|  |
| --- |
| 2017-2018 |
| Rapport de projet Big Data |
| M2 MIAGE |

|  |
| --- |
| Arthur Rozo  19/03/2018 |

Table des matières

[Question 1 3](#_Toc509212440)

[Question 2 4](#_Toc509212441)

[Question 3 6](#_Toc509212442)

[Question 4 8](#_Toc509212443)

[Question 5 9](#_Toc509212444)

Introduction

Dans le cadre du cours de Big Data, nous avons eu à répondre à certaines questions sur un fichier de données volumineux. Nous avons pu voir lors de ce cours deux technologies basé sur le map reduce pour répondre à ces questions : Hadoop en java, et Spark en scala. Nous étions libres sur le choix de la technologie. J’ai choisi d’utiliser Spark malgré mon manque de connaissance en Scala car cette solution était à la fois plus simple à coder, mais aussi plus rapide à exécuter.

Dans un premier temps, il nous a été demandé de tester nos codes sur un échantillon des données avant de lancer le traitement sur le jeu de données entier. Le jeu de données étant volumineux, nous avions à notre disposition un cluster Spark ou Hadoop suivant la technologie choisie, afin d’exécuter nos traitements de façon efficace.

Ce rapport présente les codes développés, les temps d’exécution ainsi qu’une visualisation des données. A cause d’un problème de droit sur le cluster, je n’ai pas pu exécuter les instructions *saveAsTextFile* qui m’aurait permis de récupérer les données. Les temps d’exécution présent sont donc ceux obtenu lors d’un traitement local des données.

Lien vers les sources :

<https://github.com/azoro3/spark_project.git>

# Question 1

Donnez le classement décroissant des catégories de crime

val debut1 = System.currentTimeMillis();

val data =sc.textFile("Z:/Doc/Cours/progDev/Perrin/datamin.csv")

val mappedData =data.map(line => line.split(","))

val selectedData =mappedData.map(line=> (line(5),1))

val reducedData =selectedData.reduceByKey(\_+\_)

val sortedData =reducedData.sortBy(\_.\_2)

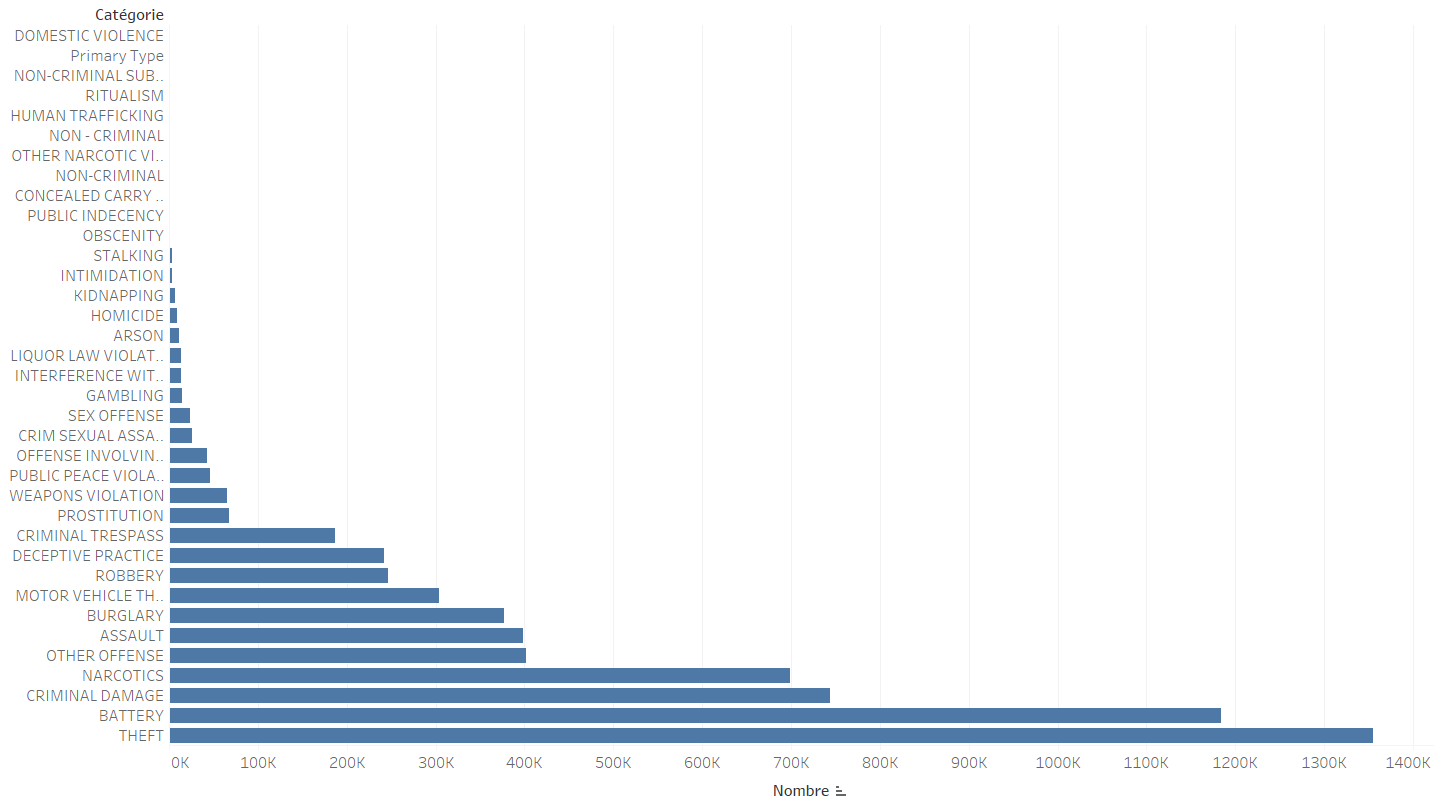
sortedData.saveAsTextFile("Z:/Doc/Cours/progDev/Perrin/out\_question1")

Q1:60270 ms

Pour répondre à cette question, nous allons splitter les données et ne récupérer uniquement que les catégories de crimes (index 5). Nous allons ensuite ajouter le numéro 1 à la catégorie pour obtenir

Le couple .

Nous appliquons ensuite les opérations reduceByKey afin de sommer le nombre de crimes par catégorie de crimes, et l’instruction sortBy afin de trier notre résultat.



# Question 2

Donnez le nombre de crime en fonction de plages horaires

val debut2 = System.currentTimeMillis();

val data =sc.textFile("Z:/Doc/Cours/progDev/Perrin/data.csv")

val mappedData =data.map(line => line.split(","))

val selectedData =mappedData.map(line=> line(2))

val pmData =selectedData.filter(line => line.contains("PM"))

val amData =selectedData.filter(line => line.contains("AM"))

val pmData2 =pmData.map(line => line.split(" "))

val amData2= amData.map(line => line.split(" "))

val pmData3= pmData2.map(line => line(1).split(":"))

val amData3= amData2.map(line => line(1).split(":"))

val pmData4 =pmData3.map(line => line(0).toInt+12)

val amData4 =amData3.map(line => line(0).toInt)

val selectedData3 =amData4.union(pmData4)

val data0\_4 =selectedData3.filter(line => (line < 5)).map(line =>("0\_4",1)).reduceByKey(\_+\_)

val data4\_8=selectedData3.filter(line => (line < 9 & line>4)).map(line=>("4\_8",1)).reduceByKey(\_+\_)

val data8\_12=selectedData3.filter(line => (line < 13 & line>8)).map(line=>("8\_12",1)).reduceByKey(\_+\_)

val data12\_16=selectedData3.filter(line => (line < 17 & line>12)).map(line=>("12\_16",1)).reduceByKey(\_+\_)

val data16\_20=selectedData3.filter(line => (line < 21 & line>16)).map(line=>("16\_20",1)).reduceByKey(\_+\_)

val data20\_24=selectedData3.filter(line => (line < 25 & line>20)).map(line=>("20\_24",1)).reduceByKey(\_+\_)

val data\_all=data0\_4.union(data4\_8).union(data8\_12).union(data12\_16).union(data16\_20).union(data20\_24)

data\_all.saveAsTextFile("Z:/Doc/Cours/progDev/Perrin/out\_question2")

Q2:75135 m

Pour cette question, nous allons récupérer uniquement le primer champ qui contient le jour ou le crime a été commis ainsi que l’heure. Nous allons ensuite récupérer uniquement l’heure du crime, sans les minutes, en faisant attention de bien séparer les heures PM et AM. Nous allons rajouter 12 au heures PM pour les faire correspondre au système français. Une fois cela fait, nous créons un dataset pour chaque plage horaire, sur le même principe que la question précédente. Une fois cela fait, nous allons sommer chaque dataset, pour obtenir le nombre de crimes commis par tranche horaires, et nous concaténons finalement tous les datasets pour obtenir un seul dataset avec la répartition finale.

# Question 3

Donnez les trois zones les plus dangereuses, et les trois zones les moins dangereuse

import org.apache.spark.mllib.clustering.{KMeans, KMeansModel}

import org.apache.spark.mllib.linalg.Vectors

import scala.util.Try

val debut3 = System.currentTimeMillis();

val data =sc.textFile("Z:/Doc/Cours/progDev/Perrin/data.csv")

val cleanData = data.map(line => {

val splitedLine = line.split(",")

val splittedArrayLength = splitedLine.length

(splitedLine(splittedArrayLength-4),splitedLine(splittedArrayLength-3))

})

val prePreData=cleanData.filter(line =>(!line.\_1.isEmpty && !line.\_2.isEmpty

&& Try(line.\_1.toDouble).isSuccess

&& Try(line.\_2.toDouble).isSuccess))

val preData = prePreData.map(s => Vectors.dense(s.\_1.toDouble, s.\_2.toDouble)).cache()

val numClusters = 10

val numIterations = 50

val clusters = KMeans.train(preData, numClusters, numIterations)

val WSSSE = clusters.computeCost(preData)

val centers = clusters.clusterCenters

val assignedData = clusters.predict(preData)

val clusterCountDesc = assignedData.map(t => (t, 1)).reduceByKey(\_ + \_).sortBy(\_.\_2,false)

val clusterCountAsc = assignedData.map(t => (t, 1)).reduceByKey(\_ + \_).sortBy(\_.\_2,true)

val troisP = clusterCountDesc.take(3)

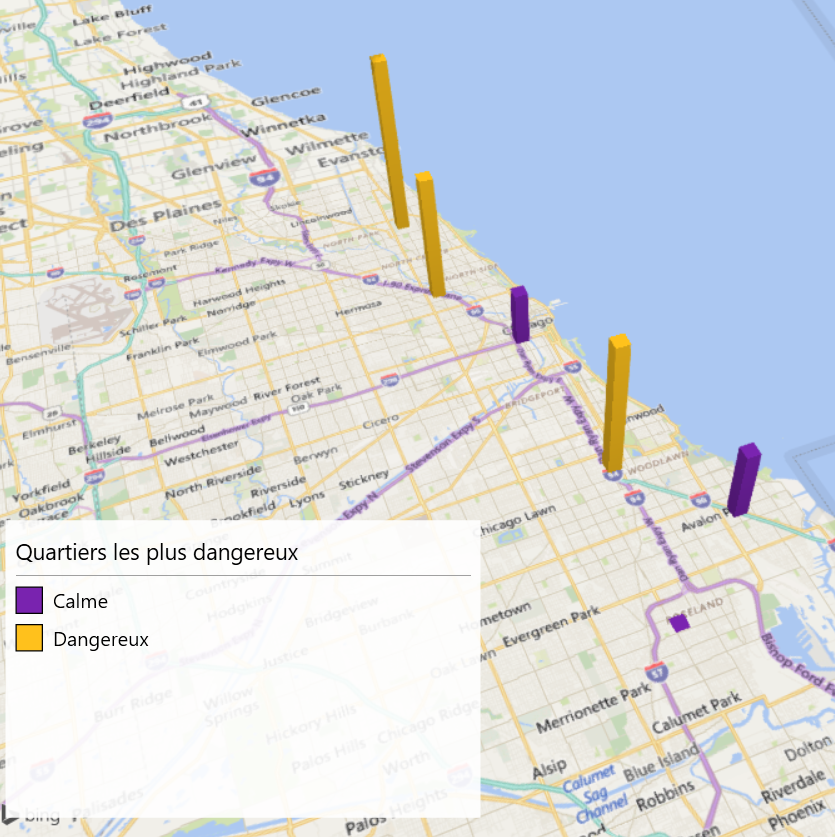
val troisM=clusterCountAsc.take(3)

clusterCountDesc.saveAsTextFile("Z:/Doc/Cours/progDev/Perrin/out\_question3\_M")

clusterCountAsc.saveAsTextFile("Z:/Doc/Cours/progDev/Perrin/out\_question3\_P")

Q3:151854 ms

Pour cette question, nous allons appliquer l’algorithme K-means afin de clustériser nos données. Nous avons choisi 10 cluster suite à un échange avec notre professeur. Pour cela nous utilisons une librairie la partie machine learning de Spark. Une fois la clusterisation faite, nous récupérons les clusters avant de les trier dans l’ordre croissant et descendant afin de récupérer les 3 plus dangereux et les 3 moins dangereux.



# Question 4

Donnez la répartition des crimes commis

val debut4 = System.currentTimeMillis();

val data =sc.textFile("Z:/Doc/Cours/progDev/Perrin/data.csv")

val splittedData = data.map(line => {

val splitedLine = line.split(",")

val splittedArrayLength = splitedLine.length

(splitedLine(splittedArrayLength-15),splitedLine(splittedArrayLength-4),splitedLine(splittedArrayLength-3))

})

val filteredData = splittedData.filter(t => (t.\_1.equals("true")))

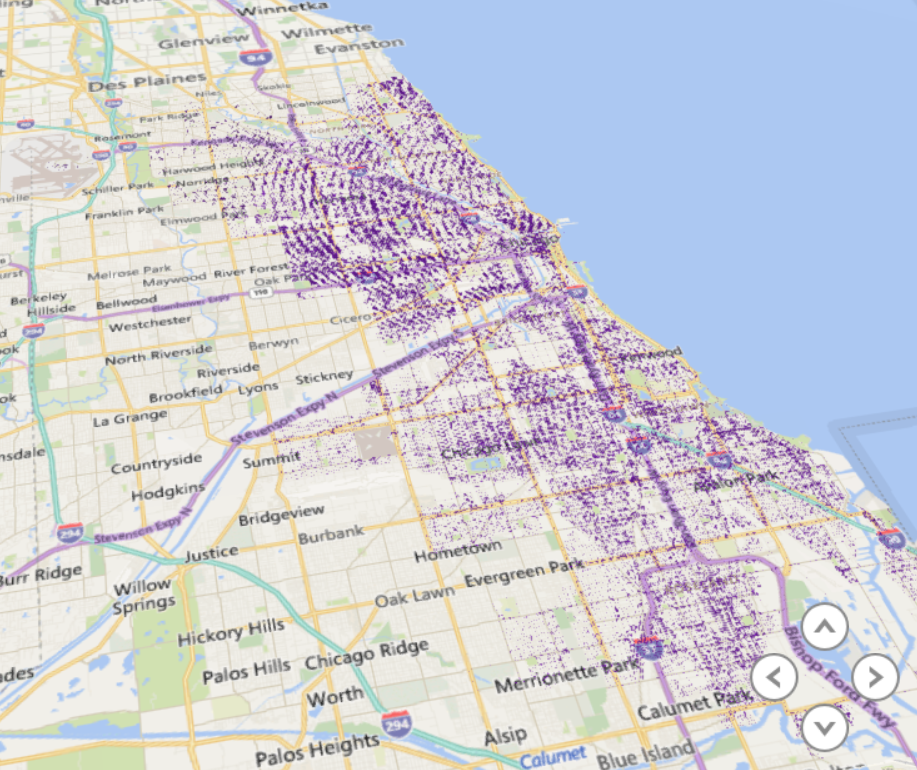
val mappedData = filteredData.map(line => (line,1)).reduceByKey(\_+\_)

mappedData.saveAsTextFile("Z:/Doc/Cours/progDev/Perrin/out\_question4")

Q4:55500 ms

Dans cette question, nous récupérons l’informations sur la clôture de l’enquête. Nous allons ensuite récupérer les crimes élucidés.

A cause de la volumétrie importantes, il ne m’a pas été possible de proposer une représentation satisfaisante des données. Ci-dessous une répartition des crimes élucidés.



# Question 5

Donnez les trois mois les plus dangereux

val debut5 = System.currentTimeMillis();

val data =sc.textFile("Z:/Doc/Cours/progDev/Perrin/data.csv")

val mappedData =data.map(line => line.split(","))

val selectedData =mappedData.map(line=> (line(2)))

val mappedData2=selectedData.map(line => line.split(" "))

val selectedData2=mappedData2.map(line=>(line(0)))

val mappedData3=selectedData2.map(line => line.split("/"))

val selectedData3=mappedData3.map(line=>(line(0),1))

val reducedData =selectedData3.reduceByKey(\_+\_)

val sortedData =reducedData.sortBy(\_.\_2,false)

val res=sortedData.take(3)

sortedData.saveAsTextFile("Z:/Doc/Cours/progDev/Perrin/out\_question5")

Q5:8627 ms

Pour répondre à cette question, nous récupérons comme pour la question 2 la colonne avec la date du crime, sauf que nous récupérons cette fois le mois du crime. Nous allons ensuite calculer le nombre de crimes dans un mois, trier les données par ordre décroissant, et enfin récupérer uniquement les trois premiers mois.

Conclusion

Ce projet nous a permis de mettre en pratique le concept de map reduce, que ce soit en Spark ou en Hadoop. Nous avons aussi eu la chance de pouvoir travailler sur un cluster, et de profiter d’une puissance de calcul que nous ne possédons pas sur nos machines.

Une autre difficulté pour ma part a été d’apprendre un nouveau langage en un temps court. Mais une fois les concepts de la programmation fonctionnelle acquis, Spark a été un réel gain de temps. Certaines opérations qui sont longues et fastidieuses, mais aussi chronophages, sont optimisés en Spark.