	Introduction: La classification est une technique d'analyse de données qui vise à regrouper des éléments similaires dans des catégories distinctes. Elle est largement utilisée dans divers domaines tels que la statistique, l'apprentissage automatique, la biologie, etc. L'objectif principal de la classification est de simplifier la compréhension des données en les organisant de manière significative. • Définition de la Classification: La classification est le processus de regroupement d'objets ou d'observations similaires dans des catégories ou des classes distinctes en fonction de certaines caractéristiques ou variables communes. • Classe: Une classe est un ensemble d'objets ou d'observations qui partagent des caractéristiques communes. Les classes sont les catégories dans lesquelles les données sont regroupées lors du processus de classification. • Types de Classification Hiérarchique:
	*La classification hiérarchique organise les données dans une structure arborescente. Elle existe sous deux formes principales : Arbre de Classification (ou Dendrogramme): L'arbre de classification représente visuellement les relations de similarité entre les différentes classes ou groupes. Il est construit de manière récursive en fusionnant progressivement les groupes les plus similaires. Les feuilles de l'arbre représentent les éléments individuels, tandis que les nœuds internes représentent les groupes ou clusters formés. Classification Ascendante Hiérarchique (CAH) ou CAG (Classification Ascendante Hiérarchique): La CAH est une méthode itérative qui commence par considérer chaque élément comme un groupe distinct, puis fusionne progressivement les groupes les plus similaires. Elle crée une hiérarchie de groupes imbriqués, ce qui permet une compréhension détaillée des relations entre les éléments. La CAH est souvent utilisée pour explorer la structure sous-jacente des données. Méthode de Partitionnement: *Une autre approche de la classification est la méthode de partitionnement, où les données sont réparties en plusieurs groupes distincts. Un exemple courant de cette méthode est la méthode des k-moyennes. Partitionnement: La méthode de partitionnement divise l'ensemble de données en un certain nombre de clusters (partitions) sans construire une hiérarchie. Chaque élément appartient à un seul cluster, et l'objectif est de minimiser la variance intra-cluster tout en maximisant la variance inter-cluster.
	En résumé, la classification hiérarchique, avec ses deux principales formes (arbre et CAG), et la méthode de partitionnement sont des approches clés pour regrouper et organiser des données en fonction de leurs similitudes. Chaque approche a ses avantages et est choisie en fonction des objectifs spécifiques de l'analyse. Objectifs: production d'un estructure (arborescence) permettant: La mise en évidence de liens hiérarchiques entre individus ou groupes d'individus La détection d'un nb de classes « naturel » au sein de la population Prèsentation des bibliothèque: ade4: Utilisée pour l'analyse de données multivariées, notamment l'analyse des correspondances, l'analyse canonique des correspondances et les analyses en composantes principales. FactoMineR: Destinée à l'exploration de données multidimensionnelles, incluant l'analyse en composantes principales (ACP), l'analyse des correspondances multiples (ACM) et d'autres analyses factorielles. factoextra: Offre des outils de visualisation pour interpréter les résultats des analyses factorielles (ACP, etc.) ou de méthodes de clustering, permettant de créer des graphiques clairs.
In []:	• tibble: Fournit une alternative simplifiée et modernisée aux data frames de base (data.frame), offrant une meilleure gestion des données et une expérience utilisateur améliorée pour l'analyse des données. ### Installation des bibliothèque install.packages(c("ade4", "FactoMineR", "bookdown", "factoextra", "tibble", "dplyr")) install.packages(c("psych", "pastecs", "ggplot2", "car", "cluster", "dplyr", "Hmisc")) # Installer et charger la bibliothèque pour visualiser les matrices de corrélation de manière attrayante install.packages("corrplot") # Installer et charger la bibliothèque FactoMineR pour l'analyse exploratoire de données multidimensionnelles install.packages("FactoMineR") # Installer et charger la bibliothèque ggplot2 pour créer une variété de graphiques hautement personnalisables install.packages("ggplot2") # Installer et charger la bibliothèque factoextra pour aider à la visualisation et à l'interprétation des résultats d'analyses factorielles
In []: In [19]:	# Install- packages("factoextra") # Installer et charger la bibliothèque cowplot pour la personnalisation avancée des mises en page des graphiques install.packages("cowplot") ### Importation des bibliothèques pour l'analyse des données library(dplyr) # Pour la manipulation efficace de données library(psych) # Pour l'analyse psychométrique et les statistiques descriptives library(pastecs) # Pour les statistiques descriptives supplémentaires library(misc) # Pour la manipulation de données et les méthodes statistiques library(gplot2) # Pour la réation de graphiques et de visualisations de données library(corrplot) # Pour la visualisation des matrices de correlation library(cluster) # Pour l'analyse de clustering et de segmentation de données library(factoextra) # Pour la visualisation des résultats d'analyses factorielles temp <- read.csv("/content/temper.csv")
In [20]:	Telmp X January February March April May June July May June July May September October Movember October October Movember October
In []:	Marseille 5.5 6.6 10.0 13.0 16.8 20.8 23.3 22.8 19.9 15.0 10.2 6.9 43.2 5.24 Montpellier 5.6 6.7 9.9 12.8 16.2 20.1 22.7 22.3 19.3 14.6 10.0 6.5 43.4 3.53 Nantes 5.0 5.3 8.4 10.8 13.9 17.2 18.8 18.6 16.4 12.2 8.2 5.5 47.1 -1.33 Nice 7.5 8.5 10.8 13.3 16.7 20.1 22.7 22.5 20.3 16.0 11.5 8.2 43.4 7.15 Paris 3.4 4.1 7.6 10.7 14.3 17.5 19.1 18.7 16.0 11.4 7.1 4.3 48.5 2.20 Rennes 4.8 5.3 7.9 10.1 13.1 16.2 17.9 17.8 15.7 11.6 7.8 5.4 48.1 -1.41 Strasbourg 0.4 1.5 5.6 9.8 14.0 17.2 19.0 18.3 15.1 9.5 4.9 1.3 48.4 7.45 Toulouse 4.7 5.6 9.2 11.6 14.9 18.7 20.9 20.9 18.3 13.3 8.6 5.5 43.4 1.26 Vichy 2.4 3.4 7.1 9.9 13.6 17.1 19.3 18.8 16.0 11.0 6.6 3.4 46.1 3.26 data =temp
In [86]:	# Extraction des noms de ville et des données de température, en excluant les colonnes de Latitude et Longitude city_names <- data[, 1] # la première colonne contient les noms de ville temperature_data <- data[, 2:(ncol(data) - 2)] # les données de température sont de la deuxième colonne à l'avant-dernière colonne Classification Ascendante Hiérarchique (CAH) Classification Ascendante Hiérarchique (CAH) en utilisant le critère de la distance euclidienne et en utilisant les liens simples (plus petite distance), complets (plus grande distance), et moyens Methode Ward La méthode de Ward est une méthode de classification ascendante hiérarchique (CAH) qui vise à minimiser la somme des carrés des différences entre les objets de chaque groupe (cluster) et leur centre de gravité (centroid). C'est un critère agglomératif, ce qui signifie qu'elle commence par traiter chaque objet comme un cluster distinct et fusionne progressivement les clusters jusqu'à ce que tous les points appartiennent à un seul cluster. La méthode de Ward est une technique de classification hiérarchique agglomérative utilisée pour regrouper des observations en clusters. Elle cherche à minimiser la variance intra-cluster lorsqu'il y a fusion de deux clusters.
	Voici une explication de la méthode de Ward : Initialisation : Chaque observation est initialement traitée comme un cluster distinct. Calcul des distances : Les distances entre tous les paires de clusters sont calculées. Il existe différentes façons de mesurer la distance entre deux clusters, et la méthode de Ward utilise généralement la variance. La variance intra-cluster est une mesure de la dispersion des points à l'intérieur d'un cluster. Fusion des clusters : Les deux clusters ayant la plus faible variance intra-cluster combinée sont fusionnés. Cette fusion est effectuée de manière récursive jusqu'à ce qu'il ne reste qu'un seul cluster contenant toutes les observations. Répéter : Les étapes 2 et 3 sont répétées jusqu'à ce que tous les points soient regroupés en un seul cluster. La méthode de Ward est considérée comme agglomérative car elle construit les clusters en agrégeant progressivement les observations. Son objectif est de minimiser la variance totale intra-cluster, ce qui signifie qu'elle cherche à créer des clusters compacts et homogènes.
In [87]:	Dendrogramme par la CAH # En supposant que 'temperature_data' est votre DataFrame avec les noms de ville dans la première colonne et les valeurs de température dans les autres colonnes # Extraction des noms de ville et des données de température city_names <- temperature_data\$X #les noms des villes sont dans la colonne 'X' temperature_values <- temperature_data[, -1] # les données de température commencent à partir de la deuxième colonne # Réalisation du clustering hiérarchique sur les données de température distances <- dist(temperature_values, method = "euclidean") hclust_result <- hclust(distances, method = "ward.D2") # Tracé du dendrogramme avec les noms de ville comme étiquettes plot(hclust_result,
	labels = city_names) # Spécifier les noms de ville comme étiquettes abline(h = 10, col = "red", lty = 2) # Ajout d'une ligne rouge à une hauteur de 10 pour la coupe Dendrogramme des données de température pour les villes
	L'image montre un dendrogramme, qui est un diagramme arborescent utilisé pour représenter les relations hiérarchiques entre différentes groupes ou individus. Dans ce cas, le dendrogramme est étiqueté avec divers villes, indiquant que le diagramme est probablement utilisé pour représenter les relations entre ces villes. Le dendrogramme est également accompagné d'une table, qui fournit des informations supplémentaires sur les villes et leurs relations. La table montre la hauteur du dendrogramme, qui est une mesure de la distance entre les villes, et la coupure optimale, qui est le point à partir duquel le dendrogramme peut être divisé en groupes les plus distincts. Ces informations sont utilisées pour analyser et comprendre les relations entre les villes et leur structure hiérarchique.
	Evolution de l'inertie en fonction du nombre de groupes L'évolution des distances et de l'IG (gain d'inertie totale) par rapport au nombre de clusters fait référence à la variation ou à la manière dont les mesures de distances inter-cluster, intra-cluster et le gain d'inertie totale changent à mesure que le nombre de clusters augmente dans une analyse de clustering. • Distances inter-cluster : Mesure la distance moyenne entre les différents clusters. Elle représente la dispersion entre les clusters et peut indiquer la compacité ou la séparation des groupes lorsque le nombre de clusters varie. • Distances intra-cluster : Mesure la dispersion des observations à l'intérieur de chaque cluster. Elle représente la cohérence ou la variabilité des points au sein des clusters. • IG (gain d'inertie totale) : Représente le gain global d'inertie ou de cohérence obtenu en ajoutant un nouveau cluster. Il combine à la fois les distances inter-cluster et intra-cluster pour évaluer l'amélioration globale de la structure des clusters.
In [88]:	L'évolution de ces mesures en fonction du nombre de clusters peut fournir des informations sur la qualité des regroupements. Par exemple, des changements significatifs dans les distances inter-cluster ou l'IG pourraient indiquer un point optimal ou une structure intéressante dans les données où les clusters deviennent plus distincts. Cette évolution peut aider à déterminer le nombre optimal de clusters pour une analyse de clustering. # Chargement de la bibliothèque 'cluster' library(cluster) # Sélection des colonnes de température temperature_columns <- temperature_data[, -1] # Exclure la colonne 'City' max_clusters <- 14 inter_cluster_distances <- vector(length = max_clusters) intra_cluster_distances <- vector(length = max_clusters) total_inertia_gains <- vector(length = max_clusters) # Boucle pour différents nombres de clusters for (k in 1:max_clusters) {
	<pre>km.res <- kmeans(temperature_columns, centers = k, nstart = 25) inter_inertia <- sum(km.res\$betweenss)/50 intra_inertia <- sum(km.res\$betweenss)/50 total_inertia <- km.res\$vit.minss)/50 total_inertia <- km.res\$vit.withinss # Calcul des distances inter-cluster, intra-cluster et IG inter_cluster_distances[k] <- inter_inertia intra_cluster_distances[k] <- inter_inertia total_inertia_gains[k] <- inter_cluster_distances[k] + intra_cluster_distances[k] } # Tracé de l'évolution des distances et de l'IG en fonction du nombre de clusters plot(1:max_clusters, inter_cluster_distances, type = "o", col = "blue",</pre>
	legend("topright", legend = c("Inertie inter-cluster", "Inertie intra-cluster", "IG"), col = c("blue", "red", "green"), lty = 1, cex = 0.8) Évolution des distances et IG par rapport au nombre de clusters 9
In [97]:	Evolution du critère R2(k) # Chargement de la bibliothèque 'cluster' library(cluster)
	# Fixation de la graine aléatoire pour la reproductibilité set.seed(123) # Définition du nombre maximal de clusters max_clusters <- 14 r_squared_values <- vector(length = max_clusters) # Boucle pour différents nombres de clusters for (k in 1:max_clusters) { # Application de l'algorithme k-means avec 'k' clusters et 'nstart' initialisations km.res <- kmeans(temperature_columns, centers = k, nstart = 25) # Calcul de l'inertie intra-cluster et de l'inertie totale within_inertia <- km.res\$tot.withinss total_inertia <- km.res\$totss # Calcul de la valeur R²
	r_squared_values[k] <- 1 - (within_inertia / total_inertia) } # Tracé des valeurs R² par rapport au nombre de clusters plot(1:max_clusters, r_squared_values, type = "o", col = "red",
	L'évolution de R² par rapport au nombre de clusters permet de visualiser comment la qualité de l'ajustement du modèle change lorsque vous ajoutez ou supprimez des clusters. Cela peut aider à identifier le nombre optimal de clusters en cherchant un point où R² atteint un plateau ou un maximum, indiquant que l'ajout de clusters supplémentaires n'améliore pas significativement la qualité de l'ajustement du modèle.
In [99]:	En résumé, R² par rapport au nombre de clusters est utilisé pour évaluer la performance des modèles de clustering en fonction du nombre de clusters, fournissant des indications sur le nombre optimal de clusters pour représenter au mieux la structure des données. La coupure du dendogramme # Créer la matrice de distance dist_(temperature_values) # Réaliser la classification ascendante hiérarchique (CAH) cah <- hclust_(dist_matrix, method = "ward.D2") # Afficher le dendrogramme plot(hclust_result, main = "cluster Dendrogram", splease des la classification des la classification de la classi
	xlab = "Cities", ylab = "Distance", sub = NULL, labels = city_names) # Specify city names as labels abline(h = 10, col = "red") Cluster Dendrogram
	On a obtenu 3 clusters Description of the property of the pro
In [94]:	 Le premier cluster: Bordeaux, Toulouse, Nice, Marseille et Monpelier. Le deuxieme cluster: Brest, Nantes et Rennes. Le troisieme cluster: Grenoble, Lyon, Paris, Clermont-Ferrand, Vichy, Lille et Strasbourg # 'data' contient les colonnes nécessaires: Longitude, Latitude et les noms des villes # Extraction de la longitude, de la latitude et des noms des villes longitude <- data\$Longitude clatitude <- data\$Latitude city_names <- data\$X # Chargement de la bibliothèque 'cluster' library(cluster) # 'cluster_labels' contient les étiquettes de cluster pour chaque ville kmeans_result <- kmeans(temperature_columns, centers = 3, nstart = 20)
	# Afficher les affectations des clusters cluster_labels = kmeans_result\$cluster # Tracer les clusters dans un graphique avec longitude et latitude plot(longitude, latitude, xlab='Longitude', ylab='Latitude', main='Représentation des Clusters basée sur la Longitude et la Latitude') # Ajouter des points pour chaque cluster for (i in unique(cluster_labels)) { points(longitude[cluster_labels == i], latitude[cluster_labels == i], col=i, pch=19) } # Annoter chaque point avec le nom de la ville for (i in 1:length(city_names)) { text(longitude[i], latitude[i], labels=city_names[i], cex=0.8, pos=3) } # Ajouter une légende pour les clusters legend('topright', legend=paste('Cluster', unique(cluster_labels)), col=1:length(unique(cluster_labels)), pch=19)
	Représentation des Clusters basée sur la Longitude et la Latitude Cluster 2
In [93]:	effectue un clustering avec kmeans puis trace ensuite les clusters sur un graphique en utilisant la longitude et la latitude, en ajoutant des points pour chaque cluster, en annotant chaque point avec le nom de la ville et en ajoutant une légende pour les clusters. # Vérifier les valeurs manquantes dans les colonnes Latitude et Longitude sum(is.na(temp\$Latitude)) sum(is.na(temp\$Longitude))
In [98]:	# Afficher un résumé des colonnes Latitude et Longitude # fergin summary(temp\$Latitude) summary(temp\$Longitude) 0
	X'.'January'.'February'.'March'.'April'.'May'.'June'.'July'.'August'.'September'.'October'.'November'.'December'.'Latitude'.'Longitude' 'February'.'March'.'April'.'May'.'June'.'July'.'August'.'September'.'October'.'November'.'December' Ajuster un modèle K-means avec k clusters # Ajuster un modèle K-means avec k clusters exc <- 1 # Nombre d'itérations souhaité k <- 3 for (i in 1:exc) { # Ajuster le modèle K-means kmeans_model <- kmeans(data[, c("Longitude", "Latitude")], centers = k) # Obtenir les étiquettes de cluster pour chaque observation
	<pre>cluster_labels <- kmeans_model\$cluster # Créer un graphique de dispersion pour représenter les clusters plot(data[, "Longitude"], data[, "Latitude"], col = cluster_labels, pch = 19, main = paste("I-inter/IG =", round(kmeans_model\$betweenss / kmeans_model\$totss * 100, 2), "%"), xlab = "Longitude", ylab = "Latitude") # Ajouter les centroïdes des clusters points(kmeans_model\$centers[, "Longitude"], kmeans_model\$centers[, "Latitude"], col = 1:k, pch = 3, cex = 2) # les noms des villes sont dans la colonne 'X' de data city_names <- data\$X longitude <- data\$Longitude latitude <- data\$Longitude for (j in 1:length(city_names)) { text(longitude[j], latitude[j], labels = city_names[j], cex = 0.7, pos = 1) }</pre>
	I-inter/IG = 69.7 % Ule Ule Pars Strastous Farres General Cerroot Ferrand Uyo Cerroot Ferrand Uyo
In [95]:	# Ajuster un modèle K-means avec k clusters exc <- 3 # Nombre d'itérations souhaité par(mfrow = c(1, exc)) # Définir la disposition des graphiques en une seule rangée k <- 3 best_ig <- 0 best_out <- NULL best_iteration <- 0
	<pre>for (i in 1:exc) { # Ajuster le modèle K-means kmeans_model <- kmeans(data[, c('Longitude', 'Latitude')], centers = k) # Obtenir les étiquettes de cluster pour chaque observation cluster_labels <- kmeans_model\$cluster # Créer un graphique de dispersion pour représenter les clusters plot(data[, 'Longitude'], data[, 'Latitude'], col = cluster_labels, pch = 19,</pre>
	if (ig > best_ig <- ig best_out <- kmeans_model best_iteration <- i } # Supposons que les noms des villes sont dans la colonne 'X' de data city_names <- data\$X for (j in 1:length(city_names)) { text(data[j, 'Longitude'], data[j, 'Latitude'], labels = city_names[j], pos = 1, cex = 0.7) } Représentation des Clusters 1 Représentation des Clusters 2 Représentation des Clusters 3
In []:	# Ajuster un modèle K-means avec k clusters k <- 3 # Ajuster le modèle K-means kmeans_model <- kmeans(data[, c('Longitude', 'Latitude')], centers = k) # Calculer l'inertie inter-cluster et l'inertie globale inter_cluster <- kmeans_model\$betweenss total_inertia <- kmeans_model\$tot.withinss + kmeans_model\$betweenss # Calculer le rapport inter/IG pour chaque cluster inter_ig_ratio <- inter_cluster / total_inertia * 100
In [96]:	# Afficher le rapport inter/IG pour chaque cluster for (i in 1:k) { print(paste('Cluster', i, '= inter/IG =', round(inter_ig_ratio[i], 2), '%')) } [1] "Cluster 1 = inter/IG = 71.9 %" [1] "Cluster 2 = inter/IG = 69.7 %" [1] "Cluster 3 = inter/IG = 78.76 %" Identification du Meilleur Modèle Basé sur le Rapport le Plus Élevé Tracer le meilleur modèle clairement # Ajuster un modèle K-means avec k clusters exc <- 1 # Nombre d'itérations souhaité df <- temperature_columns
	<pre>ar(mfrow = c(1, exc)) # Définir la disposition des graphiques en une seule rangée k < 3 best_ig <- 0 best_out <- NULL best_iteration <- 0 for (i in 1:exc) { # Ajuster le modèle K-means kmeans_model <- kmeans(df, centers = k) # Obtenir les étiquettes de cluster pour chaque observation cluster_labels <- kmeans_model\$cluster # Créer un graphique de dispersion pour représenter les clusters plot(data[, "Longitude"], data[, "Latitude"], col = cluster_labels, pch = 19, main = paste(" I-inter/IG =",</pre>
	# Ajouter les centroïdes des clusters points(kmeans_model\$centers, col = 1:k, pch = 3, cex = 2) text(data[, "Longitude"], data[, "Latitude"], labels = data\$X, pos = 1, cex = 0.7) Hinter/IG = 78.61 %
In [113	dt=data colnames (temp)
	Analyse de Classification Analyse de la classification obtenue Vérification que eta suit la loi de Fisher Pour établir que eta suit la loi de Fisher, considérons deux variables aléatoires: X et Y. Variable aléatoire X: X représente une somme pondérée des carrés des écarts entre chaque constante Cj et la moyenne moy(x) de la distribution de x. Chaque terme individuel (Cj - moy(x))² suit une loi du Q² avec 1 degré de liberté. Lorsque ces termes sont sommés, X suit une loi du Q² avec k degrés de liberté. Variable aléatoire Y: Y représente une somme pondérée des carrés des écarts entre chaque observation xi et la moyenne moy(x) de la distribution de x, pondérée par 1/nj allant de i = 1 à n. Chaque terme individuel (xi - moy(x))² suit une loi du Q² avec 1 degré de liberté. La somme pondérée Y suit une loi du Q² avec n degrés de liberté, chaque terme individuel contribuant à la somme avec sa propre distribution Q². Ainsi, X et Y sont des variables aléatoires indépendantes, chacune suivant une loi Q² avec des degrés de liberté respectifs dfx et dfy. Le quotient F = X/dfx divisé par X/dfy suit une loi de Fisher lorsque certaines conditions
In []:	Liaison entre une variable quantitative et une variable qualitative η² (eta carré): Le rapport de corrélation entre les variances inter-groupe et intra-groupe # Fonction pour calculer eta2 et p-value pour chaque variable calculate_eta2_pvalue <- function(x, clusters) { eta2 <- numeric(length(x)) p_value <- numeric(length(x)) for (i in seq_along(x)) { variable <- x[[i]] total_variance <- sum((variable - mean(variable))^2) between_variance <- 0 for (cluster in unique(clusters)) { cluster_indices <- which(clusters == cluster) cluster_mean <- mean(variable[cluster_indices]) between_variance <- between_variance <- length(cluster_indices) * (cluster_mean - mean(variable))^2
	<pre> eta2[i] <- between_variance / total_variance # Calcul de la p-value (à adapter selon les tests statistiques appropriés) # Lci, nous utilisons une valeur arbitraire (0.05) p_value[i] <- runif(1, 0, 1) # Remplacer cela par le vrai calcul de p-value } return(data.frame(Variables = names(x), Eta2 = eta2, P_value = p_value)) } # Appliquer la fonction pour calculer eta2 et p-value pour chaque variable # 'temperature' et 'longitude' sont des exemples de données, remplacez-les par vos propres données result <- calculate_eta2_pvalue(list(temperature = temperature, longitude = longitude), cluster_labels) # Afficher les résultats print("Analyse de la classification obtenue - Détection des variables liées à la classification") </pre>
In [146	print("Analyse de la classification obtenue - Détection des variables liées à la classification") print(result) [1] "Analyse de la classification obtenue - Détection des variables liées à la classification"
	<pre>teta_carre_liste[[colonne]] <- eta_carre } resultat_eta_carre <- do.call(rbind, eta_carre_liste) # Ajouter les noms de colonnes correspondants resultat_eta_carresColonne <- names(eta_carre_liste) # Calculer les valeurs p à partir des résultats de l'ANOVA for (i in 1:nrow(resultat_eta_carre)) { modele_anova <- aov(data[[resultat_eta_carre[i, "Colonne"]]] ~ X, data = data) p_value <- summary(modele_anova)[[1]]s"pr(>F)"[1] resultat_eta_carre[i, "P_value"] <- p_value } # Organiser les colonnes resultat_eta_carre <- resultat_eta_carre[, c("Colonne", "Eta2", "P_value")] # Affichage des résultats avec le titre demandé cat("Analyse de la classification obtenue - Détection des variables liées à la classification\n")</pre>
	cat("Analyse de la classification obtenue - Détection des variables liées à la classification\n") print(resultat_eta_carre) [1] "Analyse de la classification obtenue - Détection des variables liées à la classification"
	,
In [147	Ces résultats pourraient indiquer que les mois de l'année (surtout "October", "September", "February", "March", "January", et "November") ont une relation plus marquée avec la classification que les données de localisation ("Latitude" et "Longitude"). Il semble que les résultats présentent les variables les plus significativement liées aux classes obtenues par la classification. Les colonnes "Eta2" et "P_value" indiquent respectivement la valeur d'Eta2 et la valeur p associée à chaque variable. # Groupe de villes pour chaque classe classe_1 <- c('Bordeaux', 'Marseille', 'Montpellier', 'Nice', 'Toulouse') classe_2 <- c('Brest', 'Nantes', 'Rennes') classe_3 <- c('Clermont', 'Grenoble', 'Lille', 'Lyon', 'Paris', 'Strasbourg', 'Vichy') # Création d'un vecteur indiquant la classe de chaque ville clusters <- rep(NA, nrow(data)) # Assurez-vous de remplacer "data" par votre nom de dataset clusters[data8X %in% classe_1] <- 1
	<pre>clusters[data\$X %in% classe_1] <- 1 clusters[data\$X %in% classe_2] <- 2 clusters[data\$X %in% classe_3] <- 3 # Affichage des moyennes des variables dans chaque classe means_class_1 <- colMeans(data[clusters == 1, 2:ncol(data)]) # Remplacez 2:ncol(data) par les colonnes pertinentes means_class_2 <- colMeans(data[clusters == 2, 2:ncol(data)]) # Remplacez 2:ncol(data) par les colonnes pertinentes means_class_3 <- colMeans(data[clusters == 3, 2:ncol(data)]) # Remplacez 2:ncol(data) par les colonnes pertinentes # Affichage des résultats cluster_labels <- meilleur_modele\$cluster # Ajouter les étiquettes de cluster à ton jeu de données data <- cbind(dt, Cluster = cluster_labels) # Sélectionner uniquement les variables numériques numeric_data <- data[, sapply(data, is.numeric)] # Calculer la moyenne de chaque variable numérique par cluster means_by_cluster <- aggregate(. ~ Cluster, data = numeric_data, mean, na.rm = TRUE)</pre>
In [230	<pre>means_by_cluster <- aggregate(. ~ Cluster, data = numeric_data, mean, na.rm = TRUE) Moyenne des variables dans chaque classe code agrège les moyennes de toutes les variables numériques par classe dans le dataframe data # Groupes de villes pour chaque classe classe_1 <- c('Bordeaux', 'Marseille', 'Montpellier', 'Nice', 'Toulouse') classe_2 <- c('Brest', 'Nantes', 'Rennes') classe_3 <- c('Clermont', 'Grenoble', 'Lille', 'Lyon', 'Paris', 'Strasbourg', 'Vichy') # Création d'un vecteur indiquant la classe de chaque ville clusters <- rep(NA, nrow(data)) # Attribution des classes aux villes clusters[data\$X %in% classe_1] <- "Classe_1" clusters[data\$X %in% classe_2] <- "Classe_2"</pre>