Regression_logistique

Tchouenkou David Nathanäel

2024-05-22

#ETAPE 1: Définition du problème -Développer un modèle qui permet de prédire la présence d'une maladie cardiarques chez les patients

Quels sont les principaux facteurs prédictifs d'une maladie cardiaque?

#Etape 2: Collecte des données

Source: Il s'agit jeu de données sur les maladies cardiaques.Vous pouvez accéder à ces données en visitant le site de l'UCI dédié au machine learning et en cherchant le jeu de données en question: http://archive.ics.uci.edu/ml/datasets/Heart+Disease (http://archive.ics.uci.edu/ml/datasets/Heart+Disease). La base est aussi disponible sur Kaggle en cliquant sur le lien suivant: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction (https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction)

```
url <- "https://raw.githubusercontent.com/davetchouen/Logistic_Regression-/main/heart.csv"
data <- read.csv(url, header=TRUE)
```

L'analyse va se dérouler comme suit:

Etape 3: prétraitement des données

Etape 4: Sélection des Caractéristiques pertinentes

Etape 5: Partitionnement des données

Etape 6: Entrainement

Etape 7: Interprétation des résultats

Etape 8: Evaluation et prédiction

PACKAGES NECESAIRE

```
library(ggplot2)
library(cowplot)
```

```
head(data) # you see data, but no column names
```

```
##   Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR
## 1  40   M           ATA       140         289         0     Normal   172
## 2  49   F           NAP       160         180         0     Normal   156
## 3  37   M           ATA       130         283         0         ST    98
## 4  48   F           ASY       138         214         0     Normal   108
## 5  54   M           NAP       150         195         0     Normal   122
## 6  39   M           NAP       120         339         0     Normal   170
##   ExerciseAngina Oldpeak ST_Slope HeartDisease
## 1              N     0.0       Up            0
## 2              N     1.0     Flat            1
## 3              N     0.0       Up            0
## 4              Y     1.5     Flat            1
## 5              N     0.0       Up            0
## 6              N     0.0       Up            0
```

```
colnames(data) <- c(
  "Age",             # Âge
  "Sexe",            # Sexe
  "TypeDouleurThoracique",  # Type de Douleur Thoracique
  "TensionRepos",    # Tension au Repos
  "Cholesterol",     # Cholestérol
  "GlycemieAJeun",   # Glycémie à Jeun
  "ECGRepos",        # Electrocardiogramme au repos
  "FreqCardiaqueMax",# Fréquence Cardiaque Maximale
  "AngineExercice",  # Angine d'Exercice
  "DepressionST",    # Dépression ST
  "PenteST",         # Pente ST (aspect de l'électrocardiogramme)
  "MaladieCardiaque" # Maladie Cardiaque
)
```

Afficher la structure des données pour vérifier les nouveaux noms de colonnes

```
str(data)
```

```
## 'data.frame':    918 obs. of  12 variables:
##  $ Age                 : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ Sexe                : chr  "M" "F" "M" "F" ...
##  $ TypeDouleurThoracique: chr  "ATA" "NAP" "ATA" "ASY" ...
##  $ TensionRepos        : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol         : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ GlycemieAJeun       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ECGRepos            : chr  "Normal" "Normal" "ST" "Normal" ...
##  $ FreqCardiaqueMax    : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ AngineExercice      : chr  "N" "N" "N" "Y" ...
##  $ DepressionST        : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ PenteST             : chr  "Up" "Flat" "Up" "Flat" ...
##  $ MaladieCardiaque    : int  0 1 0 1 0 0 0 0 1 0 ...
```

Convertir la variable ‘MaladieCardiaque’ en facteur

```
data$MaladieCardiaque <- factor(data$MaladieCardiaque)
```

Etape 3: prétraitement des données

Ici il faut analyser la distribution des variables, analyser les valeurs manquantes, analyser les valeurs aberrantes etc.

Analyse exploratoire

Analyse des valeurs manquantes

```
summary(data) # Résumé basique pour voir les valeurs manquantes
```

```
##       Age            Sexe           TypeDouleurThoracique  TensionRepos
##  Min.   :28.00   Length:918         Length:918            Min.   :  0.0
##  1st Qu.:47.00   Class :character   Class :character      1st Qu.:120.0
##  Median :54.00   Mode  :character   Mode  :character      Median :130.0
##  Mean   :53.51                                            Mean   :132.4
##  3rd Qu.:60.00                                            3rd Qu.:140.0
##  Max.   :77.00                                            Max.   :200.0
##   Cholesterol    GlycemieAJeun      ECGRepos         FreqCardiaqueMax
##  Min.   :  0.0   Min.   :0.0000   Length:918         Min.   : 60.0
##  1st Qu.:173.2   1st Qu.:0.0000   Class :character   1st Qu.:120.0
##  Median :223.0   Median :0.0000   Mode  :character   Median :138.0
##  Mean   :198.8   Mean   :0.2331                      Mean   :136.8
##  3rd Qu.:267.0   3rd Qu.:0.0000                      3rd Qu.:156.0
##  Max.   :603.0   Max.   :1.0000                      Max.   :202.0
##  AngineExercice      DepressionST       PenteST          MaladieCardiaque
##  Length:918         Min.   :-2.6000   Length:918         0:410
##  Class :character   1st Qu.: 0.0000   Class :character   1:508
##  Mode  :character   Median : 0.6000   Mode  :character
##                     Mean   : 0.8874
##                     3rd Qu.: 1.5000
##                     Max.   : 6.2000
```

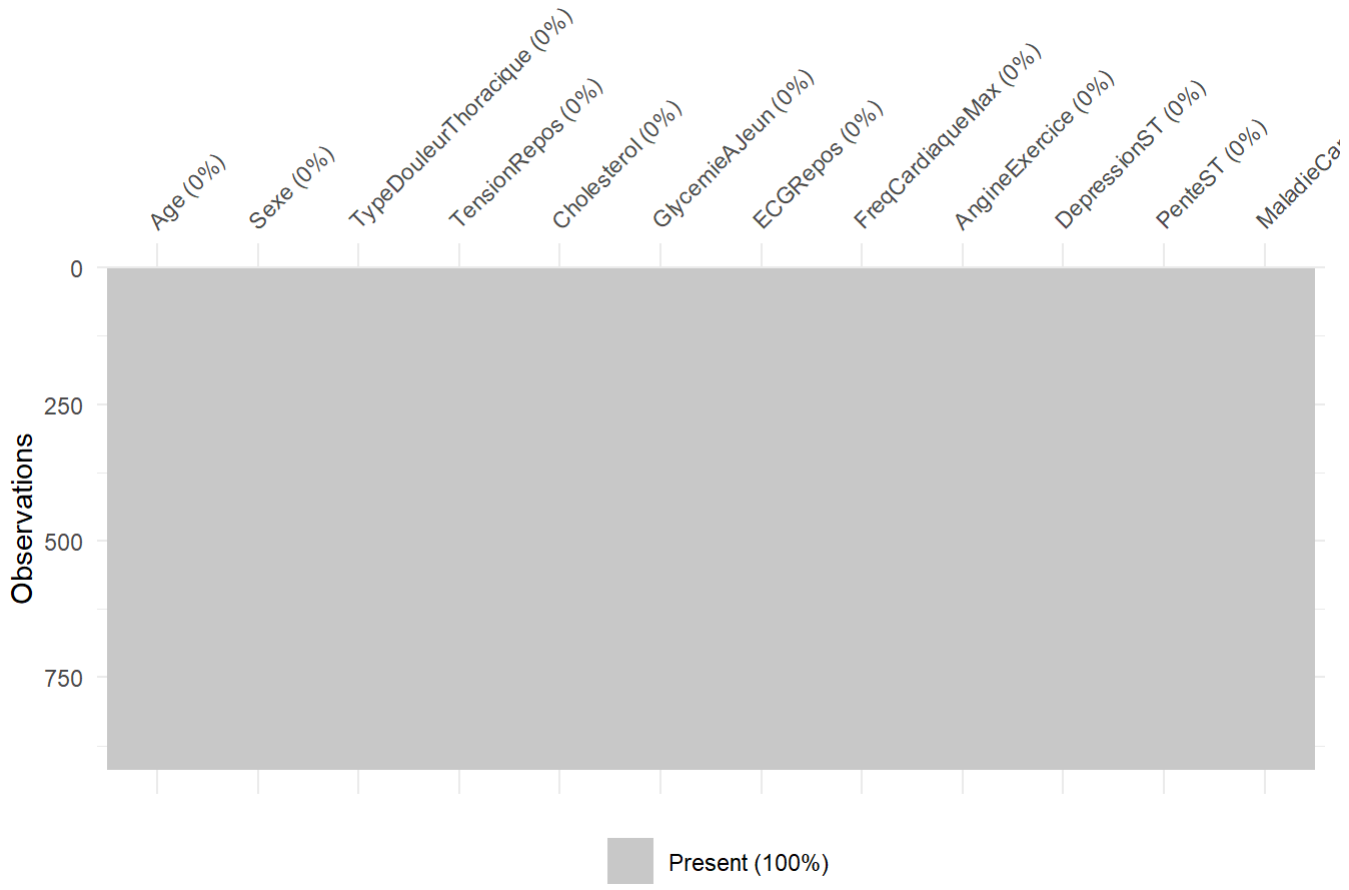
Analyse des valeurs manquantes

```
# Charger la bibliothèque pour la gestion des données manquantes
if (!require("naniar")) {
  # Si elle n'est pas installée, installer la bibliothèque
  install.packages("naniar")
}
```

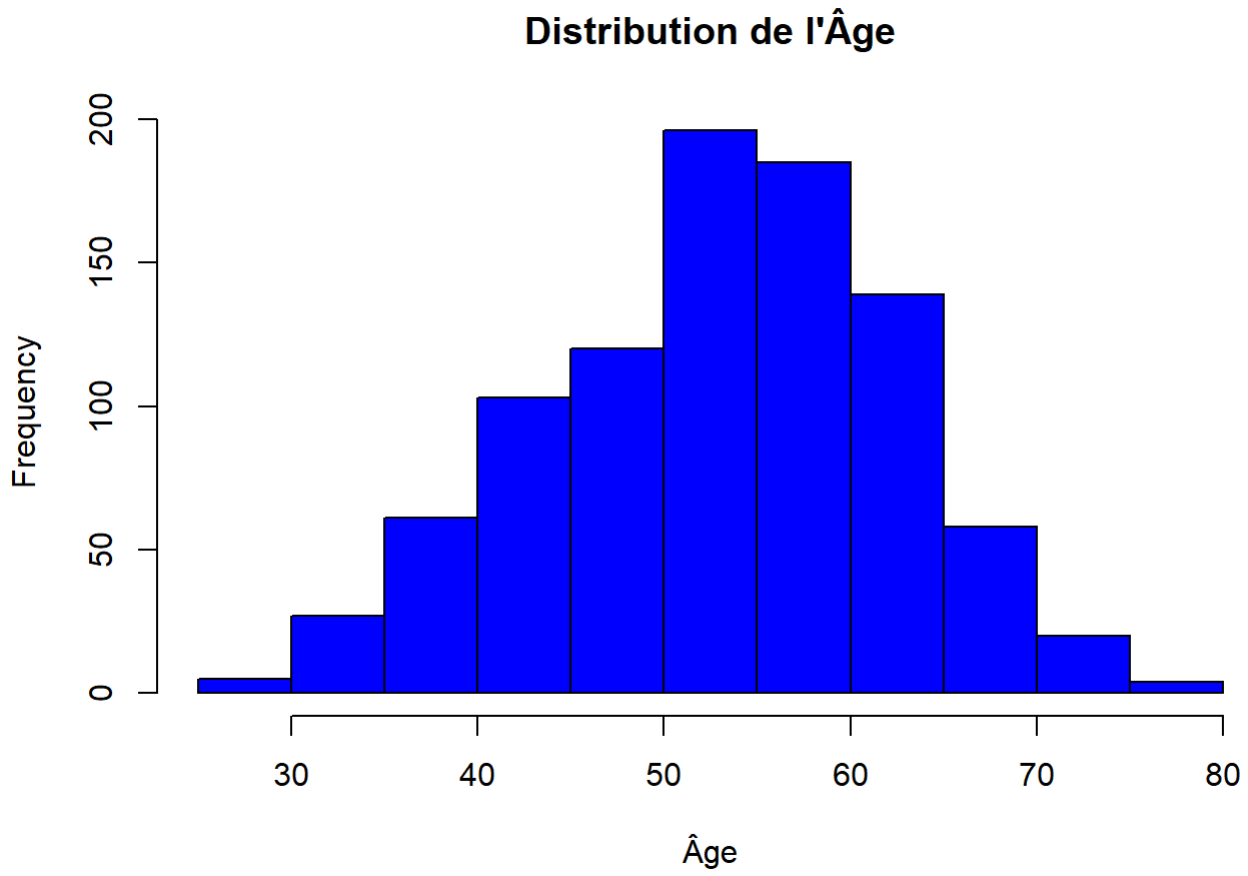
```
## Le chargement a nécessité le package : naniar
```

```
## Warning: le package 'naniar' a été compilé avec la version R 4.3.3
```

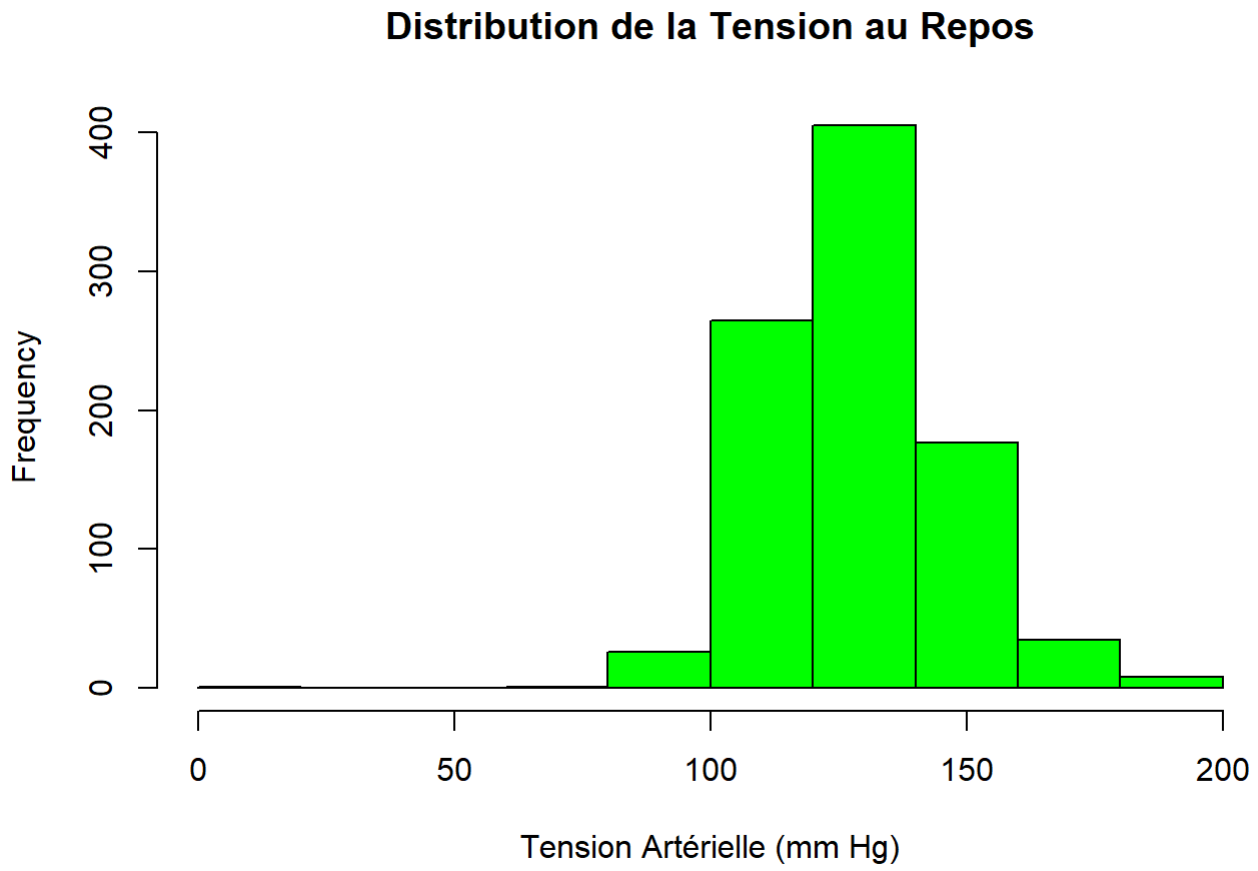
```
library(naniar)
# Analyse des valeurs manquantes
vis_miss(data) # Visualisation des données manquantes
```



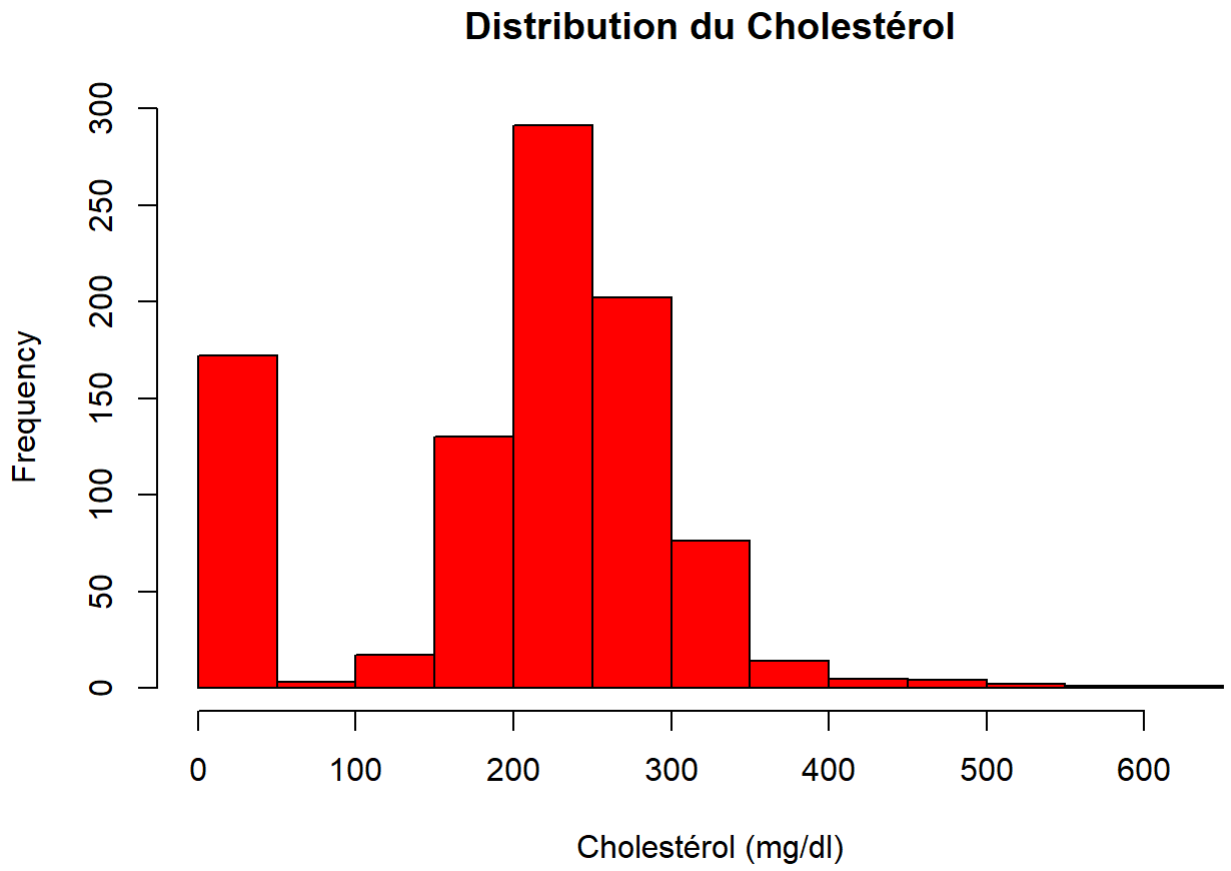

```
## Analyse des distributions des variables quantitatives
# Histogramme pour l'Age
hist(data$Age, main = "Distribution de l'Âge", xlab = "Âge", col = "blue")
```



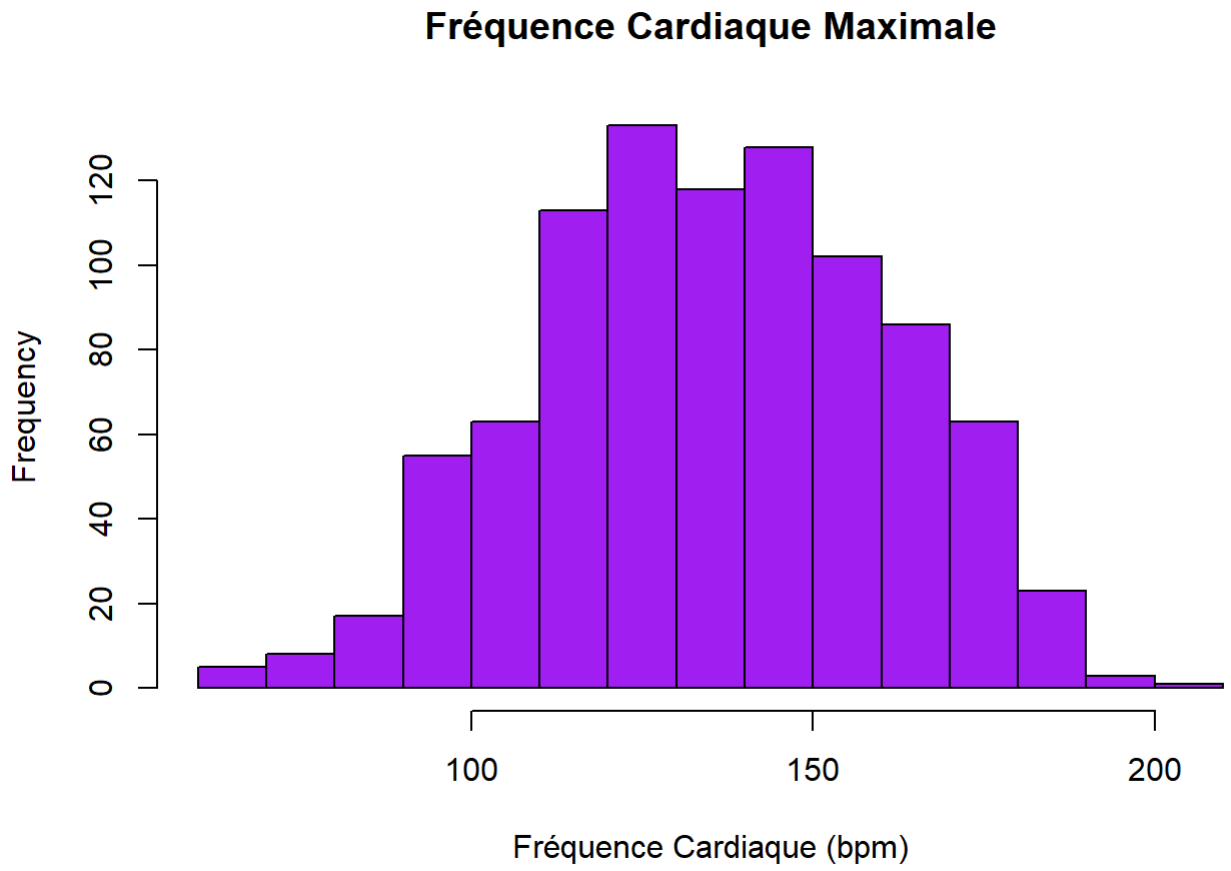

```
# Histogramme pour la Tension au Repos
hist(data$TensionRepos, main = "Distribution de la Tension au Repos", xlab = "Tension Artérie
lle (mm Hg)", col = "green")
```



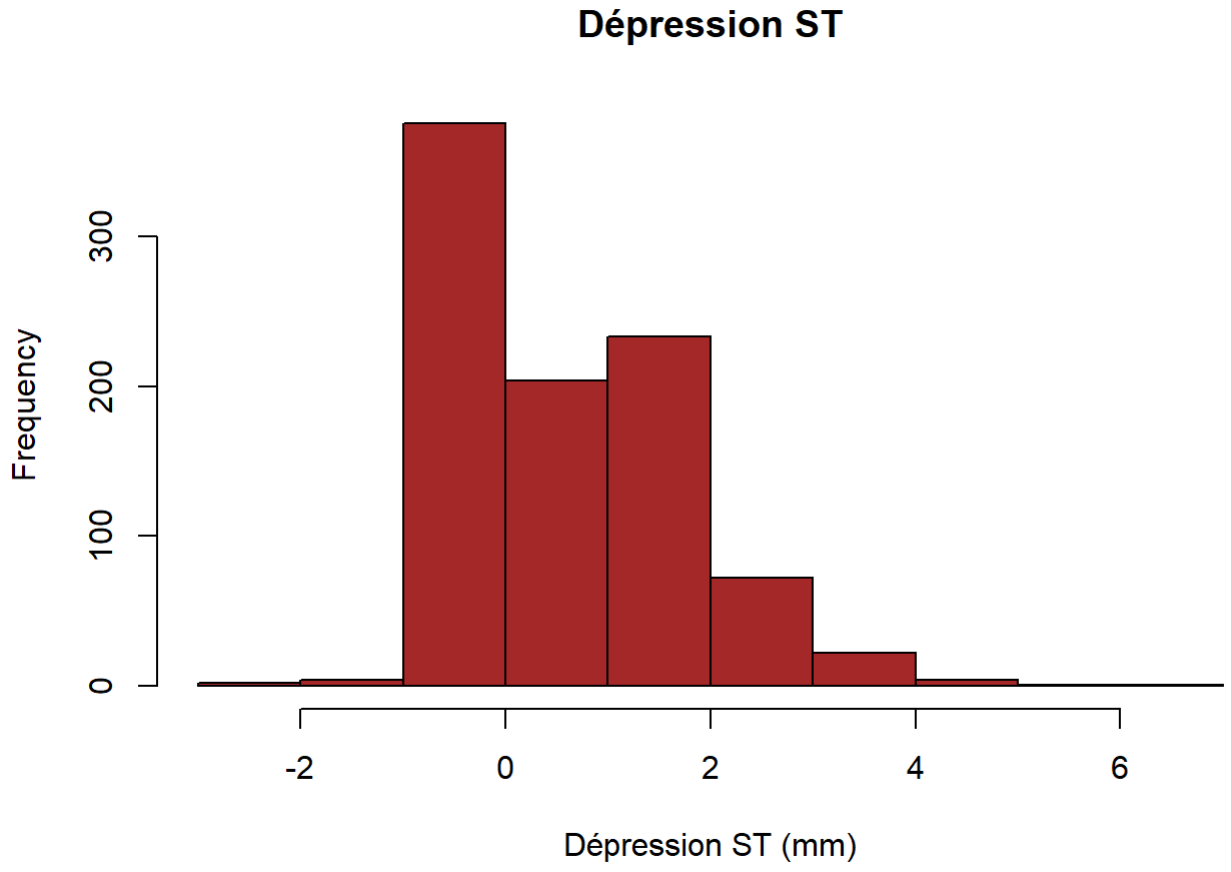

```
# Histogramme pour le Cholesterol
hist(data$Cholesterol, main = "Distribution du Cholestérol", xlab = "Cholestérol (mg/dl)", co
l = "red")
```




```
# Histogramme pour la Fréquence Cardiaque Maximale
hist(data$FreqCardiaqueMax, main = "Fréquence Cardiaque Maximale", xlab = "Fréquence Cardiaqu
e (bpm)", col = "purple")
```

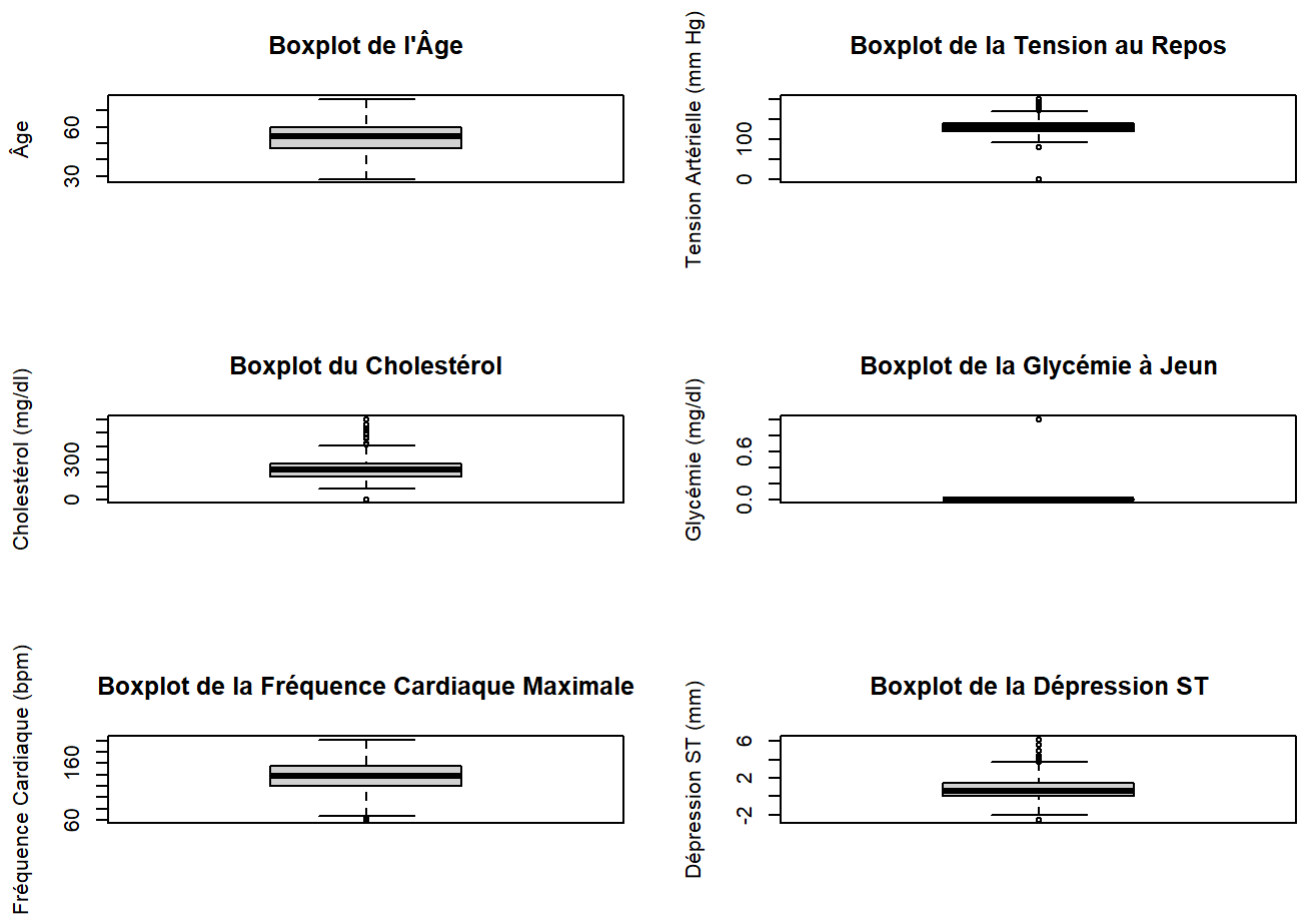



```
# Histogramme pour la Depression ST
hist(data$DepressionST, main = "Dépression ST", xlab = "Dépression ST (mm)", col = "brown")
```



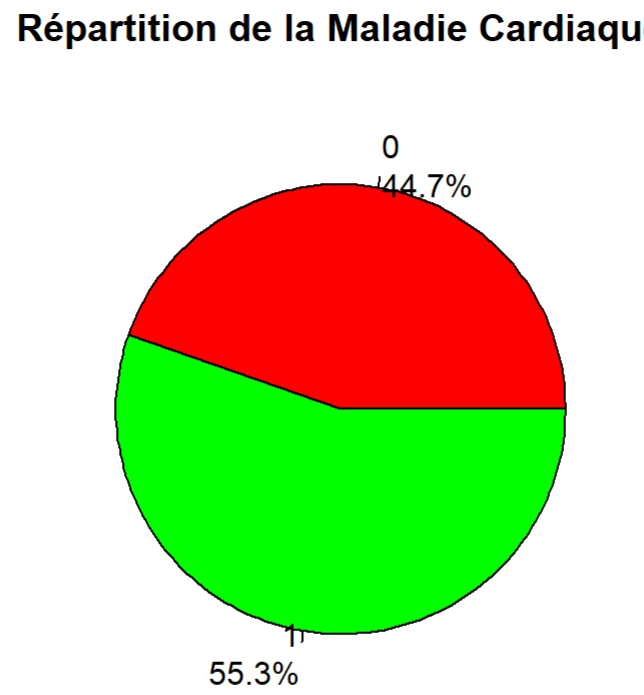
Analyse des boxplots des variables numériques

```
## Analyse des boxplots des variables numériques

# Tracer les boxplots
par(mfrow = c(3, 2)) # Organiser les graphiques en 3 lignes et 2 colonnes

# Boxplot pour l'Age
boxplot(data$Age, main = "Boxplot de l'Âge", ylab = "Âge")

# Boxplot pour la Tension au Repos
boxplot(data$TensionRepos, main = "Boxplot de la Tension au Repos", ylab = "Tension Artériell
e (mm Hg)")

# Boxplot pour le Cholesterol
boxplot(data$Cholesterol, main = "Boxplot du Cholestérol", ylab = "Cholestérol (mg/dl)")

# Boxplot pour la Glycemie à Jeun
boxplot(data$GlycemieAJeun, main = "Boxplot de la Glycémie à Jeun", ylab = "Glycémie (mg/d
l)")

# Boxplot pour la Fréquence Cardiaque Maximale
boxplot(data$FreqCardiaqueMax, main = "Boxplot de la Fréquence Cardiaque Maximale", ylab = "F
réquence Cardiaque (bpm)")

# Boxplot pour la Depression ST
boxplot(data$DepressionST, main = "Boxplot de la Dépression ST", ylab = "Dépression ST (mm)")
```

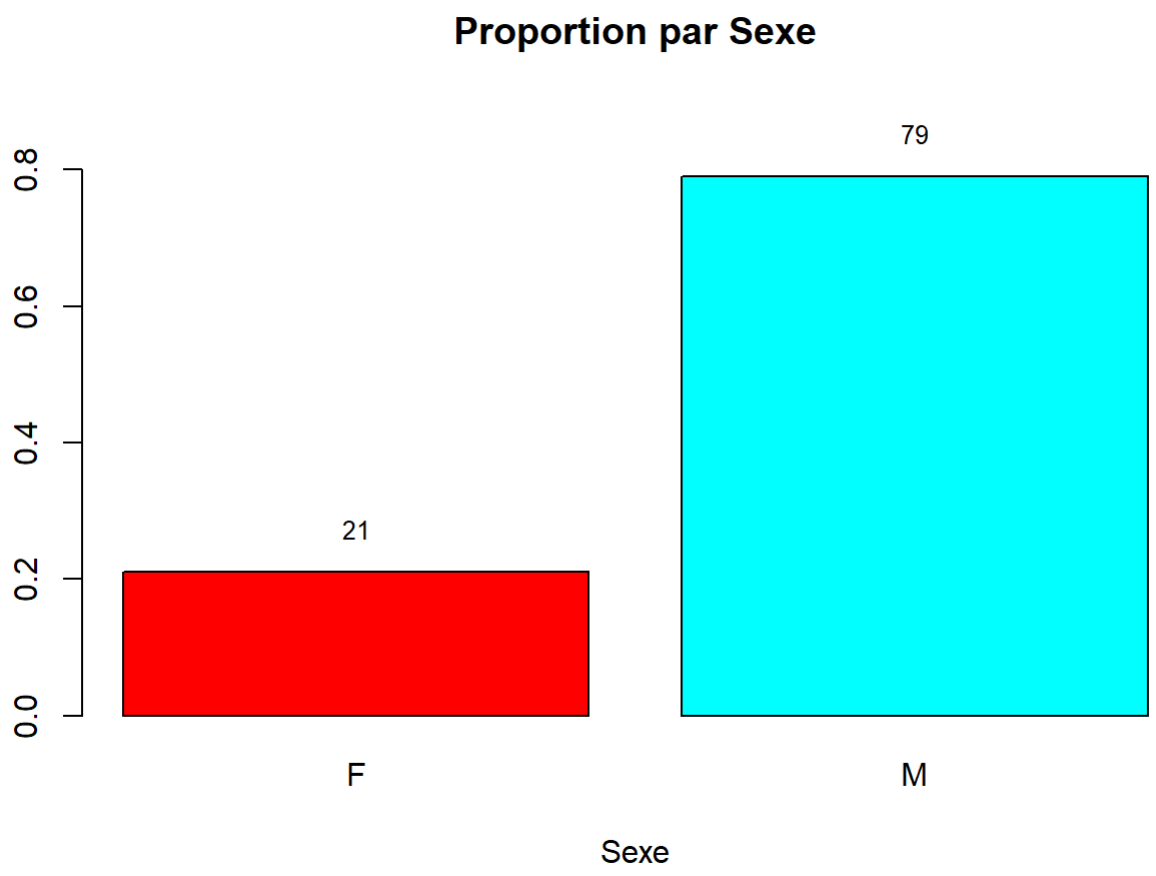


Analyse des variables qualitatives

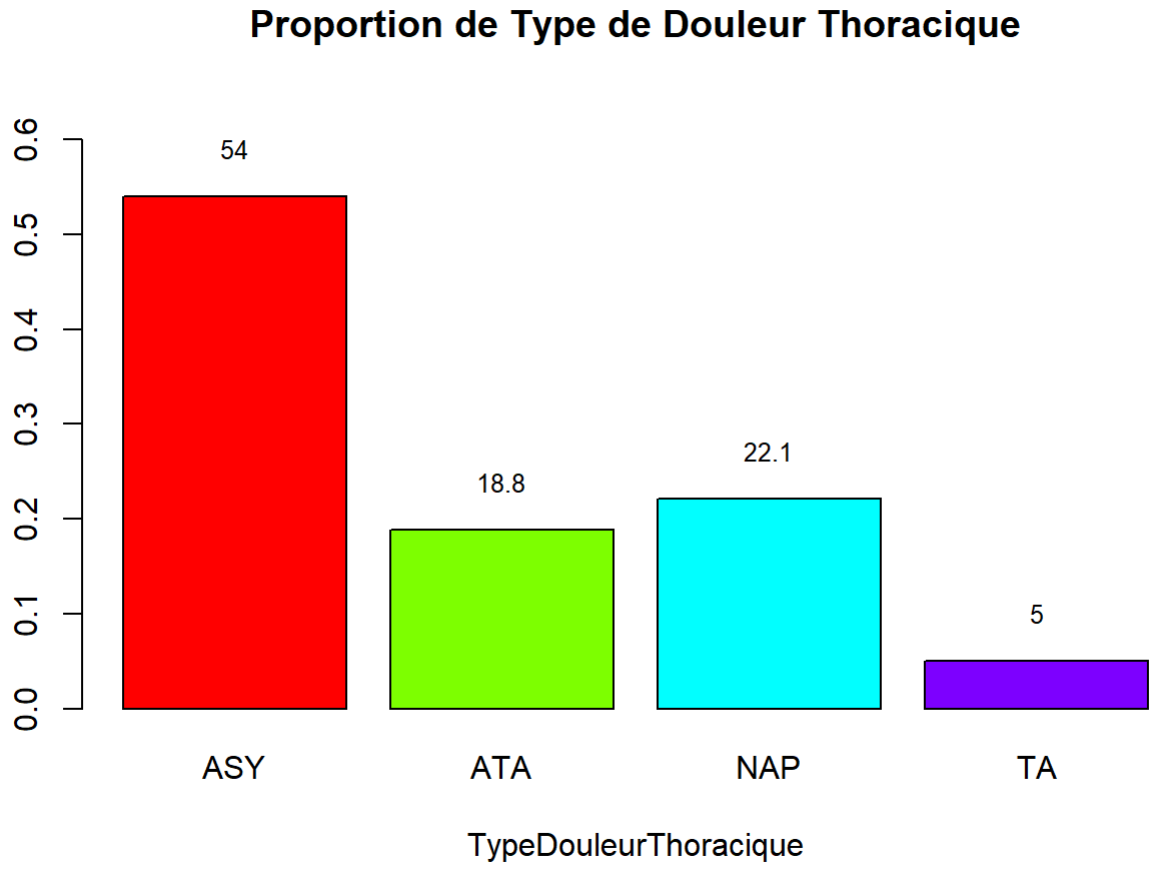
```
# Table et proportions pour 'MaladieCardiaque'
maladieCardiaqueTable <- table(data$MaladieCardiaque)
proportionsMaladieCardiaque <- round(prop.table(maladieCardiaqueTable) * 100, 1)
labels <- paste(names(maladieCardiaqueTable), "\n", proportionsMaladieCardiaque, "%", sep="")

# Diagramme en camembert avec proportions
pie(maladieCardiaqueTable, labels = labels, main = "Répartition de la Maladie Cardiaque", col
= c("red", "green"))
```



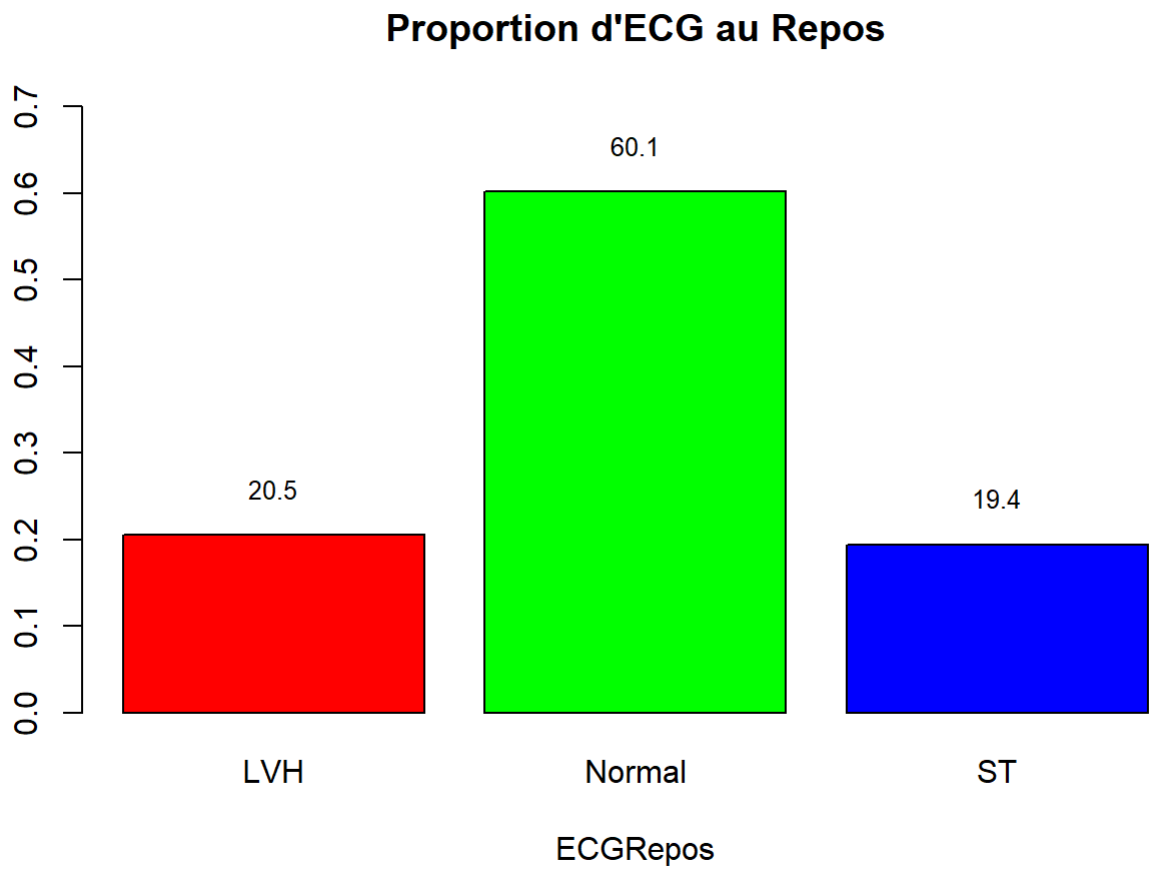

```
# Fonction pour créer un barplot de proportion avec valeurs
barplot_proportion <- function(variable, data, title) {
  table_var <- table(data[[variable]])
  prop_table <- prop.table(table_var)
  bp <- barplot(prop_table, main = title, xlab = variable, col = rainbow(length(prop_table)),
ylim = c(0, max(prop_table) + 0.1))

  # Ajouter les valeurs sur les barres
  text(bp, prop_table + 0.02, round(prop_table*100, 1), cex = 0.8, pos = 3)
}

# Barplot pour 'Sexe'
barplot_proportion("Sexe", data, "Proportion par Sexe")
```



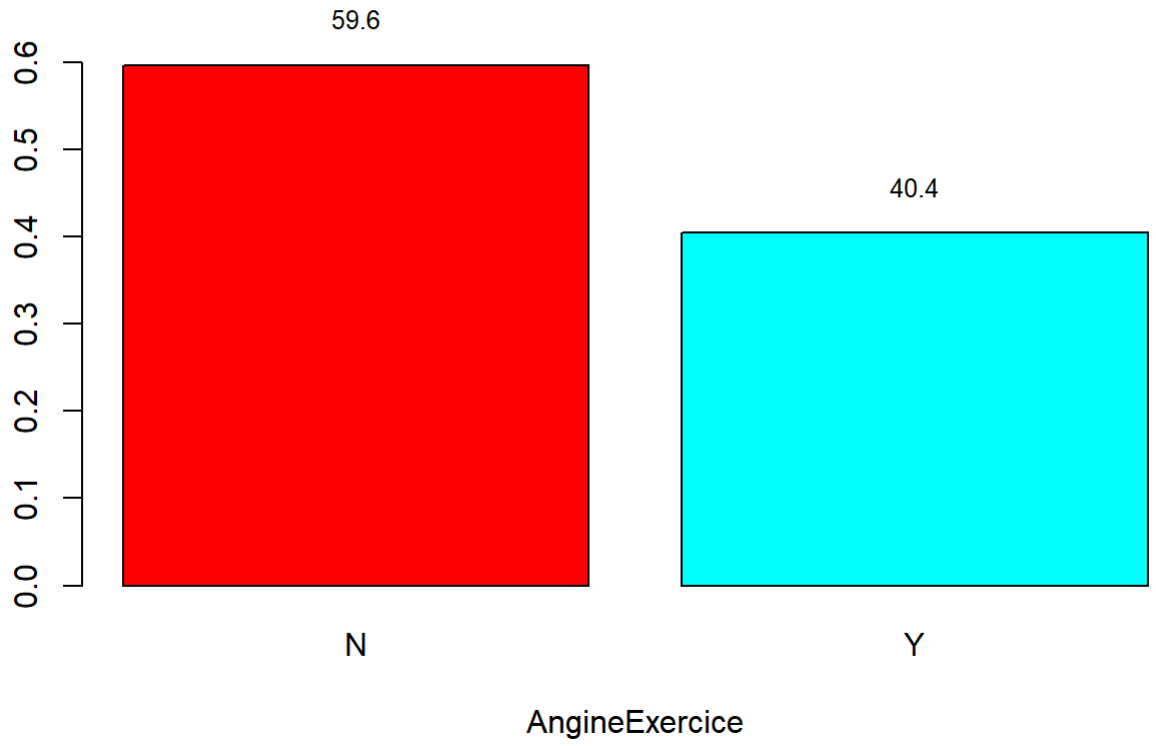

```
# Barplot pour 'TypeDouleurThoracique'
barplot_proportion("TypeDouleurThoracique", data, "Proportion de Type de Douleur Thoracique")
```



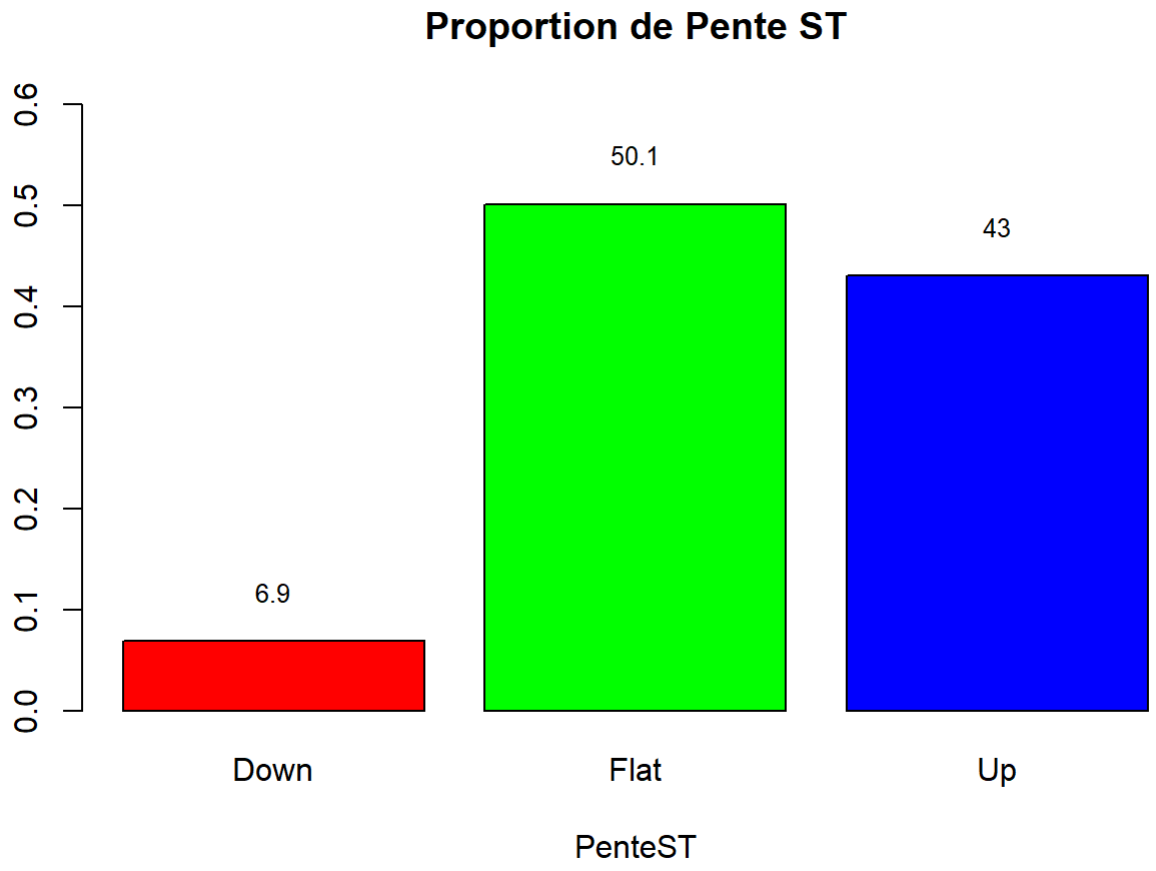
```
# Barplot pour 'ECGRepos'
barplot_proportion("ECGRepos", data, "Proportion d'ECG au Repos")
```




```
# Barplot pour 'AngineExercice'
barplot_proportion("AngineExercice", data, "Proportion d'Angine d'Exercice")
```



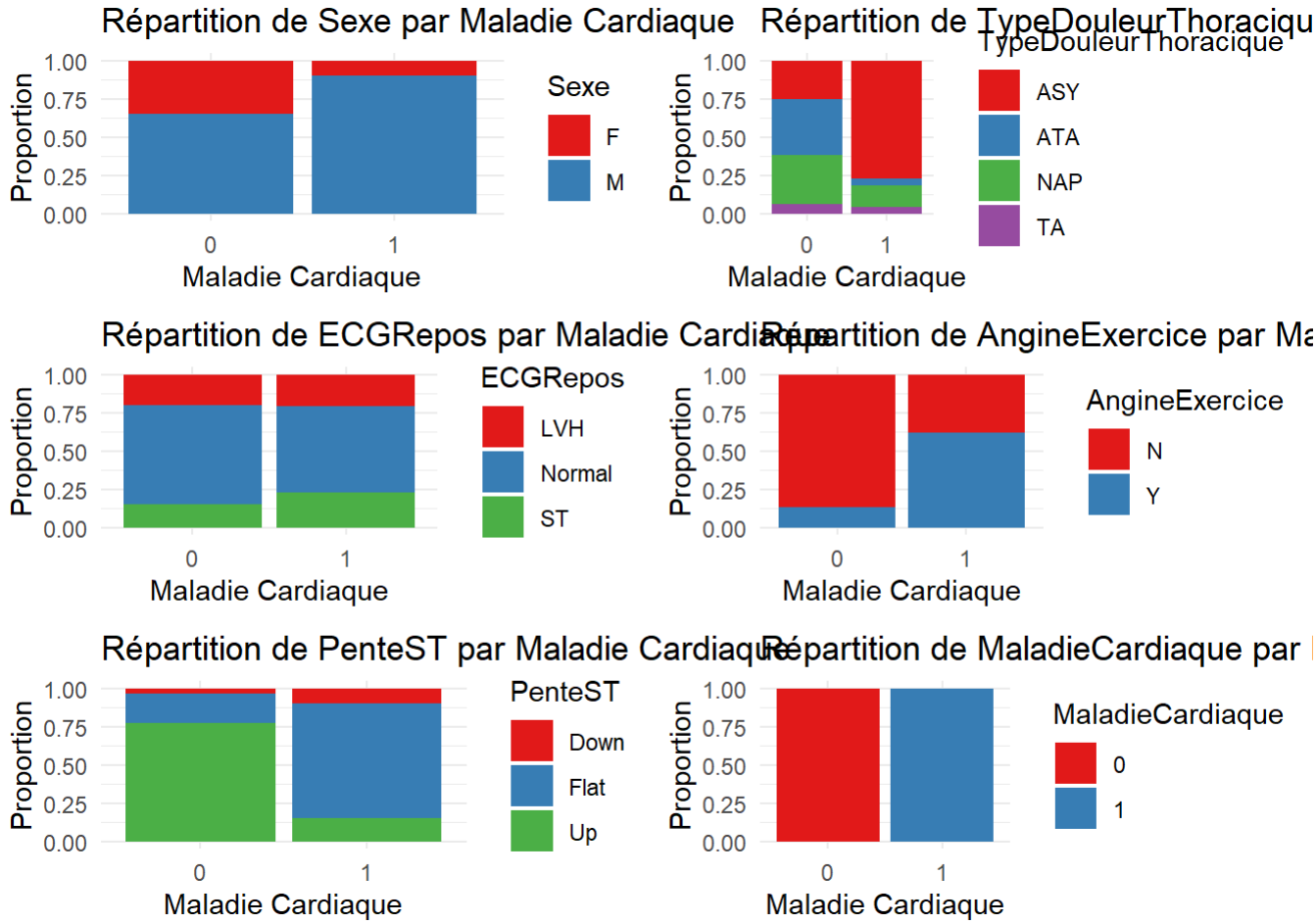

```
# Barplot pour 'PenteST'
barplot_proportion("PenteST", data, "Proportion de Pente ST")
```



Analyse de barplot bivarié

```
if (!require("ggplot2")) {
  # Si elle n'est pas installée, installer la bibliothèque
  install.packages("ggplot2")
}

library(ggplot2)
# Création des barplots bivariés
# Assurez-vous que les variables qualitatives sont du bon type
data$Sexe <- factor(data$Sexe, ordered = TRUE)
data$TypeDouleurThoracique <- factor(data$TypeDouleurThoracique, ordered = TRUE)
data$ECGRepos <- factor(data$ECGRepos, ordered = TRUE)
data$AngineExercice <- factor(data$AngineExercice, ordered = TRUE)
data$PenteST <- factor(data$PenteST, ordered = TRUE)

# Créer des barplots pour chaque variable qualitative par rapport à MaladieCardiaque
barplot_plots <- lapply(names(data)[sapply(data, is.factor)],
                        function(variable) {
                          ggplot(data, aes(x = factor(MaladieCardiaque), fill = .data[[variab
le]])) +
                            geom_bar(position = "fill") +
                            labs(title = paste("Répartition de", variable, "par Maladie Cardi
aque"),
                                 x = "Maladie Cardiaque",
                                 y = "Proportion") +
                            theme_minimal() +
                            scale_fill_brewer(palette = "Set1")
                        })
```

```
# Afficher les barplots dans une grille (2 par ligne)
library(gridExtra)
do.call(grid.arrange, c(barplot_plots, ncol = 2))
```



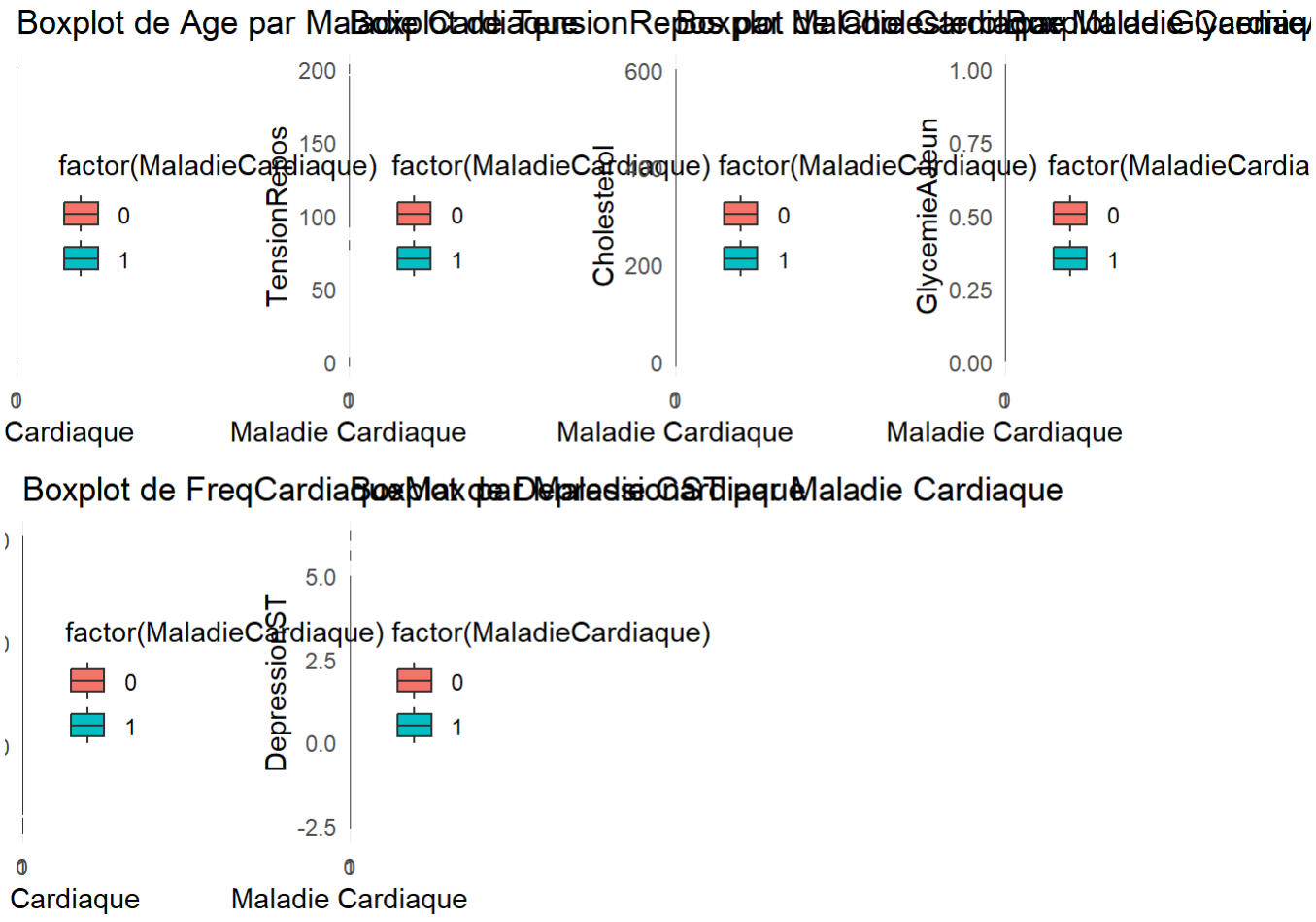
```
# Création du tableau pour les résultats des tests
results <- data.frame(Variable = character(), Chi_square = numeric(), P_value = numeric(), Cr
amers_V = numeric())

# Variables qualitatives
variables_qualitatives <- c("Sexe", "TypeDouleurThoracique", "ECGRepos", "AngineExercice", "P
enteST")

for (var in variables_qualitatives) {
  # Création du tableau de contingence
  contingency_table <- table(data[[var]], data$MaladieCardiaque)

  # Test du chi-carré
  chi_squared_test <- chisq.test(contingency_table)

  # Calcul du V de Cramer
  cramer_v <- sqrt(chi_squared_test$statistic / (nrow(data) * (min(nrow(contingency_table), n
col(contingency_table)) - 1)))

  # Ajouter les résultats au tableau
  results <- rbind(results, data.frame(Variable = var, Chi_square = chi_squared_test$statisti
c, P_value = chi_squared_test$p.value, Cramers_V = cramer_v))
}

# Trier les résultats par V de Cramer croissant
results <- results[order(-results$Cramers_V), ]

# Afficher les résultats
print(results)
```

```
##                         Variable Chi_square      P_value Cramers_V
## X-squared4               PenteST  355.91844 5.167638e-78 0.6226642
## X-squared1 TypeDouleurThoracique  268.06724 8.083728e-58 0.5403816
## X-squared3        AngineExercice  222.25938 2.907808e-50 0.4920494
## X-squared                   Sexe   84.14510 4.597617e-20 0.3027562
## X-squared2              ECGRepos   10.93147 4.229233e-03 0.1091234
```

Sélection des caractéristiques numérique

```
# Créer une fonction pour générer des boxplots bivariés
boxplot_bivarie <- function(data, variable_x, variable_y) {
  ggplot(data, aes(x = factor(variable_x), y = variable_y)) +
    geom_boxplot(fill = factor(variable_x)) +
    labs(title = paste("Boxplot de", variable_y, "par", variable_x),
         x = variable_x, y = variable_y) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
}
```

```
# Variables quantitatives à explorer
variables_quantitatives <- c("Age", "TensionRepos", "Cholesterol", "GlycemieAJeun", "FreqCard
iaqueMax", "DepressionST")

# Créer un boxplot bivarié pour chaque variable quantitative par rapport à MaladieCardiaque
boxplot_plots <- lapply(names(data)[sapply(data, is.numeric)],
                        function(variable) {
                          ggplot(data, aes(x = factor(MaladieCardiaque), y = .data[[variabl
e]], fill = factor(MaladieCardiaque))) +
                            geom_boxplot() +
                            labs(title = paste("Boxplot de", variable, "par Maladie Cardiaqu
e"),
                                 x = "Maladie Cardiaque", y = variable) +
                            theme_minimal()
                        })

# Afficher les boxplots dans une grille (2 par ligne)
library(gridExtra)
do.call(grid.arrange, c(boxplot_plots, ncol =4))
```



#LIEN ENTRE LES VARIABLES QUANTI ET LA VARIABLE QUALI(REPONSE)

```
# Variables numériques
variables_numeriques <- c("Age", "TensionRepos", "Cholesterol", "GlycemieAJeun", "FreqCardiaq
ueMax", "DepressionST")

# Créer un tableau pour les résultats
results <- data.frame(Variable = character(), Kruskal_Wallis = numeric(), P_value = numeric
())

for (var in variables_numeriques) {
  # Effectuer le test de Kruskal-Wallis
  kruskal_test <- kruskal.test(data[[var]] ~ data$MaladieCardiaque)

  # Ajouter les résultats au tableau
  results <- rbind(results, data.frame(Variable = var, Kruskal_Wallis = kruskal_test$statisti
c, P_value = kruskal_test$p.value))
}

# Trier les résultats par la statistique de test décroissant
results <- results[order(results$Kruskal_Wallis, decreasing = TRUE), ]

# Afficher les résultats
print(results)
```

```
##                                   Variable Kruskal_Wallis      P_value
## Kruskal-Wallis chi-squared5     DepressionST      161.02486 6.756660e-37
## Kruskal-Wallis chi-squared4 FreqCardiaqueMax      150.28229 1.504033e-34
## Kruskal-Wallis chi-squared               Age       76.89417 1.803687e-18
## Kruskal-Wallis chi-squared3    GlycemieAJeun       65.51468 5.768275e-16
## Kruskal-Wallis chi-squared2      Cholesterol       17.94063 2.279038e-05
## Kruskal-Wallis chi-squared1     TensionRepos       11.88939 5.645448e-04
```

ANALYSE DE LA MULTICOLINEARITE DES VARIABLES QUATITATIVE

```
### Corrélation entre les grandeur numérique

# Sélectionner les variables numériques
variables_numeriques <- data[, c("Age", "TensionRepos", "Cholesterol", "GlycemieAJeun", "Freq
CardiaqueMax", "DepressionST")]

# Calculer la matrice de corrélation
correlation_matrix <- cor(variables_numeriques, use = "complete.obs")

# Installer et charger corrplot
if (!require(corrplot)) install.packages("corrplot")
```

```
## Le chargement a nécessité le package : corrplot
```

```
## corrplot 0.92 loaded
```

```
library(corrplot)
```

Etape 5: Partitionnement des données

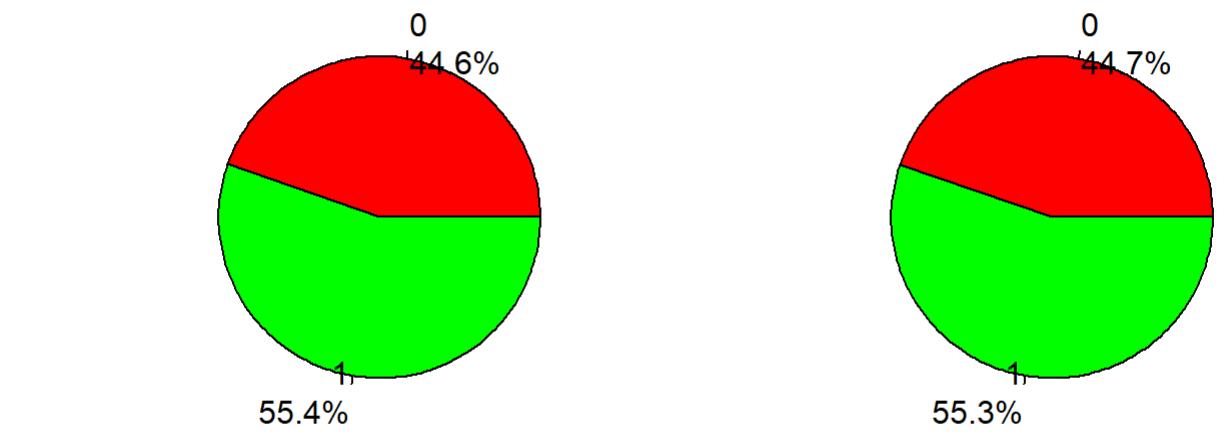
```
# Installer et charger le package caret si nécessaire
if (!require(caret)) install.packages("caret")
```

```
## Le chargement a nécessité le package : caret
```

```
## Le chargement a nécessité le package : lattice
```

```
library(caret)

# Définir la proportion de données à garder dans l'ensemble d'entraînement (par exemple, 70%)
proportion_entrainement <- 0.7

# Créer des indices pour un partitionnement stratifié
set.seed(123) # Pour la reproductibilité
indices_entrainement <- createDataPartition(data$MaladieCardiaque, p = proportion_entrainemen
t, list = FALSE)

# Créer les ensembles d'entraînement et de test
data_entrainement <- data[indices_entrainement, ]
data_test <- data[-indices_entrainement, ]

# Fonction pour créer un pie chart avec proportions
creer_pie_chart <- function(data_subset, title) {
  counts <- table(data_subset$MaladieCardiaque)
  proportions <- round(100 * counts / sum(counts), 1)
  labels <- paste(names(counts), "\n", proportions, "%", sep="")

  pie(counts, labels = labels, main = title, col = c("red", "green"))
}
```

```
# Créer un pie chart pour l'ensemble d'entraînement
par(mfrow = c(1, 2)) # Pour afficher les deux diagrammes côte à côte
creer_pie_chart(data_entrainement, "Répartition de Maladie Cardiaque (Entraînement)")

# Créer un pie chart pour l'ensemble de test
creer_pie_chart(data_test, "Répartition de Maladie Cardiaque (Test)")
```



Etape 6: Entrainement du modèle

```
# Entraînement du modèle de régression logistique avec glm
modele_logistique <- glm(MaladieCardiaque ~ ., data = data_entrainement, family = binomial)

# Afficher le résumé du modèle
summary(modele_logistique)
```

```
##
## Call:
## glm(formula = MaladieCardiaque ~ ., family = binomial, data = data_entrainement)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.123279   1.556343  -0.722 0.470453
## Age                      0.015273   0.015495   0.986 0.324290
## Sexe.L                   1.065630   0.228587   4.662 3.13e-06 ***
## TypeDouleurThoracique.L -1.303275   0.380098  -3.429 0.000606 ***
## TypeDouleurThoracique.Q  0.912365   0.333586   2.735 0.006238 **
## TypeDouleurThoracique.C -0.347737   0.304910  -1.140 0.254096
## TensionRepos             0.004104   0.006789   0.605 0.545510
## Cholesterol             -0.004185   0.001235  -3.388 0.000703 ***
## GlycemieAJeun            1.183915   0.321640   3.681 0.000232 ***
## ECGRepos.L              -0.135058   0.278745  -0.485 0.628014
## ECGRepos.Q               0.239964   0.209354   1.146 0.251706
## FreqCardiaqueMax        -0.004912   0.006014  -0.817 0.414069
## AngineExercice.L         0.458643   0.198906   2.306 0.021121 *
## DepressionST             0.286305   0.140361   2.040 0.041373 *
## PenteST.L               -0.613710   0.371561  -1.652 0.098594 .
## PenteST.Q               -1.657676   0.257716  -6.432 1.26e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 883.97  on 642  degrees of freedom
## Residual deviance: 438.99  on 627  degrees of freedom
## AIC: 470.99
##
## Number of Fisher Scoring iterations: 5
```

```
#install.packages("lmtest")
library(lmtest) ## qui permet de mettre en place le teste de rapport de vraissamblance de sig
nificativité global du modele
```

```
## Warning: le package 'lmtest' a été compilé avec la version R 4.3.3
```

```
## Le chargement a nécessité le package : zoo
```

```
##
## Attachement du package : 'zoo'
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# Effectuer le test du rapport de vraisemblance (Likelihood Ratio Test)
test_lr <- lrtest(modele_logistique)

# Afficher les résultats du test
print(test_lr) ## il ya au moins un coeficient qui est statistiquement different de 0
```

```
## Likelihood ratio test
##
## Model 1: MaladieCardiaque ~ Age + Sexe + TypeDouleurThoracique + TensionRepos +
##     Cholesterol + GlycemieAJeun + ECGRepos + FreqCardiaqueMax +
##     AngineExercice + DepressionST + PenteST
## Model 2: MaladieCardiaque ~ 1
##   #Df  LogLik  Df  Chisq Pr(>Chisq)
## 1  16 -219.50
## 2   1 -441.98 -15 444.98  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
### MESURE DU POUVOIR EXPLICATIF DU MODELE

# Calculer les déviations nulles et proposées
ll.null <- modele_logistique$null.deviance / -2
ll.proposed <- modele_logistique$deviance / -2

# Calculer le pseudo R-carré de McFadden
pseudo_r_squared_mcfadden <- 1 - (ll.proposed / ll.null)

# Afficher le pseudo R-carré de McFadden
print(pseudo_r_squared_mcfadden) ## plus c'est élévé plus notre modèle est bien
```

```
## [1] 0.5033842
```

Etape 7: Interprétation des résultats

```
# Obtenir les coefficients estimés du modèle
coefficients <- coef(modele_logistique)

# Calculer les rapports de cotes en exponentiant les coefficients
odds_ratios <- exp(coefficients)

# Créer un tableau avec les noms des variables et leurs rapports de cotes
variables <- names(coefficients)
tableau_odds_ratios <- data.frame(Variable = variables, OddsRatio = odds_ratios)

# Afficher le tableau des rapports de cotes
tableau_odds_ratios
```

```
##                                       Variable OddsRatio
## (Intercept)                         (Intercept) 0.3252116
## Age                                         Age 1.0153907
## Sexe.L                                   Sexe.L 2.9026680
## TypeDouleurThoracique.L TypeDouleurThoracique.L 0.2716408
## TypeDouleurThoracique.Q TypeDouleurThoracique.Q 2.4902049
## TypeDouleurThoracique.C TypeDouleurThoracique.C 0.7062848
## TensionRepos                         TensionRepos 1.0041126
## Cholesterol                           Cholesterol 0.9958234
## GlycemieAJeun                       GlycemieAJeun 3.2671408
## ECGRepos.L                             ECGRepos.L 0.8736649
## ECGRepos.Q                             ECGRepos.Q 1.2712038
## FreqCardiaqueMax                 FreqCardiaqueMax 0.9951004
## AngineExercice.L                 AngineExercice.L 1.5819260
## DepressionST                         DepressionST 1.3314986
## PenteST.L                               PenteST.L 0.5413390
## PenteST.Q                               PenteST.Q 0.1905814
```

INTERPRETATION

la cote d'avoir une maladie cardiaque lorsqu'on est un homme sont multiplié par 2.90 lorsque l'age augmente d'un an la cote d'avoir une maladie cardiaque est de 1.01 à toutes choses chose égale la cote d'avoir une maladie cardiaque est de 0.32

Etape 8: Evaluation et prédiction

```
## Evaluation du modèle

# Vérifier si la bibliothèque pROC est déjà installée, sinon l'installer
if (!require(pROC)) {
  install.packages("pROC")
  library(pROC)
}
```

```
## Le chargement a nécessité le package : pROC
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attachement du package : 'pROC'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     cov, smooth, var
```

```
probas_train <- predict(modele_logistique, data_entrainement, type = "response") ## response indique que je veut predire des probabilité
#class aurait voulu dire que je sohaite predire des class
probas_test <- predict(modele_logistique, data_test, type = "response")

roc_train <- roc(response = data_entrainement$MaladieCardiaque, predictor = probas_train)
```

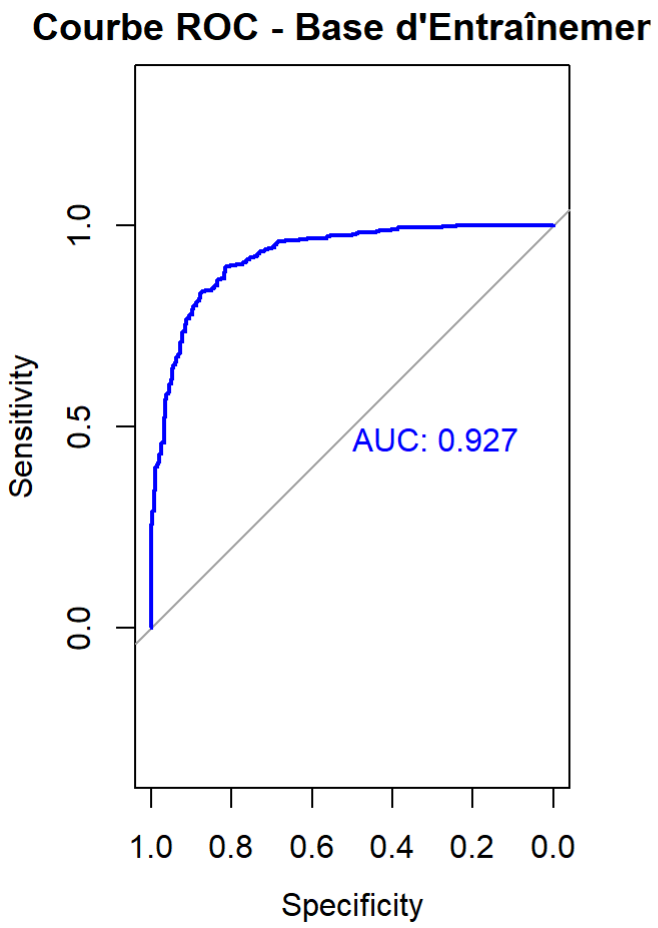
```
## Setting levels: control = 0, case = 1
```

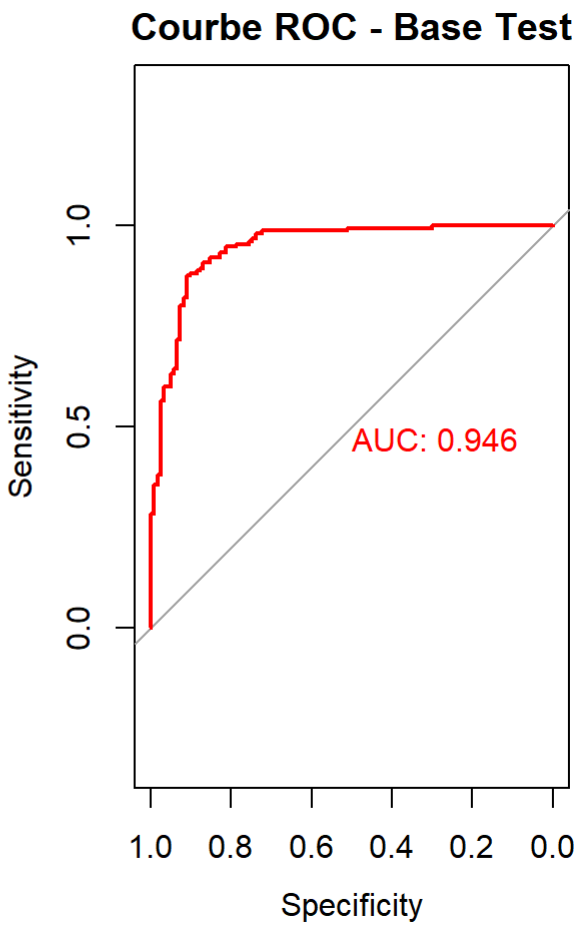
```
## Setting direction: controls < cases
```

```
roc_test <- roc(response = data_test$MaladieCardiaque, predictor = probas_test)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
# Afficher les courbes ROC avec AUC
par(mfrow=c(1,2))  # Afficher deux graphiques côte à côte
plot(roc_train, main = "Courbe ROC - Base d'Entraînement", col = "blue", print.auc = TRUE)
plot(roc_test, main = "Courbe ROC - Base Test", col = "red", print.auc = TRUE)
```

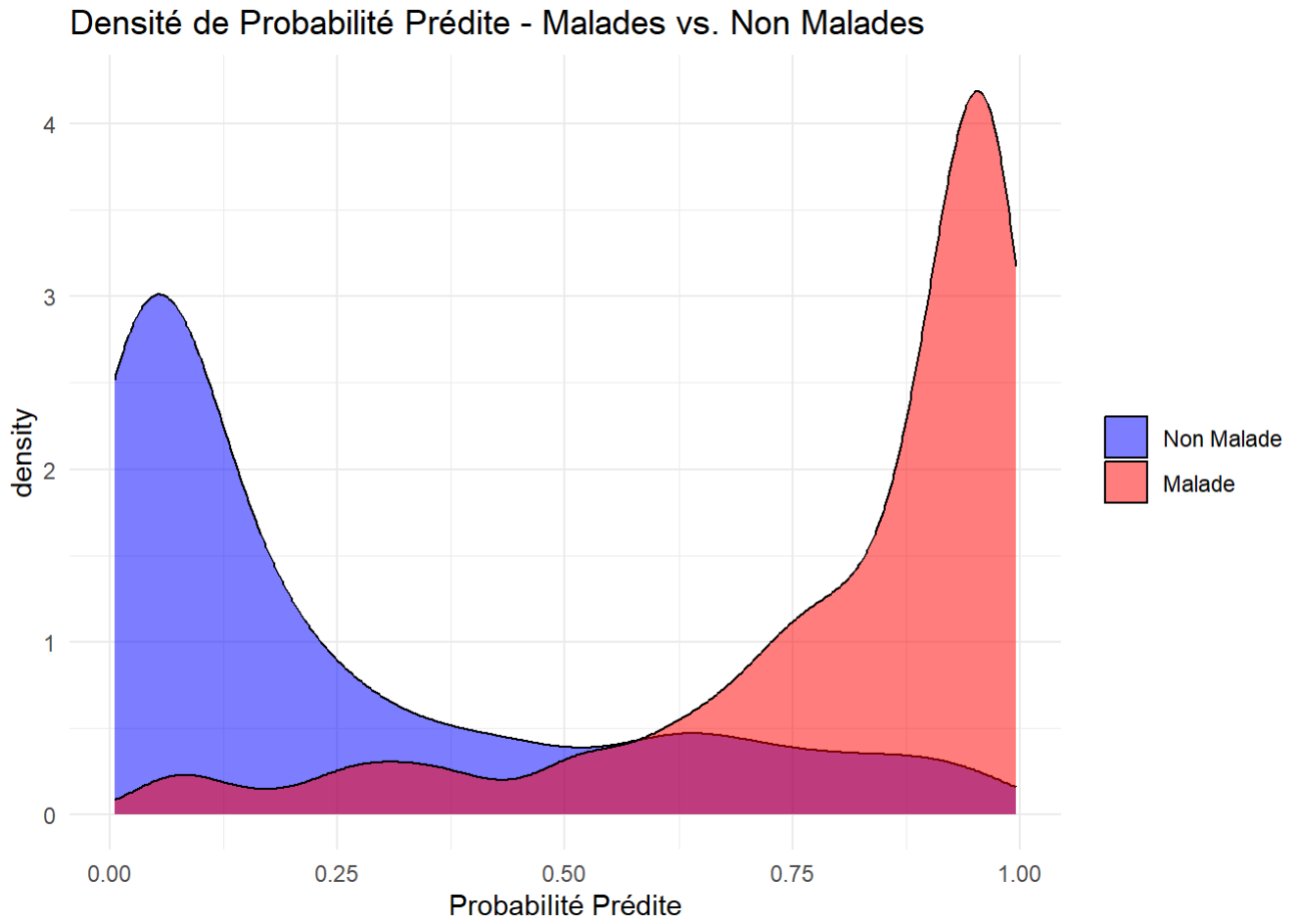




```
auc_train <- auc(roc_train)
auc_test <- auc(roc_test)

auc_table <- data.frame(Base = c("Entraînement", "Test"), AUC = c(auc_train, auc_test))

print(auc_table)
```

```
##           Base       AUC
## 1 Entraînement 0.9266042
## 2         Test 0.9455499
```

```
# Prédire les probabilités sur la base d'entraînement
probas_train <- predict(modele_logistique, data_entrainement, type = "response")

# Créer un data frame avec les probabilités prédites et les étiquettes de maladie cardiaque
predicted_data <- data.frame(Probabilite = probas_train, MaladieCardiaque = data_entrainement
$MaladieCardiaque)

# Remplacer les valeurs de MaladieCardiaque (0 par "Non Malade" et 1 par "Malade")
predicted_data$MaladieCardiaque <- factor(predicted_data$MaladieCardiaque, levels = c(0, 1),
labels = c("Non Malade", "Malade"))

# Créer un graphique de densité pour les malades et les non malades
ggplot(predicted_data, aes(x = Probabilite, fill = MaladieCardiaque)) +
  geom_density(alpha = 0.5) +
  labs(title = "Densité de Probabilité Prédite - Malades vs. Non Malades", x = "Probabilité P
rédite") +
  scale_fill_manual(values = c("Non Malade" = "blue", "Malade" = "red")) +
  theme_minimal() +
  theme(legend.title = element_blank()) +  # Supprimer le titre de la légende
  labs(fill = "Maladie Cardiaque")  # Renommer la légende
```



MATRICE DE PRECISION

```
# Installer et charger la bibliothèque caret si elle n'est pas déjà installée
if (!require(caret)) {
  install.packages("caret")
  library(caret)
}
```

```
data_entrainement$MaladieCardiaque <- as.factor(data_entrainement$MaladieCardiaque)
data_test$MaladieCardiaque <- as.factor(data_test$MaladieCardiaque)
# Prédire les classes en utilisant un seuil de probabilité de 0.5 pour la base d'entraînement
seuil <- 0.5
predictions_train <- ifelse(probas_train >= seuil, 1, 0)
predictions_train <- factor(predictions_train, levels = c(0, 1))
```

```
# Créer la matrice de confusion pour la base d'entraînement
confusion_matrix_train <- confusionMatrix(predictions_train, data_entrainement$MaladieCardiaq
ue)
```

```
# Prédire les classes en utilisant un seuil de probabilité de 0.5 pour la base de test
predictions_test <- ifelse(probas_test >= seuil, 1, 0)
predictions_test <- factor(predictions_test, levels = c(0, 1))
# Créer la matrice de confusion pour la base de test
confusion_matrix_test <- confusionMatrix(predictions_test, data_test$MaladieCardiaque)

# Afficher les matrices de confusion
confusion_matrix_train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 234  37
##          1  53 319
##
##                Accuracy : 0.86
##                  95% CI : (0.8308, 0.8859)
##     No Information Rate : 0.5537
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7153
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.8153
##             Specificity : 0.8961
##          Pos Pred Value : 0.8635
##          Neg Pred Value : 0.8575
##              Prevalence : 0.4463
##          Detection Rate : 0.3639
##    Detection Prevalence : 0.4215
##       Balanced Accuracy : 0.8557
##
##        'Positive' Class : 0
##
```

```
confusion_matrix_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 107  16
##          1  16 136
##
##                Accuracy : 0.8836
##                  95% CI : (0.8397, 0.919)
##     No Information Rate : 0.5527
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7647
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8699
##             Specificity : 0.8947
##          Pos Pred Value : 0.8699
##          Neg Pred Value : 0.8947
##              Prevalence : 0.4473
##          Detection Rate : 0.3891
##    Detection Prevalence : 0.4473
##       Balanced Accuracy : 0.8823
##
##        'Positive' Class : 0
##
```