Compte rendu Travaux pratiques en langage R

**Pour charger les données :**

data = *read.csv*("psychology.csv", *TRUE*)  
*print*(data)

Le code permet de lire un fichier CSV nommé “psychology.csv” qui contient des données sur la psychologie des individus. Le paramètre TRUE (ou header = TRUE) indique que la première ligne du fichier contient les noms des variables. C’est-à-dire que dans le fichier csv à la 1er ligne on remarque : IBE, IP et qu’il est identifié comme un header. La fonction print (data) affiche le contenu du fichier dans la console.

**Question 1 :**

# ---- Question 1 ----  
# Statistiques descriptives pour l'IBE  
summary\_IBE = *summary*(data$IBE)  
mean\_IBE = *mean*(data$IBE)  
median\_IBE = *median*(data$IBE)  
sd\_IBE = *sd*(data$IBE)  
IQR\_IBE = *IQR*(data$IBE)  
  
# Affichage des résultats  
*cat*("Statistiques descriptives pour l'IBE:\n")  
*cat*("Sommaire:", summary\_IBE, "\n")  
*cat*("Moyenne:", mean\_IBE, "\n")  
*cat*("Médiane:", median\_IBE, "\n")  
*cat*("Écart-type:", sd\_IBE, "\n")  
*cat*("IQR:", IQR\_IBE, "\n")  
  
# Statistiques descriptives pour l'IP  
summary\_IP = *summary*(data$IP)  
mean\_IP = *mean*(data$IP)  
median\_IP = *median*(data$IP)  
sd\_IP = *sd*(data$IP)  
IQR\_IP = *IQR*(data$IP)  
  
# Affichage des résultats  
*cat*("\nStatistiques descriptives pour l'IP:\n")  
*cat*("Sommaire:", summary\_IP, "\n")  
*cat*("Moyenne:", mean\_IP, "\n")  
*cat*("Médiane:", median\_IP, "\n")  
*cat*("Écart-type:", sd\_IP, "\n")  
*cat*("IQR:", IQR\_IP, "\n")

Cette partie du code permet de calculer et d’afficher les statistiques descriptives pour la variable IBE et IP, qui représente l’indice de bien-être des individus (IBE) et de productivité (IP). La fonction summary(data$IBE) renvoie les valeurs minimale, maximale, moyenne, médiane et les quartiles de la variable. Les fonctions mean(data$IBE), median(data$IBE), sd(data$IBE) et IQR(data$IBE) renvoient respectivement la moyenne, la médiane, l’écart-type et l’intervalle interquartile de la variable. La fonction cat() permet d’afficher les résultats dans la console avec des sauts de ligne (\n) et des séparateurs (:) de la même manière qu’un print.Pour IP on utilise la même chose ce qui renvoie en console :

Statistiques descriptives pour l'IBE:

Sommaire: 1 24.25 50 48.29333 74.5 100

Moyenne: 48.29333

Médiane: 50

Écart-type: 29.28626

IQR: 50.25

Statistiques descriptives pour l'IP:

Sommaire: 0 4.09425 4.6505 4.475667 5.264 6.092

Moyenne: 4.475667

Médiane: 4.6505

Écart-type: 1.201006

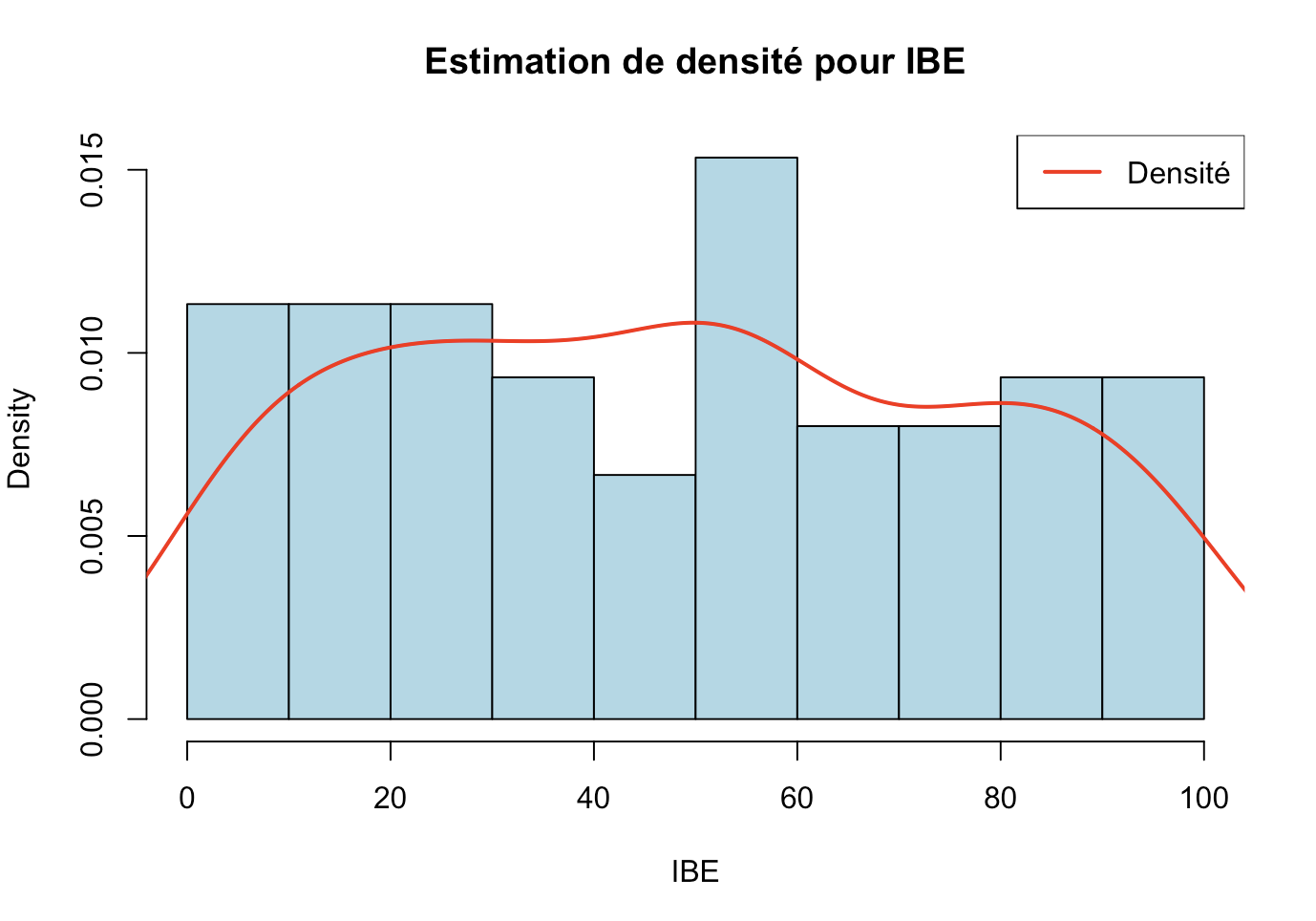
IQR: 1.16975

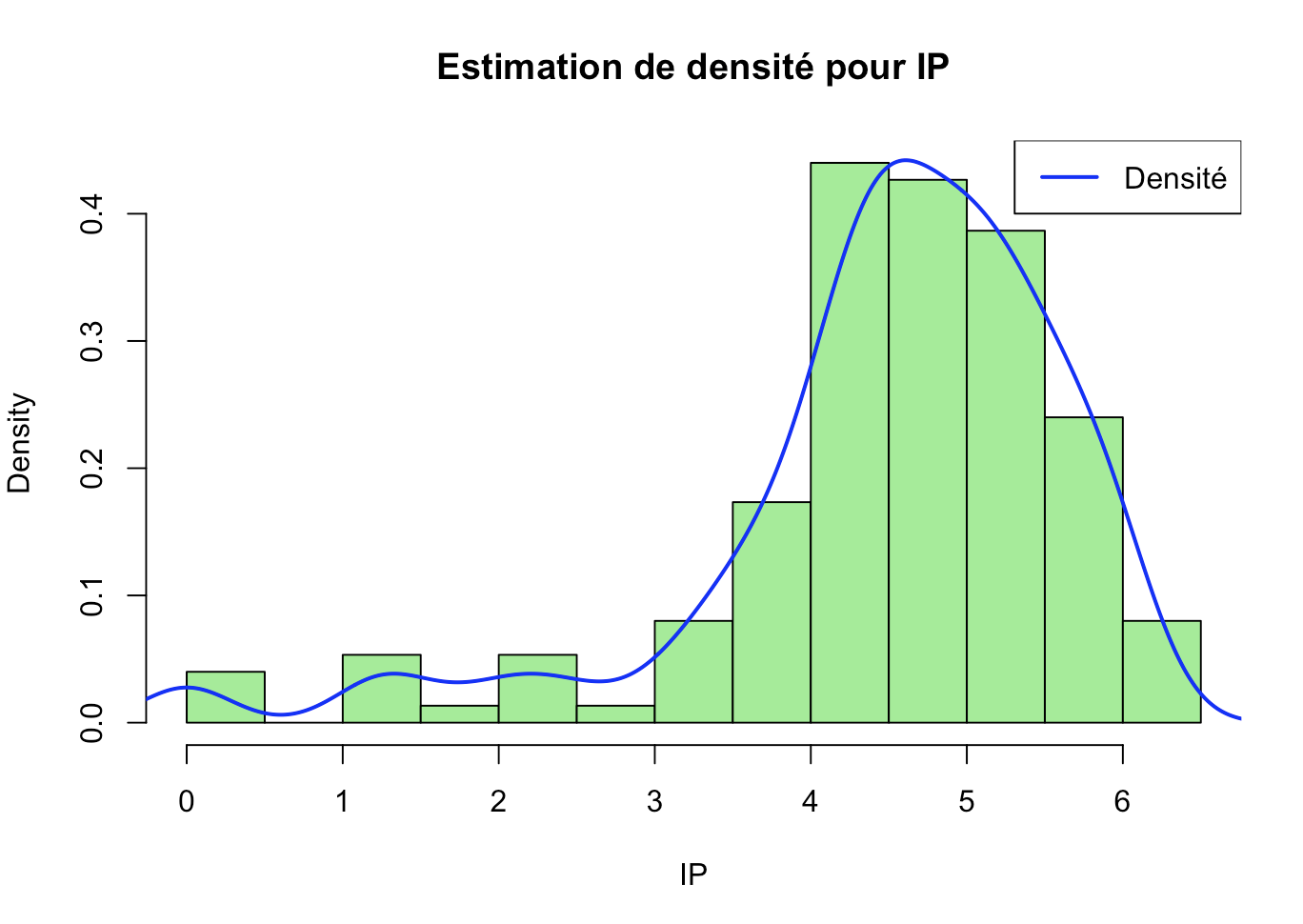
**Question 2 :**

# ---- Question 2 ----  
# Créer un histogramme pour la variable IBE  
*hist*(data$IBE, main="Estimation de densité pour IBE", xlab="IBE", col="lightblue", border="black", probability=*TRUE*)  
*lines*(*density*(data$IBE), col="red", lwd=2)  
  
# Ajouter une légende  
*legend*("topright", legend="Densité", col="red", lwd=2)  
  
# Créer un histogramme pour la variable IP  
*hist*(data$IP, main="Estimation de densité pour IP", xlab="IP", col="lightgreen", border="black", probability=*TRUE*)  
*lines*(*density*(data$IP), col="blue", lwd=2)  
  
# Ajouter une légende  
*legend*("topright", legend="Densité", col="blue", lwd=2)

Cette partie du code permet de créer des histogrammes pour les variables IBE et IP, afin de visualiser la distribution des données. La fonction hist() prend en paramètre la variable à représenter, le titre du graphique (main), le nom de l’axe des abscisses (xlab), la couleur des barres (col), la couleur des bordures (border) et le type de fréquence (probability=TRUE pour avoir des fréquences relatives). La fonction lines() permet de tracer la courbe de densité estimée par noyau (KDE) sur l’histogramme, en précisant la couleur (col) et l’épaisseur (lwd) de la ligne. La fonction legend() permet d’ajouter une légende au graphique, en indiquant la position (topright), le texte (legend), la couleur (col) et l’épaisseur (lwd) de la ligne.

Ainsi on a graphiquement pour IBE et IP avec les couleurs appliquées :

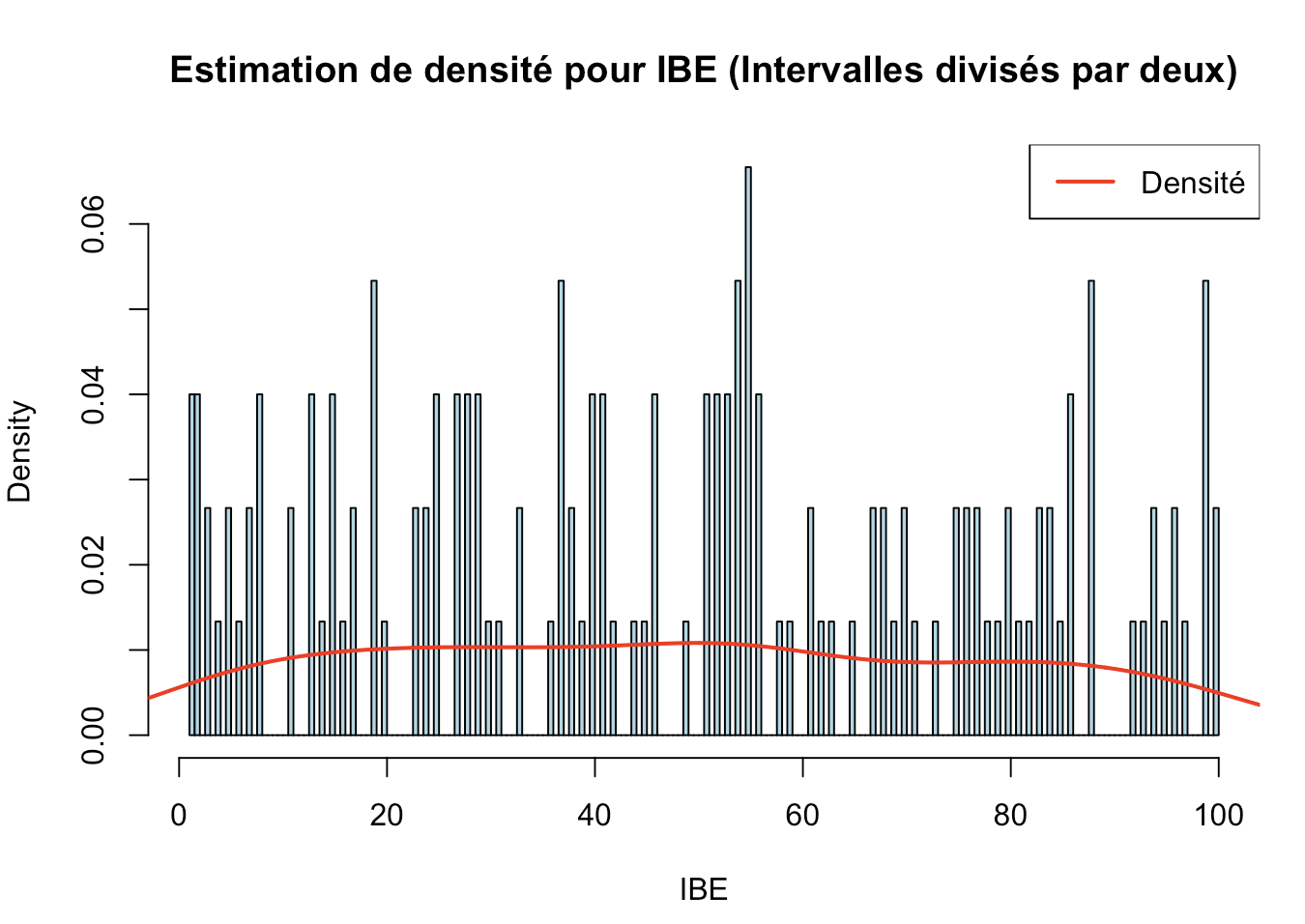


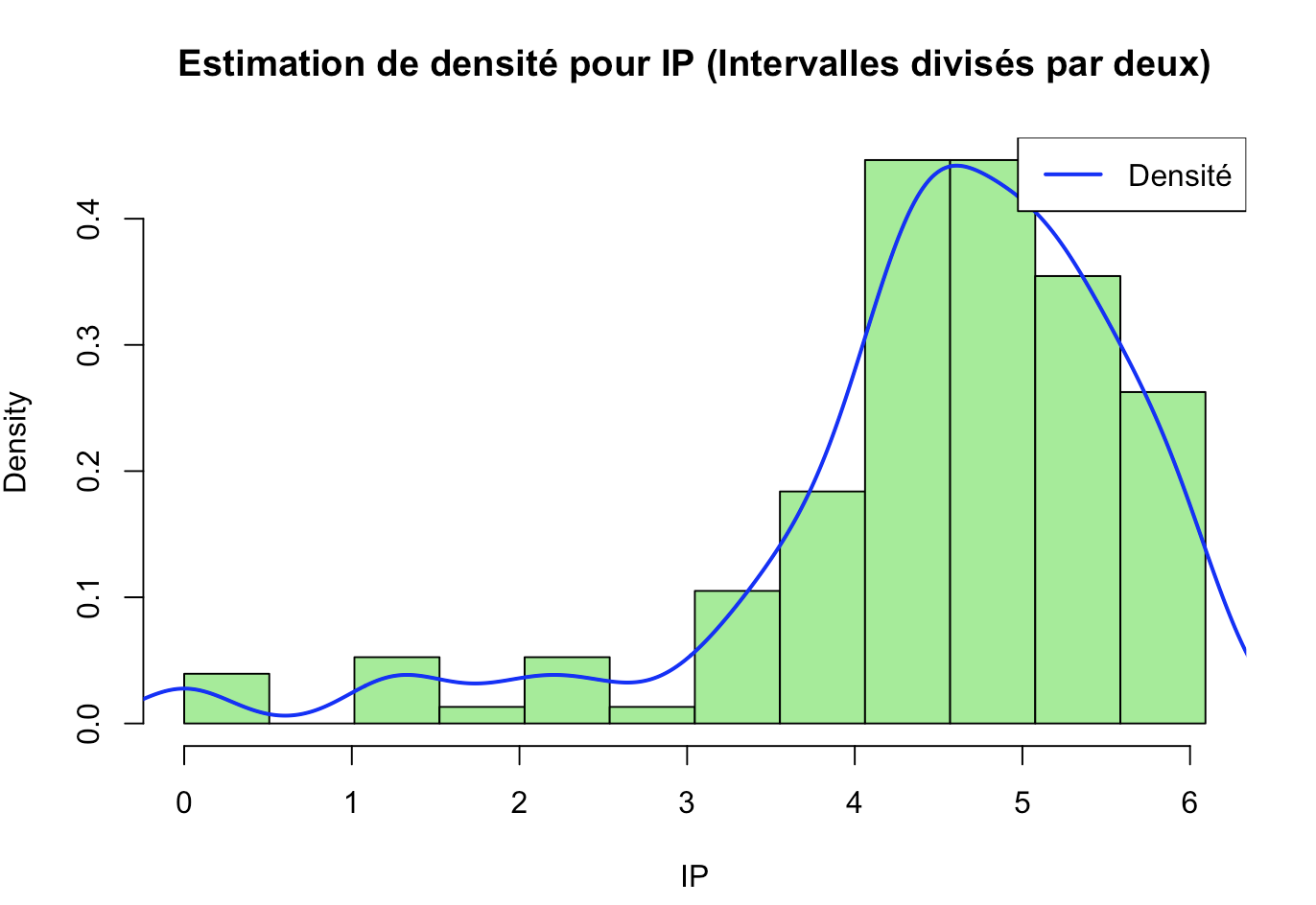


**Question 3 :**

# ---- Question 3 ----  
  
# Diviser par deux la largeur des intervalles pour IBE  
new\_breaks\_ibe = *seq*(*min*(data$IBE), *max*(data$IBE), length.out = (*length*(*seq*(*min*(data$IBE), *max*(data$IBE))) - 1) \* 2 + 1)  
  
# Créer un histogramme pour la variable IBE avec des intervalles divisés par deux  
*hist*(data$IBE, breaks = new\_breaks\_ibe, main="Estimation de densité pour IBE (Intervalles divisés par deux)", xlab="IBE", col="lightblue", border="black", probability=*TRUE*)  
*lines*(*density*(data$IBE), col="red", lwd=2)  
  
# Ajouter une légende  
*legend*("topright", legend="Densité", col="red", lwd=2)  
  
# Diviser par deux la largeur des intervalles pour IP  
new\_breaks\_ip = *seq*(*min*(data$IP), *max*(data$IP), length.out = (*length*(*seq*(*min*(data$IP), *max*(data$IP))) - 1) \* 2 + 1)  
  
# Créer un histogramme pour la variable IP avec des intervalles divisés par deux  
*hist*(data$IP, breaks = new\_breaks\_ip, main="Estimation de densité pour IP (Intervalles divisés par deux)", xlab="IP", col="lightgreen", border="black", probability=*TRUE*)  
*lines*(*density*(data$IP), col="blue", lwd=2)  
  
# Ajouter une légende  
*legend*("topright", legend="Densité", col="blue", lwd=2)

Ce code permet de créer des histogrammes avec des intervalles plus fins pour les variables IBE et IP, afin de mieux apprécier la forme de la distribution. La fonction seq() permet de créer une séquence de valeurs régulièrement espacées entre le minimum et le maximum de la variable, en précisant le nombre de valeurs (length.out). Ce nombre est calculé en multipliant par deux le nombre d’intervalles de l’histogramme précédent, moins un. La fonction hist() prend en paramètre le vecteur des intervalles (breaks) à utiliser pour créer l’histogramme. Les autres fonctions sont les mêmes que pour la question 2. Ce qui donne graphiquement :

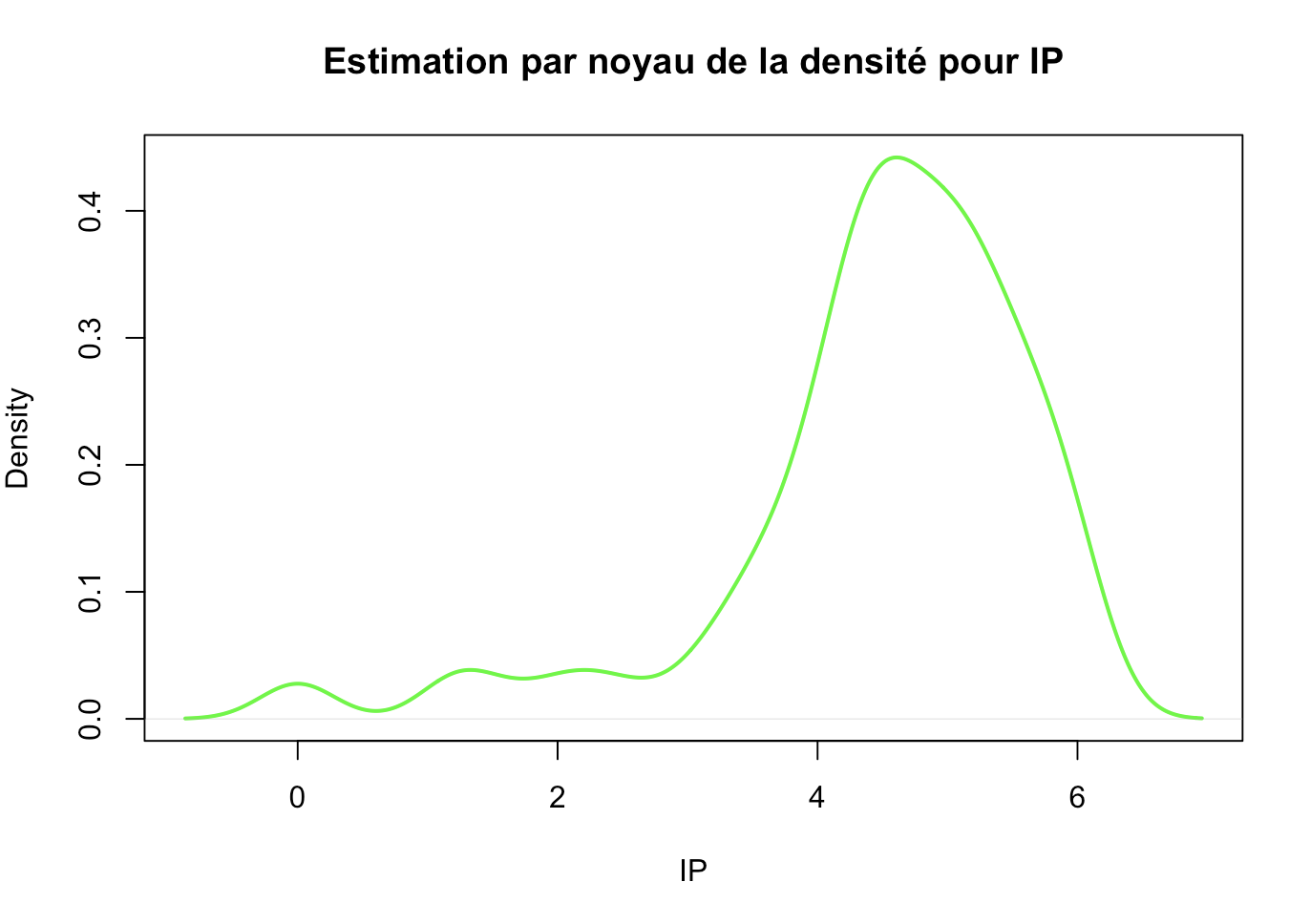
****

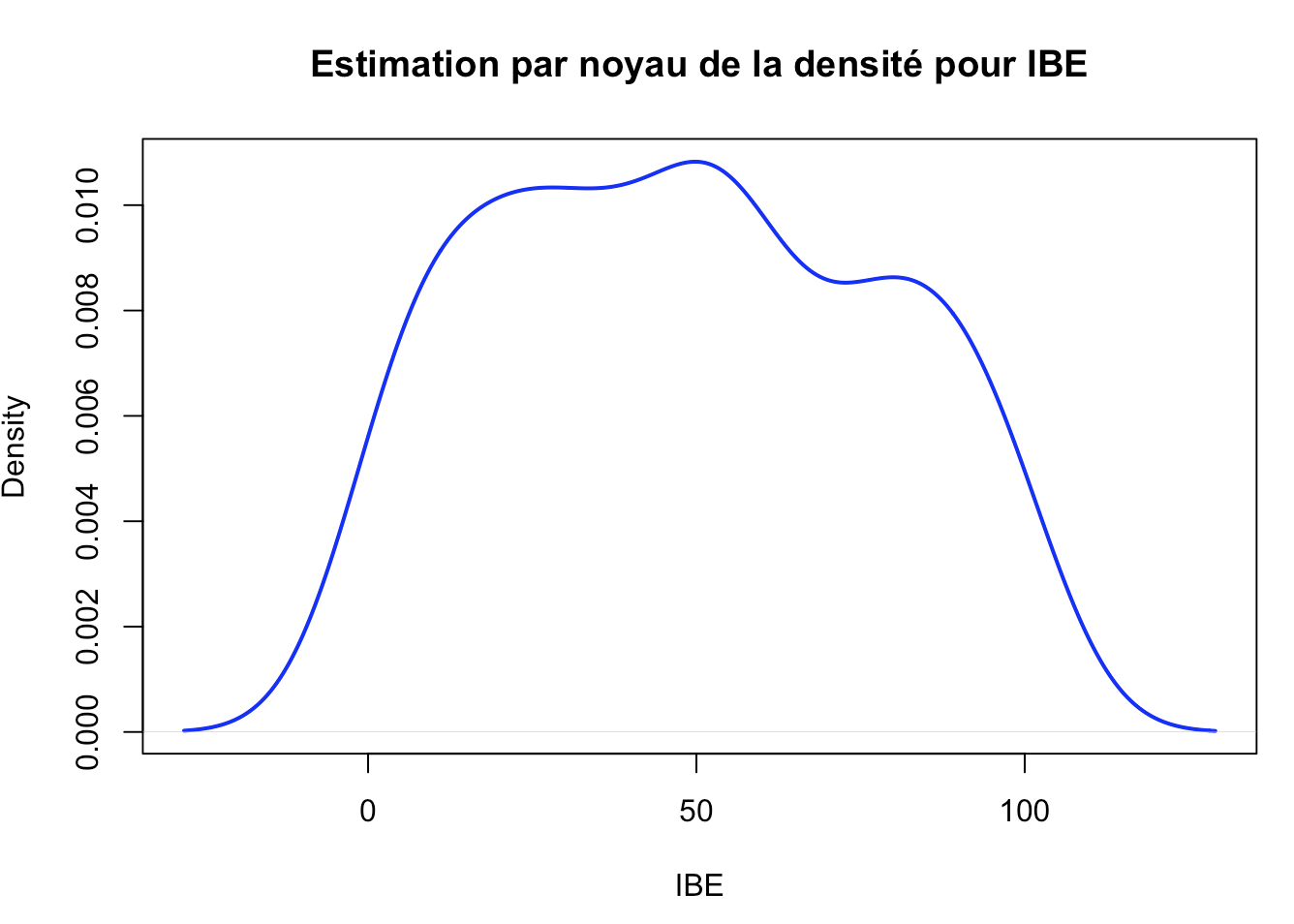
****

**Question 4 :**

# ---- Question 4 ----  
# Estimation par noyau de la densité pour IBE  
density\_ibe = *density*(data$IBE)  
  
# Tracer le KDE pour IBE  
*plot*(density\_ibe, main="Estimation par noyau de la densité pour IBE", xlab="IBE", col="blue", lwd=2)  
  
# Estimation par noyau de la densité pour IP  
density\_ip = *density*(data$IP)  
  
# Tracer le KDE pour IP  
*plot*(density\_ip, main="Estimation par noyau de la densité pour IP", xlab="IP", col="green", lwd=2)

Cette partie du code permet de tracer les courbes de densité estimées par noyau (KDE) pour les variables IBE et IP, sans utiliser d’histogramme. La fonction density() permet de calculer la densité par noyau pour une variable. La fonction plot() permet de tracer la courbe de densité, en précisant le titre du graphique (main), le nom de l’axe des abscisses (xlab), la couleur (col) et l’épaisseur (lwd) de la ligne.

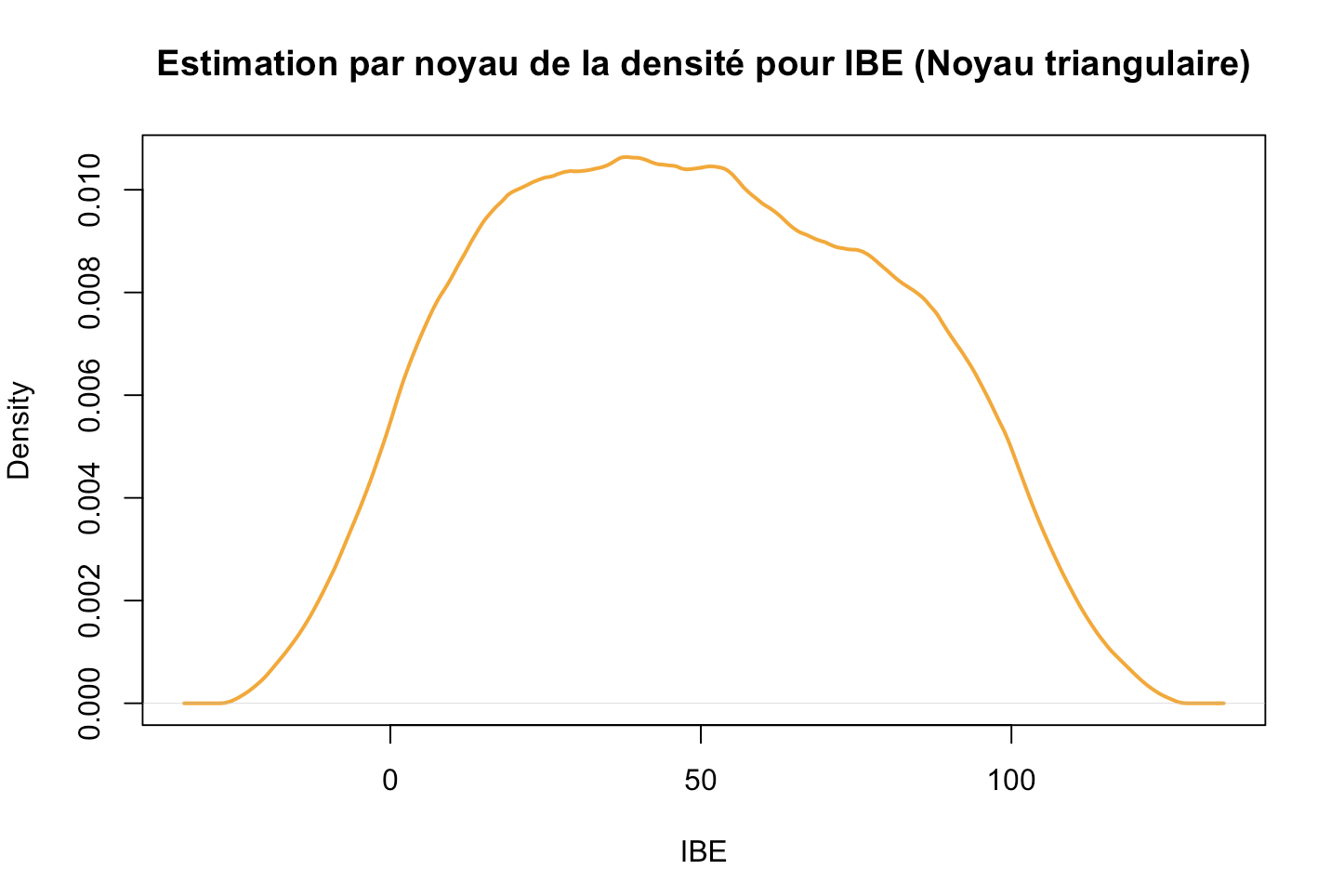
****

****

**Question 5 :**

# ---- Question 5 ----  
  
# Estimation par noyau de la densité pour IBE avec noyau triangulaire et validation croisée biaisée  
density\_ibe\_triangular = *density*(data$IBE, kernel = "triangular", bw = "nrd")  
  
# Tracer le KDE pour IBE avec noyau triangulaire  
*plot*(density\_ibe\_triangular, main="Estimation par noyau de la densité pour IBE (Noyau triangulaire)", xlab="IBE", col="orange", lwd=2)  
  
# Estimation par noyau de la densité pour IP avec noyau triangulaire et validation croisée biaisée  
density\_ip\_triangular = *density*(data$IP, kernel = "triangular", bw = "nrd")  
  
# Tracer le KDE pour IP avec noyau triangulaire  
*plot*(density\_ip\_triangular, main="Estimation par noyau de la densité pour IP (Noyau triangulaire)", xlab="IP", col="purple", lwd=2)

Cette partie du code permet d’estimer la densité par noyau (KDE) pour les variables IBE et IP, en utilisant un noyau triangulaire et une validation croisée biaisée. La fonction density() permet de calculer la densité par noyau pour une variable, en spécifiant le type de noyau (kernel) et la largeur de bande (bw). La fonction plot() permet de tracer la courbe de densité, en précisant le titre du graphique (main), le nom de l’axe des abscisses (xlab), la couleur (col) et l’épaisseur (lwd) de la ligne. Le code est répété pour les deux variables IBE et IP, en changeant la couleur de la courbe.



Une image contenant texte, Tracé, diagramme, ligne

Description générée automatiquement

**Question 6 :**

# ---- Question 6 ----  
  
# Afficher le nuage de points entre les deux variables IBE et IP  
*plot*(data$IBE, data$IP, main="Nuage de points entre IBE et IP", xlab="IBE", ylab="IP", col=*c*("blue", "green"), pch=16)  
  
# Ajouter une régression linéaire pour illustrer la tendance  
*abline*(*lm*(IP ~ IBE, data=data), col="red", lwd=2)  
  
# Ajouter une légende  
*legend*("topright", legend=*c*("IBE", "IP", "Régression linéaire"), col=*c*("blue", "green", "red"), lwd=2)  
  
# Calculer et afficher la corrélation  
correlation\_coefficient = *cor*(data$IBE, data$IP)  
*cat*("Coefficient de corrélation entre IBE et IP:", correlation\_coefficient, "\n")  
*cat*("La ligne de régression est inclinée vers le haut, cela suggère une relation positive entre les variables (quand l'une augmente, l'autre a tendance à augmenter).")

Cette partie du code permet d’afficher le nuage de points entre les deux variables IBE et IP, pour visualiser la relation entre elles. La fonction plot() permet de tracer le nuage de points, en précisant les coordonnées des points (data$IBE et data$IP), le titre du graphique (main), le nom des axes (xlab et ylab), la couleur (col) et la forme (pch) des points. La fonction abline() permet d’ajouter une régression linéaire pour illustrer la tendance, en utilisant la fonction lm() pour ajuster le modèle linéaire (IP ~ IBE). La fonction legend() permet d’ajouter une légende, en précisant la position (topright), le texte (legend), la couleur (col) et l’épaisseur (lwd) des lignes. Les fonctions cor() et cat() permettent de calculer et d’afficher le coefficient de corrélation entre IBE et IP, ainsi qu’un commentaire sur la relation entre les variables. Ce qui renvoie en console :

Coefficient de corrélation entre IBE et IP: 0.7235905

La ligne de régression est inclinée vers le haut, cela suggère une relation positive entre les variables (quand l'une augmente, l'autre a tendance à augmenter).

Une image contenant texte, ligne, diagramme, capture d’écran

Description générée automatiquement

**Question 7 :**

# ---- Question 7 ----  
  
# Filtrer les valeurs positives avant la transformation logarithmique  
positive\_values = data$IP > 0  
transformed\_IBE = *log*(data$IBE[positive\_values])  
transformed\_IP = *log*(data$IP[positive\_values])  
  
# Afficher le nuage de points avec les données transformées  
*plot*(transformed\_IBE, transformed\_IP, main="Nuage de points entre IBE et IP (Transformées logarithmiques)", xlab="log(IBE)", ylab="log(IP)", col=*c*("purple", "orange"), pch=16)  
  
# Ajouter une régression linéaire pour illustrer la tendance  
*abline*(*lm*(transformed\_IP ~ transformed\_IBE), col="red", lwd=2)  
  
# Ajouter une légende avec position manuelle  
*legend*("topright", legend=*c*("log(IBE)", "log(IP)", "Régression linéaire"), col=*c*("purple", "orange", "red"), lwd=2)  
  
# Calculer et afficher la corrélation pour les données transformées  
correlation\_coefficient\_transformed = *cor*(transformed\_IBE, transformed\_IP)  
*cat*("Coefficient de corrélation entre log(IBE) et log(IP):", correlation\_coefficient\_transformed, "\n")

Cette partie du code permet de transformer les variables IBE et IP par le logarithme, pour réduire l’effet des valeurs extrêmes et rendre la relation plus linéaire. Avant la transformation, il faut filtrer les valeurs positives, car le logarithme n’est pas défini pour les valeurs négatives ou nulles. La fonction log() permet de calculer le logarithme naturel d’une variable. Le code est ensuite similaire à la question 6, en utilisant les variables transformées (transformed\_IBE et transformed\_IP) et en changeant les titres et les légendes des graphiques. Le coefficient de corrélation est également recalculé pour les données transformées. Ce qui renvoie en console :

Coefficient de corrélation entre log(IBE) et log(IP): 0.8549226

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

**Question 8 :**

# ---- Question 8 ----  
  
# Calculer le coefficient de corrélation de Pearson  
correlation\_pearson = *cor*(transformed\_IBE, transformed\_IP, method = "pearson")  
*cat*("Coefficient de corrélation de Pearson entre log(IBE) et log(IP):", correlation\_pearson, "\n")  
  
# Calculer le coefficient de corrélation de Spearman  
correlation\_spearman = *cor*(transformed\_IBE, transformed\_IP, method = "spearman")  
*cat*("Coefficient de corrélation de Spearman entre log(IBE) et log(IP):", correlation\_spearman, "\n")  
  
# Calculer le coefficient de corrélation de Kendall  
correlation\_kendall = *cor*(transformed\_IBE, transformed\_IP, method = "kendall")  
*cat*("Coefficient de corrélation de Kendall entre log(IBE) et log(IP):", correlation\_kendall, "\n")

Cette partie du code permet de comparer différents coefficients de corrélation entre les variables transformées, en utilisant différentes méthodes : Pearson, Spearman et Kendall. La fonction cor() permet de calculer le coefficient de corrélation, en spécifiant la méthode (method) à utiliser. La fonction cat() permet d’afficher les résultats. Le coefficient de Pearson mesure la corrélation linéaire entre les variables, c’est-à-dire le degré de variation conjointe par rapport à la moyenne. Le coefficient de Spearman mesure la corrélation de rang entre les variables, c’est-à-dire le degré de concordance entre l’ordre des valeurs. Le coefficient de Kendall mesure la corrélation de concordance entre les variables, c’est-à-dire la probabilité que les paires de valeurs soient ordonnées de la même façon. Ainsi on a :

Coefficient de corrélation de Pearson entre log(IBE) et log(IP): 0.8549226

Coefficient de corrélation de Spearman entre log(IBE) et log(IP): 0.7410984

Coefficient de corrélation de Kendall entre log(IBE) et log(IP): 0.5551457

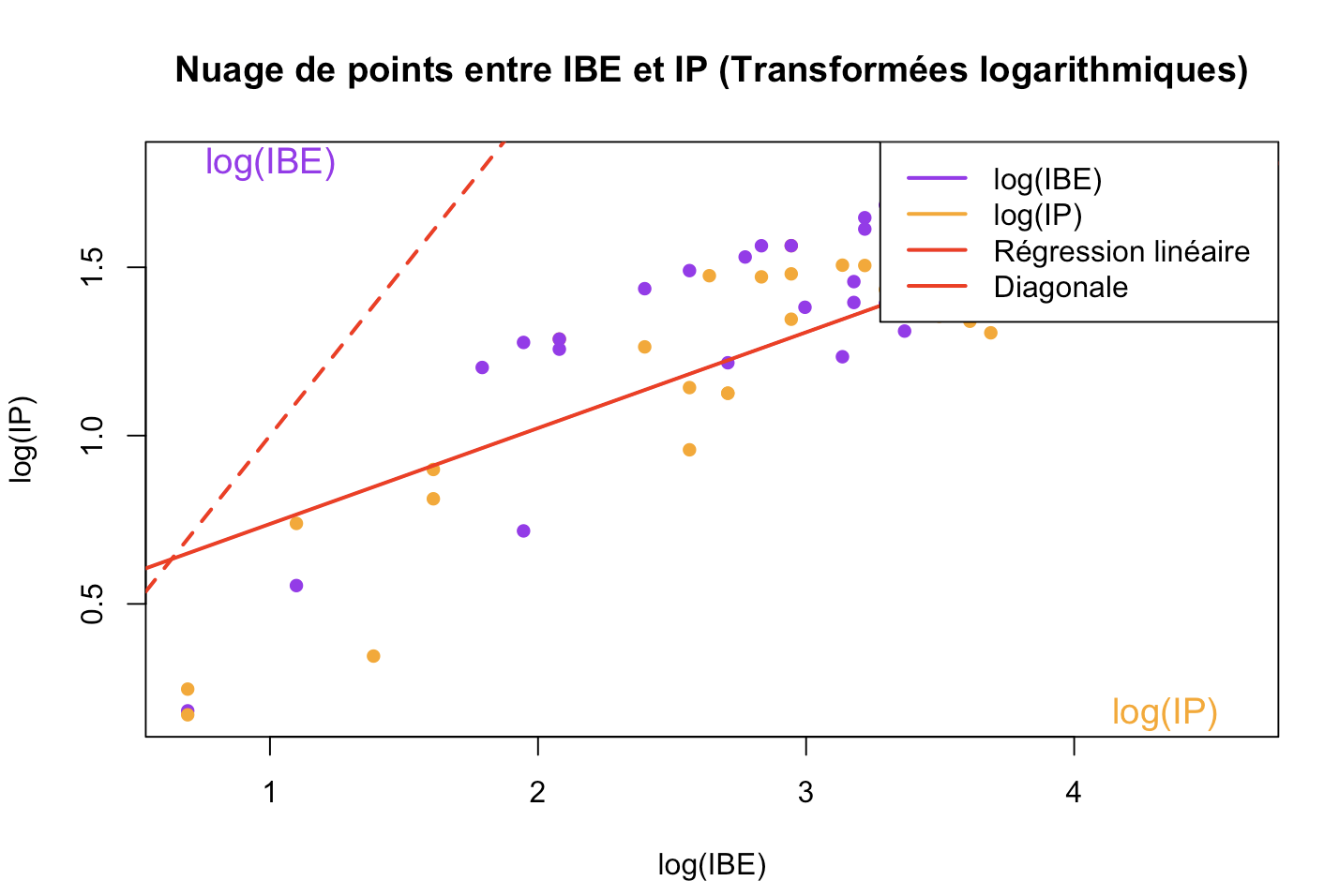
Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

**Question 9 :**

# ---- Question 9 ----  
  
# Afficher le nuage de points avec les données transformées  
*plot*(transformed\_IBE, transformed\_IP, main="Nuage de points entre IBE et IP (Transformées logarithmiques)", xlab="log(IBE)", ylab="log(IP)", col=*c*("purple", "orange"), pch=16)  
  
# Ajouter une régression linéaire pour illustrer la tendance  
*abline*(*lm*(transformed\_IP ~ transformed\_IBE), col="red", lwd=2)  
  
# Ajouter une ligne diagonale (pente = 1) en rouge  
*abline*(a = 0, b = 1, col = "red", lwd = 2, lty = 2)  
  
# Ajouter les noms des axes  
*text*(x = *min*(transformed\_IBE), y = *max*(transformed\_IP), labels = "log(IBE)", pos = 4, col = "purple", cex = 1.2)  
*text*(x = *max*(transformed\_IBE), y = *min*(transformed\_IP), labels = "log(IP)", pos = 2, col = "orange", cex = 1.2)  
  
# Ajouter une légende avec position manuelle  
*legend*("topright", legend=*c*("log(IBE)", "log(IP)", "Régression linéaire", "Diagonale"), col=*c*("purple", "orange", "red", "red"), lwd=2)  
  
# Calculer et afficher la corrélation pour les données transformées  
correlation\_coefficient\_transformed = *cor*(transformed\_IBE, transformed\_IP)  
*cat*("Coefficient de corrélation entre log(IBE) et log(IP):", correlation\_coefficient\_transformed, "\n")

Cette partie du code permet d’afficher le nuage de points entre les deux variables IBE et IP, transformées par le logarithme, pour visualiser la relation entre elles. La fonction plot() permet de tracer le nuage de points, en précisant les coordonnées des points (transformed\_IBE et transformed\_IP), le titre du graphique (main), le nom des axes (xlab et ylab), la couleur (col) et la forme (pch) des points. La fonction abline() permet d’ajouter une régression linéaire pour illustrer la tendance, en utilisant la fonction lm() pour ajuster le modèle linéaire (transformed\_IP ~ transformed\_IBE). La fonction abline() permet également d’ajouter une ligne diagonale (pente = 1) en rouge, pour comparer la régression linéaire avec une relation proportionnelle entre les variables. La fonction text() permet d’ajouter les noms des axes, en précisant les coordonnées (x et y), le texte (labels), la position (pos), la couleur (col) et la taille (cex) du texte. La fonction legend() permet d’ajouter une légende, en précisant la position (topright), le texte (legend), la couleur (col) et l’épaisseur (lwd) des lignes. Les fonctions cor() et cat() permettent de calculer et d’afficher le coefficient de corrélation entre les variables transformées.



**Question 10 :**

# ---- Question 10 ----  
# Ouvrir le périphérique graphique pour le format PNG  
*png*("question10\_plot.png", width = 900, height = 700)  
  
# Afficher le nuage de points avec les données transformées  
*plot*(transformed\_IBE, transformed\_IP, main="Nuage de points entre IBE et IP (Transformées logarithmiques)", xlab="log(IBE)", ylab="log(IP)", col=*c*("purple", "orange"), pch=16)  
  
# Ajouter une régression linéaire pour illustrer la tendance  
*abline*(*lm*(transformed\_IP ~ transformed\_IBE), col="red", lwd=2)  
  
# Ajouter une ligne diagonale (pente = 1) en rouge  
*abline*(a = 0, b = 1, col = "red", lwd = 2, lty = 2)  
  
# Ajouter les noms des axes  
*text*(x = *min*(transformed\_IBE), y = *max*(transformed\_IP), labels = "log(IBE)", pos = 4, col = "purple", cex = 1.2)  
*text*(x = *max*(transformed\_IBE), y = *min*(transformed\_IP), labels = "log(IP)", pos = 2, col = "orange", cex = 1.2)  
  
# Ajouter une légende avec position manuelle  
*legend*("topright", legend=*c*("log(IBE)", "log(IP)", "Régression linéaire", "Diagonale"), col=*c*("purple", "orange", "red", "red"), lwd=2)  
  
# Calculer et afficher la corrélation pour les données transformées  
correlation\_coefficient\_transformed = *cor*(transformed\_IBE, transformed\_IP)  
*cat*("Coefficient de corrélation entre log(IBE) et log(IP):", correlation\_coefficient\_transformed, "\n")  
  
# Fermer le périphérique graphique et enregistrer le fichier PNG  
*dev.off*()

La question 10 ouvre et ferme un périphérique graphique pour enregistrer le nuage de points au format PNG. La fonction png() permet d’ouvrir le périphérique graphique, en précisant le nom du fichier (question10\_plot.png), la largeur (width) et la hauteur (height) de l’image. La fonction dev.off() permet de fermer le périphérique graphique et d’enregistrer le fichier PNG.

Une image contenant capture d’écran, texte, ligne, diagramme

Description générée automatiquement

Problème 2 :

**Question 1 :**

# ---- Question 1----  
  
# Charger les données  
data\_2012 = *read.table*("company2012.dat", header = *FALSE*)  
data\_2017 = *read.table*("company2017.dat", header = *FALSE*)  
data\_2022 = *read.table*("company2022.dat", header = *FALSE*)  
  
# Renommer les colonnes  
*colnames*(data\_2012) = *colnames*(data\_2017) = *colnames*(data\_2022) = "Performances"  
  
# Créer une liste de données  
data\_list = *list*(data\_2012, data\_2017, data\_2022)  
years = *c*(2012, 2017, 2022)  
  
# Afficher chaque histogramme séparément avec la densité de probabilité sur l'axe y  
for (i in *seq\_along*(data\_list)) {  
 # Créer un nouvel espace graphique  
 *par*(mfrow = *c*(1, 1))  
   
 # Créer un histogramme des performances avec densité sur l'axe y  
 *hist*(data\_list[[i]]$Performances, main = *paste*("Distribution des performances en", years[i]),  
 xlab = "Performances", ylab = "Density", col = "lightblue", border = "black", freq = *FALSE*)  
   
 # Ajouter une estimation par noyau de densité  
 *lines*(*density*(data\_list[[i]]$Performances), col = "red", lwd = 2)  
}

La première partie du code consiste à charger les données à partir de trois fichiers différents, nommés company2012.dat, company2017.dat et company2022.dat. Ces fichiers contiennent probablement des données sur les performances d’une entreprise sur différentes années. Le code utilise la fonction read.table pour lire les fichiers et les stocker dans des objets appelés data\_2012, data\_2017 et data\_2022. Le paramètre header = FALSE indique que les fichiers n’ont pas de ligne d’en-tête avec les noms des colonnes.

La deuxième partie du code consiste à renommer les colonnes des trois objets créés précédemment. Le code utilise la fonction colnames pour attribuer le nom “Performances” à la seule colonne de chaque objet. Le code utilise l’opérateur = pour assigner le même nom aux trois objets en une seule ligne.

La troisième partie du code consiste à créer une liste de données contenant les trois objets créés précédemment. Le code utilise la fonction list pour regrouper les objets dans un seul objet appelé data\_list. Le code crée également un vecteur nommé years contenant les années correspondant aux données, à savoir 2012, 2017 et 2022. Le code utilise la fonction c pour concaténer les valeurs dans un vecteur.

La quatrième partie du code consiste à afficher chaque histogramme séparément avec la densité de probabilité sur l’axe y. Le code utilise une boucle for pour parcourir les éléments de la liste de données, en utilisant la variable i comme indice. Le code utilise la fonction seq\_along pour générer une séquence d’indices allant de 1 à la longueur de la liste de données. Pour chaque itération de la boucle, le code fait les actions suivantes :

Le code utilise la fonction par pour créer un nouvel espace graphique avec une seule ligne et une seule colonne, en utilisant le paramètre mfrow = c(1, 1). Cela permet d’afficher un seul histogramme à la fois.

Le code utilise la fonction hist pour créer un histogramme des performances pour l’année correspondante, en utilisant le paramètre freq = FALSE pour afficher la densité au lieu de la fréquence sur l’axe y. Le code utilise également les paramètres main, xlab, ylab, col et border pour personnaliser le titre, les étiquettes des axes, la couleur et le contour de l’histogramme. Le code utilise la fonction paste pour concaténer le texte “Distribution des performances en” avec l’année correspondante, en utilisant la variable years[i] pour accéder à la valeur du vecteur years à l’indice i.

Le code utilise la fonction lines pour ajouter une estimation par noyau de densité à l’histogramme, en utilisant la fonction density pour calculer la densité des performances pour l’année correspondante, en utilisant le paramètre data\_list[[i]]$Performances pour accéder à la colonne “Performances” de l’objet de la liste de données à l’indice i. Le code utilise également les paramètres col et lwd pour personnaliser la couleur et l’épaisseur de la courbe de densité.

Ce qui renvoie :

Une image contenant diagramme, Tracé, ligne, texte

Description générée automatiquement

Une image contenant diagramme, Tracé, ligne, texte

Description générée automatiquement

Une image contenant diagramme, Tracé, ligne, pente

Description générée automatiquement

**Question 2 :**

# ---- Question 2 ----  
  
# Le test de Shapiro-Wilk pour chaque ensemble de données  
for (i in *seq\_along*(data\_list)) {  
 result <- *shapiro.test*(data\_list[[i]]$Performances)  
   
 # Afficher le résultat du test  
 *cat*("Shapiro-Wilk test for", years[i], ":", "\n")  
 *cat*("w (Test statistic):", *round*(result$statistic, 4), "\n")  
 *cat*("p-value:", *round*(result$p.value, 4), "\n")  
   
 # Interprétation du résultat  
 if (result$p.value > 0.05) {  
 *cat*("Pas de preuve significative pour rejeter l'hypothèse nulle (distribution normale).\n\n")  
 } else {  
 *cat*("Preuve significative pour rejeter l'hypothèse nulle (non-distribution normale).\n\n")  
 }  
}

Cette partie du code utilise le test de Shapiro-Wilk pour tester si les performances de chaque année suivent une distribution normale. Le test de Shapiro-Wilk est un test statistique qui compare les données observées à une distribution normale théorique, en utilisant une mesure appelée w. Plus w est proche de 1, plus les données sont proches de la normalité. Le test produit également une valeur de p, qui représente la probabilité d’obtenir un résultat aussi extrême que celui observé si l’hypothèse nulle (distribution normale) était vraie. Plus la valeur de p est petite, plus il y a de preuves pour rejeter l’hypothèse nulle et conclure que les données ne suivent pas une distribution normale.

Le code utilise une boucle for pour appliquer le test de Shapiro-Wilk à chaque ensemble de données, en utilisant la même variable i que dans la partie précédente. Pour chaque itération de la boucle, le code fait les actions suivantes :

Le code utilise la fonction shapiro.test pour effectuer le test de Shapiro-Wilk sur la colonne “Performances” de l’objet de la liste de données à l’indice i, en utilisant le même paramètre que dans la partie précédente. Le code stocke le résultat du test dans un objet appelé result.

Le code utilise la fonction cat pour afficher le résultat du test, en utilisant le paramètre years[i] pour indiquer l’année correspondante. Le code utilise également la fonction round pour arrondir le résultat à quatre décimales, et le paramètre "\n" pour créer des sauts de ligne. Le code affiche la valeur de w (statistique du test) et la valeur de p, en utilisant le symbole $ pour accéder aux éléments de l’objet result.

Le code utilise une condition if pour interpréter le résultat du test, en utilisant la valeur de p comme critère. Le code utilise le paramètre 0.05 comme seuil de signification, ce qui signifie que si la valeur de p est supérieure à 0.05, on ne peut pas rejeter l’hypothèse nulle avec un niveau de confiance de 95%. Le code utilise la fonction cat pour afficher l’interprétation du résultat, en utilisant des phrases conditionnelles pour exprimer le degré de certitude. Le code utilise également le paramètre "\n\n" pour créer des espaces entre les résultats.

On a donc en console :

Shapiro-Wilk test for 2012 :

w (Test statistic): 0.9978

p-value: 0.639

Pas de preuve significative pour rejeter l'hypothèse nulle (distribution normale).

Shapiro-Wilk test for 2017 :

w (Test statistic): 0.9978

p-value: 0.639

Pas de preuve significative pour rejeter l'hypothèse nulle (distribution normale).

Shapiro-Wilk test for 2022 :

w (Test statistic): 0.9978

p-value: 0.639

Pas de preuve significative pour rejeter l'hypothèse nulle (distribution normale).