Stage
Cool

GANZA Mykhailo
M2 GEII

Développement d'un outil d'exploration des séries temporelles provenant de différentes sources

03/12/15

Remerciements

Introduction

Table des matières

Partie 1 : Présentation

1 Présentation de l'entreprise

Présenter Maya

2 Présentation de contexte de stage

2.1 IoT

IoT, Internet of Things, (internet des objets, objets connectées) est une extension d'internet classique (« un réseau des réseaux ») vers le monde physique grâce aux systèmes embarquées. La plupart des temps ces systèmes réalisent l'acquisition des données de leur environnement qui sont ensuite partagée via réseaux. L'exemple d'un objet connecte peut être un moniteur de fréquence cardiaque, ou un capteur de luminosité connecté, smartphone, etc. Peu importe quel équipement électronique ayant une adresse IP, et la possibilité d'échanger des données via réseau, peut être considéré comme un acteur dans l'IoT.

Le concept de connexion à internet des équipements électroniques pour fournir certains données est apparu pour la première fois à 1982 à l'Univeristé Carnegie-Mellon (états-Unis). Il s'agisse d'un bricolage de distributeur de Coca Cola pour sonder à distance (via réseau APRANET) la présence (ou absence) et la température des bouteilles de coca dans la machine[1]. Ensuite, depuis les années 1999 le domaine des objets connectées à commencée de prendre d'ampleur avec la présentation par Bill Joy (Unix BSD) d'un protocole de communication D2D (Device to Device) : un réseau des capteurs qui vont « fusionner les systèmes embarquées avec la vie quotidienne »[2]. En même année le terme « Internet of Things » a été inventée par Kevin Ashton[3]. Un autre éventement important dans l'histoire d'IoT est la création en 2005 d'une carte Arduino, qui avait beaucoup d'impact sur les objets connectées[4], surtout pour les amateurs de domotique. Depuis les années 2013 le marche d'IoT est en plein expansion avec la croissance grandiose des smartphones, montres connectés, capteurs cardiaques, différentes sortes des objets « smart » ayant une connexion internet. Et de plus en plus des acteurs principales d'industrie numérique (Google, Samsung, etc) participent pour apporter des innovations et gagner une marge de se secteur.

Les objets connectées ont pénétrées différentes domaines remplissant divers tâches. Les plus notables sont la médecine (« santé connectée ») où les objets connectées sont utilisées pour le monitoring en temps réel des indices de santé d'un patient. Un autre domaine important d'IoT est la domotique, avec toute une panoplie des capteurs de luminosité, vidéosurveillance, frigos

1 https://www.cs.cmu.edu/~coke/history_long.txt
2 https://www.technologyreview.com/s/404694/etc-bill-joys-six-webs/
3 http://www.rfidjournal.com/articles/view?4986
4 http://www.forbes.com/sites/gilpress/2014/06/18/a-very-short-history-of-the-internet-of-things/3/#540eb98143c5

connectées, etc. Un autre exemple d'application des objets connectées sont les capteurs environnementaux, qui observent l'état de certains grandeurs physiques (qualité de l'eau, indices de séisme) pour un but préventive et protection d'environnement

Vue le nombre des applications possibles et la quantité des objets connectées actuel (6.4 milliard objets prévus pour 2016[5]) une des évolutions logiques est la standardisation d'IoT. Sachant que le principe même de fonctionnement d'internet est basé sur l'utilisation des mêmes protocoles standards par toutes les équipements, les objets connectes pour être considéré comme des vrais « objets d'internet », doivent être standardisées, et être capables de se parler entre eux. Actuellement la communication des objets connectées est basée sure les protocoles définis et standards comme TCP/IP, HTTP, etc. Mais au dessous des protocoles connus, dans un couche des données métier, le protocole de communication entre les objets connectées n'est pas du tout homogène d'un solution à autre. Les protocoles de niveau le plus haut sont différent d'une solution à autre et sont très homogènes. Par exemple, la nature des données échangées par un réseau des capteurs de luminosité et une infrastructure des frigos connectées est complètement différente. Il y a un besoin d'un langage standard pour que toutes les objets connectes, malgré le fournisseur, puissent se parler entre eux. Une des avancement dans cet direction est le système EPC[6], qui en perspective pourra être utilisée comme une base commun pour développer un tel langage universel.

Il est important de préciser que l'IoT ne correspond pas à une seule technologie, il s'agit plutôt d'une systèmes des systèmes, une infrastructure des plusieurs tiers. Et comme les objets connectées sont en plein développement, la panoplie des secteurs pour R&D dans l'IoT est très vaste : développement des microprocesseurs et cartes spécifiques (ex : Intel Edison), différentes sortes des capteurs miniaturisées (ex : accéléromètre, thermomètre etc), solutions middleware, sécurité des données, etc. Une des autres aspects importantes de l'IoT est l'analyse de données acquis.

2.2 Analyse des données

L'aspect d'analyse données dans l'IoT prend de plus en plus d'ampleur. Les informations acquis ont beaucoup de valeur, car ils peuvent être analysées afin de tirer des indices importantes pour la prise des décisions stratégiques pour l'entreprise. Par exemple, clôturer l'utilisation des certains cellules dans un réseau des capteurs de luminosité à cause de la puissance négligente mesurée. Dans ce cas il est possible d'analyser les données manuellement : il suffit juste d'analyser toutes les relevés. Aussi, un module d'analyse des données peut faire une partie d'un produit final. Par exemple la détection de chute de rythme cardiaque mesurée par un montre connectée d'un patient.

Si on voudra concevoir une système automatisé d'analyse des données pour l'IoT on sera face aux trois problèmes les plus pertinentes: volume, approche, et l'hétérogénéité des types de stockage et formats. Vue la quantité

[5] http://www.gartner.com/newsroom/id/3165317
[6] https://fr.wikipedia.org/wiki/Code_produit_%C3%A9lectronique

des objets connectées et haute fréquence des mesures effectuée, une dimension des données acquis peut être assez volumineux. De se fait l'IoT se pose fortement sur les pratiques de Big Data pour la gestion des énormes massives des données. Une autre point problématique est le choix d'approche. C'est une question de recherche d'un algorithme le plus adaptée pour tirer les conclusions nécessaires à partir d'une ensemble des données acquis. Il s'agit d'un domaine des statistiques, l'utilisation des classificateurs, recherche des trends, étude de similitude, etc. Ces techniques sont très souvent utilisé dans l'IoT pour le but de l'exploration des données.

La dernière problématique, ou si on va se mettre dans un contexte d'un outil automatisée d'analyse des données, une point à considérer, est la diversité des formats et façons de stocker les données. Comme on a vu dans un chapitre précédent, le marché des objets connectées actuel se pose sur les protocoles commun de communication, mais pas jusqu'au point d'utiliser le format des données standard, c'est-à-dire même pour toutes les types des objets connectées. Pour avoir plus des exemples sur les formats des données, cf chapitre suivante. Cela se complexifié aussi par le fait que les domaines d'utilisation d'IoT sont très variées, et pour chaque cas d'utilisation la nature des données métiers sera complètement différente.

Cet heterogenite se complexifie encore plus par l'utilisation des différentes outils de stockage : souvent les données sont stockées dans les bases des données relationnelles (comme MySQL, Oracle), ou les bases des données orientées document (MongoDB, Couchbase), qui sont de plus en plus utilisée vu leur structure des données flexible. Rien n'empêche d'utiliser les fichiers texte (JSON, CSV, ARFF), ou d'autres formats rares ou exotiques (photos, fichiers binaires, etc).

La question de multitude des formats et types de stockage est importante à résoudre, si on veut concevoir un outil automatique **et generique** d'analyse des données, car cet outil doit pouvoir accéder à plusieurs types de stockage et lire les données de structure variable.

Partie 2 : Étude de problématique

1 Objectif initial

Des le début, l'objectif de stage est le suivante :

« Concevoir un outil de l'exploration des relations entre les données ayant une nature différente provenant de sources différentes»

Il s'agit d'un application qui sera capable de faciliter l'exploration des relations entre les données qui ont été générées par les sources différents. Par exemple, comparer l'évolution d'une mesurée horaire de la température dans une pièce avec les relevés de la température venant d'un site météorologique.

La problématique de « sources différentes » peut se conclure à trois point principaux, lorsque on veut concevoir un outil d'exploration qui travail avec

plusieurs sources. Il s'agit de désynchronisation des instant de prise de mesures, les stockages varie, le format différent. L'application doit pouvoir gérer cet nature heterogene des données.

Les concepts « exploration des données» et « relation entre les données» sont à définir par stagiaire. Une seule contrainte à suivre est que l'outil doit pouvoir travailler avec au moins deux types des stockage : les bases de données MongoDB et MySQL. Bien sur le format et la structure des données n'est pas imposée. Aucune autre indice n'est pas présentée.

Cela revient à stagiaire de choisir les technologies les plus adoptés pour résoudre la problématique posée, définir le cahier de charges de l'application. Vue la nature generique de l'objectif, le stagiaire est amenée a explorer la problématique posée, afin de définir les frontières et spécificités nécessaires pour apporter une solution concret et satisfaisante.

2 Étude initial

L'objectif initial est assez étroit et impose que l'utilisation des systèmes des stockage (MongoDB et MySQL). Mais il y a un autre contrainte sous attendu : le stage se porte dans un domaine d'IoT, et on peut déjà y tirer une conclusion : l'outil d'exploration doit au moins être capable de travailler avec les données spécifiques à l'IoT.

Vue cet première précision importante, le milieu de recherche initial avait pour le but d'explorer et apprendre le façon dont les observations provenant des objets connectées sont sauvegardées. Il s'agit des séries temporelles : une manière intuitive d'organiser les mesures au fil de temps.

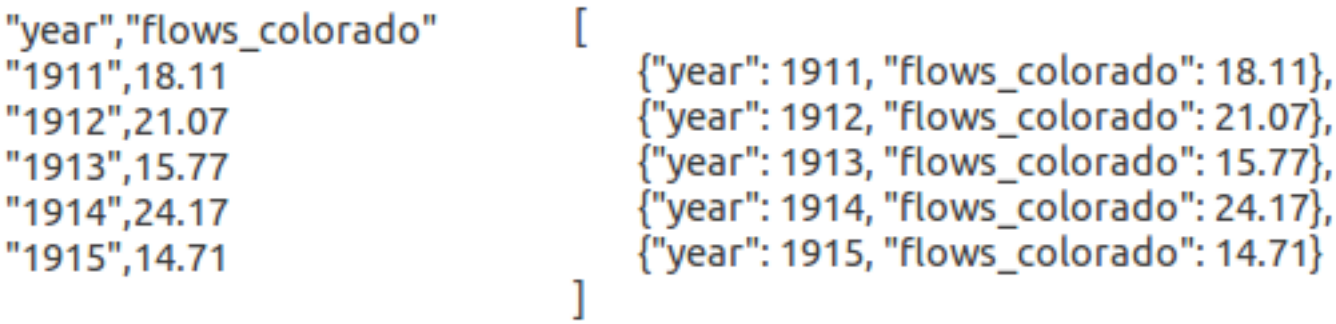
2.1 Séries temporelles

Une série temporelle, ou une série chronologique, est une suite des valeurs numériques qui évolue suivant le temps. La plupart de temps les séries temporelles sont représentées par une ensemble des associations clef-valeur avec une clef-valeur spécifique d'horodatage qu'on appel timestamp.

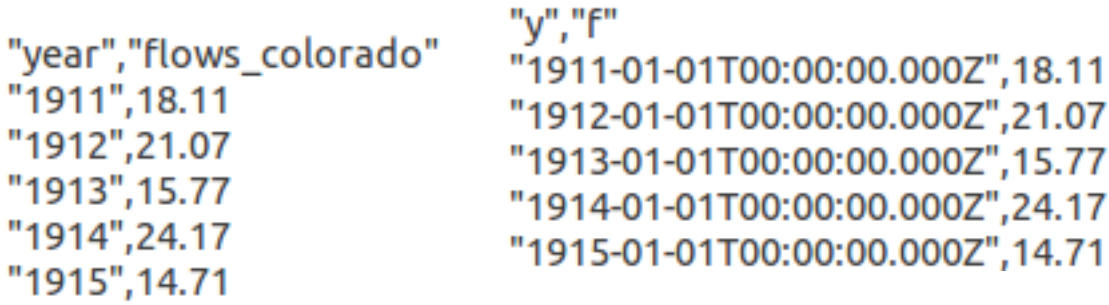
La plupart de temps les objets connectées observent l'évolution de certains grandeurs physiques au file de temps. Du coup l'utilisation des séries temporelles dans l'IoT est adapté : chaque observation est accompagnée d'un timestamp avec le temps de prise de mesure. Il faut préciser bien sûr, qu'il s'agit des domaines d'application ou l'utilisation des séries temporelles est optimale, par exemple, pour une caméra de vidéosurveillance connectée l'utilisation de ce type d'organisation des données ne sera pas vraiment adaptée et naturelle.

2.2 Problématique

Comme il était dit dans un chapitre précédent, les formats des données et les types de stockage peuvent être différentes. Par exemple voici une même série temporelle, mais présentée par différentes stockages comme les fichiers JSON (à gauche) et CSV (à droite) :



Ou on peut imaginer un cas ou les containeurs sont identiques, mais la structure des données et format ne sont pas les mêmes :



Dans une exemple à droite on peut voir une autre problématique mineur qui se pause, et peut être rencontrée lors de gestion des séries temporelles : le format de timestamp est aussi variable. La plupart de temps le timestamp suis un format standard ISO8601, ou souvent on utilise un timestamp unix : le nombre des secondes écoulées depuis le 1er janvier 1970. Mais de temps en temps le format de temps peut être spécifique, par exemple dans un cas d'une série temporelle avec les mesures mensuelles de niveau de précipitation. Donc, cet aspect variable des séries temporelles doit aussi être géré si on veut concevoir un outil qui pourra travailler avec plusieurs format et sources des données.

2.3 Première palier

Issue de cet étude initial on a un invariant de base, sur lequel l'outil d'exploration des données peut se fonder : on travaille uniquement avec les séries temporelles car c'est une façon naturelle de stocker le mesures. C'est qui implique, par la définition des séries temporelles, que l'application va fonctionner seulement avec les datasets ayant les caractéristiques suivantes :

- chaque échantillon d'une série temporelle doit être accompagné par un seul timestamp
- chaque échantillon doit avoir au moins un attribut associée (mesure)
- les timestamps doivent suivre un ordre chronologique

Bien sûr, même si les séries temporelles est une façon très répandu de stockage des observations, il peut y avoir d'autres solutions pour organiser les données. Évidement ces formats spécifiques ne seront pas acceptée par l'application. Mais le choix de se concentrer sur les séries temporelles est

argumentée par :

- pour arriver à un outil réel, il faut spécifier ces fonctionnalités
- c'est un façon d'organiser les données largement répandu
- est compatible avec MayaNet

Issue de cet étude initial on a un pierre d'angle commun pour toutes les types de stockage et formats possibles: les données sont représentée par les séries temporelles.

3 Première approche

En se basant sur le premier mini étude sur les séries temporelles, la première application naïve est apparu.

3.1 Module unique

De le début un outil à développer était vu comme un seul unité de traitement, qui peut recevoir en entrée des séries temporelles provenant de différents types de stockages et ayant des formats différent, et en sortie pourra fournir, sous un certain format, les indicateurs caractérisant les données. Et s'il le faut, les indices caractérisant les relations entre les données.

Gardant cet idée dans la tête, la première tentative pour répondre à une problème a était proposée. L'application était écrit sous Java, utilisée Morphia, et travaillée avec MongoDB.

MongoDB

Le choix de MongoDB, comme un type des stockage de base, est argumenté par le fait que c'est une base dés données orientée document, et par rapport à les bases de données classiques, MongoDB permet d'avoir un format flexible des données. En quelque sorte, MongoDB (avec Morphia) était considérée comme une solution pour la problématique de gestion de plusieurs formats, parce que cet base de données peut facilement travailler avec les format différentes des données. ~~La raison est que la structure des données stockée dans cet base des données n'est pas fixe (). car elle ne nécessite pas la structure des données rigide.~~

Aussi, MongoDB est une base de données sur laquelle MayaNet vise de migrer en future (encore une fois, à cause de dynamisme de format). Cet base de données est en perspective un seul et unique type de stockage qui sera utilisée par MayaNet.

Vue ces deux arguments, la première application à stockée les séries temporelles dans une base des données MongoDB.

Java

L'utilisation de Java comme une langage de programmation a était aussi influencé par MayaNet car elle utilise Kura[7]. Kura est un framework Java « basé sur OSGi pour les applications M2M s'exécutant dans les passerelles de services »[8]. Il s'agit d'un solution logicielle faisant un intermédiaire entre réseau filaire (ou sans fil) des objets connectées (capteurs, etc) attachés à un concentrateur et un réseau local (ou internet).

Donc, en se basant sur le principe que l'outil qu'on vise à développer, sera utilisée (ou pourra servir) pour l'IoT et notamment MayaNet, l'utilisation de Java semblé appropriée.

---

7 http://www.eclipse.org/kura/

8 http://www.electronique-mag.com/article8518.html

Morphia

Encore une fois, l'utilisation de Morphia était argumenté pour être compatible avec MayaNet. MayaNet utilise les objets POJO pour définir les données métiers. POJO (Plain Old Java Object) est un simple classe Java, qui par rapport aux JavaBeans, ne suis pas des conventions strictes.

Morphia est un wrappeur de driver Java pour MongoDB qui s'en sert des POJO (en ajoutant quelques annotations) pour communiquer avec une base de donnés. Morphia est un couche d'abstraction de MongoDB qui facilite beaucoup l'interface entre les objets métier et la base des données.

L'idée derrière l'utilisation de Morphia est que pour peut importe quel type d'objets connectes que MayaNet pourra servir, il faudra créer des objets métiers (des POJO). Donc, comme les POJO sont crées dans toutes les cas, pour toutes les « things », un outil automatisée d'analyse des donnée pourra utiliser Morphia comme un interface entre peu importe quel format des objets métiers et la base des données.

Du coup, l'utilisation de Morphia est vue comme une réponse à une problématique de multitude des formats des données. Car par la suite, chaque format sera défini dans les POJO, et Morphia offre de façon universelle (même pour toutes les POJO) d'écriture et lecture dans la base des données MongoDB.

Récapitulatif

La première tentative naïve de répondre à une problématique posée est largement inspirée par les choix technologiques de MayaNet, pour un but d'être compatible avec la dernière. C'est un application qui est capable de travailler avec les formats des données différentes grâce à l'utilisation de Morphia et MongoDB. Le but d'un étape de développement qui devez suivre est d'englober cet l'application par un couche de traitement des données ~~qui sera aussi uniforme pour toutes les formats possibles des données~~.

3.2 Problématique

Le première approche de développer un outil d'analyse des données a permit de se familiariser avec la problématique de multitude des formats et avoir une première initiation à les données réels. Mais la solution proposée n'était pas du tout satisfaisante à cause des plusieurs points.

Tout d'abord, l'utilisation de Java. L'objectif initial n’impose pas la compatibilité avec MayaNet, c'était une décision de stagiaire de partir vers une solution qui sera à priori compatible. De plus, MayaNet utilise Kura (et donc Java) pour les couches très bases de communication entre les objets connectés, concentrateur, l'IHM. Par contre l'application qu'on vise à développer doit servir à l'utilisateur pour explorer les données, et donc elle se place logiquement dans un couche plus haut, loin des objets métiers de MayaNet.

Une autre problème évidente de première approche c'est le non respect d'une seule consigne imposée : l'utilisation d'au moins deux types de

stockages : MongoDB et MySQL. Morphia est utilisable seulement avec MongoDB. Par contre il existe une DAO (un outil d'abstraction des données comme Morphia) Hibernate OGM, qui permet de travailler avec MongoDB et MySQL, mais les POJO ne sont pas interchangeables entre les deux[9]. Donc cet approche à résoudre la problématique de multitude des formats et stockages n'est pas valable.

Mais la conclusion très importante issue de cet première approche, est une réflexion sur le façon comment l'utilisateur pourrai explorer les données, et surtout, les relations entre les données. Cet fonctionnalité n'a pas était rajouté à cet premier application parce que, évidement, l'application ne correspond vraiment à l'objectif posée, mais encore une fois, elle a permit d'initier la recherche des solutions pour offrir la possibilité d'explorer les données.

4 Exploration des donnés

En général le terme « exploration des données » correspond à une acte d'observation et analyse des données acquis, afin de tirer des informations supplémentaires sur les données. En contexte de stage, l'étendue de terme « exploration des données » n'est pas vraiment précisée dans l'objectif initial. Il peut s'agir d'utilisation des techniques statistiques, ou juste simple visualisation par une table, c'est qui est aussi une manière d'exploration des données sous forme des valeurs numériques. Il va falloir fixer un sens concret pour se terme afin d'apporter une solution réel.

Il y plusieurs façons pour analyser et explorer les données, mais on va étudier seulement les deux les plus communs et convenables. Tout d'abord, l'utilisation des techniques statistiques avancées de machâge des données pour y tirer des connaissances. Un autre façon, plus simple mais aussi efficace, est une visualisation des données via différentes sortes de graphes pour mettre en évidence leurs caractéristiques importantes afin d'être facilement analysables par l'humain. Les deux approches ont ces défauts et avantages.

~~En fait, deux façons sur lesquelles la question est : est-ce que l'outil qu'on vise à développer va posséder des techniques avancées de mâchage des données pour y tirer de connaissances, ou l'outil va visualiser les données de tel sorte, pour mettre en évidence leurs caractéristiques importantes. Chaque approche à ces défauts et avantages.~~

Les techniques statistiques qu'on peut utiliser pour explorer les données, et surtout relation entre les données, peuvent être simples comme recherche de max ou min, filtrage, moyenne glissante. Mais ils peuvent être aussi avancées comme, par exemple, l'étude des trends, l'utilisation des estimateurs, techniques d'extrapolation, classification, etc. L'avantage d'utilisation des méthodes statistiques est qu'ils sont très performantes pour trouver du valeur cachée dans les données. La problème est que, si on vise à développer un outil purement « statistique », une base de cet outil doit présenter des techniques avancées et complexes de traitement des données, et donc, la mise en place des ces méthodes risque d'être assez complexe. Surtout pour quelqu'un non-spécialiste de domaine. Par contre, rien n'empêche d'appliquer au moins les caractéristiques statistiques simples, pour tirer plus d'information à partir des

[9] http://hibernate.org/ogm/faq/

données.

La façon la plus simple et évidente, pour interpréter et explorer les données est l'utilisation des graphiques de différent sorte comme les histogrammes, nuage des points, etc. Les avantages et défauts de cet approche sont opposés à celles d'une approche statistique : la visualisation par les graphiques est simple à mettre en place, mais l’étendu d’exploration des données, et relation entre eux, n'est pas autant avancée.

Les critères sur lesquelles le stagiaire se base pour définir le terme « l'exploration des données » en contexte d'un outil à développer sont:

- question de temps de développement
- compétence de stagiaire
- l’utilité pour un cas d'usage réel
- découverte de DC.js (chapitre suivante)

Vue toutes ces points, et surtout les aptitudes naturelles de stagiaire, l'exploration des données sera réalisée grâce à un outil de visualisation avancée DC.js (qui en quelque sorte, permet de mettre en évidence la relation entre les données) et quelques statistiques simples à mettre en place pour fournir plus d'information. Cet solution est un mix entre les deux approches exprimées ci-dessous car on utilisera une visualisation de graphiques, mais pas simple et avancée, et on va utiliser certains techniques statistiques, mais pas avancée et simples. Et surtout, c'est une solution le plus réaliste et possible à produire pour fournir un outil fonctionnelle.

5 DC.js

DC.js est une librairie JavaScript basée sur Crossfilter.js et D3.js.

5.1 Crossfilter.js

Crossfileter.js est une librairie open source d'exploration de massives des données multi variantes Elle permet d'explorer les relations entre plusieurs attributs au sein d'une suite des mesures.

Pour expliquer l'intérêt d'utilisation de Crossfilter.js on invite à consulter une page officielle de Crossfilter[10] avec un exemple d'utilisation concret qu'on va exposer ici. On précise que le chargement de cet page pourra prendre quelques secondes. Un autre remarque à noter et que les graphes ont était générées via D3.js et sont cliquables (expliquée par la suite).

L'exemple est construit à partir d'un extrait de dataset[11] des vols réalisés entre 1 janvier et 31 avril 2001. Chaque instance de cet base à 29 attributs (mesures), dans l'exemple seulement 4 ont été étudie : heure de départ, retard à l'arrivée (en minutes), distance parcouru (en milles) et une date de départ.

Chaque attribut est représenté par une histogramme. Sur chaque histogramme il est possible de choisir une plage des valeur, c'est-à-dire appliquer un filtre. Grâce à Crossfilter.js, le filtre sera affectée à toutes les autres histogrammes. Il est possible d'utiliser plusieurs filtres à la fois.

Tout cela nous permet, par exemple, d'explorer les relations suivantes entre plusieurs mesures (cf Figure 1):

- quels sont les journées de la semaine avec les vols les plus longues ?
- quels sont les distances de vols à choisir (imaginons que c'est ce qu'on cherche), pour avoir le moins de retard ?
- quels heures de départ il faut choisir pour avoir le plus de retard (si on aime bien passer du temps, par exemple)?

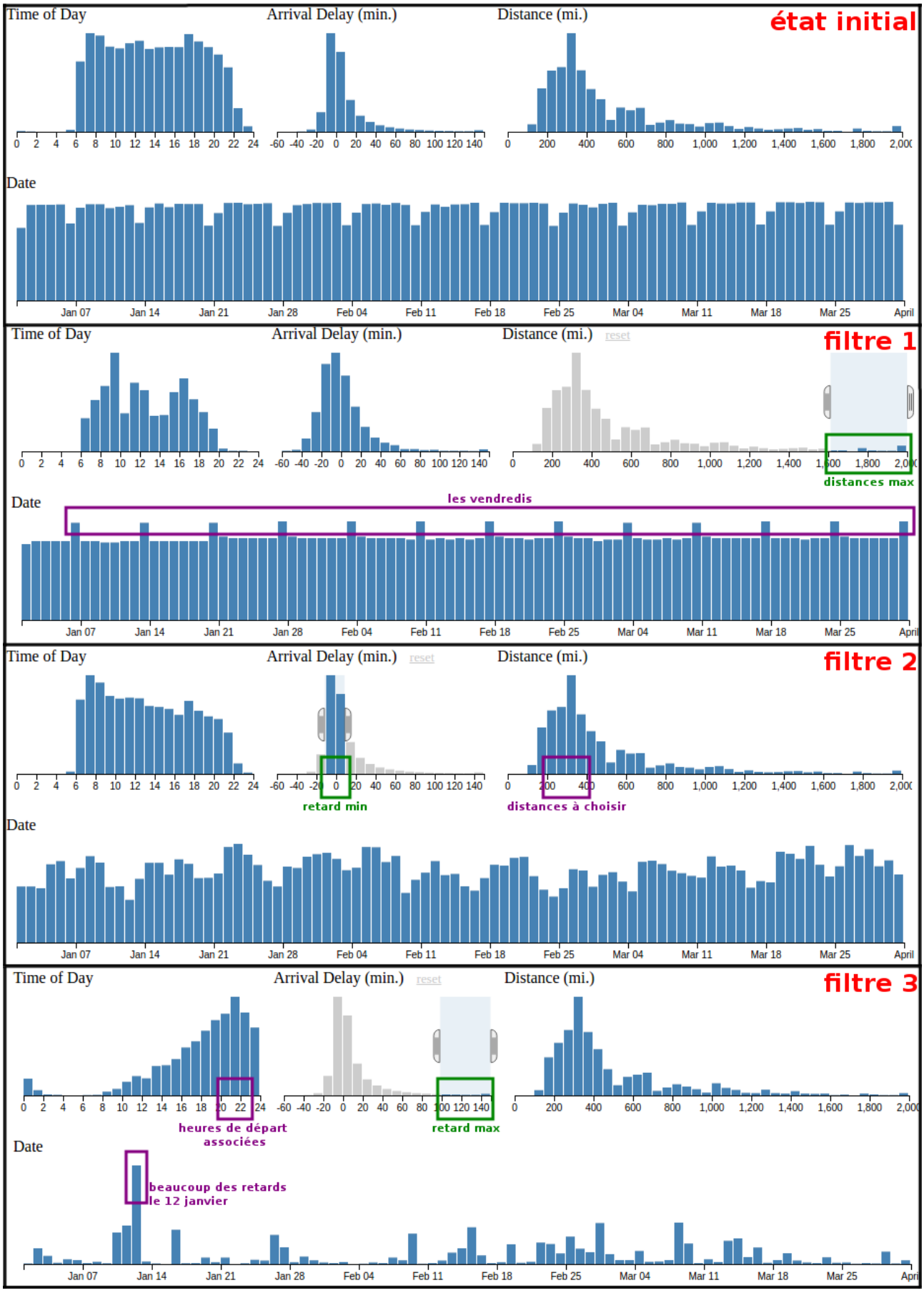
On peut dire que certains relations sont absurdes, mais ils existent, et donc sont explorables par Crossfilter.js. En fait, les correspondances entre peut importe quels attributs au sein de même mesure sont explorables par cet librairie.

La puissance de Crossfilter.js et que les filtres sont appliquée en temps réel, c'est qui implique que lorsque les données en entrée ont était modifiées, les caractéristiques calculées seront modifiées instantanément. Et cela permet d'explorer l'influence de chaque mesure sur le résultat final. Par illustrer cela, prenons un exemple d'une série temporelle, et on va utiliser crossfilter pour compter le nombre d'occurrence de chaque valeur mesurée (Figure chepa). A l'entrée on a un dataset, en sortie on a un dataset. Si en instant t les données en entrée sont modifiées (on appliquée un filtre), le dataset en sortie sera instantanément modifiée.

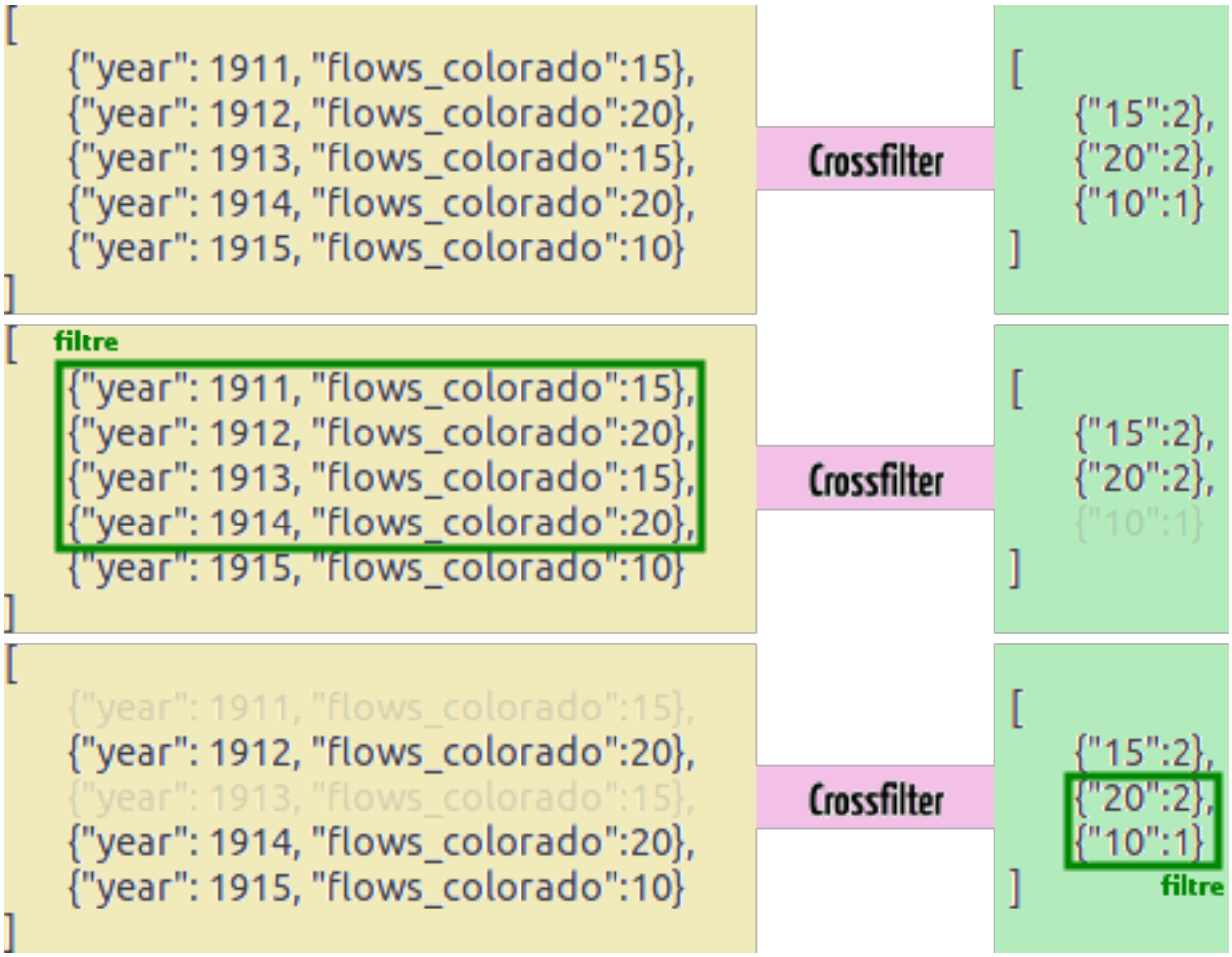
Comme il était dit avant, crossfilter support les filtres

[10] http://square.github.io/crossfilter/

[11] http://stat-computing.org/dataexpo/2009/the-data.html


Time of Day
Arrival Delay (min.)
Distance (mi.)
état initial
Date
Jan 07
Jan 14
Jan 21
Jan 28
Feb 04
Feb 11
Feb 18
Feb 25
Mar 04
Mar 11
Mar 18
Mar 25
April
reset
filtre 1
distances max
les vendredis
filtre 2
retard min
distances à choisir
filtre 3
heures de départ associées
retard max
beaucoup des retards le 12 janvier

multidirectionnelles, c'est-à-dire dans le cas de cet exemple, on peut appliquer un flitre sur les caracteristiques (les sorties), pour voir comment les données à l'entrée vont être changée.



En conclusion, Crossfilter.js est un outil puissante d'analyse des massives des données, mais il ne s'agit pas d'une bibliothèque graphiqe, c'est plutôt un engine d'exploration. Pour visualiser les relations dynamiques mis en évidence par crossfilter, il faut utiliser une autre librairie JavaScript pour la rendu des graphes. Il est possible d'utiliser peu importe quel bibliothèque graphique JavaScript, mais pour tirer le max de Crossfilter.js, il faut considérer deux point importantes à suivre pour choisir une bibliothèque graphique appropriée: l'interfaçage et dynamisme.

Comme Crossfilter.js travaille directement avec les données, si on veut l'utiliser avec une librairie de rendu visuel, la dernière doit être capable de générer les graphes directement (ou avec des efforts minimales) à partir des données. Donc, une question d'interfaçage simple est importante, et dans le cas de crossfilter il s'agit d'utiliser les données brutes pour la rélier avec un outil de rendu visuelle. Un autre point est le dynamisme. Pour se profiter d'utilisation des filtres interactives de crossfilter, la bibliothèque graphique doit supporter de transitions, morphing, passage des objets visuelles d'un état à un autre. En plus, comme l'application des filtres crossfilter se traduit par la

modification des données, la librairie graphique doit être capable de détecter un tel changement.

Un bon candidat ayant une bonne compatibilité avec Crossfilter.js est D3.js. Ces deux bibliothèques JavaScript sont souvent utilisées ensemble (comme dans un première exemple de ce chapitre), mais il peut être utilisée avec d'autres librairies graphiques comme Chart.js[12] ou NVD3[13].

5.2 D3.js

D3.js (Data-Driven Documents) est une bibliothèque JavaScript graphique qui permet la visualisation des données numériques en forme des graphiques dynamiques. Elle est conforme W3C et utilise les technologies SVG et CSS. Les point forts de cet librairie graphique est dynamisme (les objets graphiques peuvent réaliser des transitions) et le fait qu'elle est « data-driven ».

Les objets visuelles rendu par D3.js via les SVG : les images vectorielles et dont le style peut être modifiable même par un les selectors CSS. C'est qui est la raison pourquoi tout sur les graphes dans D3.js peuvent « bouger ».

Pour expliquer pourquoi D3.js est « data-driven » il faut tout d'abord se rappeler de quelques notions : toutes les éléments d'une page web (les tags html) sont organisée dans une hiérarchie qu'on appel DOM. Le DOM et « accessible » et peut être programmée (ex : jQuery). La puissance de D3.js est qu'elle utilise le « data-binding » : elle relie les données bruts (sous un format JSON) et les éléments de DOM, dont les SVG pour le rendu visuel des objets graphiques. Tout cela garantit que lorsque les données ont était changée, les graphes vont changer.

Pour conclure sur D3.js, il faut dire que ce n'est pas une librairie de génération des graphiques scientifiques, mais plutôt de rendering des objets visuelles. On invite à consulter la page principale de D3.js[14] pour se rendre compte de toutes les domaines d'application de cet bibliothèque graphique. Cet flexibilité de génération des objets visuelles implique que langage graphique de D3.js est assez bas: on travaille avec les rectangles, cercles, sélecteurs des éléments html et donc le temps d'apprentissage nécessaire pour visualiser des graphes simples est importante.

5.3 DC.js

DC.js (Dimensional Charting) et basé sur deux bibliothèques JavaScript : Crossfilter.js pour filtrage multidimensionnelle et multidirectionnelle entre les données, et D3.js pour le rendu des objets visuelles dynamiques. Ces deux librairies vont naturellement bien ensemble, et peuvent même être utilisées en tant que tel sans aide de DC.js. Ce qui permettra de générer des visualisations plus flexibles et configurables, mais il y a quelques raisons pour l'utilisation de DC.js.

Tout d'abord, la courbe d'apprentissage de D3.js est exponentielle, il faut passer beaucoup de temps pour apprendre comment l'utiliser pour rendre des

12 https://github.com/nsubordin81/Chart.dc.js/tree/master

13

14 https://d3js.org/

graphiques simples. De plus, comme D3 travail avec les objets graphiques simples comme rectangle ou cercle, et de plus, les sélecteurs, la mise en place des graphes peu être assez complexe. Bien sûr le temps fourni à cet apprentissage va permettre à développer des visualisations splendides et complexes, comme par exemple celle-ci[15] , réalisée en pure D3.js. Mais comme en chapitre 2.3 on a décidé de travailler qu'avec les séries temporelles, pour les analyser il nous suffit seulement d'une ensemble des graphiques classiques et simples (histogramme, nuage des points, etc). Vue qu'on a fixée dans un chapitre 2.3 qu'on va travailler uniquement avec les séries temporelles, et pour cet raison, l'utilisation des graphes simples comme l'histogramme, nuage des points, etc.

Et donc la deuxième point pour l'utilisation de DC.js est qu'il offre assez des graphes (la liste peut être consultée ici[16]) pour satisfaire nos besoins d'exploration des données via les graphes. Par contre il y a des inconvénients par rapport à DC.js aussi .

?Par contre DC.js cache cet complexité et permet la mise en place plus simple et rapide des graphiques prédéfini et dynamiques. ?

5.4 SVG

DC.js est basée sur D3.js et comme on a dit avant D3.js utilise des SVG pour générer les objets visuelles. La puissance de SVG vient de fait que c'est une image vectorielle, et chaque son composant et référençable via DOM. C'est qui permet de mettre à jour facilement certains éléments d'image, mais cela a un inconvénient. Pour être repérable via DOM, chaque composant d'image doit porter des métadonnées. Et si l'image à rendre et assez complexe comportant milliers des éléments vectorielles, la génération sera ralenti. Donc la problème peut venir lorsque nombre des formes à dessiner est importante.

Pour illustrer ceci, on peut comparer[17] SVG avec un autre technologie souvent utilisé pour les images : les canvas. Pour le DOM, canvas représente un trou avec les pixels. On ne peut pas référencer les formes dans le canvas, pour mettre à jour l'image il faut le recharger. Par contre comme il s'agit juste d'une ensemble des pixels, sachant aussi qu'ils sont générée par le processeur graphique, le temps de réponse est presque invariable en fonction de nombre des éléments à produire.

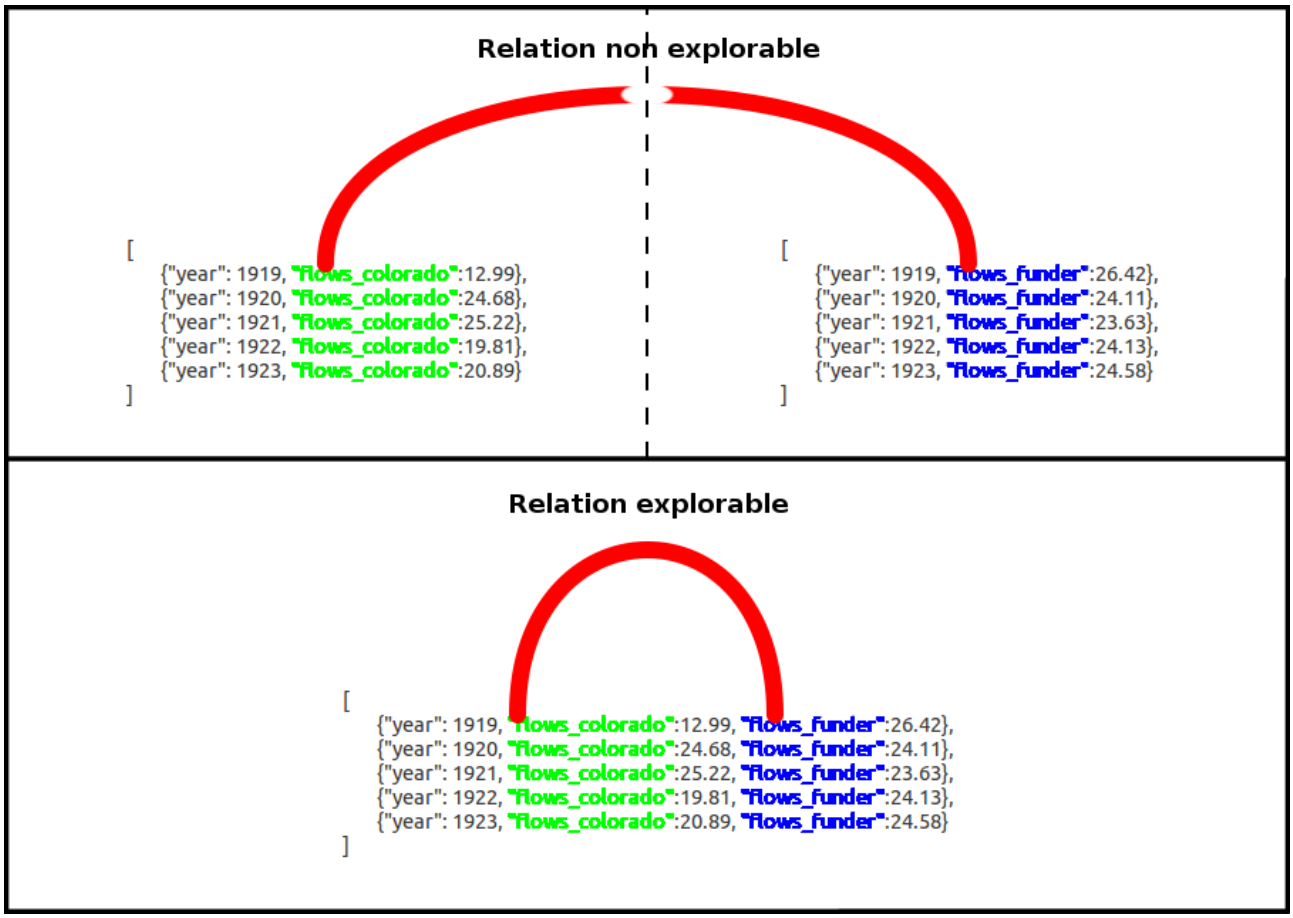
5.5 Format des données

La structure des données avec lesquelles travaille Crossfilter.js, et donc DC.js, est une table des JSONs. JSON, comme XML, est un format des données textuelles. Il comporte des paires clef-valeur, listes des valeurs. Dans un contexte des séries temporelles, chaque mesure est représente par un JSON avec un clef timestamp, et les valeurs mesurées.

---

15 http://www.nytimes.com/newsgraphics/2013/09/07/director-star-chart/
16 http://dc-js.github.io/dc.js/examples/
17 http://smus.com/canvas-vs-svg-performance/

Une précision importante à noter à propos de crossfilter est que, en fait, il



est possible d'explorer les relations entre plusieurs valeurs mesurés mais seulement au sein de la même mesure (cf la première exemple de chapitre 5.1). En langage JavaScript cela vaut dire que seulement la comparaison entre les valeurs au sein d'un seul JSON (voir figure chepa) est possible.

Concrètement cela vaut dire que Crossfilter.js peut établir des filtres interactives seulement au sein d'une seule série temporelle. Encore une fois, en langage JavaScript cela vaut dire que crossfilter n'accepte qu'une seule table de JSONs. Et donc, un peu comme dans une Figure chepa, il va falloir fusionner plusieurs séries temporelles en une, si on veut pouvoir les comparer. Plus de détails sur les aspects subtiles de cet fusion sont précisée dans un chapitre 6.

6 Deuxième palier

Le premier palier a figée les données avec lesquelles l'application va travailler. Il s'agit des séries temporelles. La deuxième vise à définir comment ces données seront explorée par l'utilisateur.

Et donc, des relations les séries temporelles et leurs caractéristiques seront explorée, tout simplement, via l'utilisation des graphiques. De plus, on se fixe d'utiliser la librairie DC.js pour le rendu visuel, car cet bibliothèque graphique permet d'explorer les données avec la plus grande profondeur.

7 Fusion des séries temporelles

Si on se fixe sur l'utilisation de DC.js, et on veut se profiter de ces capacités pour comparer plusieurs séries temporelles, il va falloir les combiner de sorte pour en avoir qu'un seul (cf chapitre 5.5). Une façon de le faire est de réaliser une fusion horizontale, un peu comme dans une figure chepa.

Cet fusion n'est une opération compliquée à mettre en place, mais il y a quelques problèmes à résoudre. En fonction des conditions dans lesquels les objets connectes réalisent les mesures, la période de la prise de mesure peut varier. De plus, il se peut que les timestamps d'une séries temporelles soient complètement irrégulières. Et donc si on veut comparer les deux suites des mesures provenant de différents sources, on risque de réaliser des fusions fausses si on ne respect pas la correspondance de timestamp des deux mesures.

Donc il faut traiter toutes les séries temporelles avant de les fusionner, afin qu'ils contiennes que des échantillons avec les instants de mesure homogènes. C'est-à-dire avant de fussionner toutes les séries temporelles en une, il faut s'assurer tout d'abord qu'ils ont toutes les mêmes instant de prise de mesure. Et cela est rarement le cas si les séries proviennent de différents sources. Du coup, il va falloir pretraiter les données, pour qu'ils soient homogènes.

Par exemple, entre une ensemble des séries temporelles, on peut un laisser seulement les mesures qui ont était faite à les instants inclut dans toutes les séries temporelles. Ou on peut passer par l'interpolation des points (mesures) manquant d'une série mais présentes dans une autre. Ou on peut remplacer chaque trou (mesure manquante) par des mesures zero. Ou choisir de ne pas accepter à traiter les suite des mesurer qui ne sont pas compatible.

Du coup, cet question fais partie de problématiques auxquelles le stagiaire doit répondre et trouver la solution la plus approchée de l'objectif initial.

8 Nouveau objectif concret

Un nouveau objectif est une itération d'un objectif initial qui apport plus de concret sur c'est qui est à accomplir. Les termes « données » et « exploration » ont était eclairci. Aussi, l'approfondissement de la problématique a permit d'ajouter certains nouveaux éléments.

Le but est de développer une application qui se compose des deux composants indépendants :

- un module de traitement des séries temporelles qui ; une boit à outil qui prends en entrée une ensemble des séries temporelles ayant un format fixe, et qui offre différentes possibilités de traiter les données
- une application graphique qui s'en sert d'un module précédent pour traiter les données, offre à l'utilisateur l'interface qui permet de choisir plusieurs systèmes de stockage (toujours au moins deux MongoDB et MySQL), offre à l’utilisateur des graphiques DC.js pour explorer les données et les options de traitement à faire par le module précedent

- aucune autre contrainte technique n'est pas précisée

Cet idée de séparer une application unique en deux sous-applications indépendantes est motivé par plusieurs raisons. Tout d'abord, avoir une application modulaire facilitera la travail pour ceux qui vont reprendre le travail. Et en général, c'est toujours bien de séparer le code dans les brickes logiques afin de mieux l'organiser.

Une autre raison est encore une fois la compatibilité avec MayaNet et la question de utilisabilité. Il est plus simple de rajouter un module dans un projet existent que « casser » une autre application pour y tirer du code. Aussi, cela ne rend pas au sujet de stage, mais dans un contexte industrielle, certains client pourront imposer l'utilisation d'une bibliothèque de visualisation spécifique et pas seulement DC.js. Du coup, un module de traitement des séries temporelles est une sorte de noyau, autour laquelle on peut construire une application exigée.

Par contre, travail qu'avec seul format des séries temporelles, et cela revient à une application-wrapper de transformer toutes les formats acceptée vers un format compatible avec le module de traitement.

Et aussi donc, cet application-wrapper doit offrir à l'utilisateur de choisir plusieurs types des stockages possibles, graphes DC.js à utiliser, options nécessaires etc. Un autre intérêt de cet application wrapper est de donner des exemples d'utilisation de module, et surtout la génération de fichier config acceptable par le module.

Ayant un cahier des charges bien claire, on peut commencer à concevoir une application réel.

Partie 3 : Travail réalisé

Dans se chapitre on explique toutes les fonctionnalités réalisée afin de répondre aux exigences de cahier de charges.

1 Module de traitement

Un module de traitement des séries temporelles, qu'on appelle tsproc (TimeSeries Processor), est un module Node.js qui prend en entrée une ensemble des séries temporelles et offre plusieurs façons de les traiter.

De ça conception le module était conçu pour prétraiter les données avant de les servir à DC.js. C'est la raison pourquoi il contient certains fonctionnalités spécifiques comme l'interpolation ou fusion horizontal des séries, etc. Mais il est fini par être une sorte de boite à outil pour les séries temporelles. L'utilisation par décrier ne se limite pas seulement sur l'utilisation des graphes.

1.1 Choix technologiques

Le module est écrit en JavaScript, et comme c'est un module de traitement des données. Du coup il s'agit d'application backend, d'où viens la raison d'utilisation de Node.js.

La raison de choix de JavaScript pour développer un tel module est argumentée, encore un fois, par le choix technologique MayaNet. La partie IHM de MayaNet tourne sous un serveur Node.js, d'où viens la raison d'utiliser JavaScript. En plus, Node.js est parfait pour un petit module comme celui qu'on vise à développer.

1.2 Première approche

Delta

Comment on a vu dans un chapitre 7 de première partie, pour être capable d'explorer plusieurs séries temporelles par DC.js ils doivent être fusionnée vers une. La première idée pour résoudre cet problème est d'utiliser un delta de temps. Il s'agit de découper le temps entre la mesure la plus ancienne et plus récente des toutes les séries temporelles en morceaux de taille delta. Ensuite, toutes les mesures qui se trouvent à l'intérieur d'une delta donnée vont être composée par une certain opération mathématique (calcul de somme ou moyenne), à une seule mesure.

Cet approche peut être utile pour certains types de mesures, par exemple les relèves de la pluie annuelles (et donc si la delta est égale 5 ans ça de la sens), mais pour les séries temporelles où les mesures ne sont pas cumulatives (par exemple les niveau de tension hebdomadaires) cela n'a pas beaucoup d’intérêt.

Exemple réel

fsdfqsdf fdqsdf