NF26

RAPPORT DE PROJET

Groupe TD2 : 17-18

Hugo DURET

Haojie LU

Jeu de données

France de 2005 à 2014

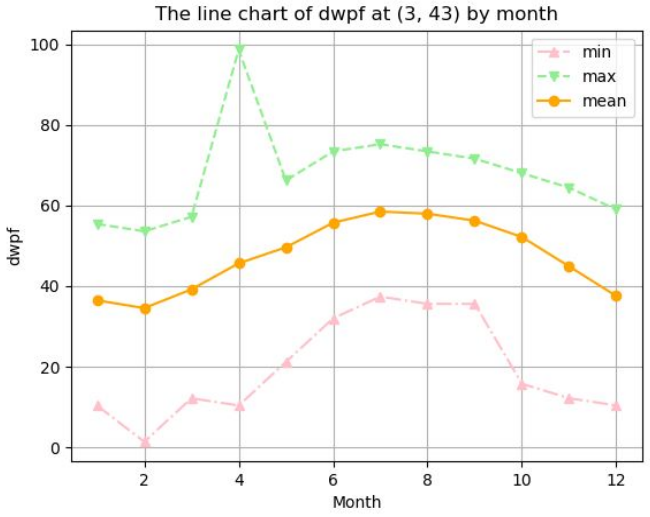
P19

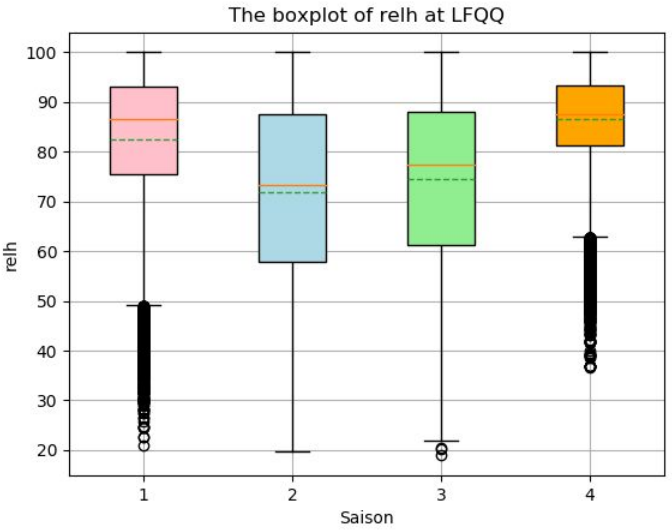
I. Résultats du projet

1) *Pour un point donné de l'espace, je veux pouvoir avoir un historique du passé, avec des courbes adaptés. Je vous pouvoir mettre en évidence la saisonnalité et les écarts à la saisonnalité. (référencée comme* Question 1 *dans la suite)*

Un point donné de l'espace est un nom de station, ou des coordonnées GPS.

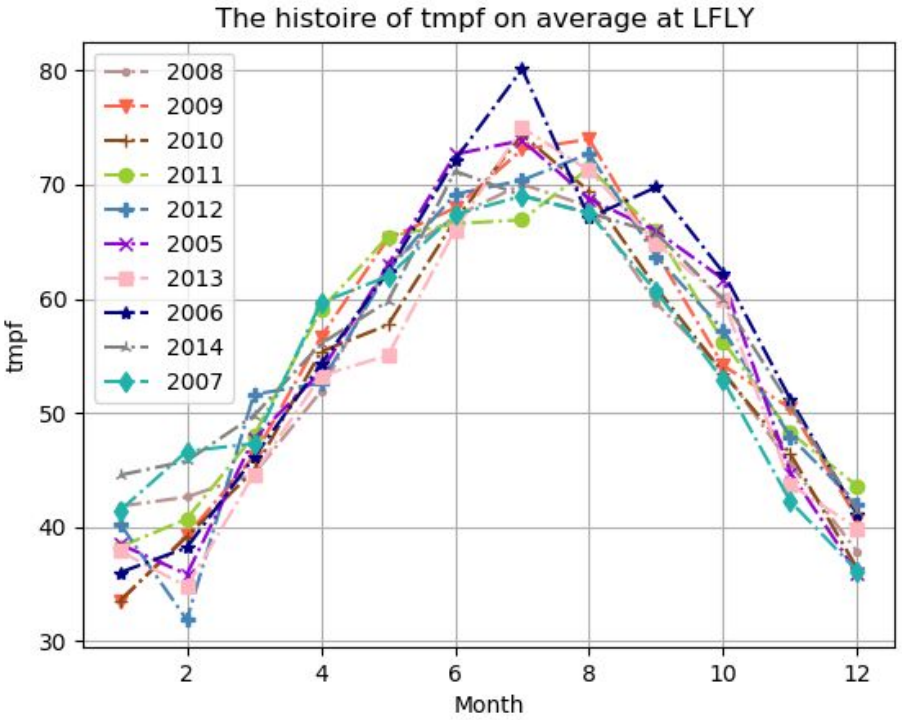
➔ *linechartparmois(point, indicator) ⇒ linechartparmois((3,43), 'dwpf')*





➔ *boxplot(point, indicator) ⇒ boxplot('LFQQ', 'relh')*

➔ *linechart_histoire(point, indicator) ⇒ linechart_histoire('LFLY', 'tmpf')*



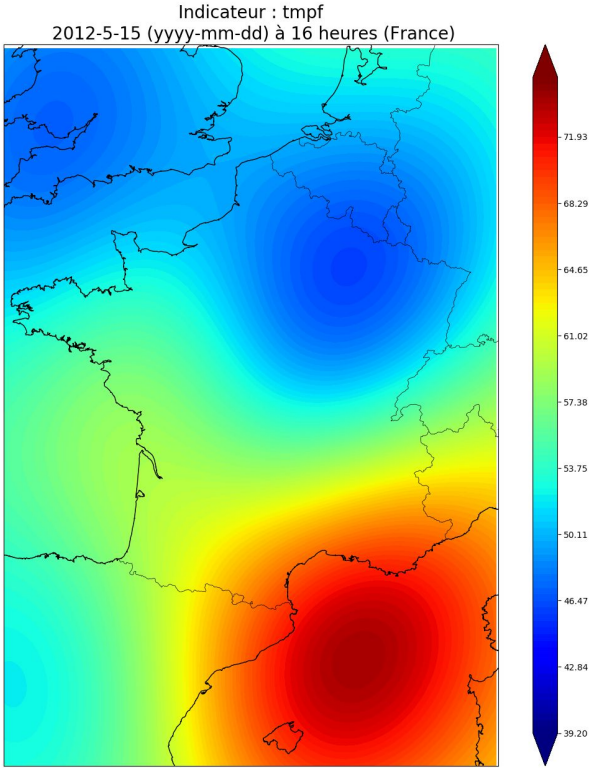
2) *À un instant donné je veux pouvoir obtenir une carte me représentant n'importe quel indicateur. (référencée comme* Question 2 *dans la suite)*

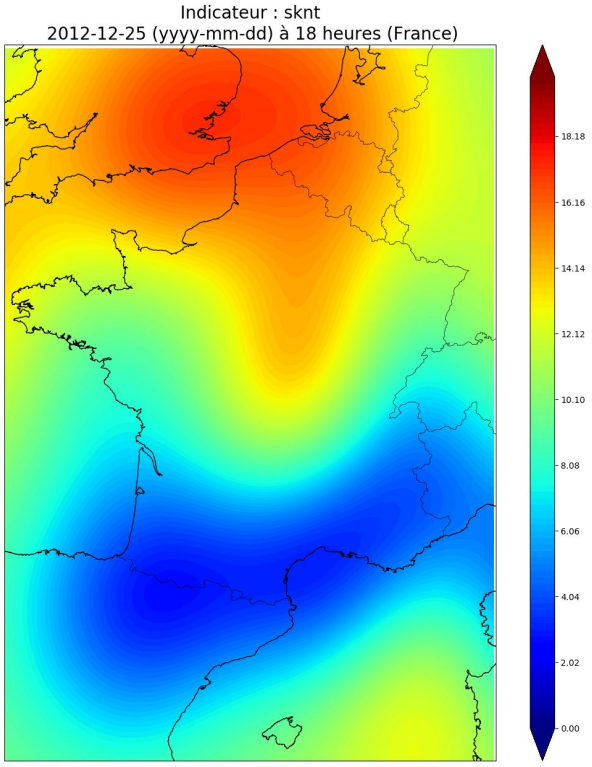
➔ *get_plot_per_hour_and_indicator(year, month, day, hour, indicator)*

⇒ *get_plot_per_hour_and_indicator(2012, 5, 15, 16, 'tmpf')*

➔ *get_plot_per_hour_and_indicator(year, month,day, hour, indicator)*

⇒ *get_plot_per_hour_and_indicator(2012, 12, 25, 18, 'sknt')*





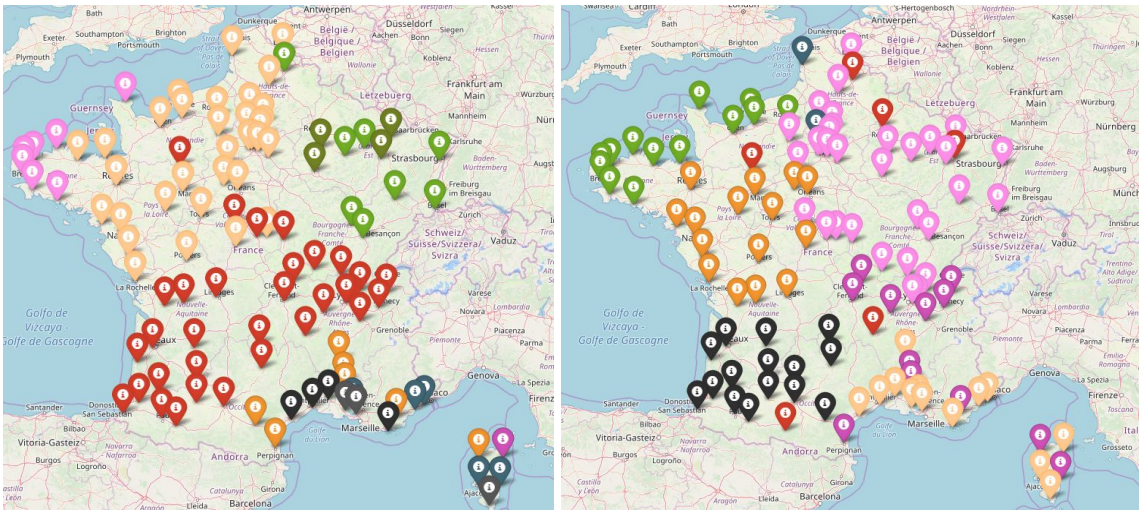
3) *Pour une période de temps donnée, je veux pouvoir clusteriser l'espace, et représenter cette clusterisation. (référencée comme* Question 3 *dans la suite)*

➔ *mapofclusterisation(start_date, end_date, nombre_de_gruope(10 par défaut))*

⇒ *mapofclusterisation('2012-06-09', '2012-08-09')*

➔ *mapofclusterisation(start_date, end_date, nombre_de_gruopes)*

⇒ *mapofclusterisation('2012-03-09', '2012-08-22',8)*



II. Description des données

Les données METAR sont utilisées (*https://fr.wikipedia.org/wiki/METAR*). Ce sont des rapports d'observations de données météorologiques. Elles sont téléchargées depuis *https://mesonet.agron.iastate.edu/request/download.phtml*

La taille du jeu de données total était de 2,7 Go. Pour les besoins du projet, et la rapidité des calculs, nous avons réduit ce jeu selon les questions, pour être de l'ordre de 300 Mo. Le détail des données insérées est fourni plus loin pour chaque question.

III. Choix de stockage

Pour identifier un enregistrement, les informations nécessaires sont :
station, year, month, day, hour, minute

C'est donc ces données que l'on retrouvera dans les clés de partitions et de tris.

La station peut aussi être identifiée par les coordonnées (longitude,latitude), puisqu'il n'existe pas deux stations positionnées exactement au même endroit.

La station *LFVP ST PIERRE-FRANCE*, la seule hors de la France métropolitaine, a été ignorée pour cette étude. Ses données étant très éloignées des autres.

1) ***Question 1 :*** toutes les données de dix stations pour toute la période 2005-2014. (800 Mo)

Clé de partitionnement	Clés de tri
station	year, month, day, hour, minute

Puisque la question demande des informations pour un point donné de l'espace, la station est choisie comme clé de partitionnement. Les informations sur le temps servent alors de clé de tri.

La clé de partitionnement pourrait être les coordonnées (longitude, latitude) de la station. Mais comme chaque station a des coordonnées uniques, il est plus simple d'utiliser le nom de la station directement.

Cependant, on nous demande de donner la réponse en tout point de l'espace (et non pas juste selon le nom de la station). On a donc décidé de laisser à l'utilisateur le choix entre utiliser le code de la station, ou des coordonnées quelconques.

Dans le cas où il utilise le code de la station, on va chercher directement la partition correspondant à cette station. Dans le cas d'un point donné (de coordonnées donc), on va aller parcourir un fichier contenant les stations et leurs coordonnées puis comparer les distances avec le point donné. On retient alors la station la plus proche du point, et on va chercher ses informations.

Un inconvénient de ce stockage est que les partitions grandissent avec l'ajout de données car chacune contient les informations sur toute la période. Mais vu que la question demande de faire un historique du passé, le stockage est quand même bon car cohérent avec la question posée.

2) ***Question 2*** : toutes les données de toutes les stations pour 2012(350 Mo)

Clé de partitionnement	Clés de tri
year, month, day, hour	minute, station

Puisque la question demande des informations pour un instant donné, le partitionnement se fera en fonction du temps.

Le choix de la dimension d'un "instant" s'est fait sur l'heure. Partitionner par minute n'aurait pas beaucoup de sens car les partitions seraient trop petites, et on n'a pas d'enregistrement toutes les minutes. Partitionner par heure aurait donné des partitions plus grandes, mais il aurait été plus compliqué de faire des cartes par heure.

Ce choix permet donc de faire facilement une carte avec les informations actualisées toutes les heures. C'est cohérent puisque les conditions météorologiques changent rarement d'une minute à l'autre, mais peuvent beaucoup changer d'un jour à l'autre. L'heure est un bon compromis.

3) ***Question 3 :*** toutes les données de toutes les stations pour 2012. (350 Mo)

Clé de partitionnement	Clés de tri
timestamp_day	hour, minute, station

Puisque la question demande de clusteriser pour une période donnée, on crée alors *timestamp_day*, une colonne de type timestamp, à partir de année-mois-jour. Ainsi on a une partition par jour, donc des partitions ni trop grandes, ni trop petites, dont la taille n'évolue pas avec le temps. Au début on souhaite pouvoir chercher les données par des requête du type : *SELECT ... WHERE date > '2007-10-12' and date < '2008-10-12* . Dans ce cas, c'est

obligatoire d'utiliser Allow Flitering. Cassandra connaît cette requête mais il ne peut pas l'exécuter de façon efficace.

Par conséquent, on utilise la fonction *pandas.date_range()* afin d'obtenir la liste des dates entre ces deux dates saises. Ensuite, on unit toutes les résultats obtenues par *SELECT ... WHERE date ='xxxx-xx-xx'*. De plus, il est beaucoup plus rapide de faire les requêtes sous la forme décrite ci-dessus.

IV. Traitement

1) *Question 1*

Au début du traitement, on a récupéré les informations d'une station, sur toute la période.
On choisit un indicateur à étudier.

Line Charts par mois :

On veut le minimum, la moyenne, et le maximum pour chaque mois.

1) **mapping**

(station, année, mois, jour, indicateur) ⇒ (mois, (1, indicateur, indicateur, indicateur))

2) **réduction**

r : (a , b) ⇒ (a[0]+b[0], min(a[1],b[1]), max(a[2],b[2]), a[3]+b[3])

3) **mapping**

(mois, (s0, s1, s2, s3))⇒ (mois, (s1, s2, s3/s0))

Boxplots pour la saisonnalité :

On veut un boxplot d'un indicateur pour chaque saison de l'année.

1) **mapping et groupe**

(station, année, mois, jour, indicateur) ⇒ (saison, indicateur) => (saison,*[itérable]*)

En fonction de *groupeByKey()*, pour chaque saison, on a un itérable dont on peut parcourir les valeurs, à l'aide d'un for par exemple.

2) **Boxplots**

On crée les boxplots à partir du mapping obtenu précédemment.

Line Charts par mois pour toute la période :

On calcule la moyenne pour un indicateur pour chaque mois de l'année, pour chaque année de la période d'étude.

1) **mapping**

(station, année, mois, jour, indicateur) ⇒ (année, mois, (1, indicateur))

2) **réduction**

r : (a , b) ⇒ (a+b)

3) **mapping**

((année, mois), (s0, s1))⇒ ((année, mois), (s1/s0))

4) **mapping et groupe**

((année, mois), s0)⇒ (année, (mois, s0)) =>(année, [itérable])

2) *Question 2*

De la base de données, on récupère les valeurs d'un indicateur, pour toutes les stations, pour une certaine heure (ainsi que les coordonnées de stations).

On veut pouvoir obtenir une carte pour représenter ces indicateurs. Or, les stations fournissent des informations en certains points. On utilise donc du krigeage pour extrapoler les valeurs des stations au reste de la carte.

La librairie utilisée pour le krigeage est PyKrige. On lui fournit les coordonnées des stations et leurs valeurs des indicateurs, ainsi qu'une grille de la France. Elle attribue ensuite une valeur à chaque point de la grille, basée sur les points alentours. Le krigeage utilisé est le krigeage ordinaire, avec un modèle de variogramme gaussien (cohérent par rapport aux indicateurs météorologiques).

On obtient alors une grille de la France avec une valeur pour l'indicateur choisi en chaque point. On crée alors une heatmap, sur un fond de carte du pays.

3) Question 3

Traitement de données

Dans la base de données, on récupère tous les enregistrements pour une certain période donnée mais avec une sélection de variables. Notre principe est de choisir les variables numériques dont la plupart des valeurs sont non nulles. Par conséquent, on prend (station,lon,lat) pour indiquer la station et prend (*tmpf, dwpf, relh, drct, sknt, alti, vsby ,skyl1, feel)* comme les indicateurs pour faire la clusterisation.

Dans une période, pour une station, on a plusieurs enregistrements. Néanmoins, l'idée est de clusteriser les stations. Du fait, on a pris la moyenne de chaque variable sauf (station,lon,lat) avec Map-Reduce. Dans la suite, l'algorithme de Kmeans s'applique sur ces données traitées.

Clusterisation : Kmeans

Pour la méthode de clusterisation, on d'abord fait une normalisation de Min-Max sur le jeu de données afin d'assurer le même poids de chaque variable. Ensuite, on réalise la méthode Kmeans à l'aide de Map-Reduce.

Input : data, K : nombre de classes , max itération, eps

1) *Initialiser au hasard K centres des catégories.*
2) *Mapping : Pour chaque station (un enregistrement), trouver le centre i le plus proche par la distance d'Euler et retourner une paire de clé - valeur : (i,(1,enregistrement)).*
3) *Réduction et Mapping : Dans cette étape, on a mis à jour le centre de chaque catégorie avec le moyen de tous les enregistrements actuels de cette catégorie.*
4) *Si les centres de K catégories sont stables et changent pas trop ou si on arrive déjà au maximum d'itération, on quitte la boucle et retourne le résultat de chaque enregistrement. Sinon, retourner à l'étape 2.*

Représentation

En utilisant la librairie Folium, on marque les stations sur une carte française par ses coordonnées (latitude, longitude) et indique leurs catégories par des marques de couleurs différents. Notre fonction générera à la fin un fichier au format HTML et on peut le visualiser sur navigateur.

Lien vers gitlab : *https://gitlab.utc.fr/durethug/nf26-part2-project* (bienvenu, vous êtes développeur)