NF26 - Projet

Yann Ladeve

16 juin 2020

1 Contexte

L'objectif de ce projet est de mettre en application les connaissances vu en cours (Base orientée colonnes, Distribution de calculs) en répondant à des questions données par l'ASA lors d'une compétition sur les données de transport entre des villes aux États-Unis (http://stat-computing.org/dataexpo/2009/).

Les deux questions choisies pour ce projet sont :
— Do older planes suffer more delays ?
— How does the number of people flying between different location change over time ?

2 Fichiers utilisés

— 2005.csv
— plane-data.csv
— plane-seats.csv (ici)

3 Modélisation

Dans l'environnement de travail plusieurs fichiers .csv sont disponibles apportant des informations sur les vols, les avions, les compagnies aériennes ou encore les aéroports. Pour mieux visualiser le lien entre chaque élément voici une modélisation des données.

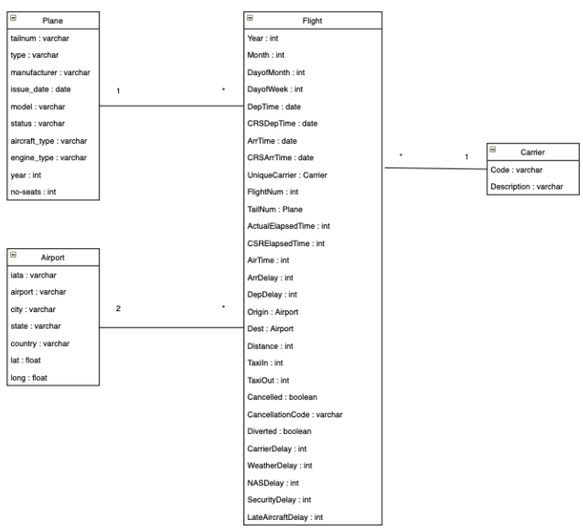


FIGURE 1 – Modélisation jeu de données

À un vol est associé un avion, une compagnie aérienne et deux aéroports (départ, arrivée).

4 Do older planes suffer more delays ?

Cette question demande de récupérer des informations sur l'ancienneté de l'avion et les retards pour chaque vol. L'âge de l'avion est donné dans le fichier *plane-data.csv* avec la colonne *year*. Tous les vols possèdent des informations concernant les retards selon différents critères. Tout d'abord il faut différencier le retard au départ et le retard à l'arrivée. Ces deux valeurs peuvent être différentes si l'avion prend de l'avance pendant le vol. Ces valeurs sont directement données par *ArrDelay et DepDelay*. Dans notre cas, le retard à l'arrivé a été considéré. Des informations avec la cause du retard (météo, problème de sécurité ...) sont aussi indiquée par les variables : *CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay*. La somme des variables est égale à *ArrDelay*. La table implémentée sur cassandra pour cette question est la suivante :

$$\text{flight}(\underbrace{creation\_year}_{\text{clé de partition}}, \underbrace{tailnum, year, month, day, crs}_{\text{clé de tri}}, \text{arr\_delay, dep\_delay, manufacturer})$$

Avec cassandra, il faut toujours pouvoir indiquer la clé de partition lorsque l'on fait une recherche. Ici on veut pouvoir chercher des vols selon l'année de création de l'avion. C'est pour ça que c'est cette variable qui est choisie comme clé de partition. Le fichier plane-data.csv ne donnant pas une date plus précise pour l'ancienneté, on pourra seulement faire des recherches selon une année. Ensuite dans chaque partition, il faut pouvoir lister tous les vols. *tailnum* qui correspond à l'identité de chaque avion est unique. Donc à un instant t (%Y/%m/%d 00 : 00), l'avion avec le tailnum x ne peut être que sur un vol. La précision du t va jusqu'au minute car un même avion peut faire plusieurs vols le même jour et si l'heure de départ n'est pas indiquée alors des valeurs seraient écrasées lors de l'insertion en base.

Pour récupérer ces données depuis ces fichiers csv, la fonction *read_csv_flight_data()* fourni un générateur. Pour chaque ligne du fichier 2005.csv, la fonction récupère les éléments nécessaires pour remplir la table flight. Certains vols sont annulés ou n'ont pas d'heure d'arrivé. Ces cas là ne sont pas considérés. Après utilisation de la fonction *check_na_flight()*, seulement 0.2% des vols sont concernés, ce qui est très faible. La récupération des éléments *creation_year* et *manufacturer* doit se faire à partir du fichier *plane-data.csv*.

Au départ un deuxième générateur était imbriqué dans la fonction principale mais le temps d'éxecution était très long. De ce fait j'ai décidé de mettre en place un dictionnaire avec comme clé tailnum et en valeur un $tuple(manufacturer, year)$. Ce dictionnaire est de petite taille, car il possède moins de 5030 couples. Pour confirmer la performance de cette solution, l'utilisation de la fonction *compute_time*() permet de savoir le temps d'itération sur le générateur final. En passant par un dictionnaire l'ensemble du générateur, pour les 7 millions de lignes du fichier csv, était itérer en 197.2s. À titre de comparaison la même fonction en utilisant des yields imbriqués n'avait pas fini d'itérer après 10 min et ceux pour une limite de 100 mille lignes.

Le fichier plane-data est parfois incomplet (seulement le tailnum), dans ce cas les valeurs par défaut Unknown et 0 sont appliquées pour le constructeur et la date de création. Puis toutes ces valeurs sont mises dans un namedtuple *Flight()* qui permettra une facilité de manipulation par la suite. Le générateur retourné est ensuite utilisé dans la fonction *_insert_datastream()* pour insérer chaque *Flight()* dans cassandra.

La troisième étape consiste à analyser ces données insérées en base pour pouvoir répondre à la question. Dans le fichier *analyse_cassandra.py* une classe *FlightTrip* est crée avec comme initialisation la création de la session cassandra. Cela a pour objectif de ne pas ouvrir un nouveau thread à chaque appel de la base.

Tout d'abord la fonction *get_flight_one_year()* est implémentée. Cette fonction retourne un générateur de Flight() pour une année de création d'avion donnée. Le fichier csv pris pour les vols étant celui de 2005, une vérification est faite sur l'année rentrée en paramètres. L'objectif étant de pouvoir évaluer le retard moyen sur plusieurs années, la fonction *get_flight()* appelle n fois la fonction précédente pour chaque année.

Pour pouvoir répondre à la question, il faut donc calculer le retard moyen accumulé pour chaque année de création. Ce calcul est effectué avec la méthode *get_ cumulated_ delay_ by_ year()*. L'objectif est de créer une matrice $(yearSup - yearInf) * 3$ où la première colonne indique l'année, la deuxième la somme des retards accumulés pour cette année et la dernière le nombre de vols concernés par ces retards. Lors de l'affichage, les éléments qui ont les valeurs par défaut définies dans *read_ csv_ flight_ data()* (Unknown et 0) ne sont pas affichés.

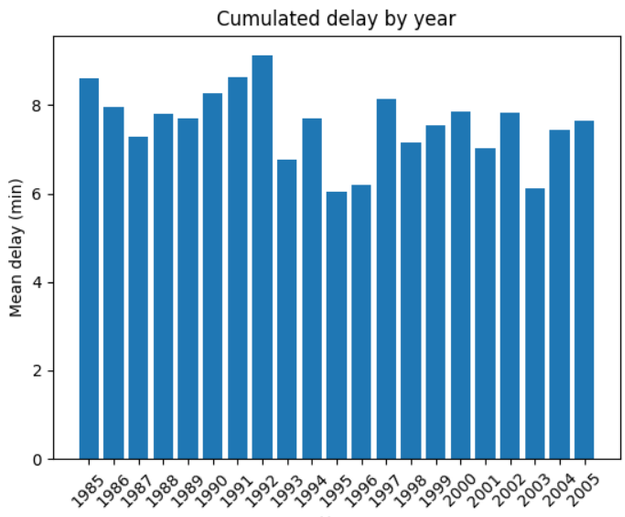
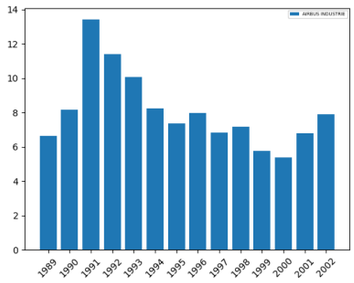


FIGURE 2 – Retard moyen par ancienneté d'avion d'un vol

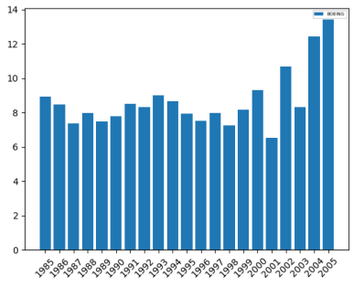
Effectivement les retards moyens par vol les plus élévés sont les plus haut pour les années les plus anciennes (entre 1985 et 1991). Cependant ce retard ne tend pas à diminuer de manière constante par la suite.

À la suite de ces résultats, il me semble intéressant de voir si cette tendance est identique lorsque l'on regarde les retards moyens par vol par constructeur (Boeing, AirBus ...). Pour obtenir ce résultat, la méthode *get_ cumulated_ delay_ by_ year_ manufacturer()* est implémentée. Elle possède le même comportement que la fonction précédente à ça près que la matrice retourne une matrice à 4 colonnes, la quatrième étant le constructeur.

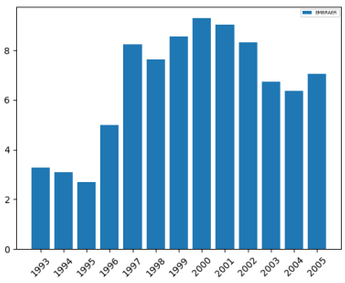
Pour l'affichage, il faut faire une distinction entre les avions d'année x et de constructeur y qui n'ont pas eu de retard (un delay de 0) et les avions d'année x et de constructeur y' qui n'ont pas de retard car aucune données les concernant n'a été récupérée. Dans le premier cas l'année x sera affichée avec un bar null alors que pour le deuxième cas l'année x n'apparaîtra pas sur l'axe. L'axe x étant propre à chaque constructeur un barplot par constructeur est donc crée.



(a) Airbus



(b) Boeing



(c) Embraer

FIGURE 3 – Retard moyen par vol par constructeur par ancienneté d'avion

On peut voir que pour Airbus ce retard a diminué au fur et à mesure des années alors que Boeing et Embraer connaissent une augmentation de ce retard à partir de la fin des années 90.

5 How does the number of people flying between different location change over time ?

Cette question va être répondue avec deux niveaux de précision. Le premier sera le nombre de passagers par moi, tandis que le deuxième sera le nombre de passagers par jour de la semaine et par heure. De plus ce qui va servir de référence pour le nombre de passagers, et le nombre de sièges dans l'avion. Cela implique que l'on considère que 100% des vols sont complets. Bien évidemment ce n'est pas toujours le cas dans la vie réel, mais c'est le moyen le plus précis que j'ai pu trouver pour associer une valeur sur le nombre des passagers. L'autre solution pensée, aurait été de juste se baser sur la fréquence des vols pour définir le nombre de passagers indépendemment de la taille de l'avion et de son taux de remplissage, les résultats auraient été encore plus biaisés.

Initialement, les informations sur le nombre de sièges par type d'avion n'étaient pas fournies. Un fichier externe a été donc importé (voir 2).

Pour récupérer un namedtuple contenant le vol, le modèle d'avion et le nombre de sièges, il faudrait itérer sur 3 fichiers. Si l'on passe par des yields cela reviendrait à $7140597 * 5030 * 87067$ itérations, ce chiffre est considérable et non envisageable à cette échelle. Pour palier à ce problème l'utilisation d'un dictionnaire a été faite. Ce dictionnaire créé par la fonction *dico_plane_seat()* a pour clé le tailnum et pour valeur le nombre de sièges. Grâce à cette méthode, lors de l'itération du fichier 2005.csv, il n'y a pas besoin d'itérer sur les deux autres fichiers mais simplement de récupérer la valeur associée à la clé tailnum de ce vol. Pour certains avions, il n'y avait pas d'information concernant le nombre de sièges. La moyenne de sièges, calculée parmi tous les modèles ayant l'information, leur a été donc affectée. Le namedtuple retourné ($Flight\_w\_seat()$) est légèrement différent du $Flight()$ vu pour la première question. Les informations sur le retard des vols n'ont pas été gardées n'étant pas utile pour répondre à cette question.

Pour le premier niveau de précision, la méthode implémentée est *get_ nb_ passengers_ per_ month()*. Dans le cas de la distribution de calcul, un map reduce est employé. Le map ne garde que le mois et le nombre de siège. Puis le tout est sommé avec le reduce, la clé étant le mois. La méthode *display_ per_ month()* permet d'afficher les résultats.

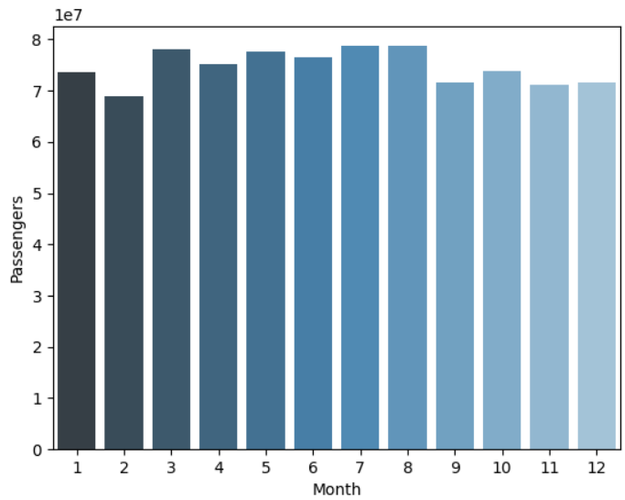


FIGURE 4 – Nombre de passagers par mois

On peut voir logiquement que les mois de Juillet et Août sont les mois avec le plus de passagers. Cependant le nombre de voyageurs ne varie pas de manière importante. Il est étonnant aussi de remarquer que le mois de Décembre ne se détache pas avec les leaders, on pourrait pourtant penser que pour les fêtes de Noël, plus de voyageurs prendrait l'avion.

Pour le deuxième grain, on veut voir si des tendances se dessinent en fonction du jour de la semaine et de l'heure dans ce jour. Pour ce faire, un autre map reduce est effectué, cette fois-ci la clé est composée du jour de la semaine et de l'heure de départ.

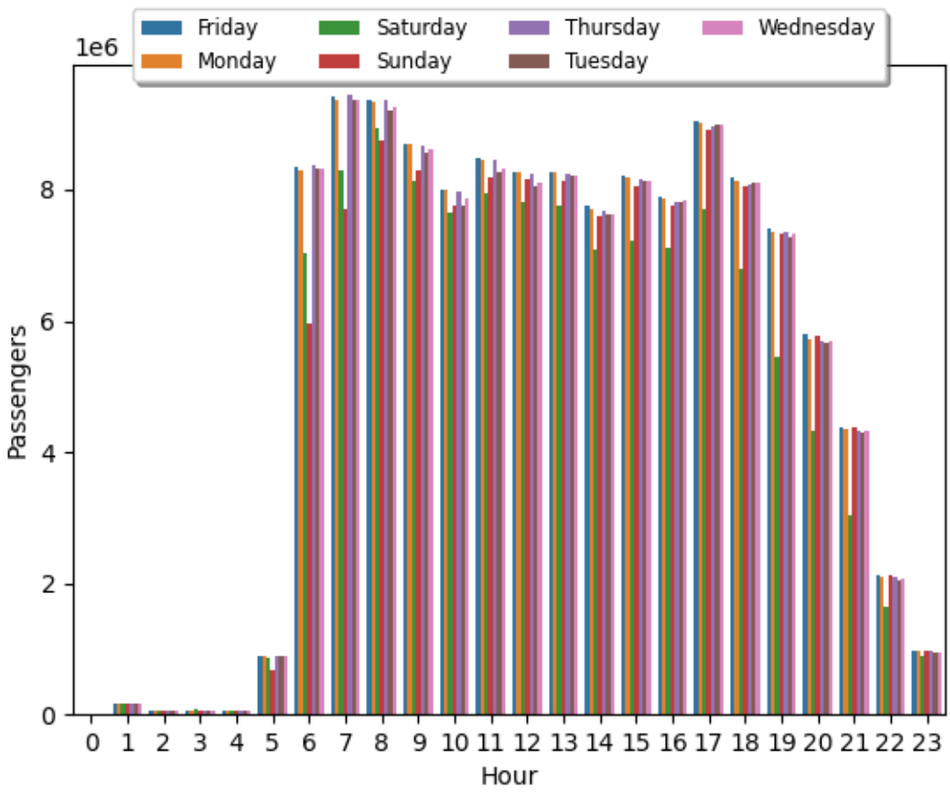


FIGURE 5 – Nombre de passagers par jour et heure

Ici on peut voir que le nombre de passagers est plus important les lundi, mercredi, jeudi et vendredi. Le lundi et vendredi peuvent s'expliquer par les départs et retour en week end. Ce nombre est d'autant plus important à 7, 8 et 17 heures. Le mercredi et le jeudi doivent correspondre plus à des déplacements professionnels. On peut également voir qu'il n'y a pratiquement aucun vol de minuit à 5h.