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

Chapitre 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 innovation 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'environnementalement.

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 standard 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 homogene 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 toutes les objets connectes, malgré le fournisseur, puissent se parler entre eux. Une des avancement dans cet domaine 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 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 a 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 but de l'exploration des données.

La dernière problématique, ou si on va se mettre dans un contexte d'un outil automatise 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.

Chapitre 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 terminologique.

La problèmatique posée est vaste parce que les données de diferentes sources peuvent avoir des instant de mésure différent, 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 satisfaisante.

2 Étude initial

Le première milieu de recherche était une étude de façon de sauvegarde des observations par les objets connectées (et les observations en général). Il s'agit des séries temporelles.

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) :

```
"year","flows_colorado"
"1911",18.11
"1912",21.07
"1913",15.77
"1914",24.17
"1915",14.71
```

```
[
    {"year": 1911, "flows_colorado": 18.11},
    {"year": 1912, "flows_colorado": 21.07},
    {"year": 1913, "flows_colorado": 15.77},
    {"year": 1914, "flows_colorado": 24.17},
    {"year": 1915, "flows_colorado": 14.71}
]
```

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

```
"year","flows_colorado"
"1911",18.11
"1912",21.07
"1913",15.77
"1914",24.17
"1915",14.71
```

```
"y","f"
"1911-01-01T00:00:00.000Z",18.11
"1912-01-01T00:00:00.000Z",21.07
"1913-01-01T00:00:00.000Z",15.77
"1914-01-01T00:00:00.000Z",24.17
"1915-01-01T00:00:00.000Z",14.71
```

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.

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 caracteristiques 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 timestamp 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

3.1 Module unique

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

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

MongoDB

Le choix de MongoDB comme un type des stockage est argumenté par le fait que c'est une base dés données orientée document, et par rapport les bases de données relationnelles, MongoDB permet d'avoir un format flexible des données. En quelque sorte, MongoDB était considérée comme une solution pour la problématique de gestion de plusieurs formats, 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 utilisée par MayaNet.

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 l'internet).

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

Morphia

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

---

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

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

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 connecte que MayaNet pourra servir, il faudra crées des objets métiers (des POJO). Donc, comme les POJO sont crées dans toutes les cas, pour toutes « things », un outil automatisée d'analyse des donnée poura utiliser Morphia comme un interface entre pour importe quel format des objets metiers, et la base des données.

Du coup, l'utilisation de Morphia est vue comme une réponse à une problèmatique de multitude des formats des données. Car par la suite, chaque format sera défini dans les POJO, et Morphia offre de façon universelle (mêmes 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 grace à l'utilisation de Morphia et MongoDB. Le but d'un étape de développement qui devez suivre est d'englober cet l'application existante 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 choses.

Tout d'abord, l'utilisation de Java. L'objectif initial l’imposée pas la compatibilité avec MayaNet, c'était une décision de stagiaire de partir vers une solution qui sera a 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 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 seront pas interchangeables entre les deux[9]. Donc cet approche à resoudre la problématique de multitude des formats et stockages n'est pas valable.

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

Mais la conclusion très importante issue de cet première approche, est une réflexion de 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

Tout d'abord, 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. En fait, 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. Les techniques avancées sont, par exemple, étude des trends, utilisation des estimateurs, techniques d'extrapolation, classification, etc. L'avantage de cet approche est que il est très performante pour trouver du valeur cachée dans les données. mais 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 du coup, la mise en place des ces méthodes risque d'être assez complexe. Par contre, rien n'empêche d'appliquer au moins les caractéristiques simples, pour tirer plus d'information à partir des données.

La façon la plus simple et évidente, pour interpréter et explorer les données est l'utilisation des graphique 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 visualisatin de graphiques, mais pas simple et avancée, et l'utilisation des carac 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 multidimensionnelle 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 github : http://square.github.io/crossfilter/ (consultée le 13/07/2016) avec un exemple 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), et Crossfilter.js sert pour le filtrage dans toutes les sens.

L'exemple est construit à partir d'un extrait de dataset des vols réalisés entre 1 janvier et 31 avril 2001. Les données d'origine comporte 29 attributs, dans l'exemple seulement 4 ont été étudie : heure de départ, retard à l'arrivée (min), distance parcouru (milles), date de départ.

Sur l'exemple de site, chaque attribut est représenté par une histogramme, et sur chaque il est possible de choisir une plage de valeurs, c'est-à-dire appliquer un filtre. Et grâce à Crossfilter.js, le filtre sera appliquée à toutes les autres histogrammes. Il est possible d'appliquer plusieurs filtres à la fois.

Tout cela nous permet, par exemple de voir les relations suivantes (cf Figure 1):

- quels sont les journées de la semaine avec les vols les plus longues (etat_longdist)

- quels sont les distances de vols à choisir (imaginons que c'est ce qu'on cherche), pour avoir le moins de retard (etat_mindelay)

- quels heure de départ est le plus génère le plus de retard ? (etat_horraire_maxdelay)

5.2 D3.js

D3.js (Data-Driven Documents) est une bibliothèque graphique en JavaScript qui permet de la visualisation des données numérique en forme des graphiques dynamique. Elle conforme W3C et utilise les technologies SVG et CSS. On invite à consulter une page principale de D3.js https://d3js.org/ (consultée le 22/07/2016) pour apprendre les cas d'utilisation possibles et se rendre compte des points forts de cet librairie : dynamisme (les objets graphiques peuvent réaliser des transitions et morpher) et le fait qu'elle est « data-driven ».

Pour expliquer pourquoi D3.js est « data-driven » il faut tout d'abord se rappeler de principe de HTML : toutes les éléments d'une page web (les tags)

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.

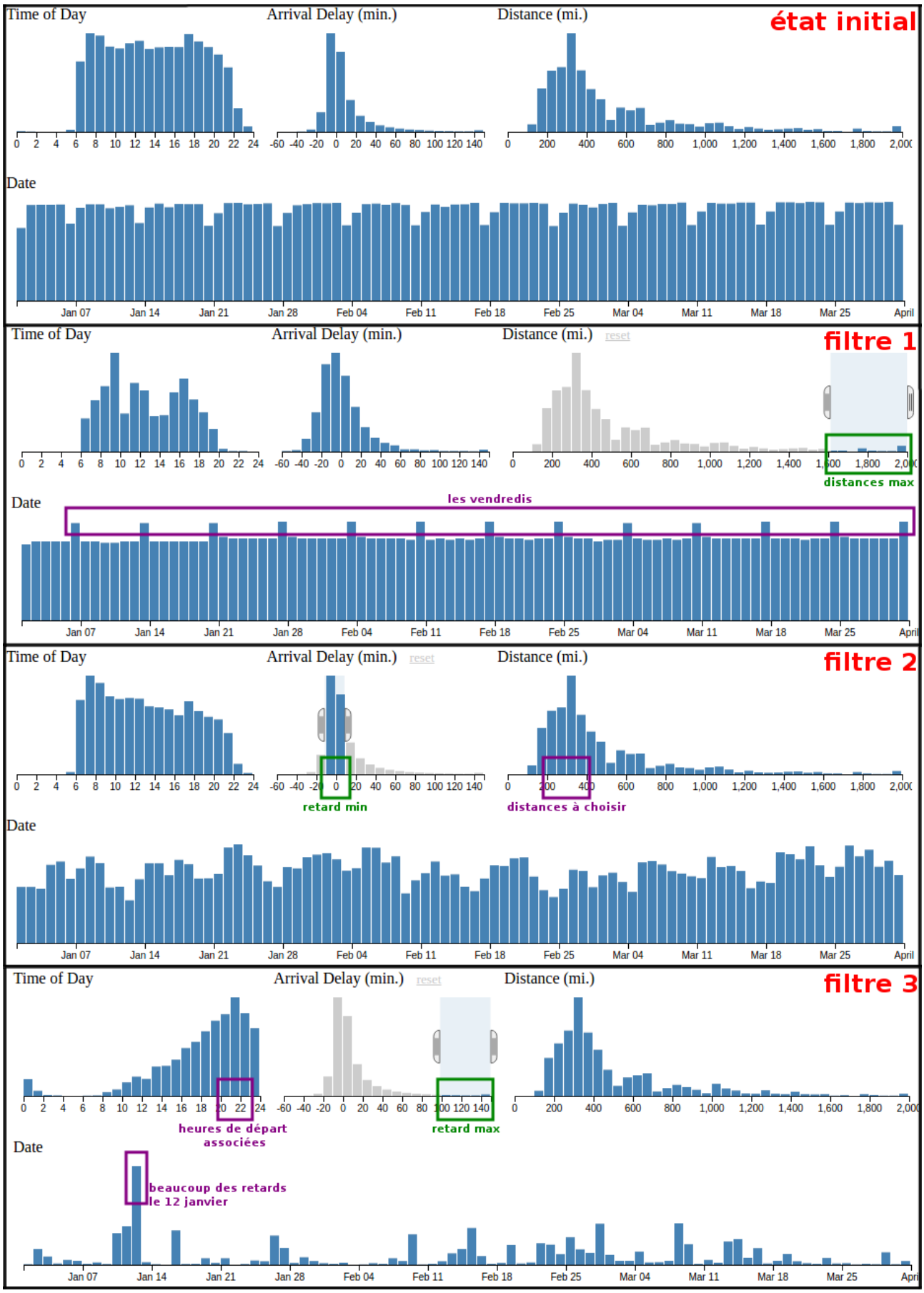
5.3 DC.js

DC.js (Dimensional Charting) et basé sur deux bibliothèques JavaScript : crossfilter.js pour interconnexion et filtrage multidimensionnelle entre les données, et D3.js pour le rendu des graphique visuelles. Ces deux librairies vont naturellement bien ensemble, et peuvent même être utilisées en tant que tel sans aide DC.js. Ce qui permettra de générer des graphes plus flexibles, mais DC.js correspond bien à le cas dans quel il sera utilisée.

Tout d'abord parce que la courbe d'apprentissage de D3.js est tres exponentielle, c'est-à-dire que il faut passer beaucoup de temps tout au début pour apprendre comment l'utiliser pour rendre des graphiques simples. En plus, cela va prendre encore plus de temps si on va relier D3.js et crossiflter.js. Par contre DC.js cache cet complexité et permet la mise en place plus simple et rapide des graphiques dynamiques.

Bien sûr le temps fourni à cet apprentissage va permettre à développer des graphs splendide et complexes, comme par exemple un graphe[10] réalisée en D3.js pure qui visualise les graphes de réalisateurs et leurs acteurs préférées. 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[11]) pour satisfaire nos besoins d'éxploration des données via les graphes.

[10] http://www.nytimes.com/newsgraphics/2013/09/07/director-star-chart/
[11] http://dc-js.github.io/dc.js/examples/



6 Deuxième palier