TD 3 : Graphiques

L'objectif de ce TD est d'approfondir les différents types de graphique et illustrer leur utilité dans une étude de Data Mining. Certains ont déjà été abordés dans les cours précédents. Le choix est libre entre l'utilisation des data frames « classiques » et des tibbles combinées au package dplyr vu au cours précédent. Si le temps le permet, nous aborderons les principes de Shiny dans le but de développer notre propre application au cours suivant.

Comme à la fin du précédent cours, nous allons travailler avec les données des stations météorologiques de Météo-France et plus particulièrement les stations de Nantes et Rennes (année 2018). Ce set est disponible sous forme de fichier csv dans le Drive ou via Teams. Il va de pair avec le fichier de metadata. Commencez par importer ces deux fichiers dans votre environnement.

1 – Traitement des données : séparation des stations, correction des valeurs manquantes, transformation en données quotidiennes

1) Une fois les fichiers .csv chargés dans votre environnement, créez deux variables pour séparer les stations de Rennes et de Nantes. Vous pourrez pour cela vous servir de la fonction ***which()*** ou ***filter()*** et du fichier de metadata.
2) Conservez les variables date, t et precip pour chacune des stations avec la fonction ***select()*** ou ***which()***.
3) Déterminez la résolution temporelle des mesures et le taux de valeurs manquantes pour chaque variable et chaque station.
4) Installer le package ***fitdistrplus***. Utiliser les fonctions ***plotdist()*** et ***descdist()*** sur les données des stations en ayant au préalable mis de côté les NAs. Tenter d'expliquer le dernier graphique, en utilisant notamment l'argument boot sur la précipitation. Sinon, si vous avez des pbl d'installation du package ci dessus utiliser les fonctions ***geom_density()*** et ***gem_histogram()*** utiliséé l'année dernière.
5) Déterminer la loi la plus adaptée pour chacun des paramètres. Utiliser la fonction ***fitdist()*** pour estimer les valeurs des paramètres de ces lois pour notre échantillon.
6) Comment pourrait-on se servir de ce travail pour remplacer les valeurs manquantes de température et de précipitation ?
7) Remplacer finalement les données manquantes de température par une interpolation linéaire à l'aide de la fonction ***na.approx()*** du package ***zoo***. Remplacer par des 0 les valeurs manquantes de précipitation. Utiliser la fonction ***summary()*** pour vous assurer de la disparition des NAs.
8) Pour chaque station, transformer les données au format quotidien en faisant la moyenne pour la température et la somme pour les précipitations.
9) De plus, faites passer les températures de °K en °C : T(°C) = T(°K) – 273,15.
10) Vous devez donc vous retrouver avec deux variables (une par station) contenant 365 lignes et 3 colonnes (date, t et precip).

2 – Plot, HighCharter, séries multiples et corrélations linéaires

1) A l'aide de la fonction ***plot()*** ou du package ***highcharter/plotly***, tracer les deux séries temporelles de température.
2) Essayez d'estimer les relations linéaires entre les deux stations à l'aide de la fonction ***cor()*** et ***lm()***. N'hésitez pas à consulter les aides des fonctions pour savoir comment les utiliser.

3 – Boxplot

1) A l'aide de la fonction ***boxplot()***, tracez les boxplot pour chacune des variables. Utiliser les arguments *col* et *names* pour les couleurs et le nom des boxes.
2) Comparer les deux stations.

4 – Barplot

1) En fixant un seuil à 15°C, créer un vecteur qualifiant les jours avec une haute température et une basse température.
2) A l'aide de la fonction ***table()***, créer une variable qui contient le nombre de jours dans l'échantillon où la température est supérieure au seuil et le nombre de jours où elle est inférieure.
3) Tracer un diagramme à bâtons de cette nouvelle variable à l'aide de la fonction ***barplot()***.
4) Afin d'avoir des statistiques mensuelles, créer une nouvelle colonne qui donne le numéro du mois. On peut se servir de la fonction ***substr()*** pour extraire le numéro du mois de la date complète.
5) En créant une table à double entrée, déterminer le nombre de jours de haute température par mois. Se servir de la fonction ***table()*** et de son argument ***deparse.level*** pour déterminer le « degré » de détail.
6) Définir les couleurs pour chaque catégorie, et afficher la légende.
7) Vous pouvez également utiliser l'argument *beside* pour clarifier le graphique.
8) Comparer directement les deux stations en recherchant le nombre de jours par mois où il fait plus chaud à Nantes qu'à Rennes.

5 – Histogramme et Density plots

1) Tracer l'histogramme des températures avec la fonction ***hist() ou geom_histogram()***. Essayer plusieurs valeurs pour l'argument *breaks*.
2) Combinez les fonctions ***density() ou geom_denisty()*** et ***plot()*** pour tracer la densité.
3) Superposer les deux en utilisant notamment l'argument *prob* de la fonction ***hist()***.
4) Comparer ensuite les densités pour les deux stations et interpréter.

6 – Scatter plots simples et multiples

1) (Faire cette question si vous êtes en avance) Définir un vecteur de couleur dont les valeurs varient en fonction de l'écart entre les températures quotidiennes entre les deux stations.

2) Tracer la série de températures de Rennes en fonction de celle de Nantes. Conseil : laisser les points, ne les reliez pas par une droite.
3) En créant un échantillon fictif où x=y, tracer la bissectrice au graphique.
4) Repérer quelle ville est la plus chaude.
5) Rassemblez dans une seule variable les données de Rennes et de Nantes. Supprimez la colonne de dates.
6) A l'aide de la fonction ***corrplot.mixed()*** de la librairie ***corrplot*** et de cette nouvelle variable, tracez des scatter plot multiples.
7) Essayez la même chose avec la fonction ***pairs()***.
8) Encore une fois avec la fonction ***pairs.panel()*** du package ***psych***.