentes solutions qui existe pour la visualisation des TS et analysé leur avantages et inconvénients. On remarque que la majorité de ces solutions ne traitent pas le coté Big Data et que d'autre qui ne traitent pas la gestion des pipelines. Notre solution **VisIo** est un logiciel web qui permet l'importation, le traitement et la visualisation des TS pour les deux types multivariées, et univariées tout en permettant la description du pipeline menant à cette visualisation par l'assemblage interactif de blocs visuels (dans la mouvance du "visual scripting" et des outils "low code"). Cette solution permet d'importer les données de n'importe quelle base de données de type SQL (Hive, MYSQL, postgres, fichiers parquet ...). Elle propose aussi d'enchaîner des algorithmes SPARK (qui sont développer en MapReduce) et/ou des algorithmes Simples. Ce qui nous permet de traiter les données massives et de visualiser les résultats dans des graphes tels que : Graphique linéaire, graphe des points, graphe des bars etc. Tout ça en utilisant une architecture web moderne, flexible et extensible. Dans ce chapitre, nous détaillons l'architecture du projet, l'analyse et la

conception, l'implémentation et le déploiement de cette solution . Enfin, nous présentons les différentes technologies utilisées.

5.2 Analyse des besoins

Avec l'apparition de plusieurs algorithmes de traitement de séries temporelles, il est primordial d'avoir un outil qui permet d'analyser et visualiser ces données. Notre problématique, est de créer une interface intuitive pour un expert de métier, prenant la forme généralement de « requêtes graphiques ». Cette solution vise donc à faire intervenir plus directement l'expert du domaine et permet une analyse rapide des résultats issus de l'apprentissage automatique. Les besoins fonctionnelles de ce projet peut être résumé en cinq éléments principales :

- De charger les données de différentes sources ;
- D'enchaîner certain algorithmes de traitement de données distribué ou non ;
- De visualiser les résultats sortantes dans des graphes ;
- D'organiser ces fonctionnalités en blocs visuel tel que chaque bloc peut prendre la sortie d'un autre bloc (voir la Figure 5.1 pour un exemple) ;
- De pouvoir exporter les graphe en PDF ou image.

Les besoins caractérisent ce système(non-fonctionnelles) sont :

- Interface intuitive ;
- Solution extensible (ajouter des nouvelles fonctionnalités facilement) ;
- Facile a déployer ;
- Utilisation des technologies moderne de développement.

Ce système regroupe plusieurs cas d'utilisations (CU), nous présentons dans le diagramme suivant les CU que nous jugeons importants (voir la Figure 5.2)

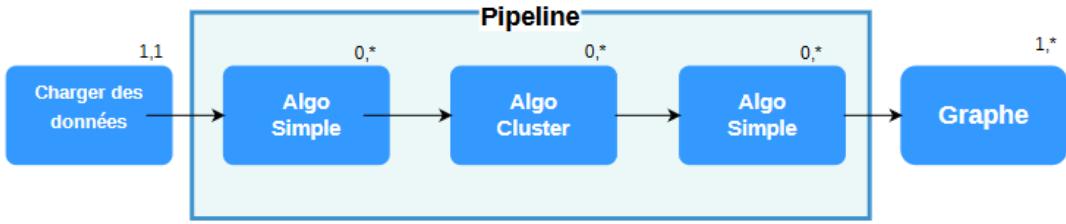


FIG. 5.1 : Illustration pour l'enchaînement des blocs visuel

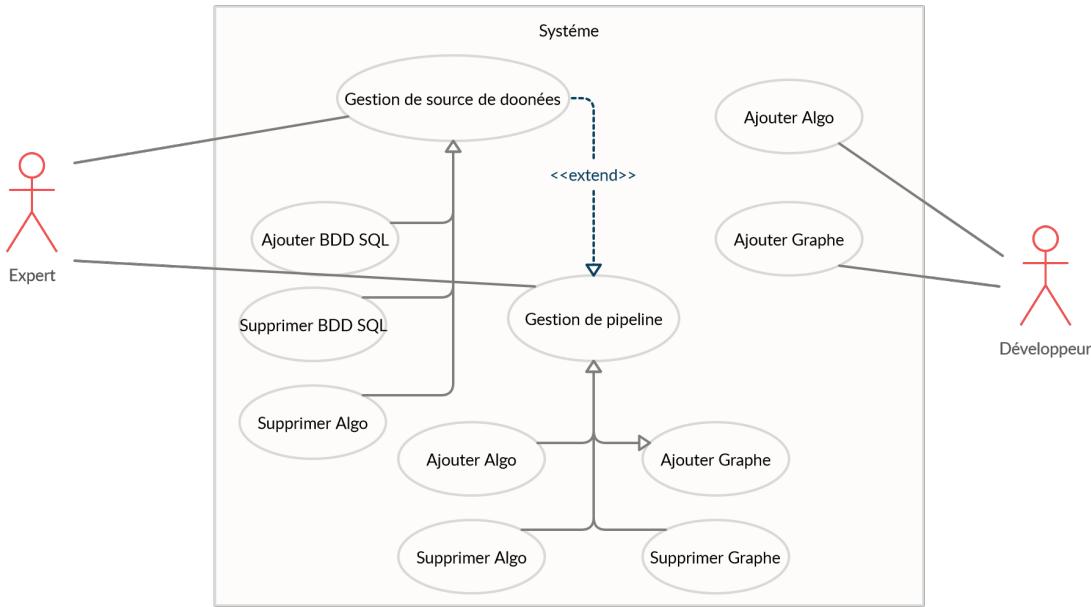


FIG. 5.2 : Diagramme de cas d'utilisations

Pour résumer la situation, nous présentons l'un des scénarios de notre projet en un diagramme de séquence (voir la Figure 5.3)

5.3 Architecture

Nous avons devisé notre architecture en six entités essentielles, qui sont :

- **L'interface graphique** : C'est bien la partie front de cette architecture qui contient tout les composants visuels ;

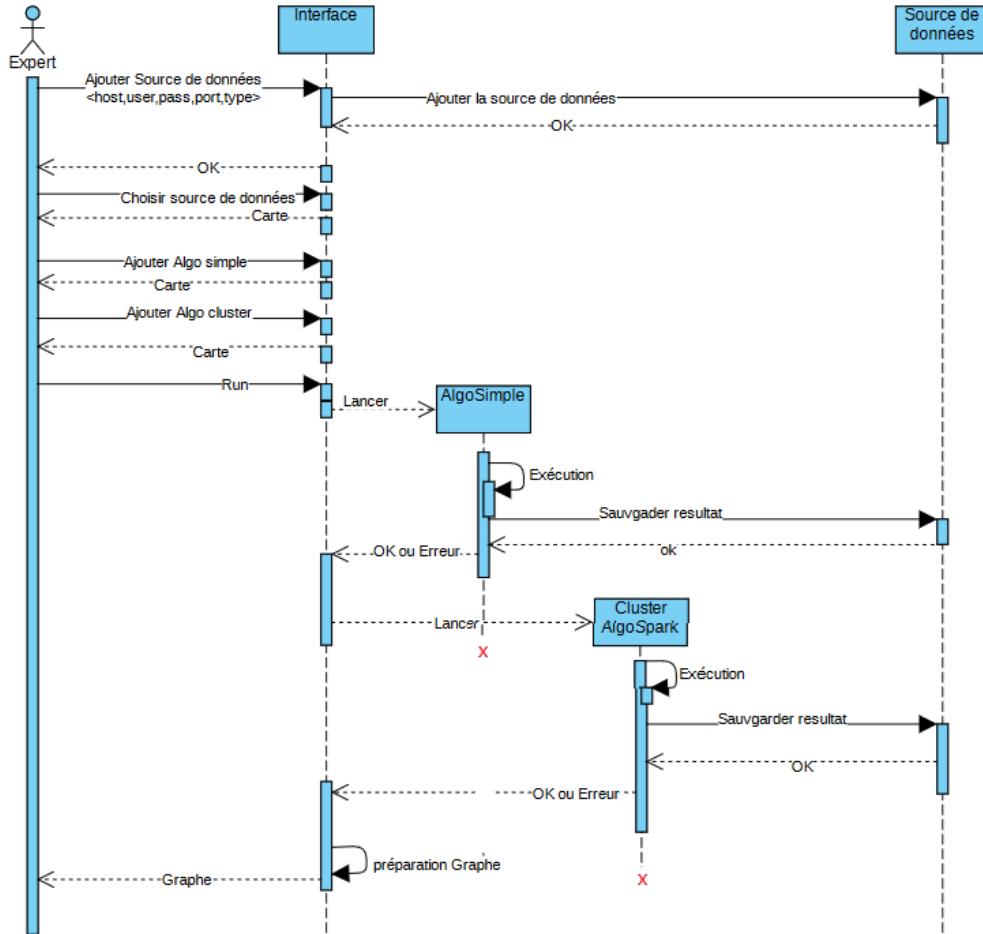


FIG. 5.3 : Diagramme de séquences pour un scénario

- **Les algorithmes distribués** : Sont les algorithmes qui tourne sur un cluster avec un moteur SPARK ;
- **Les algorithmes non-distribués (simples)** : sont les algorithmes qui tourne sur un seul serveur ;
- **la gestion d'interface** : C'est la partie qui prend en charge la gestion de sources de données et la gestion du pipeline ;
- **La gestion des échantillons** : Cette partie s'occuper de la visualisation des séries ;

- **La gestion des sessions SPARK** : cette partie se charge de créer, supprimer et afficher les sessions SPARK.

Pour l'architecture, nous avons choisi de mettre en place une architecture WEB moderne (client léger). Dans la littérature, il existe plusieurs modèles d'architectures WEB tels que l'architecture monolithique et web service. Dans notre cas, nous avons adopté une architecture microservices qui présente plusieurs avantages pour notre solution tels que :

- Séparation de la partie design et la partie traitement ;
- Une architecture extensible : pour ajouter un traitement il faut simplement ajouter un microservice et leur représentation dans l'interface ;
- Utilisation de plusieurs langages de programmation pour la partie traitement ;
- Utilisation des technologies récentes de Javascript pour les visualisations ;
- Simlicités de déploiement.

5.3.1 Microservices

Nous avons divisé notre solution en microservices, sachant que chaque algorithme est un microservice. De plus nous avons trois autres microservices, le premier sert à gérer l'interface, le deuxième sert à gérer les sessions SPARK et le troisième sert à mettre en place une gestion échantillonnage. Pour envoyer des travaux au moteur SPARK, nous avons besoin d'une solution qui s'appelle LIVY¹ qui se charge de la communication entre les deux microservices (MS des algorithmes distribués et le MS qui gère les sessions SPARK) et le moteur SPARK. Cette outil nous permet d'ouvrir une session SPARK interactive pour l'exécution des algorithmes distribués en mode SPARK². Notre architecture est développée principalement avec trois langages de programmation, Javascript (Angular 9 framework) pour la partie interface, et Scala (Finatra framework) et Python (Flask framework) pour la partie microservices. Les diagrammes dans les Figures

¹<http://livy.apache.org/>

²Le tout passe en mémoire

5.4, 5.5, 5.6 et 5.7, montre les routes et les paramètres des quatre microservices (que nous jugeons importants) AlgoSpark, AlgoSimple, gestion d'échantillonnage et celui qui gère l'interface respectivement .

FIG. 5.4 : Diagramme des routes et paramètres pour les microservices des algorithme spark

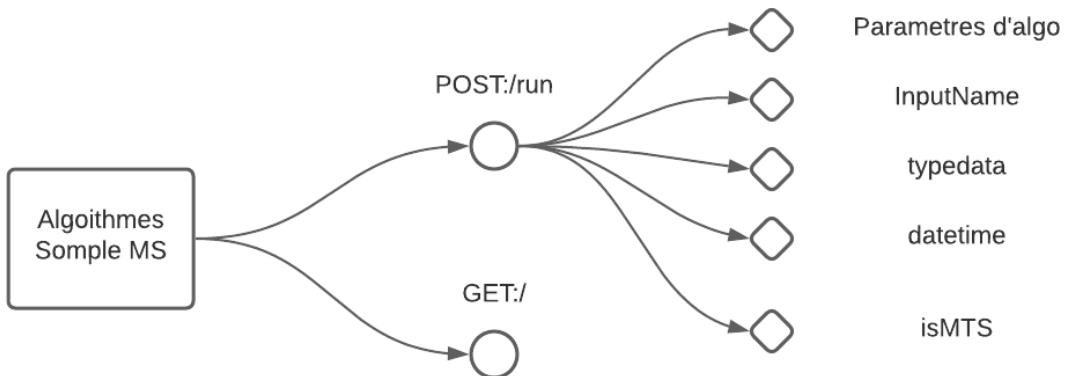


FIG. 5.5 : Diagramme des routes et paramètres pour les microservices des algorithme simple

5.3.2 Communication entre les microservices

Malgré les avantages de cette architecture MS, elle présente quelques inconvénients. L'un de ces inconvénients c'est bien la communication entre MS, qu'est un axe de recherche active actuellement. Dans la littérature il existe plusieurs solutions pour ce problème telles que : la communication http entre les MS, la communication par un protocole RPC (Remote Procedure Call), l'utilisation d'un message queue ou bien l'utilisation d'une base de données partagée. Dans notre cas, le problème ce pose dans la communication entre les MS d'algorithmes généralement. On distinct deux cas de figure :

- un algorithme qui tourne sur SPARK passe les données à autre de même nature :
Dans ce cas on utilise la nature interactive de SPARK le tout est dans la mémoire

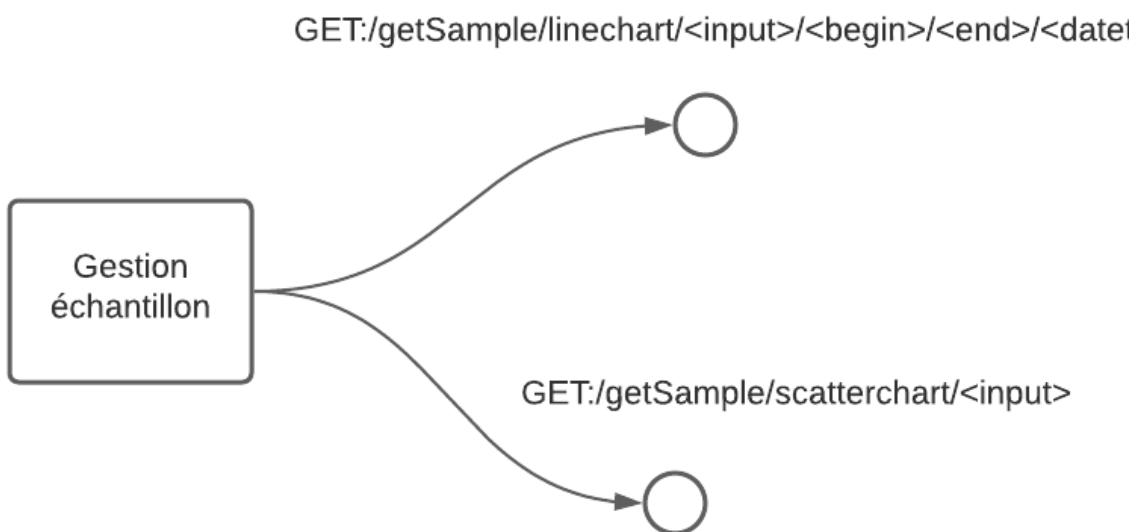


FIG. 5.6 : Diagramme des routes et paramètres pour les microservices de gestion

- un algorithme qui tourne sur SPARK passe les données à un autre qui tourne sur un serveur simple ou le contraire : dans ce cas nous avons choisi la solution de la base de données partagée après une étude comparative entre les quatre solutions

Le diagramme dans la Figure 5.8 montre un résumé de notre architecture .

5.3.3 Base de données partagée

Pour la communication entre les différents microservices, nous avons choisi de mettre en place une base de données MongoDB partagée. Elle comporte un document de gestion qui rassemble les informations du serveur et des bases de données initiales, et les résultats de algorithmes (voir la Figure 5.9). La Figure 5.10 montre le schéma du document de gestion . Nous avons organisé les données sortantes des algorithmes en lignes, tel que chaque ligne est une séries pour les UTS et une variables pour les MTS. Voici un exemple :

```

1 {Algo<id> :
2 {

```

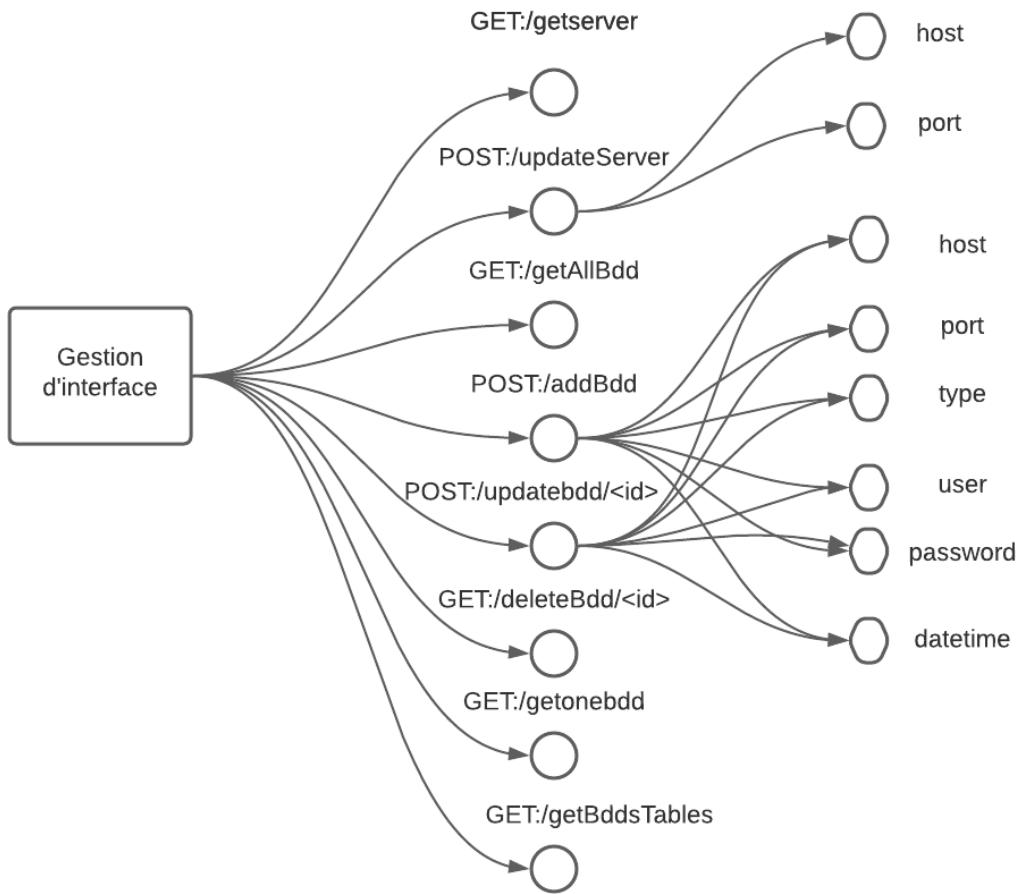


FIG. 5.7 : Diagramme des routes et paramètres pour les microservices qui gère l'interface

```

3 'id' : 1,
4 'cols' : 'series1',
5 'series' : [1.22,4.33,0.33,...]
6 },
7 {
8 'id' : 2,
9 'cols' : 'series2',
10 'series' : [1.22,4.33,0.33,...]
11 },
12 {

```

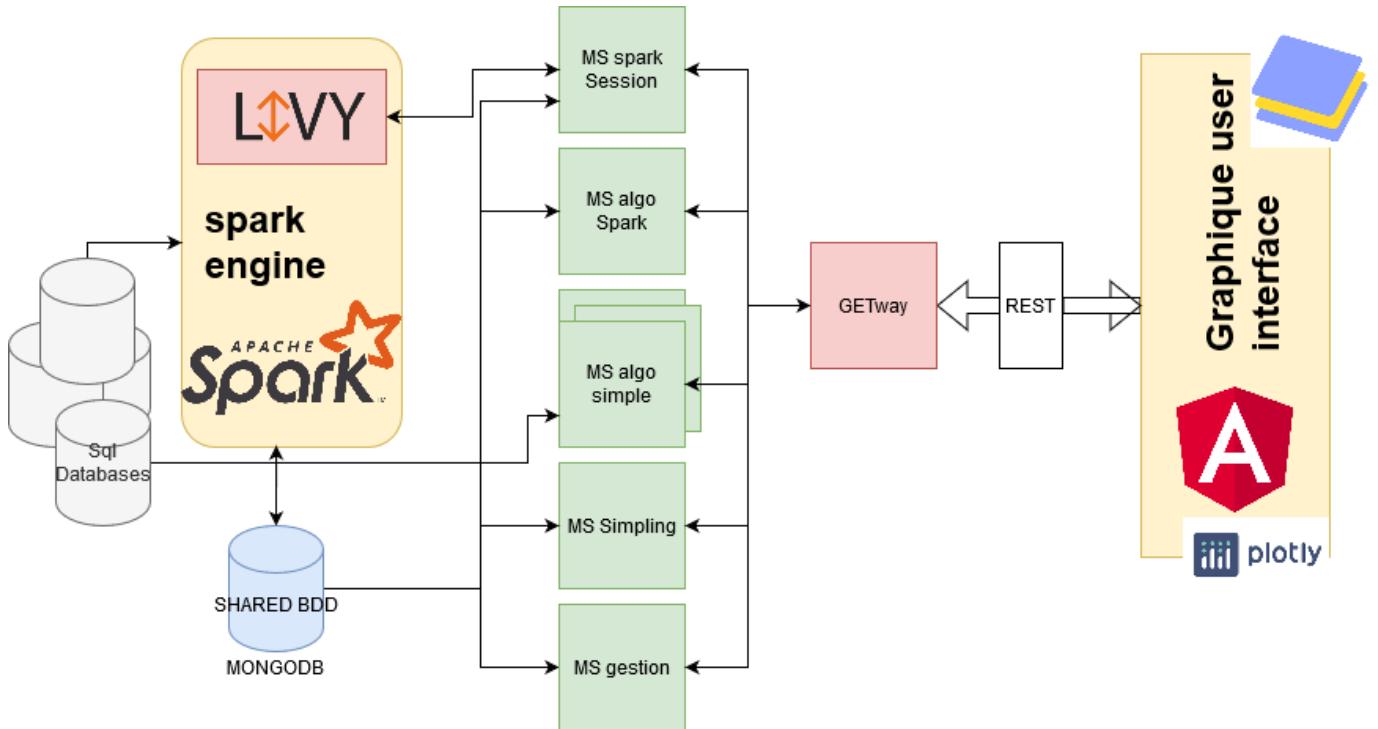


FIG. 5.8 : Architecture Microservices de VisIo

```

13 'id' : 3,
14 'cols' : 'series3',
15 'series' : [1.22,4.33,0.33,...]
16 }

```

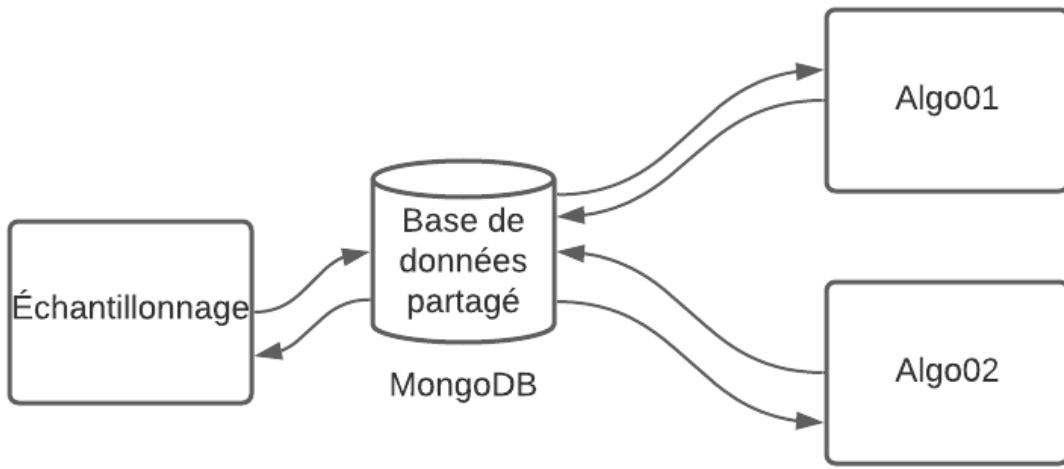


FIG. 5.9 : Communication microservices et base de données partagé

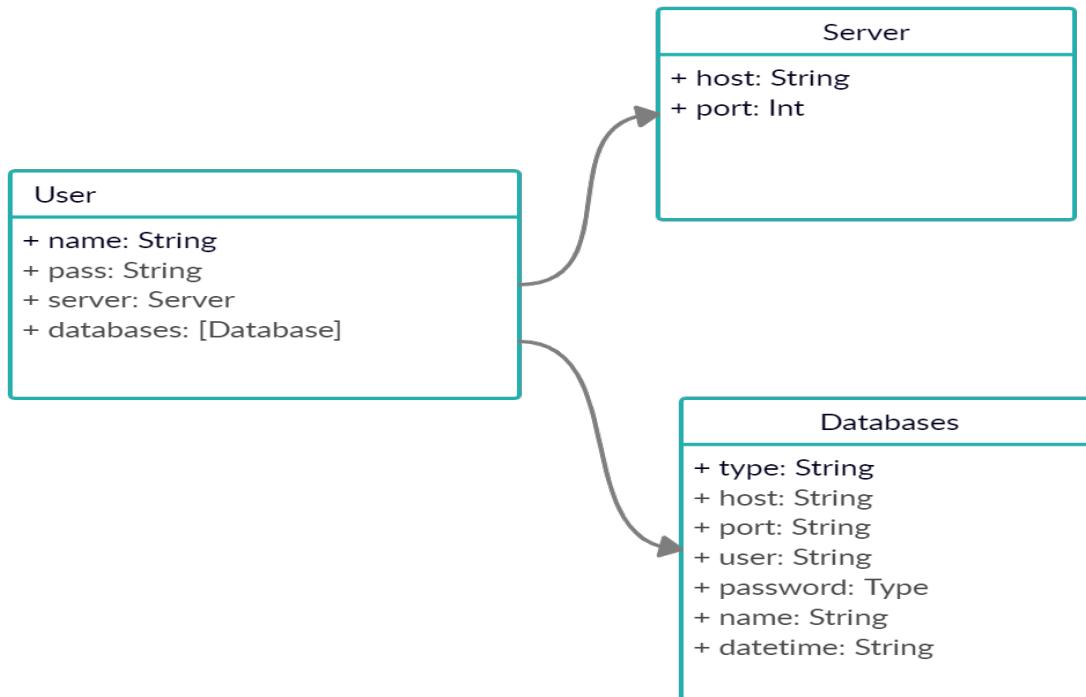


FIG. 5.10 : Schéma base de données partagé

5.4 Conception

Pour rendre notre solution extensible, nous avons implémenté une abstraction qui permet d'ajouter des nouveaux algorithmes justes en étendre l'une des deux classes *AlgoSpark* ou bien *AlgoSimple*, et redéfinissant leur fonction *run*. Après en utiliser cette dernière dans un microservice pour ajouter un algorithme (vous trouvez plus de détails sur l'ajout d'un algorithme dans l'annexe). Le diagramme de classes suivantes présentes les classes de conception du système (voir la Figure 5.11). La Figure 5.12 présente l'implémentation de deux algorithmes en microservices. Notons que cette solution est développé en deux langages Python et Scala, donc nous pouvons ajouter des algorithmes dans les deux langages.

5.5 Implémentation

Visio dans sa version bêta, contient plusieurs fonctionnalités intéressantes qu'on ne trouve pas dans d'autres outils de même nature. Dans cette section nous allons présenter la totalité de ces fonctionnalités.

5.5.1 Configuration manuellement de la partie SPAKR

Notre outil permet de configurer manuellement du moteur d'exécution SPARK, puis si le serveur est en marche l'utilisateur peut ajouter des nouvelles sessions SPARK de différents types ("SPARK", "PYSPARK") . Cette fonctionnalité est réalisé à l'aide de la solution LIVY, on indique leur endroit et leur port. Sachant que ce dernier est installé sur le moteur SPARK (voir l'annexe pour plus de détails) . Les Figures 5.13 et 5.14 l'implémentation de cette fonctionnalité

FIG. 5.13 : Capture sur la fonctionnalités : configuration du serveur

FIG. 5.14 : Capture sur la fonctionnalités : ajoute de session SPARK

5.5.2 Gestion des sources de données

Généralement, les expert du domaine de traitement de données massive disposant leur données dans un endroit précis. Par exemple qui travaillons sur SPARK disposent leur données généralement dans des base de données HIVE ou bien dans des fichier

parquet qui sont souvent dans le moteur SPARK. A cause de grande volume de données nous n'amusant pas à les déplacer vers notre outil. Par conséquence nous avons proposé une solution d'ajouter des sources de données SQL en utilisant la librairie JDBC, on citant leur endroit (url), leur port, le nom d'utilisateur, le mot de passe et le nom de la base de données (voir les Figures 5.15 et 5.16).

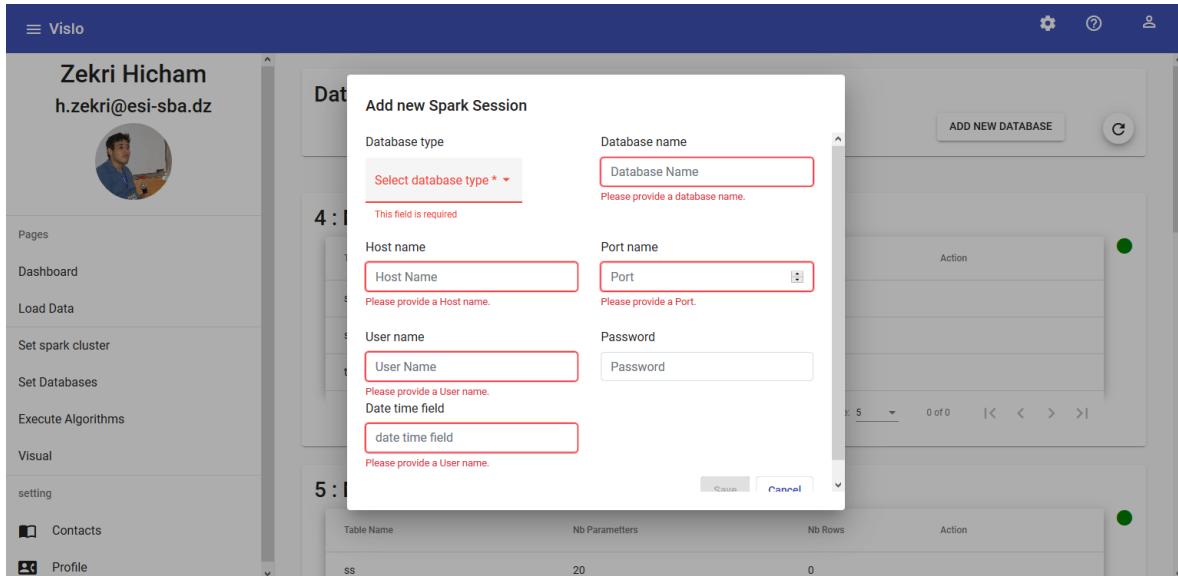


FIG. 5.15 : Capture sur la fonctionnalités : ajoute de base de données

The screenshot shows the Visio application's main interface with a sidebar containing user information (Zekri Hicham, h.zekri@esi-sba.dz) and a list of pages like Dashboard, Load Data, Set spark cluster, etc. Two tables are displayed side-by-side. Table 4 (Mysql) has rows: ss (Nb Parametters: 20, Nb Rows: 0), sss (Nb Parametters: 20, Nb Rows: 0), and timeseries (Nb Parametters: 20, Nb Rows: 7056). Table 5 (Mysql) has rows: ss (Nb Parametters: 20, Nb Rows: 0), sss (Nb Parametters: 20, Nb Rows: 0), and timeseries (Nb Parametters: 20, Nb Rows: 7056).

FIG. 5.16 : Capture sur la fonctionnalités : gestion des sources de données

5.5.3 Gestion du pipeline

Notre outil permet d'enchaîner plusieurs algorithmes de déférente nature, en indiquant la source de données et les algorithmes à exécuter. Pour une première version, nous avons implémenté quatre algorithmes qui sont :

- **PCA (Analyse des Composantes Principales)** : Ce algorithme est le PCA traditionnelle, qui sert a réduire la dimension implémenter en utilisant la lib "Unsupervised" ³, qui a été développer par l'un des membre de l'équipe A3 ;
- **DFTpre (Transformation de Fourier discrète)** : Ce algorithme est la DFT traditionnelle plus un pré-traitement plus une interpolation ;
- **UMAP (Uniform Manifold Approximation and Projection)** : ce algorithme est un algorithmes de réduction de dimension qui permettre de réduire notre série en 2D. Developper en utilisant la lib "scikit learn" ⁴ de Python ;
- **SMA (Moyenne mobile)** : c'est algorithmes de lissage .

Dans ce cas on aura quatre microservice qui représente les quatre algorithmes les deux premiers sont développer en Scala en utilisant le framework Finatra et les deux autre avec Python en utilisant le framework Flask (un tableau comparative entre les framework pour les microservice dans l'annexe). Les Figures 5.17 et 5.18 montre deux pipelinse construit avec notre solution, sachant que le premier est pour les UTS et le deuxième est pour les MTS. Cette partie a été développer avec un framework recent de Javascript qui s'appelle Rete.js ⁵

³<https://github.com/unsupervise/spark-tss>

⁴<https://scikit-learn.org/stable/>

⁵<https://rete.js.org/>

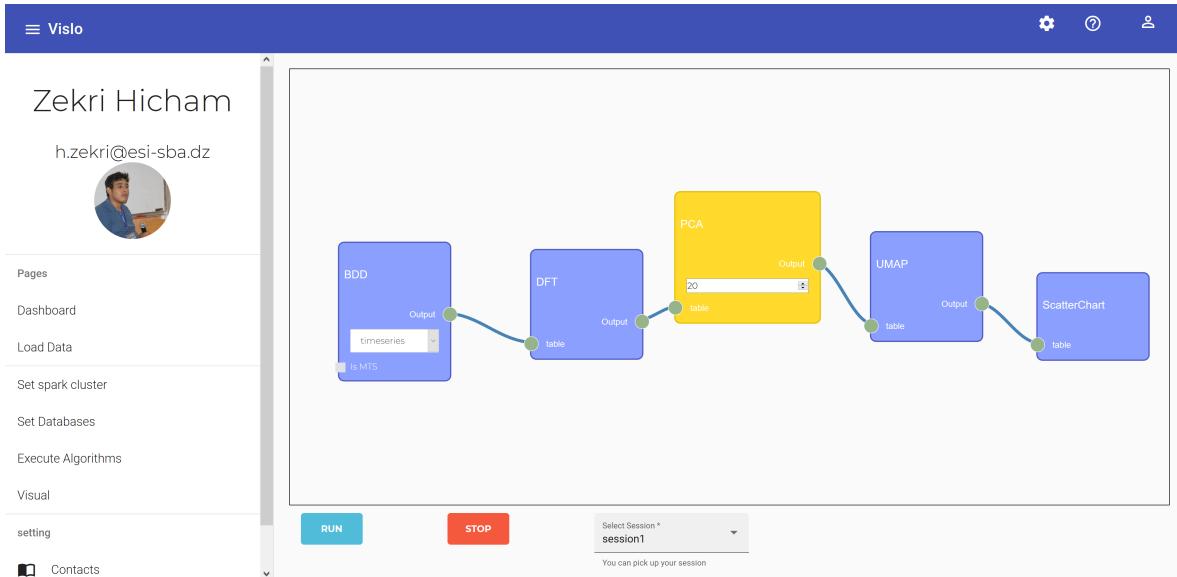


FIG. 5.17 : pipeline pour les UTS

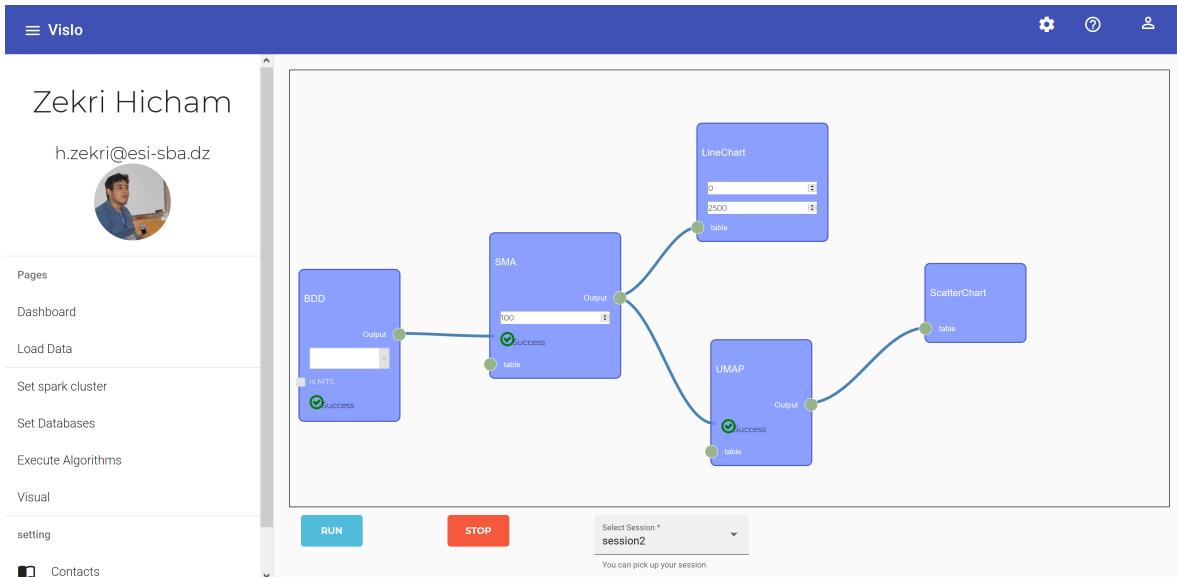


FIG. 5.18 : Pipeline pour les MTS

5.5.4 Visualisation graphique

Après la préparation du pipeline, nous pouvons ajouter des cartes pour la visualisation de data sortante dans des graphes 2D. Pour l'instant, nous avons mis en place deux types de graphes : Un graphique linéaire et le nuage de points. Les Figures suivantes

5.19, 5.20 et 5.21 montre les résultats d'exécution des deux pipelines présenter dans les Figure 5.17 et 5.18 . Ces graphes sont développer avec la bibliothèques Plotly.js⁶, d'une manier autonome de l'application. C'est ta dire que nous pouvons les invoquées de n'importe quelle autre applications (d'un notebook par exemple) .

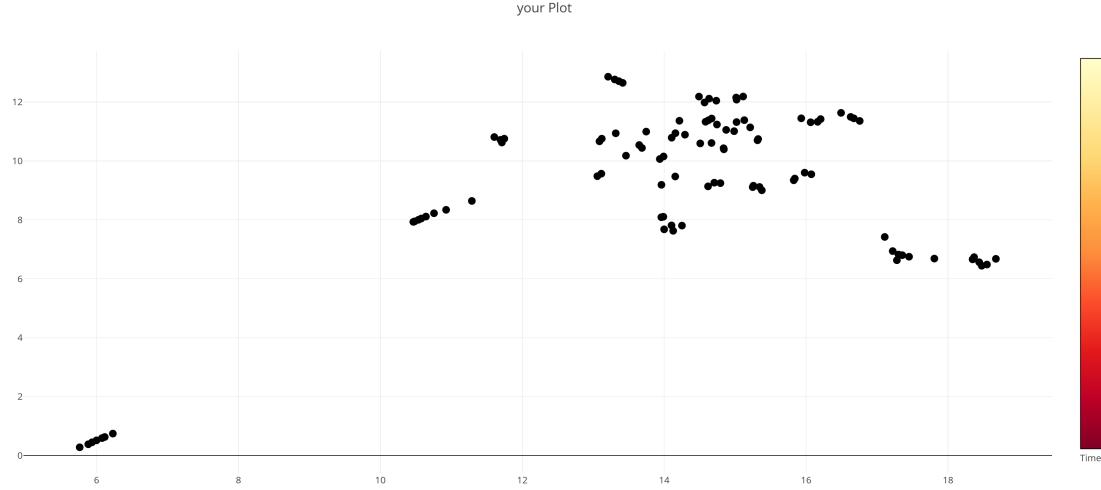


FIG. 5.19 : Résultat pipeline pour UTS

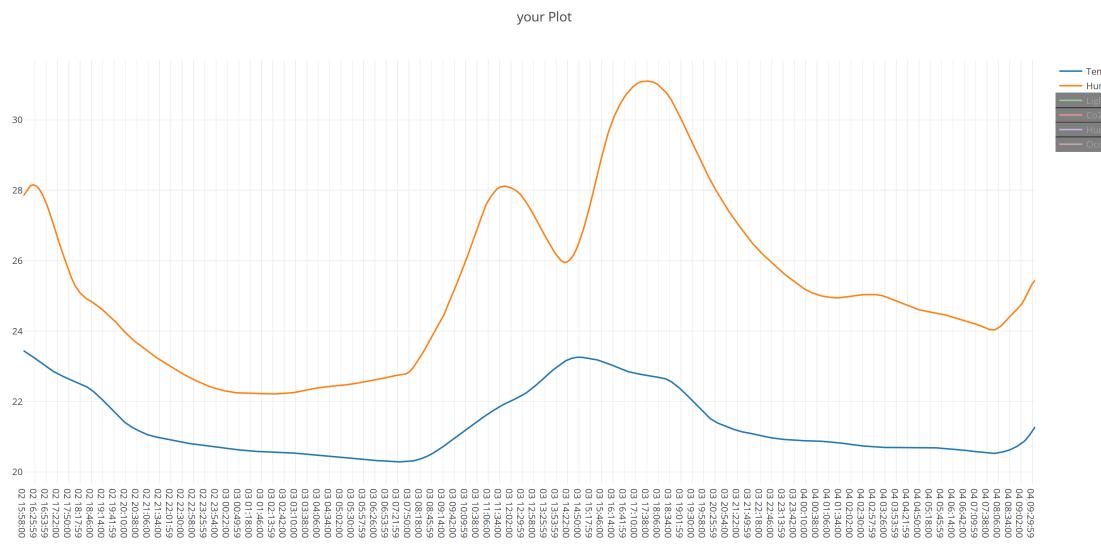


FIG. 5.20 : Résultat pipeline pour les MTS linechart

⁶<https://plotly.com/javascript/>

Technologies	Équilibrage des charges	GUI	Haute disponibilité	Évolutivité	Réseau	Simplicité
Kubernetes	besoin de configuration	+	+	automatique	manuel	-
Docker Swarm	automatique	-	+	manuel	auto	+

Technologies	open source	la tolérance aux pannes	Objectif d'optimisation
Kubernetes	yes	+	Un grand groupe
Docker Swarm	yes	-	de nombreuses petites groupes

TAB. 5.1 : Etude comparative entre Kubernetes et Docker Swarm

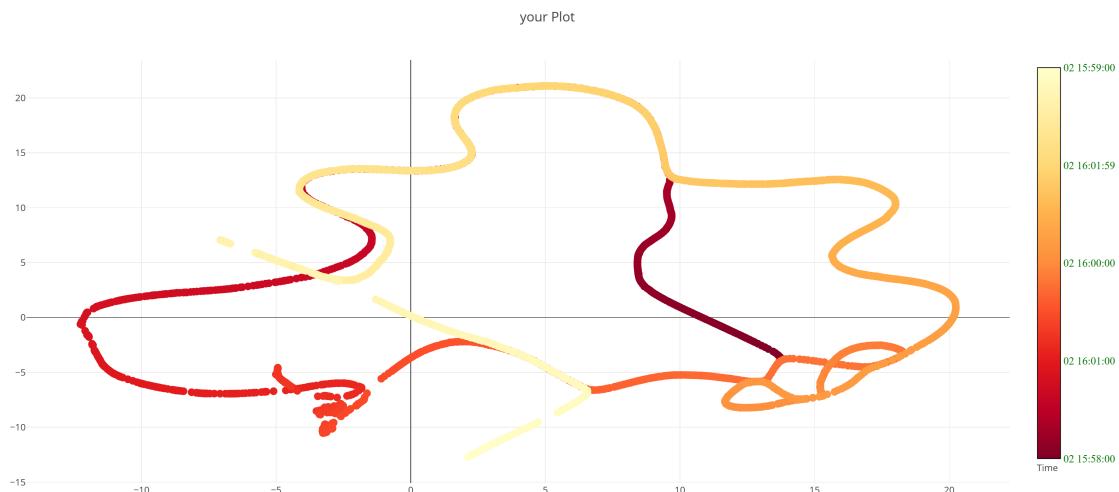


FIG. 5.21 : Résultat pipeline pour les MTS scatterchart

5.6 Déploiement

Pour avoir un déploiement facile de cette application, nous avons mis sur la table deux solutions, Kubernetes et Docker swarm le tableau suivant montrent une étude comparative des deux

Pour notre cas, nous avons choisi Kubernetes à cause de deux fonctionnalités importante, la tolérance aux pannes et la mise à l'échelle automatique. Kubernetes montre une excellente performance qu'on parle de ces deux fonctionnalités par rapport au Docker Swarm. Dans la Figure 5.22 présentons notre diagramme de déploiement pour Visio. Pour la partie SPARK elle est configurée par l'utilisateur comme nous avons indiqué précédemment.

5.7 Environnement expérimental

Apache Spark : "un moteur d'analyse unifié pour le traitement des données à grande échelle. Il fournit des API de haut niveau en Java, Scala, Python et R"⁷. Dans notre cas , il est utilisé pour exécuter des algorithmes distribués.

Apache Livy : "est un service qui permet d'interagir facilement avec un cluster Spark via une interface REST. Il permet de soumettre facilement des travaux Spark ou des bribes de code Spark, de récupérer des résultats synchrones ou asynchrones, ainsi que de gérer le contexte Spark, le tout via une simple interface REST ou une bibliothèque client RPC"⁸ . D'une autre manière, on aura un spark-shell à distance. Pour nous, Cette solution sert à communiquer les microservices avec SPARK et de gérer les sessions interactives du SPARK.

Flask : "est un framework d'application web léger du WSGI développé en python. Il est conçu pour faciliter et accélérer la programmation microservices, avec la possibilité d'évoluer vers des applications complexes"⁹.

Finatra : "Finatra est un framework léger permettant de construire des applications rapides, testables et scalables sur TwitterServer et Finagle¹⁰. Finatra fournit une API facile à utiliser pour créer et tester les serveurs et les applications Finagle, ainsi qu'une prise en charge puissante de JSON, une journalisation moderne via SLF4J, des utilitaires

⁷<https://spark.apache.org/docs/latest/index.html>

⁸<http://livy.incubator.apache.org/>

⁹<https://palletsprojects.com/p/flask/>

¹⁰<https://twitter.github.io/finagle/>

clients Finagle, et bien plus encore”¹¹. Pour les microservices en scala

Angular 8 : ”Angular est un framework de conception d’applications et une plate-forme de développement permettant de créer des applications monopages efficaces et sophistiquées”¹². Pour l’interface graphique.

Rete JS : ”Rete est un framework modulaire pour la programmation visuelle. Rete vous permet de créer un éditeur basé sur les noeuds directement dans le navigateur. Vous pouvez définir des noeuds et des travailleurs qui permettent aux utilisateurs de créer des instructions pour le traitement des données dans votre éditeur sans une seule ligne de code”¹³. Pour le visuel scripting.

Plotly : ”Bibliothèque graphique Open Source JavaScript Construit sur d3.js et stack.gl, elle est une bibliothèque graphique Open Source JavaScript Construit sur d3.js et stack.gl, déclarative de haut niveau. plotly.js propose plus de 40 types de graphiques, y compris des graphiques 3D, des graphiques statistiques et des cartes SVG. ”¹⁴. Pour les graphes de visualisation.

Docker : ”est un logiciel libre permettant de lancer des applications dans des conteneurs logiciels. Selon la firme de recherche sur l’industrie 451 Research, « Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n’importe quel serveur ». Il ne s’agit pas de virtualisation, mais de conteneurisation, une forme plus légère qui s’appuie sur certaines parties de la machine hôte pour son fonctionnement”¹⁵. Pour l’utilisation dans l’outil Kubernetes, et pour le déploiement local.

Kubernetes : ”est un système open source pour la gestion d’applications containerisées sur plusieurs hôtes. Il fournit des mécanismes de base pour le déploiement, la maintenance et la mise à l’échelle des applications”¹⁶. Pour le déploiement mode cluster.

¹¹<https://twitter.github.io/finatra/>

¹²<https://angular.io/>

¹³<https://rete.js.org/>

¹⁴<https://plotly.com/javascript/>

¹⁵[https://fr.wikipedia.org/wiki/Docker_\(logiciel\)](https://fr.wikipedia.org/wiki/Docker_(logiciel)) *cite_note – L*

¹⁶<https://github.com/kubernetes/kubernetes/>

5.8 conclusion

Notre outil, est une solution pour la visualisation des TS qui a ajouter des nouvelles fonctionnalités par rapport a d'autre solution telles que le traitement des algorithmes distribués, la gestion des pipelines et l'utilisation de plusieurs sources de données. De plus elle est extensible, donc il est facile d'ajouter des nouveaux algorithmes et graphes. Tout ça en une architecture web qui ne demandes pas d'installation localement et facile a déployer en ligne. Dans ce chapitre, nous avons présenté la l'analyse des besoins , la conception , l'implémentation et le déploiement de notre solution.

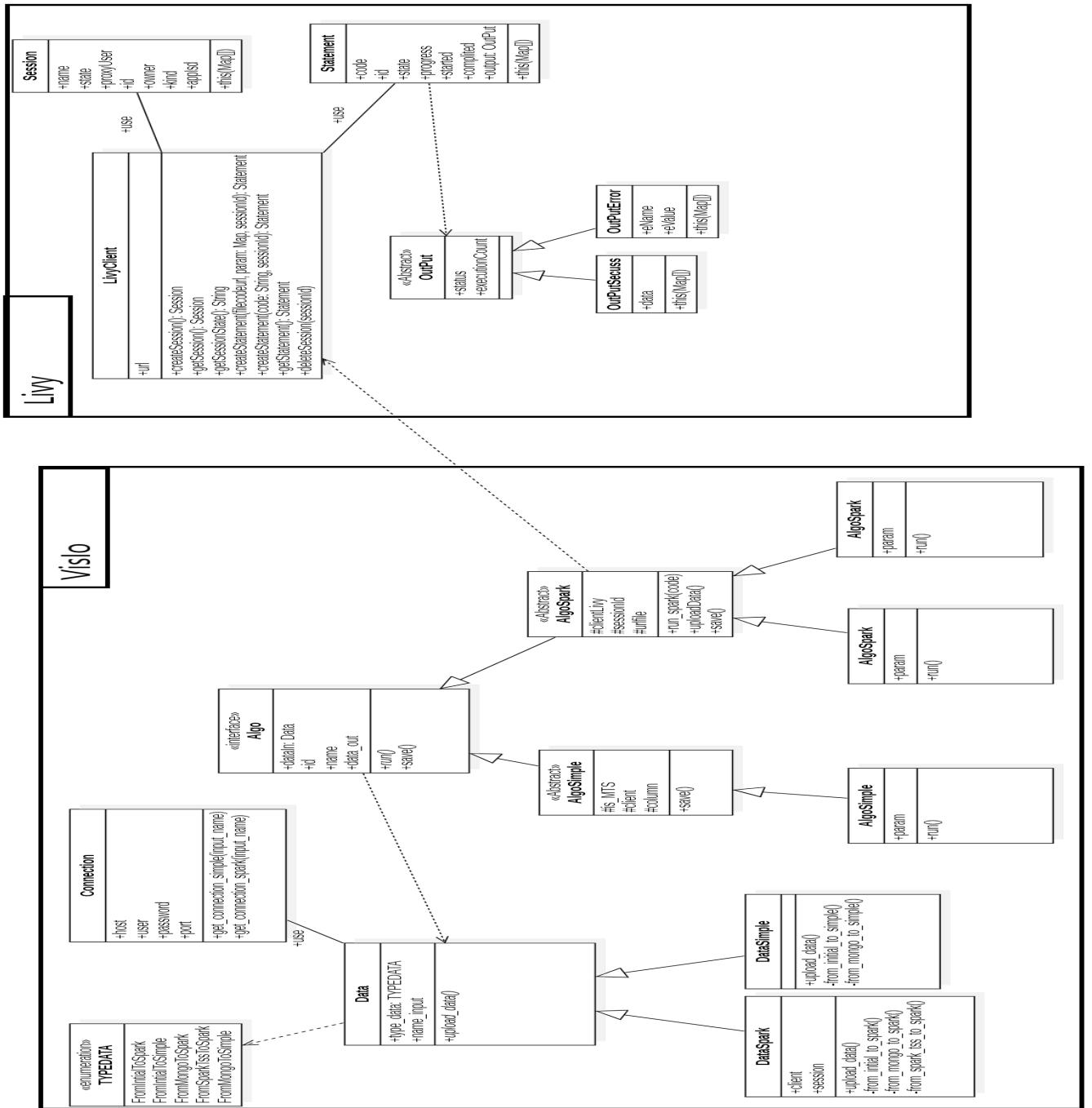


FIG. 5.11 : Diagramme de classes

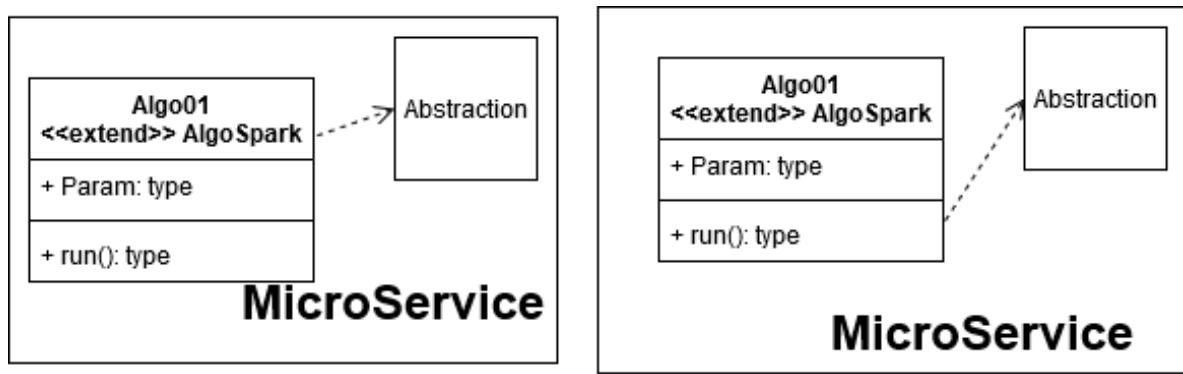


FIG. 5.12 : L'implémentation de deux algorithmes en microservice

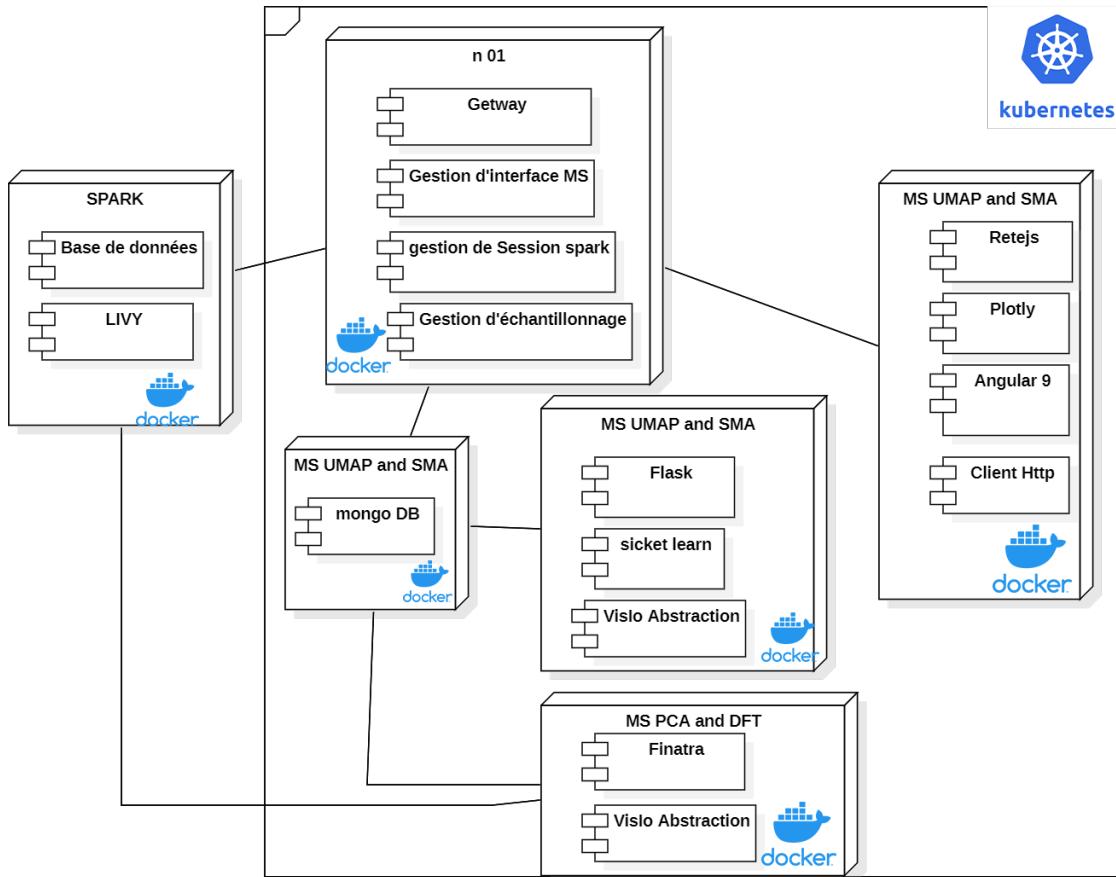


FIG. 5.22 : Diagramme de déploiement avec Kubernetes

CHAPITRE 6

CONCLUSION ET PERSPECTIVE

Tout au long de notre projet de fin d'étude, nous avons pris comme mission l'implémentation un nouveau outil pour la visualisation des TS, qui permet l'intervention d'un expert du domaine d'une fassent simple et intuitive. Cette outil comporte plusieurs fonctionnalités telles que l'extensibilité, l'interactivité , la gestion des pipelines et l'importation de plusieurs sources de données. Tous ces fonctionnalités son présenter par une interface web moderne développe en microservices. Durant l'implémentions de cet outil, nous avons appris plusieurs technologies de développement moderne en comparant les solutions existantes pour n'importe quelle tache réalise. Dans notre mémoire, nous avons présenter les différentes aspects, techniques et outils, qui nous a permis de développer cette solution.

Organiser les données massive pour la visualisation reste une tache difficile. Suit a ce travail, nous avons eu plusieurs idées pour des nouvelles méthode de visualisation des séries temporelles qui pourraient être testées. Ces idées pourraient être un sujet d'un autre travail prochainement.

ANNEXE

Comment déployer Spark ?

Pour le déploiement du moteur SPARK, il faut suivre le manuel d'utilisation dans la documentation officiel ([http ://spark.apache.org/docs/latest/](http://spark.apache.org/docs/latest/)). Notre solution ne s'occupe pas de la configuration SPARK (c'est a l'utilisateur de faire ça). Cependant elle nécessite un configuration de plus dans la partie moteur SPARK. Cette configuration consiste a installer et lancer LIVY framework. Dans la documentation officiel vous trouver comment installer et configurer LIVY ([http ://livy.io/](http://livy.io/)). Pour un test, nous avons preparer un image docker plus un script shell qui permet de lancer SPARK en local voir ([https ://github.com/PierreKieffer/docker-spark-yarn-cluster](https://github.com/PierreKieffer/docker-spark-yarn-cluster)).

Comment déployer notre solution en mode autonome

Pour lancer notre solution en mode local, il faut telecharger le code qui est dans le repo git ([https ://github.com/zekriHichem](https://github.com/zekriHichem)). Puis lancer la commande
`docker-compose up`

Comment ajouter un algorithmes ?

Pour ajouter un algorithmes il faut ajouter un microservice qui prend la forme du schéma dans les Figures 5.4 et 5.5. Nous pouvons utiliser soit Scala avec le framework Finatra, soit avec Python avec le framework Flask.

6.0.1 Ajouter un algorithme SPARK

Pour ajouter l'algorithme a notre microservice il faut suivre les étapes suivantes :

- Importer notre Abstraction.
- Crée une classe qui etend la classe *AlgoSpark* (voir le diagramme de classe dans la Figure 5.11) en ajoutons les paramètres d'algorithmes comme des attribues
- ecrire l'algorithme dans un fichier .scala en mode shell voir l'exemple du DFT ou PCA, sachant que les paramètres d'enter sans entre "<>" est l'algorithme doit retourner a la fin un variable *Algo < id >*, c'est un "Dataframe", tel que chaque colonne est une série ou bien une variable. Si l'algorithme utilise l'objet "TSS" il doit fournir en plus une variables *Algo < id > TSS*.
- redifinier la fonction *run()*. Voir le repo Github pour plus de détails.

6.0.2 Ajouter un algorithme Simple

Pour ajouter l'algorithme a notre microservice il faut suivre les étapes suivantes :

- Importer notre Abstraction.
- Crée une classe qui etend la classe *AlgoSimplek* (voir le diagramme de classe dans la Figure 5.11) en ajoutons les paramètres d'algorithmes comme des attribues
- redéfinie la fonction *run()*. sachant quelle dois remplire l'attribue *data_out* par les résultats, et *columns* par les colonnes des résultats. Voir le repo Gituhub pour plus de détails.

6.0.3 Ajouter l'algorithme a notre interface

Pour ajouter l'algorithme a notre interface il faut suivre les étapes suivantes :

- Ajouter une composant au fichier `/visualscripting/components/` qui contiens deux fonctions `builder()` et `worker()`. Voir l'exemple dans le repo Github et la documentation officiel de Rete.js (<https://rete.js.org>).
- Ajouter ce composant a liste des composants (`components`) dans le fichier `/visualscripting/visualscripting.js`.

Étude comparative entre les framework pour les microservice

TAB. 6.1 : Framework for micro-services architecture

(a) JVM Framework for micro-services architecture

APIs	Play	Spring Boot	Dropwizard	Restlet	Spark	Lagom	Scalatra	Finatra & Finagle
Operating system	Cross-platform							
Programming language	JVM like, Scala	JVM like	JVM like	JVM like	JVM like	JVM like, Java	JVM like, Scala	JVM like, Scala
Open source	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Compiled language	Yes							
Backing organization	Lightend							
Githuh stars (kStars)	12	45	8	Swagger	@perwendel	Lightbend	Scalatra	Twitter
Ease of use	Medium	Medium	N/A	9	2.4	2.5	2 & 8	
MVC	Yes	Yes	Height	Yes	Yes	Yes	Yes	
static Typing	Yes	Yes	Yes	Yes	Yes	No	Yes	
stateless	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Run speed	speed	Heavier						
Community	Limited	Large						
Scalability	has several scalability issues	Yes	Yes					
ML library	Limited Libraries	Limited librarie						
RESTFULL API	Yes	Yes	N/A	4	28	10	34	
API source weight (Mo)	119	166	112	−∞	+2	+2	+2	
Gaël Opinion	0 too heavy → machine à gas	0 too heavy	0 too heavy	proprietary framework → unopen sourced	lightweight	light warning	lightweight	lightweight production tool
Hicham Opinion	0 too heavy unwieldy	0 too heavy	good documentation	−∞ closed source	+2 Apache Foundation use it	0 some solutions not free	+2 lightweight	finatra is atop of finagle use Akka

(b) Python Framework for micro-services architecture

APIs	Flask
Operating system	Cross-platform
Programming language	Python
Open source	Yes
Compiled language	No
Backing organization	
Github stars (kStars)	49
Ease of use	Height
MVC	Yes
static Typing	No
stateless	No (built-in session)
Run speed	Heavier
Community	Large community
Scalability	has several scalability issues
ML library	Many libraries
RESTFULL API	Yes
API source weight (Mo)	10
Gaël Opinion	The python currently chosen one
Hicham Opinion	The BEST ONE

BIBLIOGRAPHIE

- [1] Ratnadip Adhikari and Ramesh K Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv :1302.6613*, 2013.
- [2] Ricardo Cachucho, Kaihua Liu, Siegfried Nijssen, and Arno Knobbe. Pipeline : a web-based visualization tool for biclustering of multivariate time series. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–16. Springer, 2016.
- [3] Sin-seok SEO CAO Anh Quan, Dohy HONG. Visualization and analysis of multi-variate time-series data. Master’s thesis, 2019.
- [4] John Corbett. Charles joseph minard, mapping napoleon’s march, 1861. csiss classics. 2001.
- [5] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction : Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1) :2859–2900, 2015.
- [6] Luther Pfahler Eisenhart. *Introduction to differential geometry*. Princeton University Press, 2015.
- [7] Haifaa Hussein Hameed. Smoothing techniques for time series forecasting. Master’s thesis, Eastern Mediterranean University (EMU)-Doğu Akdeniz Üniversitesi (DAÜ), 2015.
- [8] Dodi Heryanto and Imam Solikin. Peramalan stock motor pada pt thamrin brothers cabang tugu mulyo menggunakan weighted moving average (wma). *Media Informatika dan Komputer*, 6(1) :14–25, 2015.
- [9] Man Shan Kan, Andy CC Tan, and Joseph Mathew. A review on prognostic techniques for non-stationary and non-linear rotating systems. *Mechanical Systems and Signal Processing*, 62:1–20, 2015.
- [10] Yunqian Ma and Yun Fu. *Manifold learning theory and applications*. CRC press, 2011.
- [11] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.
- [12] Shawn Martin and Tu-Toan Quach. Interactive visualization of multivariate time series data. In *International Conference on Augmented Cognition*, pages 322–332. Springer, 2016.

- [13] J Peter May. *Simplicial objects in algebraic topology*, volume 11. University of Chicago Press, 1992.
- [14] Leland McInnes, John Healy, and James Melville. Umap : Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv :1802.03426*, 2018.
- [15] Adam Milton. Simple, exponential, and weighted moving averages. *About. com*, 2018.
- [16] Sidharth Prasad Mishra, Uttam Sarkar, Subhash Taraphder, Sanjay Datta, Devi Prasanna Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. Multivariate statistical data analysis-principal component analysis (pca). *Int J Liv Res. 2017c*, 7(5) :60–78, 2017.
- [17] V. Monbet. *Modélisation des séries temporelles*. Université de rennes 1, 2016-2017.
- [18] Nikolaos Passalis, Anastasios Tefas, Juho Kannainen, Moncef Gabbouj, and Alexandros Iosifidis. Adaptive normalization for forecasting limit order book data using convolutional neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1713–1717. IEEE, 2020.
- [19] Emily Riehl. *Category theory in context*. Courier Dover Publications, 2017.
- [20] Ian Ross. Nonlinear dimensionality reduction methods in climate data analysis. *arXiv preprint arXiv :0901.0537*, 2009.
- [21] Alexei V. Samsonovich. *Biologically Inspired Cognitive Architectures 2019: Proceedings of the Tenth Annual Meeting of the BICA Society*. Springer, July 2019. Google-Books-ID : kwSjDwAAQBAJ.
- [22] Winita Sulandari and Yudho Yudhanto. Forecasting trend data using a hybrid simple moving average-weighted fuzzy time series model. In *2015 International Conference on Science in Information Technology (ICSITech)*, pages 303–308. IEEE, 2015.
- [23] Li Wei and Eamonn Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–753, 2006.