DATE DE REMISE :

- **Remettez votre travail au plus tard le 1er décembre 2024 à 23:55 sur Remise de travaux.**
- **AUCUN DEVOIR NE SERA ACCEPTÉ À UN MOMENT ULTÉRIEUR.**

DIRECTIVES :

- Ce travail compte pour **18 % de la note finale du cours**.
- Pour ce devoir, nous demandons à l'étudiant de fournir le code de programmation qu'il aura effectué pour chaque question.
- À l'intérieur du programme, il est important de bien mettre votre nom et votre matricule au tout début, et cela en commentaire. De plus, veuillez mentionner le numéro des questions en commentaire. Par ailleurs, lorsque vous devez répondre concrètement à une question, faite le à la suite du code que vous avez écrit pour répondre à la question.
- Vous devez **obligatoirement** envoyer un fichier nommé **MATH30602.A2024_Devoir2_NomDeFamille.**

  Envoyez votre travail en format '.**TXT'** sur Zone Cours dans la section '**Remise de travaux'** car les fichiers '.**R'** sont refusés.
- ATTENTION, le non-respect de ces règles peut entrainer des points en moins sur la note du devoir.
- Enfin, ce travail est un travail individuel. AUCUNE FORME DE PLAGIAT NE SERA TOLÉRÉE DANS CE DEVOIR. Cela pouvant causer l'échec au devoir et éventuellement au cours.

CONTEXTE :

- Nous allons analyser les données issues du site de la ville de Montréal qui recensent les crimes commis à Montréal.

DONNÉES :

Description :

- https://donnees.montreal.ca/dataset/actes-criminels/resource/c6f482bf-bf0f-4960-8b2f-9982c211addd/download
- https://spvm.qc.ca/fr/Search/Quartier?Quartier=

Fichiers de données :

- Les fichiers suivants seront utilisés pour ce devoir :
  - **actes-criminels.csv** : la liste des crimes commis à Montréal
  - **pdq_description.xlsx** : la description des PDQ

Directives générales :

- Le logiciel R devra être utilisé pour réaliser ce devoir.
- Le devoir sera évalué en tenant compte des aspects suivants :
  - Bonne réponse à la question.
  - Clarté dans le code fourni.
  - Simplicité dans le code fourni.
  - Respect des directives.

Partie 1 – 40 points

Question 1 – Importation des données (5 points)

- Importez les jeux de données actes-criminels.csv et pdq_description.xlsx

Question 2 – Jointure (10 points)

- Faire une jointure entre les deux tables chargées à la question 1 afin qu'en face de chaque PDQ de la table actes-criminels figure la description du PDQ associés.

Question 3 – Test sur la jointure (5 points)

a- Quels PDQ n'ont pas été retrouvés (PDQ sans description)?

b- Combien de lignes cela représente?

c- Supprimez les lignes pour lesquelles la description du PDQ n'a pas été retrouvée.

d- Quel PDQ enregistre le plus grand nombre de crimes? Et le plus faible nombre?
Indices : utilisez la fonction summary sur une variable de type factor. Sauvegardez votre résultat dans un objet data.frame que vous ordonnerez ensuite par order croissant.

Question 4 – Boucle de partage par description (5 points)

- Écrivez une boucle **for** qui, pour chaque valeur unique de la colonne description, créera un nouvel élément qui prendra la valeur de la description en cours de traitement et assignera les valeurs. La liste s'appelera crimes_by_PDQ, elle sera initialement vide et contiendra, une fois le boucle for terminée, autant d'élément que de valeur unique de la variable description.

Question 5 – Généralisation de la fonction (10 points)

- À partir du code que vous avez développé à la question 4, créez une fonction nommée **split_df_to_list** qui prendra en paramètres d'entrée **entry_df**, un data.frame sur lequel on veut travailler, **split_var**, qui sera le nom d'une variable catégorielle à partir de laquelle on veut faire le partage de la base de données, enfin **selected_var**, un paramètre qui prendra la valeur par défaut NULL et qui pourra contenir un vecteur de valeurs représentant les colonne de la table entry_df que vous aimeriez voir apparaître dans la sortie.

Question 6 – Testez votre fonction (5 points)

- Testez votre fonction avec entry_df, le data.frame créé à la question 2, avec la split variable CATEGORIE avec en sortie les variables longitude, latitude, date et quart.

Partie 2 – 60 points (10 points par question)

La suite des étapes successives suivantes vous permettra de créer une fonction qui à partir d'une position géographique donnée par une combinaison latitude longitude, vous rendra une liste des crimes s'étant produits dans un périmètre de x kilomètre autour de ce point central et vous permettra de visualiser les crimes.

Question 1 – Adaptation de code

- À partir du code donné ici : https://stackoverflow.com/a/66210126, ajustez le code suivant :

```
get_circle_points <- function(center_long,
                              center_lat,
                              radius_in_km,
                              nb_of_points){
# Complétez les fonctions avec les bons paramètres d'entrée
 radiusLon = 1 / (111.319 * cos(XXX * (pi / 180))) * XXX
 radiusLat = 1 / 110.574 * XXX

 dTheta = 2 * pi / XXX
 theta = 0
 # Un data.frame vide dans lequel on va stocker le résultat au fur et à mesure des itérations de la boucle for
 points_on_circle =XXX

 for (i in 1:XXX){
# Contanez les résultats correctement
  points_on_circle <- XXX(points_on_circle,
                          XXX(center_long + radiusLon * cos(theta),
                            center_lat + radiusLat * sin(theta)
                          ))
  theta = theta + dTheta
 }
 points_on_circle <- data.frame(points_on_circle)
#Ajustez le nom des colonnes du dataframe de sortie
XXX(points_on_circle) <- c('LONGITUDE', 'LATITUDE')

 points_on_circle$geographical_quarter <- 'West - South'
 points_on_circle$geographical_quarter[points_on_circle$LATITUDE >= center_lat & points_on_circle$LONGITUDE
<= center_long] <- 'West - North'
 points_on_circle$geographical_quarter[points_on_circle$LATITUDE >= center_lat & points_on_circle$LONGITUDE >
center_long] <- 'East - North'
# En vous inspirant de ce qui est fait dans les 2 lignes au dessus, écrivez la dernière condition
XXXX …. XXXX <- 'East - South'
 # Ajoutez le point central défini en paramètre d'entrée
 points_on_circle = XXX(XXX, c(XXX, XXX, 'Center'))
# Ajustez les formats à numérique
 points_on_circle$LONGITUDE <- XXX (points_on_circle$LONGITUDE)
 points_on_circle$LATITUDE <- XXX (points_on_circle$LATITUDE)
# On veut retournez le dataframe que l'on a construit dans la fonction
return(XXX)
}
```

Le but étant d'obtenir une sortie qui ressemble à ça :

```
  LONGITUDE LATITUDE geographical_quarter
1 -73.84861 45.45093         East - North
2 -73.84864 45.45150         East - North
3 -73.84872 45.45206         East - North
>
```

Question 2 – Testez votre fonction

- Testez votre fonction en utilisant et ajustant le code suivant :

  test1km <- get_circle_points(**XXX**, **XXX**, **XXX**, **XXX**)
  plot(**LONGITUDE**, **LATITUDE** , col = as.factor(**geographical_quarter**))

  On veut se placer au point latitude 45.4983503 et longitude -73.6212999, pour un radius de 1 kilomètre et 100 points pour caractériser le périmètre.

Question 3 – Une fois des points du cercle déterminés, filtrez les observations de votre base de données ne sorte à ne garder que celle qui se situe dans le carré dont les limites sont données par le minimum et le maximum de la longitude et de la latitude.

Donc ne garder dans une table réduite, que les observations pour lesquelles la latitude et entre le min et le max de la latitude du cercle obtenu en utilisant la fonction get_circle_points, pareil pour la longitude.

Question 4 – Identifiez les points à l'intérieur du périmètre voulu.

- Ajustez la fonction suivante. Cette dernière prend 2 arguments en entrée :
  1- Une ligne d'observation d'un data.frame qui contient au minimum les colonnes LONGITUDE et LATITUDE
  2- Le data.frame qui contient les points du cercle et la description géographique, comme celui obtenu avec le code la question 1.

  Note :

  - La fonction any retourne TRUE si au moins un test est vrai.
  - La fonction retourne TRUE si toutes les conditions sont vraies.

```
in_the_cercle_v2 <- function(x, circle_point){
 res <- all(c(any(XXX[XXX$geographical_quarter == 'West - South',c('LONGITUDE')]<= as.numeric(XXX[['XXX']]) & XXX[XXX$geographical_quarter == 'West - South',c('LATITUDE')]< as.numeric(XXX[['XXX']])),
            any(XXX[XXX$geographical_quarter == 'West - North',c('LONGITUDE')]<= as.numeric(XXX[['XXX']]) & XXX[XXX$geographical_quarter == 'West - North',c('LATITUDE')]> as.numeric(XXX[['XXX']])),
            any(XXX[XXX$geographical_quarter == 'East - North',c('LONGITUDE')]>= as.numeric(XXX[['XXX']]) & XXX[XXX$geographical_quarter == 'East - North',c('LATITUDE')]> as.numeric(XXX[['XXX']])),
#En vous basant sur les trois précédentes conditions écrivez la dernière
            XXX…XXX
           ))
   return(XXX)
}
```

Question 5 – Rassemblez toutes les pièces

- Ajustez le code suivant afin d'obtenir la liste des crimes commis autour d'un point central et d'identifier ceux dans le rayon souhaité.

```
merge_all <-function(data_center_long, data_center_lat, entire_spatial_DB, radius_in_km, nb_radius_points=100){
  radius_points <- appel_de_la_fonction_de_la_question_1
  entire_spatial_DB_int <- la_logique_de_la_question_3
# Appliquer de façon récursive la fonction sur toutes les observations d'intérêt
  res <- data.frame(XXX (sur_quelle_base, 0 ou 1,la_fonction, circle_point = radius_points))
  return(list(res, entire_spatial_DB_int,radius_points))
}
```

Question 6 – Testez votre fonction

- Placez-vous à l'édifice Côte-Sainte-Catherine et analysez dans un rayon de 1.5 KM autour de l'édifice.

```
Latitude : 45.50333, longitude : -73.6206714
testRes <- merge_all(XXX, XXX, XXX, XXX, 100)
plot(testRes[[2]]$LONGITUDE, testRes[[2]]$LATITUDE, col = as.factor(testRes[[1]][,1]))
```