WAS 19: Régression Logistique dans la pratique

Natacha NJONGWA YEPNGA

2023-12-03

Table of Contents

# [LeCoinStat](https://youtube.com/c/LeCoinStat?sub_confirmation=1)

[![Vidéo disponible sur LeCoinStat](data:text/html; charset=utf-8;base64,)](https://www.youtube.com/watch?v=XOfyd7DuFbw&list=PLyh35eYRez8fPQY7ujGjWI4lusS4jxp2N&index=18)

# Etape 1: Définition du problème

Développer un modèle qui permet de prédire la présence d’une maladie cardiaque chez les patients.

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

# Etape 2: Collecte des données

Les informations pour la démonstration proviennent du dépôt de machine learning de l’Université de Californie à Irvine (UCI). [UCI Machine Learning Repository](http://archive.ics.uci.edu/ml/index.php)

Il s’agit plus précisément du 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: [Heart Disease Data Set](http://archive.ics.uci.edu/ml/datasets/Heart+Disease)

La base est également disponible sur Kaggle: [Heart Failure Prediction](https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction)

url <- "https://raw.githubusercontent.com/LeCoinStat/WAS/main/WAS19/data/heart.csv"  
data <- read.csv(url, header=TRUE)

head(data)

## 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

# Renommer les colonnes  
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 ...

# Afficher la structure du cadre de données pour voir le type de chaque variable  
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)  
  
# Vérifier la modification  
str(data$MaladieCardiaque)

## Factor w/ 2 levels "0","1": 1 2 1 2 1 1 1 1 2 1 ...

# Etape 3: prétraitement des données

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

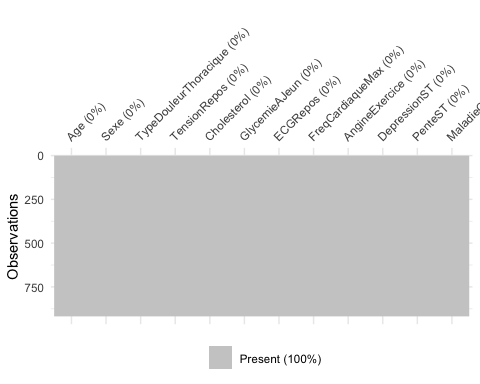
## Analyse exploratoire

# 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

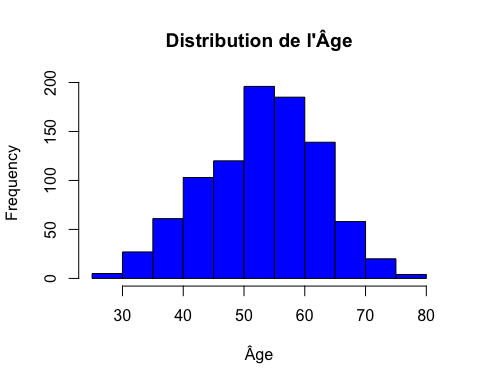
## Analyse des valeurs manquantes

# Charger la bibliothèque pour la gestion des données manquantes  
#install.packages("naniar")  
library(naniar)  
# Analyse des valeurs manquantes  
vis\_miss(data) # Visualisation des données manquantes

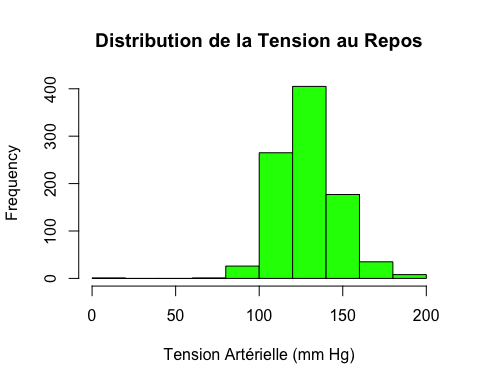


## Analyse des distributions des variables quantitatives

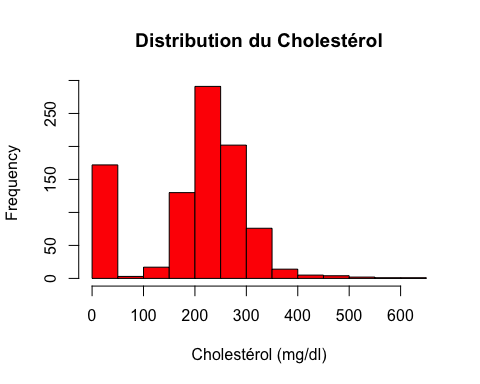
# Histogramme pour l'Age  
hist(data$Age, main = "Distribution de l'Âge", xlab = "Âge", col = "blue")



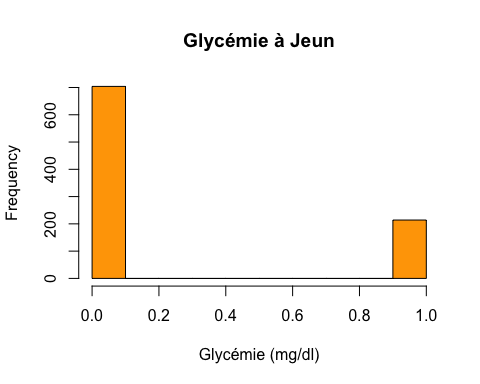
# Histogramme pour la Tension au Repos  
hist(data$TensionRepos, main = "Distribution de la Tension au Repos", xlab = "Tension Artérielle (mm Hg)", col = "green")



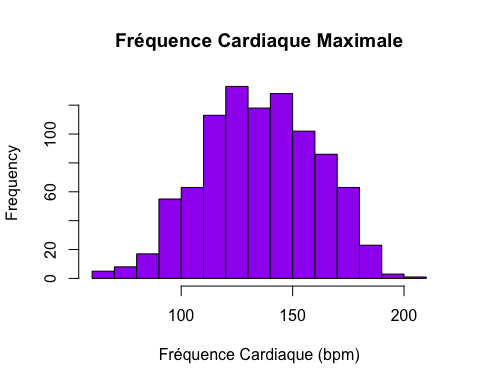
# Histogramme pour le Cholesterol  
hist(data$Cholesterol, main = "Distribution du Cholestérol", xlab = "Cholestérol (mg/dl)", col = "red")



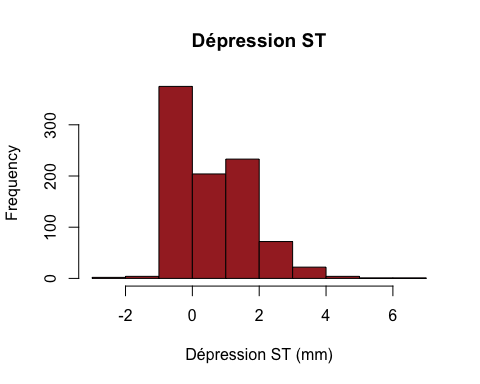
# Histogramme pour la Glycemie à Jeun  
hist(data$GlycemieAJeun, main = "Glycémie à Jeun", xlab = "Glycémie (mg/dl)", col = "orange")



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

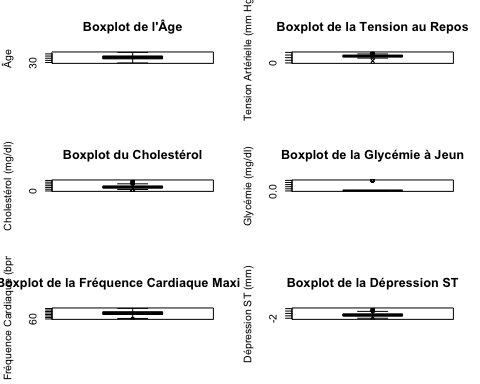


# Histogramme pour la Depression ST  
hist(data$DepressionST, main = "Dépression ST", xlab = "Dépression ST (mm)", col = "brown")



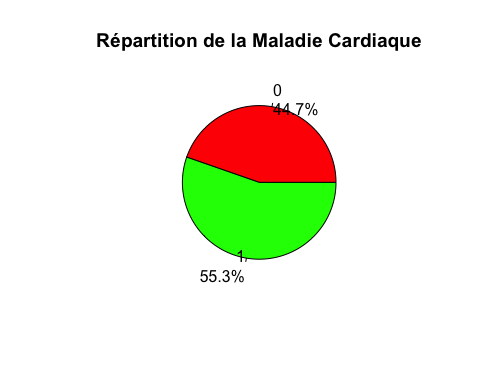
## Analyse des boxplots des variables numériques

# 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 Artérielle (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/dl)")  
  
# Boxplot pour la Fréquence Cardiaque Maximale  
boxplot(data$FreqCardiaqueMax, main = "Boxplot de la Fréquence Cardiaque Maximale", ylab = "Fréquence Cardiaque (bpm)")  
  
# Boxplot pour la Depression ST  
boxplot(data$DepressionST, main = "Boxplot de la Dépression ST", ylab = "Dépression ST (mm)")

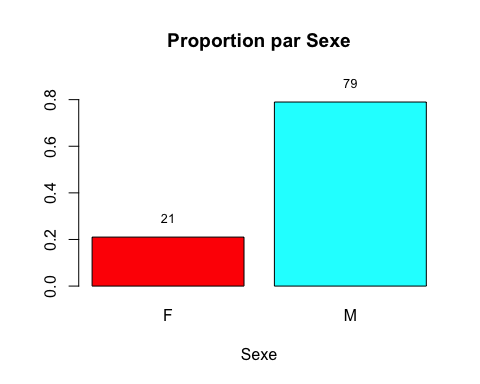


## Analyse des variables qualitatives

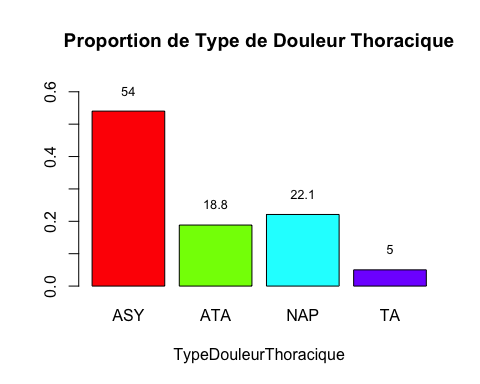
# Table et proportions pour 'MaladieCardiaque'  
maladieCardiaqueTable <- table(data$MaladieCardiaque)  
proportionsMaladieCardiaque <- round(prop.table(maladieCardiaqueTable) \* 100, 1)  
labels <- paste(names(maladieCardiaqueTable), "\n", proportionsMaladieCardiaque, "%", sep="")  
  
# Diagramme en camembert avec proportions  
pie(maladieCardiaqueTable, labels = labels, main = "Répartition de la Maladie Cardiaque", col = c("red", "green"))



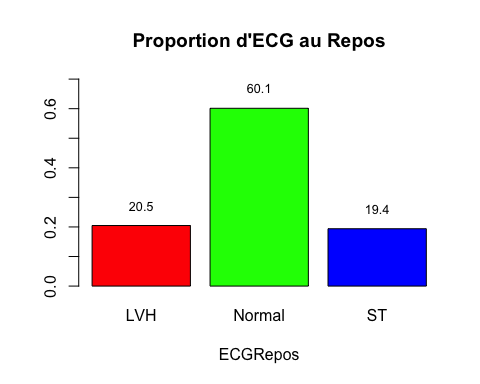
# Fonction pour créer un barplot de proportion avec valeurs  
barplot\_proportion <- function(variable, data, title) {  
 table\_var <- table(data[[variable]])  
 prop\_table <- prop.table(table\_var)  
 bp <- barplot(prop\_table, main = title, xlab = variable, col = rainbow(length(prop\_table)), ylim = 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")



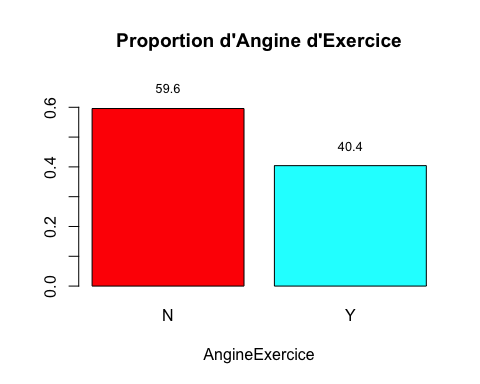
# Barplot pour 'TypeDouleurThoracique'  
barplot\_proportion("TypeDouleurThoracique", data, "Proportion de Type de Douleur Thoracique")



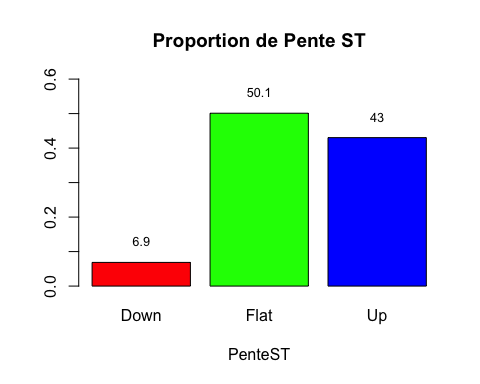
# Barplot pour 'ECGRepos'  
barplot\_proportion("ECGRepos", data, "Proportion d'ECG au Repos")



# Barplot pour 'AngineExercice'  
barplot\_proportion("AngineExercice", data, "Proportion d'Angine d'Exercice")



# Barplot pour 'PenteST'  
barplot\_proportion("PenteST", data, "Proportion de Pente ST")



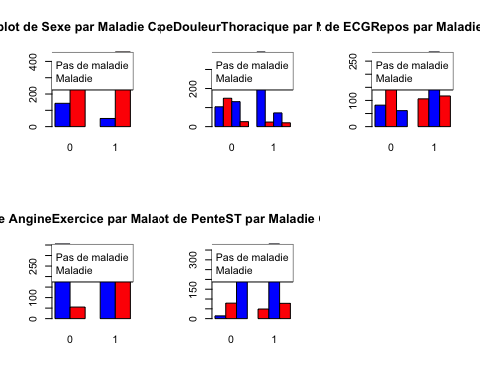
# Etape 4: Sélection des Caractéristiques pertinentes

## Sélection des variables qualitatives

-Faire les analyses graphiques -Faire des analyses de liaison via le test de chi deux, le v de cramer, analyser l’IV, le WOE et

### Analyse de barplot bivarié

# Création des barplots bivariés  
variables\_qualitatives <- c("Sexe", "TypeDouleurThoracique", "ECGRepos", "AngineExercice", "PenteST")  
par(mfrow = c(2, 3)) # Ajuster en fonction du nombre de variables qualitatives  
for (var in variables\_qualitatives) {  
 table\_var <- table(data[[var]], data$MaladieCardiaque)  
 barplot(table\_var, beside = TRUE, main = paste("Barplot de", var, "par Maladie Cardiaque"), col = c("blue", "red"))  
 legend("topright", legend = c("Pas de maladie", "Maladie"), fill = c("blue", "red"))  
}



# Création du tableau pour les résultats des tests  
results <- data.frame(Variable = character(), Chi\_square = numeric(), P\_value = numeric(), Cramers\_V = numeric())  
  
# Variables qualitatives  
variables\_qualitatives <- c("Sexe", "TypeDouleurThoracique", "ECGRepos", "AngineExercice", "PenteST")  
  
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), ncol(contingency\_table)) - 1)))  
   
 # Ajouter les résultats au tableau  
 results <- rbind(results, data.frame(Variable = var, Chi\_square = chi\_squared\_test$statistic, P\_value = chi\_squared\_test$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

# Variables numériques  
variables\_numeriques <- c("Age", "TensionRepos", "Cholesterol", "GlycemieAJeun", "FreqCardiaqueMax", "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$statistic, P\_value = kruskal\_test$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 multicolinéarité

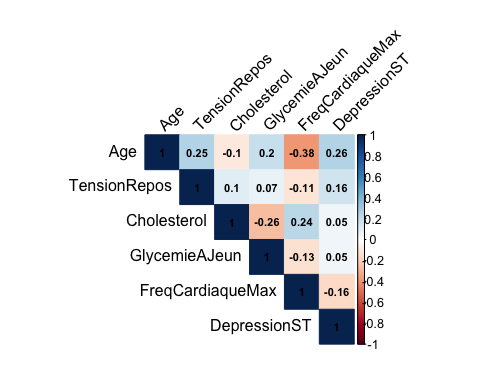
### Corrélation entre les grandeur numérique

# Sélectionner les variables numériques  
variables\_numeriques <- data[, c("Age", "TensionRepos", "Cholesterol", "GlycemieAJeun", "FreqCardiaqueMax", "DepressionST")]  
  
# Calculer la matrice de corrélation  
correlation\_matrix <- cor(variables\_numeriques, use = "complete.obs")  
  
# Installer et charger corrplot  
if (!require(corrplot)) install.packages("corrplot")

## Loading required package: corrplot

## corrplot 0.92 loaded

library(corrplot)  
  
# Créer la heatmap de corrélation  
corrplot(correlation\_matrix, method = "color", type = "upper",   
 tl.col = "black", tl.srt = 45, addCoef.col = "black",   
 number.cex = 0.7, number.digits = 2)



# Etape 5: Partitionnement des données

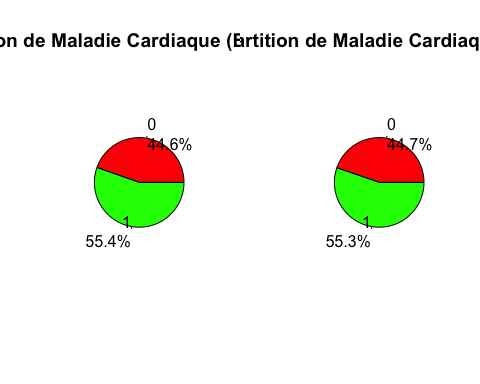
# Installer et charger le package caret si nécessaire  
if (!require(caret)) install.packages("caret")

## Loading required package: caret

## Loading required package: ggplot2

## Loading required 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\_entrainement, 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.5776 -0.3991 0.1878 0.4847 2.5233   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.842221 1.675548 -0.503 0.615208   
## Age 0.015273 0.015495 0.986 0.324290   
## SexeM 1.507029 0.323270 4.662 3.13e-06 \*\*\*  
## TypeDouleurThoraciqueATA -1.806232 0.385128 -4.690 2.73e-06 \*\*\*  
## TypeDouleurThoraciqueNAP -1.922537 0.312419 -6.154 7.57e-10 \*\*\*  
## TypeDouleurThoraciqueTA -1.904039 0.546820 -3.482 0.000498 \*\*\*  
## 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 \*\*\*  
## ECGReposNormal -0.389396 0.315018 -1.236 0.216419   
## ECGReposST -0.191001 0.394204 -0.485 0.628014   
## FreqCardiaqueMax -0.004912 0.006014 -0.817 0.414069   
## AngineExerciceY 0.648619 0.281296 2.306 0.021121 \*   
## DepressionST 0.286305 0.140361 2.040 0.041373 \*   
## PenteSTFlat 1.596272 0.500953 3.186 0.001440 \*\*   
## PenteSTUp -0.867916 0.525467 -1.652 0.098594 .   
## ---  
## 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)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from '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)

## 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

# 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)

## [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) 0.4307529  
## Age Age 1.0153907  
## SexeM SexeM 4.5133012  
## TypeDouleurThoraciqueATA TypeDouleurThoraciqueATA 0.1642719  
## TypeDouleurThoraciqueNAP TypeDouleurThoraciqueNAP 0.1462355  
## TypeDouleurThoraciqueTA TypeDouleurThoraciqueTA 0.1489657  
## TensionRepos TensionRepos 1.0041126  
## Cholesterol Cholesterol 0.9958234  
## GlycemieAJeun GlycemieAJeun 3.2671408  
## ECGReposNormal ECGReposNormal 0.6774661  
## ECGReposST ECGReposST 0.8261315  
## FreqCardiaqueMax FreqCardiaqueMax 0.9951004  
## AngineExerciceY AngineExerciceY 1.9128978  
## DepressionST DepressionST 1.3314986  
## PenteSTFlat PenteSTFlat 4.9345999  
## PenteSTUp PenteSTUp 0.4198254

# 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)  
}

## Loading required package: pROC

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

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

probas\_train <- predict(modele\_logistique, data\_entrainement, type = "response")  
probas\_test <- predict(modele\_logistique, data\_test, type = "response")  
  
roc\_train <- roc(response = data\_entrainement$MaladieCardiaque, predictor = probas\_train)

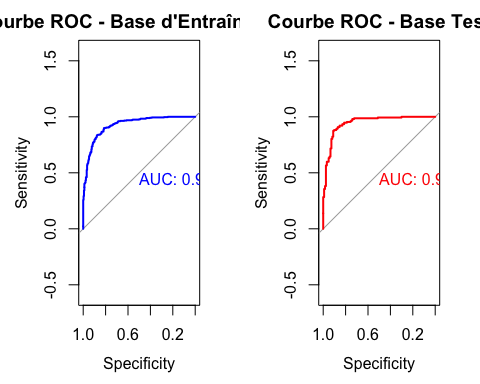
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

roc\_test <- roc(response = data\_test$MaladieCardiaque, predictor = probas\_test)

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

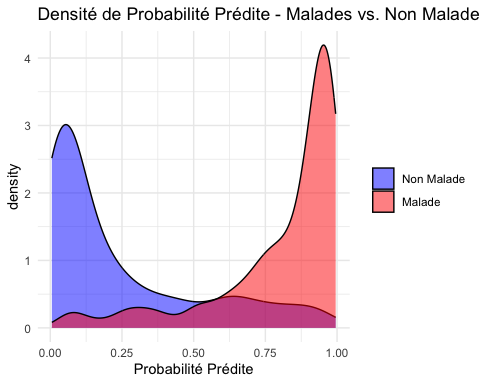
# Afficher les courbes ROC avec AUC  
par(mfrow=c(1,2)) # Afficher deux graphiques côte à côte  
plot(roc\_train, main = "Courbe ROC - Base d'Entraînement", col = "blue", print.auc = TRUE)  
plot(roc\_test, main = "Courbe ROC - Base Test", col = "red", print.auc = TRUE)



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

# Installer et charger la bibliothèque ggplot2 si elle n'est pas déjà installée  
if (!require(ggplot2)) {  
 install.packages("ggplot2")  
 library(ggplot2)  
}  
  
  
# 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 Prédite") +  
 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



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