

Projet : Introduction aux Essais Cliniques

Justine SAUCE - M2MIGS - 7 avril 2023

L'objectif général est de programmer un algorithme de randomisation dans le cadre d'un essai thérapeutique en utilisant la technique dite de minimisation. Il sera testé sur les données du fichier « projet-rando.csv ». A cet effet, trois facteurs de stratification sont à prendre en compte dans la programme :

- Le centre de suivi du patient : variable « centre »
- La résection ou non de la tumeur primitive: variable « tum_prim »
- Le critère de Köhne : variable catégorisée en trois niveaux (Bas, Intermédiaire, Elevé)

Préparation du jeu de données

```
#Importation  
Data <- read.csv2("projet-rando.csv")
```

```
#Aperçu des données  
head(Data)
```

```
##   id_patient centre leuco nb_meta oms pal tum_prim  
## 1          1    C05      1        1  1  1          2  
## 2          2    C05      1        1  1  1          2  
## 3          3    C02      1        1  1  1          2  
## 4          4    C05      1        1  1  2          2  
## 5          5    C02      1        1  0  1          2  
## 6          6    C01      1        3  0  1          1
```

On dispose déjà des variables « centre » et « tum_prim ». On va donc commencer par coder une nouvelle variable contenant le critère de Köhne.

```
# On définit n correspondant au nombre d'observations (patients)  
n = nrow(Data)  
  
# Initialisation de la variable du critere de Kohne, appelee CK  
CK = c()  
  
# On parcourt chaque observation et on determine le niveau du critere de Kohne  
# correspondant.  
for (i in 1:n){  
  
  # Niveau Bas : OMS=0/1 et 1 seul site tumoral métastatique  
  if (Data$oms[i] %in% c(0,1) & Data$nb_meta[i] == 1){  
    CK = append(CK, "Bas", after = length(CK))  
  }  
}
```

```

# Niveau Intermediaire : OMS=0/1, +1 site tumoral, phosphatases <= 300
# OU OMS > 1, globules blancs <= 10e10, 1 site tumoral
if ( ( Data$oms[i] %in% c(0,1) & Data$nb_meta[i] > 1 & Data$pal[i] == 1 ) |
      ( Data$oms[i] > 1 & Data$leuco[i] == 1 & Data$nb_meta[i] == 1 )
    ) {
  CK = append(CK, "Intermediaire", after = length(CK))
}

# Niveau Eleve : OMS=0/1, +1 site tumoral, phosphatases alcalines > 300
# OU OMS > 1, globules blancs <= 10e10, +1 site tumoral
# OU OMS > 1, globules blancs > 10e10
if ( ( Data$oms[i] %in% c(0,1) & Data$nb_meta[i] > 1 & Data$pal[i] == 2 ) |
      ( Data$oms[i] > 1 & Data$leuco[i] == 1 & Data$nb_meta[i] > 1 ) |
      ( Data$oms[i] > 1 & Data$leuco[i] == 2 )
    ) {
  CK = append(CK, "Eleve", after = length(CK))
}
}

# On combine notre variable avec le jeu de données
Data <- cbind(Data, CK)

# Aperçu
head(Data)

```

```

##   id_patient centre leuco nb_meta oms pal tum_prim      CK
## 1          1    C05     1      1  1  1          2      Bas
## 2          2    C05     1      1  1  1          2      Bas
## 3          3    C02     1      1  1  1          2      Bas
## 4          4    C05     1      1  1  2          2      Bas
## 5          5    C02     1      1  0  1          2      Bas
## 6          6    C01     1      3  0  1          1 Intermediaire

```

```

# Appel de la variable via le jeu de données
head(Data$CK)

```

```

## [1] "Bas"          "Bas"          "Bas"          "Bas"
## [5] "Bas"          "Intermediaire"

```

Maintenant que le jeu de données a été préparé on peut introduire un algorithme de randomisation.

Consigne 1

On souhaite proposer un algorithme pour une randomisation dans trois bras de traitement A, B et C suivant les ratios 3:2:1. L'algorithme sera basé sur l'approche BCM. On prendra $p^H = 0.72$ comme probabilité d'allocation du bras préféré lorsque le bras C est identifié à partir des scores de déséquilibre comme bras préféré. On programmera 2 métriques du score de déséquilibre, l'étendue et la variance.

On commence par construire une fonction calculant les probabilités d'allocation pour trois bras, avec comme bras préféré le bras C via l'approche BCM :

```

# Definition de la fonction

BCM_3bras_ABC <- function(ratio, pH){
  # Principe : Calculer les probabilités d'allocation via BCM.
  #
  # Entrees :
  #   ratio : Ratio d'allocation sous la forme (1,1) pour 1:1.
  #   pH    : Probabilité du traitement prefere (ici bras C)
  #
  # Sortie :
  #   Probas_Allo : Matrice contenant les probabilités d'allocation pour les
  #                 bras A x B x C.

  # Calcul des probabilités d'allocation
  # On considere les traitements A (1), B (2) et C (3)
  pC = pH

  pA = 1 - ( ( sum(ratio)-ratio[1] ) / ( sum(ratio)-ratio[3] ) ) * ( 1-pC )
  pB = 1 - ( ( sum(ratio)-ratio[2] ) / ( sum(ratio)-ratio[3] ) ) * ( 1-pC )

  pA_B = ( ratio[1] / ( sum(ratio)-ratio[2] ) ) * ( 1-pB )
  pB_A = ( ratio[2] / ( sum(ratio)-ratio[1] ) ) * ( 1-pA )
  pA_C = ( ratio[1] / ( sum(ratio)-ratio[3] ) ) * ( 1-pC )
  pC_A = ( ratio[3] / ( sum(ratio)-ratio[1] ) ) * ( 1-pA )
  pB_C = ( ratio[2] / ( sum(ratio)-ratio[3] ) ) * ( 1-pC )
  pC_B = ( ratio[3] / ( sum(ratio)-ratio[2] ) ) * ( 1-pB )

  # Creation d'une matrice pour les stocker
  Probas_Allo = matrix( c(pA, pA_B, pA_C, pB_A, pB, pB_C, pC_A, pC_B, pC),
                        nrow = 3 )

  # On renvoie la matrice
  return(Probas_Allo)
}

# Test d'appel pour verification
pH = 0.72
ratio = c(3,2,1)
Probas = BCM_3bras_ABC(ratio, pH)

print(Probas)

```

```

##      [,1] [,2] [,3]
## [1,] 0.832 0.112 0.056
## [2,] 0.168 0.776 0.056
## [3,] 0.168 0.112 0.720

```

On souhaite maintenant calculer les scores de déséquilibre selon trois variables de stratifications : « centre », « tum_prim » et « CK ». On doit définir pour cela une fonction, dont le but est de calculer les scores de déséquilibre pour chacun des bras, puis de renvoyer le bras pour lequel le score est minimal.

Cette fonction sera basée sur ce qu'on appellera par la suite un « tableau de traitement », en ligne on aura les trois traitements A, B et C puis en colonne les modalités des variables de stratification.

On propose de définir tout d'abord une fonction qui va créer un tableau de traitement vide avec les trois bras de traitement et les modalités des variables de stratification. A la suite de celle-ci on implémentera une fonction permettant sa mise à jour au fur et à mesure que de nouveaux patients sont affectés à un traitement.

```
# Initialisation tableau a 3 bras A, B et C, selon les variables de stratifications
```

```
Init_tab_traitement_3 <- function() {
  # Principe : Initialiser le tableau de traitement à 3 bras
  #
  # Sortie :
  #      Tableau_Traitement : Tableau de traitement vide.

  # Traitements
  Traitements <- c("A", "B", "C")

  # Variables de Stratifications, initialisees à 0
  # (Initialisation = encore aucun patient en traitement)

  # Modalites des 5 centres
  C01 <- c(0, 0, 0)
  C02 <- c(0, 0, 0)
  C03 <- c(0, 0, 0)
  C04 <- c(0, 0, 0)
  C05 <- c(0, 0, 0)

  # Modalites des trumeurs primitives 1 ou 2
  TP1 <- c(0, 0, 0)
  TP2 <- c(0, 0, 0)

  # Modalites du critere de Kohne
  Bas <- c(0, 0, 0)
  Intermediaire <- c(0, 0, 0)
  Eleve <- c(0, 0, 0)

  # On combine les données
  Tableau_Traitement <- data.frame(Traitements,
                                   C01, C02, C03, C04, C05,
                                   TP1, TP2,
                                   Bas, Intermediaire, Eleve)

  return (Tableau_Traitement)
}

# Aperçu du tableau vide
Tab_vide = Init_tab_traitement_3()
print(Tab_vide)
```

```
##   Traitements C01 C02 C03 C04 C05 TP1 TP2 Bas Intermediaire Eleve
## 1           A  0  0  0  0  0  0  0  0           0  0
## 2           B  0  0  0  0  0  0  0  0           0  0
## 3           C  0  0  0  0  0  0  0  0           0  0
```

On peut passer à la mise à jour du tableau au cours des différentes affectations des patients à un traitement. L'objectif est de remplir les modalités correspondant aux informations du patients.

C'est à dire que si un patient est, par exemple, dans le centre 1, qu'il n'a pas subi un acte de résection, que son critere de Kohne est de niveau Bas et qu'il est affecté au traitement B, alors on ajoutera 1 dans chacune de ces modalités pour la ligne correspondant au traitement B.

De cette façon, on pourra avoir accès à l'effectif des patients affectés à chacun des traitements par modalité des variables de stratification.

```
# Mise a jour du tableau de traitement

MaJ_tab_traitement_3 <- function (tab_actuel, patient, traitement) {
  # Principe : Mettre à jour le tableau de traitement à 3 bras
  #
  # Entrees :
  #   tab_traitement : Tableau actuel de traitement.
  #   patient       : Informations du patient sous le format impose par le
  #                   jeu de donnees.
  #   traitement    : Traitement pour lequel le patient a ete affecte.
  #
  # Sortie :
  #   Tableau_Traitement : Tableau de traitement mis a jour.

  # On recupere les informations en entree
  Tableau_Traitement = tab_actuel

  if ( traitement == "A" ) {
    k = 1
  }
  if ( traitement == "B" ) {
    k = 2
  }
  if ( traitement == "C" ) {
    k = 3
  }

  # On procede a la mise a jour par modalites

  # Pour la variable centre
  if ( patient$centre == "C01" ){
    Tableau_Traitement$C01[k] = Tableau_Traitement$C01[k] + 1
  }

  else if ( patient$centre == "C02" ){
    Tableau_Traitement$C02[k] = Tableau_Traitement$C02[k] + 1
  }
  else if ( patient$centre == "C03" ){
    Tableau_Traitement$C03[k] = Tableau_Traitement$C03[k] + 1
  }

  else if ( patient$centre == "C04" ){
    Tableau_Traitement$C04[k] = Tableau_Traitement$C04[k] + 1
  }
  else{
    Tableau_Traitement$C05[k] = Tableau_Traitement$C05[k] + 1
  }
}
```

```

# Pour la variable concernant la resection de la tumeur primitive
if ( patient$tum_prim == 1 ){
  Tableau_Traitement$TP1[k] = Tableau_Traitement$TP1[k] + 1
}
else{
  Tableau_Traitement$TP2[k] = Tableau_Traitement$TP2[k] + 1
}

# Pour la variable du critere de Kohne
if ( patient$CK == "Bas" ){
  Tableau_Traitement$Bas[k] = Tableau_Traitement$Bas[k] + 1
}
else if ( patient$CK == "Intermediaire" ){
  Tableau_Traitement$Int[k] = Tableau_Traitement$Intermediaire[k] + 1
}
else {
  Tableau_Traitement$Eleve[k] = Tableau_Traitement$Eleve[k] + 1
}

# On renvoie le tableau mis à jour
return (Tableau_Traitement)
}

```

On va tester cette fonction sur le premier patient, qu'on affecte au traitement A à titre d'essai.

```

Tab_test = Init_tab_traitement_3()
Tab_test = MaJ_tab_traitement_3(Tab_test, Data[1,], "A")

# Informations du patient
print(Data[1,])

```

```

##   id_patient centre leuco nb_meta oms pal tum_prim CK
## 1           1    C05      1      1  1  1          2 Bas

```

```

# Mise a jour effectuee
print(Tab_test)

```

```

##   Traitements C01 C02 C03 C04 C05 TP1 TP2 Bas Intermediaire Eleve
## 1           A  0  0  0  0  1  0  1  1           0  0
## 2           B  0  0  0  0  0  0  0  0           0  0
## 3           C  0  0  0  0  0  0  0  0           0  0

```

On doit maintenant définir une fonction calculant les scores de déséquilibre. Le but étant d'utiliser deux métriques différentes, on introduira en paramètre le choix de la métrique.

Les métriques utilisées seront la variance et l'étendue. Pour la variance on utilisera la fonction « var » de R, pour l'étendue on va tout de suite définir une petite fonction simple.

```

etendue <- function(values) {
  # Principe : Calcul l'étendue d'une liste de valeur.

  return (max(values)-min(values))
}

```

Finalement, on peut introduire notre fonction déterminant le bras de traitement préféré.

```
# Definition de la fonction determinant le traitement prefere en calculant les  
# scores de desequilibre.
```

```
Scores_3bras_ABC <- function (tab_actuel, patient, ratio, metrique){  
  # Principe : Calcul des scores de déséquilibre en fonction du tableau de  
  # traitement et renvoie le traitement prefere  
  #  
  # Entrees :  
  #   tab_actuel : Tableau de traitement actuel (k patients).  
  #   patient    : Information du prochain patient a affecter (patient k+1).  
  #   ratio      : Ratio d'allocation sous la forme (3,2,1) pour 3:2:1.  
  #   metrique   : Metrique choisie pour le calcul des scores.  
  #  
  # Sortie : Traitement prefere  
  
  # Pour calculer les scores on commence par creer trois nouveaux tableaux de  
  # traitement, un tableau ou on supposera le patient dans le bras A, un ou  
  # on le supposera dans le bras B et le dernier dans le bras C.  
  
  Tab_traitement_A = tab_actuel  
  Tab_traitement_B = tab_actuel  
  Tab_traitement_C = tab_actuel  
  
  # On effectue les mise a jour dans chacun des tableaux en fonction des  
  # informations du patient.  
  
  Tab_traitement_A = MaJ_tab_traitement_3(Tab_traitement_A, patient, "A")  
  Tab_traitement_B = MaJ_tab_traitement_3(Tab_traitement_B, patient, "B")  
  Tab_traitement_C = MaJ_tab_traitement_3(Tab_traitement_C, patient, "C")  
  
  # Division par les ratios  
  
  # Tableau pour le groupe A  
  Tab_traitement_A[1,-1] = Tab_traitement_A[1,-1]/ratio[1]  
  Tab_traitement_A[2,-1] = Tab_traitement_A[2,-1]/ratio[2]  
  Tab_traitement_A[3,-1] = Tab_traitement_A[3,-1]/ratio[3]  
  
  # Tableau pour le groupe B  
  Tab_traitement_B[1,-1] = Tab_traitement_B[1,-1]/ratio[1]  
  Tab_traitement_B[2,-1] = Tab_traitement_B[2,-1]/ratio[2]  
  Tab_traitement_B[3,-1] = Tab_traitement_B[3,-1]/ratio[3]  
  
  # Tableau pour le groupe C  
  Tab_traitement_C[1,-1] = Tab_traitement_C[1,-1]/ratio[1]  
  Tab_traitement_C[2,-1] = Tab_traitement_C[2,-1]/ratio[2]  
  Tab_traitement_C[3,-1] = Tab_traitement_C[3,-1]/ratio[3]  
  
  # Calcul des Scores  
  
  # Par centre  
  if ( patient$centre == "C01" ){  
    Score_centre_A = metrique(Tab_traitement_A$C01)
```

```

    Score_centre_B = metrique(Tab_traitement_B$C01)
    Score_centre_C = metrique(Tab_traitement_C$C01)
}

if ( patient$centre == "C02" ){
    Score_centre_A = metrique(Tab_traitement_A$C02)
    Score_centre_B = metrique(Tab_traitement_B$C02)
    Score_centre_C = metrique(Tab_traitement_C$C02)
}

if ( patient$centre == "C03" ){
    Score_centre_A = metrique(Tab_traitement_A$C03)
    Score_centre_B = metrique(Tab_traitement_B$C03)
    Score_centre_C = metrique(Tab_traitement_C$C03)
}

if ( patient$centre == "C04" ){
    Score_centre_A = metrique(Tab_traitement_A$C04)
    Score_centre_B = metrique(Tab_traitement_B$C04)
    Score_centre_C = metrique(Tab_traitement_C$C04)
}

if ( patient$centre == "C05" ){
    Score_centre_A = metrique(Tab_traitement_A$C05)
    Score_centre_B = metrique(Tab_traitement_B$C05)
    Score_centre_C = metrique(Tab_traitement_C$C05)
}

# Par tumeurs resequees ou non
if ( patient$tum_prim == 1 ){
    Score_tp_A = metrique(Tab_traitement_A$TP1)
    Score_tp_B = metrique(Tab_traitement_B$TP1)
    Score_tp_C = metrique(Tab_traitement_C$TP1)
}

if ( patient$tum_prim == 2 ){
    Score_tp_A = metrique(Tab_traitement_A$TP2)
    Score_tp_B = metrique(Tab_traitement_B$TP2)
    Score_tp_C = metrique(Tab_traitement_C$TP2)
}

# Par critere de Kohne
if ( patient$CK == "Bas" ){
    Score_ck_A = metrique(Tab_traitement_A$Bas)
    Score_ck_B = metrique(Tab_traitement_B$Bas)
    Score_ck_C = metrique(Tab_traitement_C$Bas)
}

if ( patient$CK == "Intermediaire" ){
    Score_ck_A = metrique(Tab_traitement_A$Intermediaire)
    Score_ck_B = metrique(Tab_traitement_B$Intermediaire)
    Score_ck_C = metrique(Tab_traitement_C$Intermediaire)
}

```

```

}

if ( patient$CK == "Eleve" ){
  Score_ck_A = metrique(Tab_traitement_A$Eleve)
  Score_ck_B = metrique(Tab_traitement_B$Eleve)
  Score_ck_C = metrique(Tab_traitement_C$Eleve)
}

Score_A = Score_centre_A + Score_tp_A + Score_ck_A
Score_B = Score_centre_B + Score_tp_B + Score_ck_B
Score_C = Score_centre_C + Score_tp_C + Score_ck_C

# On cherche le score minimal
Scores = c(Score_A, Score_B, Score_C)
Indice_Score_minimal = which(Scores==min(Scores))

if ( length(Indice_Score_minimal) == 1 ) {
  Tr_Score_min = Indice_Score_minimal
}

# On tire de facon equiprobable si le score minimum n'est pas unique
else {
  Tr_Score_min = sample(Indice_Score_minimal, size = 1)
}

# On renvoie le traitement d'affectation prefere
if ( Tr_Score_min == 1 ){
  return ("A")
}
else if ( Tr_Score_min == 2 ){
  return ("B")
}
else {
  return ("C")
}
}

```

On va essayer de tester cette fonction, avec notre tableau de test construit précédemment.

```

ratio = c(3,2,1)

Traitement_prefere = Scores_3bras_ABC(Tab_test, Data[2,], ratio, etendue)
print(Traitement_prefere)

```

```
## [1] "B"
```

On a maintenant à disposition les fonctions nécessaires à la création d'un algorithme de randomisation à 3 bras de traitement.

```

# Algorithme de randomisation a 3 bras

Algorithme_Radom_3bras_ABC <- function (Data, pH, ratio, metrique){

```

```

# Principe : Proposer une randomisation des patients en trois bras de
#             traitement A, B et C, selon les variables de stratification.
#
# Entrees :
#   Data      : Jeu de données.
#   pH        : Probabilite du traitement prefere.
#   ratio     : Ratio d'allocation sous la forme (3,2,1) pour 3:2:1.
#   metrique  : Metrique choisie pour le calcul des scores.
#
# Sortie :
#   Traitement : Vecteur contenant les traitements attribués par patient

# On definit n, le nombre de patients du jeu de donnees
n = nrow(Data)

# On commence par calculer les probabilites d'allocation via BCM
Probas_Allo = BCM_3bras_ABC(ratio, pH)

# On initialise le tableau de traitement
Tab_Traitement = Init_tab_traitement_3()

# On tire aleatoirement le traitement du premier patient
Random_Traitement = sample(c("A","B","C"), size = 1)

# On initialise un vecteur qui contiendra les traitements affectes
Traitement = c(Random_Traitement)

# On met a jour le tableau de traitement avec le premier patient
Tab_Traitement = MaJ_tab_traitement_3(Tab_Traitement,
                                       Data[1,],
                                       Random_Traitement)

# On parcourt le jeu de donnees pour affecter un traitement a chaque patient
for ( i in 2:n ) {

  # On cherche le traitement prefere
  Traitement_prefere = Scores_3bras_ABC (Tab_Traitement,
                                       Data[i,],
                                       ratio,
                                       metrique)

  # On retrouve les probabilites d'allocation associees a ce traitement
  if ( Traitement_prefere == "A" ) {
    indic_traitement_prefere = 1
  }
  else if ( Traitement_prefere == "B" ) {
    indic_traitement_prefere = 2
  }
  else {
    indic_traitement_prefere = 3
  }

  # On garde le traitement associe a la probabilite maximale

```

```

    Probas_traitement_pref = Probas_Allo[indic_traitement_prefere,]
    if ( which ( Probas_traitement_pref==max(Probas_traitement_pref) ) == 1 ) {
        Tr_prefere = "A"
    }
    else if ( which ( Probas_traitement_pref==max(Probas_traitement_pref) ) == 2 ) {
        Tr_prefere = "B"
    }
    else {
        Tr_prefere = "C"
    }

    # On met à jour le tableau en fonction du traitement prefere
    Tab_Traitement = MaJ_tab_traitement_3(Tab_Traitement,
                                           Data[i,],
                                           Tr_prefere)

    # On met a jour notre vecteur Traitement
    Traitement = append(Traitement, Tr_prefere, after = length(Traitement))
}

return (Traitement)
}

```

On va maintenant tester l'algorithme une première fois sur notre jeu de données.

```

pH = 0.72
ratio = c(3,2,1)

head(Algorithme_Radom_3bras_ABC(Data, pH, ratio, etendue))

```

```
## [1] "C" "A" "B" "A" "A" "A"
```

```
head(Algorithme_Radom_3bras_ABC(Data, pH, ratio, var))
```

```
## [1] "A" "B" "A" "C" "B" "A"
```

A partir de cet algorithme on peut simuler autant de randomisation que souhaité. Ici, on souhaite simuler 100 randomisations sur notre jeu de données et décrire les écarts aux ratios attendus pour l'ensemble des données, puis pour les modalités de la variable « CK ».

On va donc effectuer N=100 simulations, et calculer la moyenne des ratios obtenus via la randomisation.

```

# On definit les nombre de simulations
# Attention N = 100 ~ 5 minutes de temps de calcul
N = 100

# On definit les parametres des simulations
pH = 0.72
ratio = c(3,2,1)
metrique = etendue

# Declaration des vecteurs qui contiendront les ratios de chaque simulation

```

```

Data_ratio = c()
Data_ratio_Bas = c()
Data_ratio_Int = c()
Data_ratio_Elv = c()

# On lance N simulations
for (i in 1:N) {

  # On effectue une randomisation qu'on stocke dans le vecteur tr
  tr = Algorithme_Radom_3bras_ABC(Data, pH, ratio, metrique)

  # On ajoute une colonne a notre jeu de donnees correspondant au traitement affecte
  Data_simul = cbind(Data, tr)

  # On calcule les ratios obtenus par traitement par rapport au Bras de traitement C
  Data_ratio = rbind( Data_ratio,
                      prop.table(table(Data_simul$tr))/prop.table(table(Data_simul$tr))[3] )

  # Puis les ratios par rapport au Bras de traitement C, par modalités de la variable CK
  Data_ratio_Bas = rbind( Data_ratio_Bas,
                          prop.table(table(Data_simul$tr, Data$CK))[1,2]
                          /prop.table(table(Data_simul$tr, Data$CK))[3,1] )
  Data_ratio_Int = rbind( Data_ratio_Int,
                          prop.table(table(Data_simul$tr, Data$CK))[2,2]
                          /prop.table(table(Data_simul$tr, Data$CK))[3,2] )
  Data_ratio_Elv = rbind( Data_ratio_Elv,
                          prop.table(table(Data_simul$tr, Data$CK))[3,2]
                          /prop.table(table(Data_simul$tr, Data$CK))[3,3] )
}

```

On construit alors un petit Data frame pour stocker les écarts aux ratios comme suit.

```

Data_ecarts_ratios <- data.frame(A = abs(ratio[1] - Data_ratio[,1]),
                                B = abs(ratio[2] - Data_ratio[,2]),
                                Bas_A = abs(ratio[1] - Data_ratio_Bas[,1]),
                                Bas_B = abs(ratio[2] - Data_ratio_Bas[,2]),
                                Int_A = abs(ratio[1] - Data_ratio_Int[,1]),
                                Int_B = abs(ratio[2] - Data_ratio_Int[,2]),
                                Elv_A = abs(ratio[1] - Data_ratio_Elv[,1]),
                                Elv_B = abs(ratio[2] - Data_ratio_Elv[,2]) )

# Aperçu du Data frame
head(Data_ecarts_ratios)

```

```

##           A           B      Bas_A      Bas_B      Int_A      Int_B      Elv_A
## 1 0.00000000 0.00000000 0.03333333 0.00000000 0.14285714 0.14285714 0.02702703
## 2 0.03658537 0.03658537 0.00000000 0.03333333 0.31250000 0.18750000 0.05555556
## 3 0.04878049 0.02439024 0.00000000 0.03333333 0.31250000 0.18750000 0.02777778
## 4 0.00000000 0.00000000 0.13793103 0.03448276 0.21428571 0.07142857 0.18421053
## 5 0.04878049 0.02439024 0.00000000 0.03333333 0.14285714 0.14285714 0.15789474
## 6 0.04878049 0.02439024 0.00000000 0.03333333 0.06666667 0.06666667 0.08108108
##           Elv_B
## 1 0.05405405

```

```
## 2 0.02777778
## 3 0.05555556
## 4 0.05263158
## 5 0.07894737
## 6 0.00000000
```

On va représenter graphiquement la distribution des écarts entre les ratios observés et souhaités.

Remarque : On ne représente pas les écarts pour le traitement C, étant donné que c'est notre référence pour le calcul des ratios, ils seront nuls.

```
# On stocke la densité des écarts pour le bras A et pour le B dans 2 variables
dA <- density(Data_ecarts_ratios$A)
dB <- density(Data_ecarts_ratios$B)

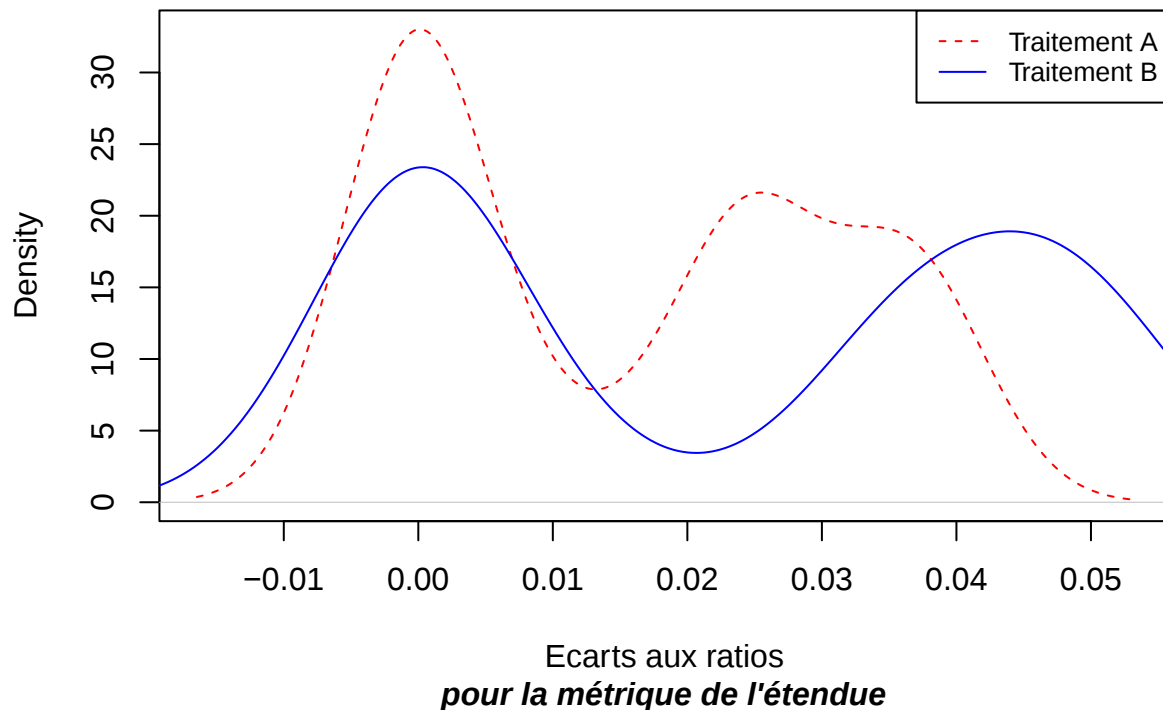
# On représente graphiquement la densité des écarts au ratio pour le bras A
plot(dB, type = "l", col = "red", pch = 4, lty = 2,
     ylim = c(0, max(max(dA$y), max(dB$y))),
     xlab = "Ecart aux ratios",
     main = " ")

# On ajoute à la courbe la densité pour le bras B
lines(dA, type = "l", col = "blue", pch = 4, lty = 1)

legend("topright", legend = c("Traitement A", "Traitement B"), col = c("red", "blue"),
      lty = 2:1, cex=0.8)

title(main = "Distribution des écarts aux ratios pour les traitements A et B",
      sub = "pour la métrique de l'étendue", font.sub=4)
```

Distribution des écarts aux ratios pour les traitement A et B



On constate une variance relativement faible, et une concentration des valeurs entre 0 et 0.05. Sur l'ensemble des patients les écarts aux ratio sont corrects.

On représente maintenant graphiquement les écarts aux ratios par traitement au sein de chaque modalité de la variable du critère de Kohne.

```
# On divise la représentation en 3 graphiques pour chaque modalités
par(mfrow=c(3,1))

# On procède de la même façon que précédemment
dBA <- density(Data_ecarts_ratios$Bas_A)
dBB <- density(Data_ecarts_ratios$Bas_B)
dIA <- density(Data_ecarts_ratios$Int_A)
dIB <- density(Data_ecarts_ratios$Int_B)
dEA <- density(Data_ecarts_ratios$Elv_A)
dEB <- density(Data_ecarts_ratios$Elv_B)

# Premier graphique pour le niveau Bas
plot(dBA, type = "l", col = "red", pch = 4, lty = 2,
     ylim = c(0, max(max(dBA$y), max(dBB$y))),
     xlab = "Ecart aux ratios",
     main = "Distribution des écarts aux ratios modalité de niveau Bas")
lines(dBB, type = "l", col = "blue", pch = 4, lty = 1)
legend("topright", legend = c("Traitement A", "Traitement B"), col = c("red", "blue"),
     lty = 2:1, cex=0.8)

# Deuxième graphique pour le niveau Intermediaire
```

```

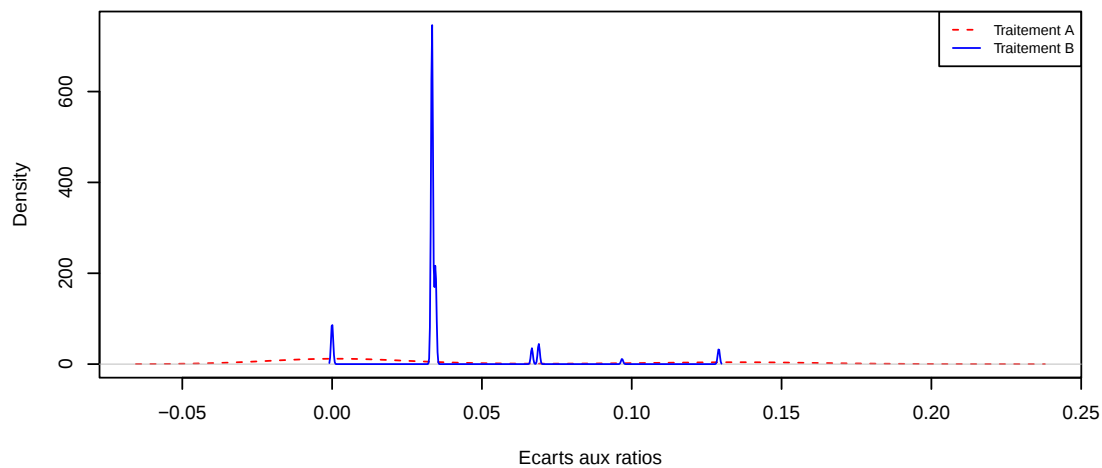
plot(dIA,type = "l", col = "red", pch = 4, lty = 2,
     ylim = c(0, max(max(dIA$y), max(dIB$y))),
     xlab = "Ecart aux ratios",
     main = "Distribution des écarts aux ratios modalité de niveau Intermédiaire")
lines(dIB, type = "l", col = "blue", pch = 4, lty = 1)
legend("topright", legend = c("Traitement A", "Traitement B"), col = c("red", "blue"),
      lty = 2:1, cex=0.8)

# Troisieme graphique pour le niveau Eleve
plot(dEA, type = "l", col = "red", pch = 4, lty = 2,
     ylim = c(0, max(max(dEA$y), max(dEB$y))),
     xlab = "Ecart aux ratios",
     main = "Distribution des écarts aux ratios modalité de niveau Elevé")
lines(dEB, type = "l", col = "blue", pch = 4, lty = 1)
legend("topright", legend = c("Traitement A", "Traitement B"), col = c("red", "blue"),
      lty = 2:1, cex=0.8)

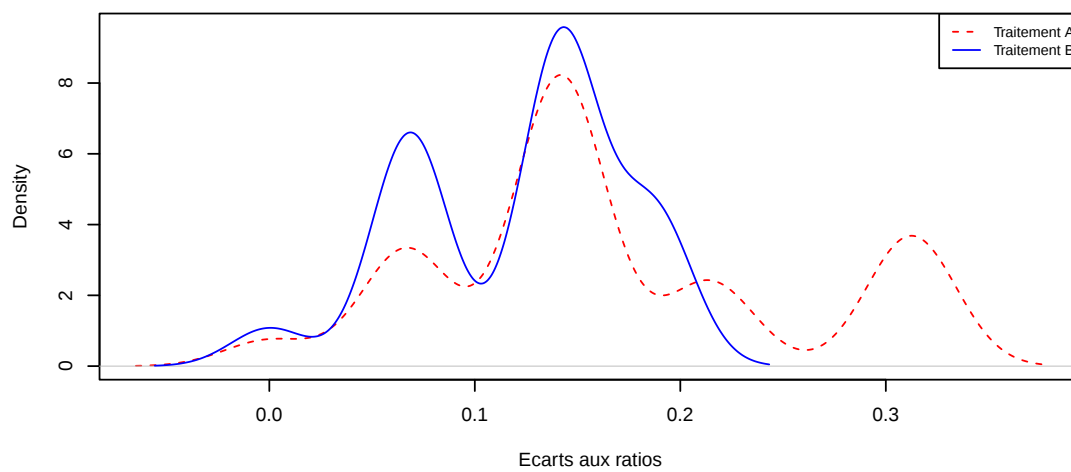
title (sub = "pour la métrique de l'étendue", font.sub=4)

```

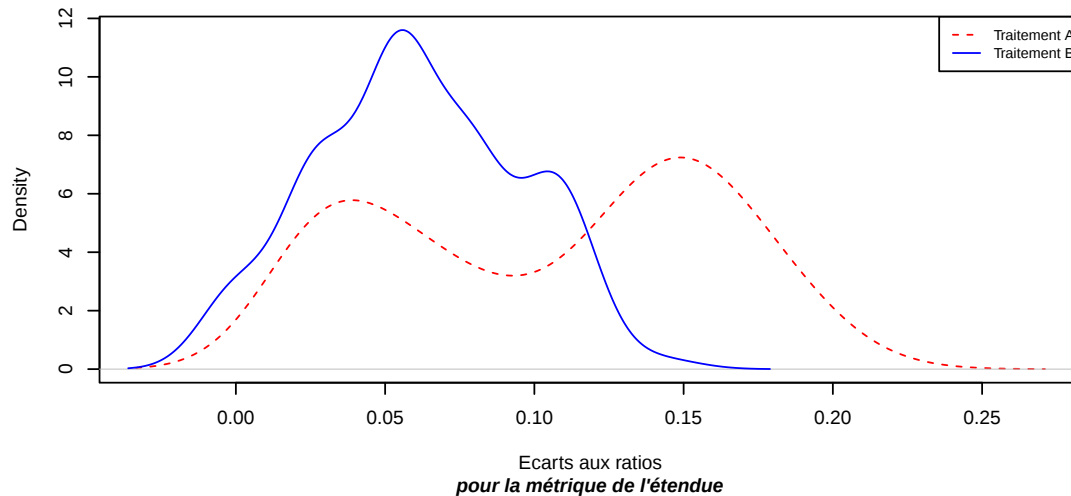
Distribution des écarts aux ratios modalité de niveau Bas



Distribution des écarts aux ratios modalité de niveau Intermédiaire



Distribution des écarts aux ratios modalité de niveau Elevé



pour la métrique de l'étendue

On constate que pour la modalité de niveau Bas, les écarts sont très concentrés autour de 0 et inférieurs à 0.05 à l'exception de quelques valeurs atypique dans le bras de traitement A.

Pour le niveau Intermédiaire, le respect des ratio est légèrement plus difficile de manière générale, puisqu'on a une plus grande variance et que la plus forte concentration se situe autour de 0.15. Cependant, cela reste relativement acceptable.

Pour le niveau Elevé, de même la variance est plus grande dans le bras A, l'algorithme a dû faire plus de compromis mais les écarts restent faibles et très corrects.

Enfin, pour clôturer notre randomisation sur notre jeu de données, on aimerait comparer les métriques de la variance et de l'étendue. Jusqu'à présent, les graphiques étaient réalisés sur la base de l'étendue. On va donc devoir créer un Data frame pour la variance, avec $N = 100$ simulations. Pour la comparaison, on se propose d'étudier les écarts aux ratios généraux et non en fonction du traitement. On souhaite aussi effectuer une comparaison par modalités de la variable CK.

```
# On definit les nombre de simulations
# Attention N = 100 ~ 5 minutes de temps de calcul
N = 100

# On definit les parametres des simulations
pH = 0.72
ratio = c(3,2,1)
metrique = var

Data_ratio = c()
Data_ratio_Bas = c()
Data_ratio_Int = c()
Data_ratio_Elv = c()

# on procede de la meme maniere que pour l'etendue, cette fois-ci avec la variance
for (i in 1:N) {
  tr = Algorithme_Radom_3bras_ABC(Data, pH, ratio, metrique)

  Data_simul = cbind(Data,tr)

  # Ratios obtenus par traitement par rapport au Bras de traitement C
  Data_ratio = rbind( Data_ratio,
                      prop.table(table(Data_simul$tr))/prop.table(table(Data_simul$tr))[3] )

  # Ratios par rapport au Bras de traitement C, par modalités de la variable CK
  Data_ratio_Bas = rbind( Data_ratio_Bas,
                          prop.table(table(Data_simul$tr, Data$CK))[,1]
                          /prop.table(table(Data_simul$tr, Data$CK))[3,1] )
  Data_ratio_Int = rbind( Data_ratio_Int,
                          prop.table(table(Data_simul$tr, Data$CK))[,2]
                          /prop.table(table(Data_simul$tr, Data$CK))[3,2] )
  Data_ratio_Elv = rbind( Data_ratio_Elv,
                          prop.table(table(Data_simul$tr, Data$CK))[,3]
                          /prop.table(table(Data_simul$tr, Data$CK))[3,3] )
}
```

On stocke nos résultats.

```

# On creee notre Data Frame pour la metrique de la variance
Data_ecarts_ratios_2 <- data.frame(A = abs(ratio[1] - Data_ratio[,1]),
                                   B = abs(ratio[2] - Data_ratio[,2]),
                                   Bas_A = abs(ratio[1] - Data_ratio_Bas[,1]),
                                   Bas_B = abs(ratio[2] - Data_ratio_Bas[,2]),
                                   Int_A = abs(ratio[1] - Data_ratio_Int[,1]),
                                   Int_B = abs(ratio[2] - Data_ratio_Int[,2]),
                                   Elv_A = abs(ratio[1] - Data_ratio_Elv[,1]),
                                   Elv_B = abs(ratio[2] - Data_ratio_Elv[,2]))

# On ne distingue plus les traitements(1)
Data_var = data.frame(General = matrix(c(Data_ecarts_ratios_2$A,
                                          Data_ecarts_ratios_2$B),ncol = 1),
                      Bas = matrix(c(Data_ecarts_ratios_2$Bas_A,
                                      Data_ecarts_ratios_2$Bas_B),ncol = 1),
                      Int = matrix(c(Data_ecarts_ratios_2$Int_A,
                                      Data_ecarts_ratios_2$Int_B),ncol = 1),
                      Elv = matrix(c(Data_ecarts_ratios_2$Elv_A,
                                      Data_ecarts_ratios_2$Elv_B),ncol = 1))

Data_etd = data.frame(General = matrix(c(Data_ecarts_ratios$A,
                                          Data_ecarts_ratios$B),ncol = 1),
                      Bas = matrix(c(Data_ecarts_ratios$Bas_A,
                                      Data_ecarts_ratios$Bas_B),ncol = 1),
                      Int = matrix(c(Data_ecarts_ratios$Int_A,
                                      Data_ecarts_ratios$Int_B),ncol = 1),
                      Elv = matrix(c(Data_ecarts_ratios$Elv_A,
                                      Data_ecarts_ratios$Elv_B),ncol = 1))

```

(1) pour plus de lisibilité sur les graphiques proposés.

On représente maintenant nos résultats graphiquement. On commence par l'ensemble des patients.

```

par(mfrow=c(1,1))

d_var <- density(Data_var$General)
d_etd <- density(Data_etd$General)

plot(d_var,type = "l", col = "green", pch = 4, lty = 2,
     ylim = c(0, max(max(d_var$y), max(d_etd$y))),
     xlab = "Ecart aux ratios",
     main = " ")

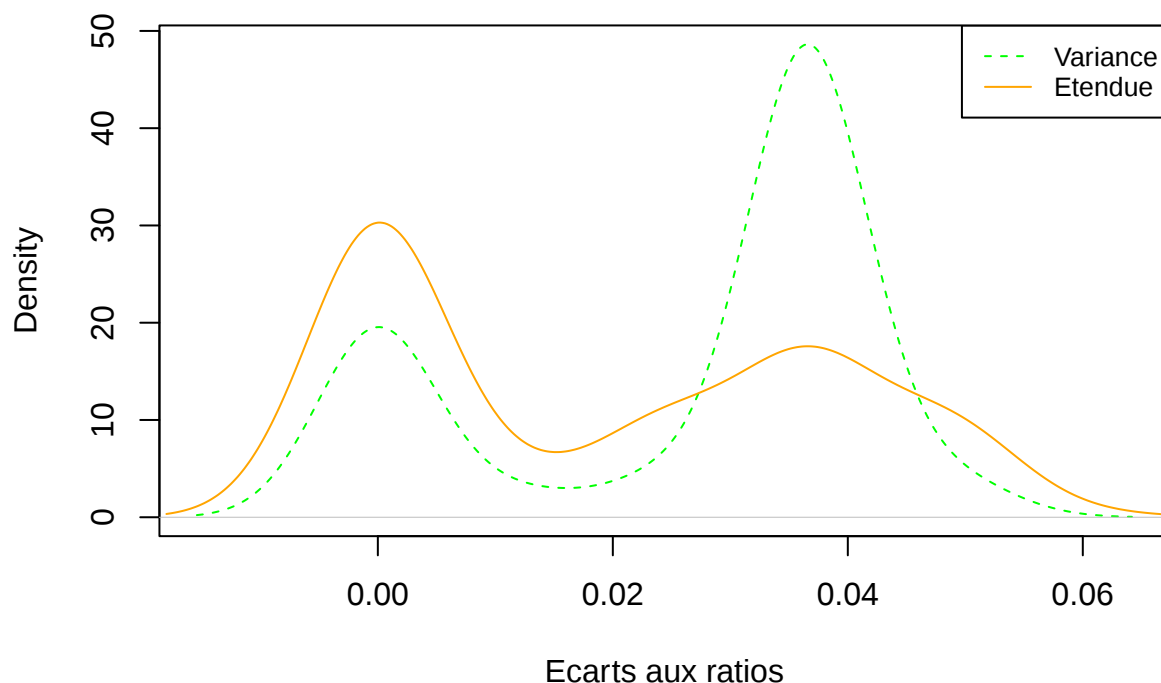
lines(d_etd, type = "l", col = "orange", pch = 4, lty = 1)

legend("topright", legend = c("Variance", "Etendue"), col = c("green", "orange"),
      lty = 2:1, cex=0.8)

title(main = "Distribution des écarts aux ratios")

```

Distribution des écarts aux ratios



D'après cette représentation, l'étendue semble être un meilleur choix de métrique pour ce jeu de données. Les écarts sont plus concentrés autour de 0 avec celle-ci qu'avec la métrique de la variance.

On peut aussi effectuer une comparaison des moyennes, comme suit.

```
moy_var <- mean(Data_var$General)
moy_etd <- mean(Data_etd$General)

cat("La moyenne avec la metrique de la variance vaut", moy_var)
```

```
## La moyenne avec la metrique de la variance vaut 0.02671183
```

```
cat("La moyenne avec la metrique de l'etendue vaut", moy_etd)
```

```
## La moyenne avec la metrique de l'etendue vaut 0.01915236
```

De cette façon, on voit à nouveau que la métrique de l'étendue est très légèrement meilleure.

On regarde maintenant par modalités de la variable CK.

```
# On divise la représentation en 3 graphiques pour chaque modalités
par(mfrow=c(3,1))

d_var_bas <- density(Data_var$Bas)
d_etd_bas <- density(Data_etd$Bas)
```

```

d_var_int <- density(Data_var$Int)
d_etd_int <- density(Data_etd$Int)
d_var_elv <- density(Data_var$Elv)
d_etd_elv <- density(Data_etd$Elv)

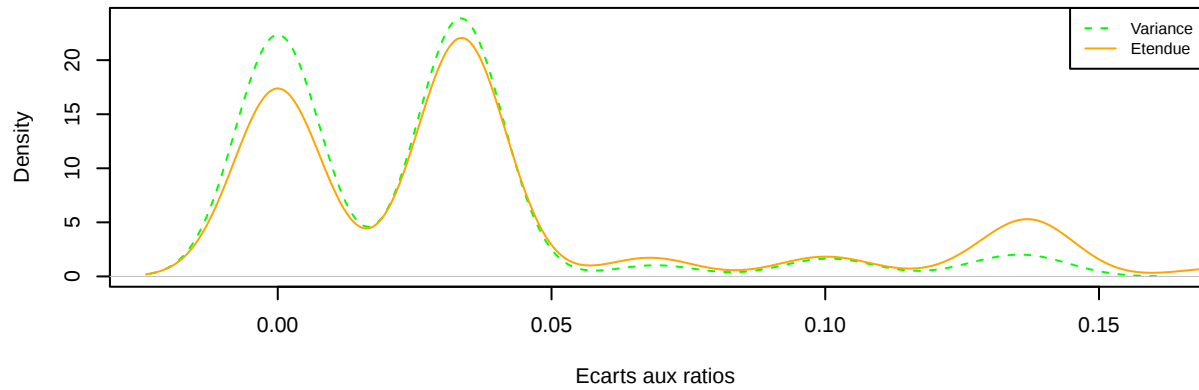
plot(d_var_bas, type = "l", col = "green", pch = 4, lty = 2,
     ylim = c(0, max(max(d_var_bas$y), max(d_etd_bas$y))),
     xlab = "Ecart aux ratios",
     main = "Distribution des écarts aux ratio modalité de niveau Bas")
lines(d_etd_bas, type = "l", col = "orange", pch = 4, lty = 1)
legend("topright", legend = c("Variance", "Etendue"), col = c("green", "orange"),
     lty = 2:1, cex=0.8)

plot(d_var_int, type = "l", col = "green", pch = 4, lty = 2,
     ylim = c(0, max(max(d_var_int$y), max(d_etd_int$y))),
     xlab = "Ecart aux ratios",
     main = "Distribution des écarts aux ratio modalité de niveau Intermédiaire")
lines(d_etd_int, type = "l", col = "orange", pch = 4, lty = 1)
legend("topright", legend = c("Variance", "Etendue"), col = c("green", "orange"),
     lty = 2:1, cex=0.8)

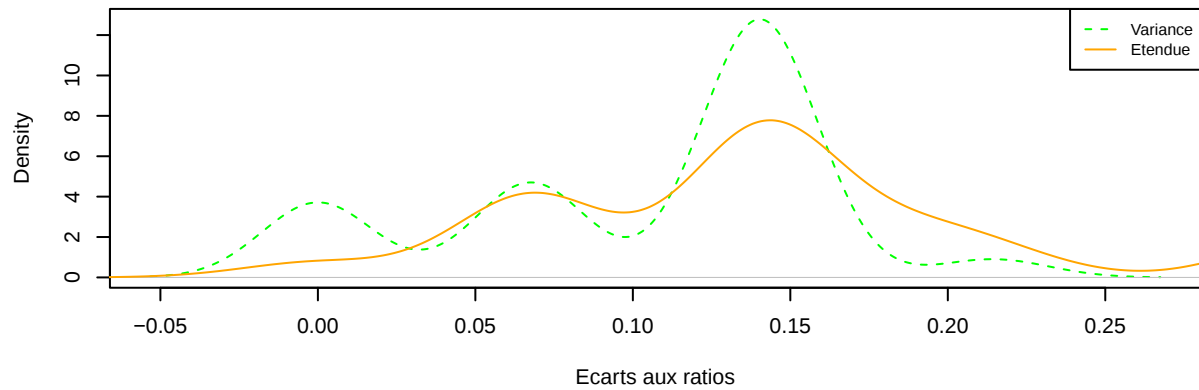
plot(d_var_elv, type = "l", col = "green", pch = 4, lty = 2,
     ylim = c(0, max(max(d_var_elv$y), max(d_etd_elv$y))),
     xlab = "Ecart aux ratios",
     main = "Distribution des écarts aux ratio modalité de niveau Elevé")
lines(d_etd_elv, type = "l", col = "orange", pch = 4, lty = 1)
legend("topright", legend = c("Variance", "Etendue"), col = c("green", "orange"),
     lty = 2:1, cex=0.8)

```

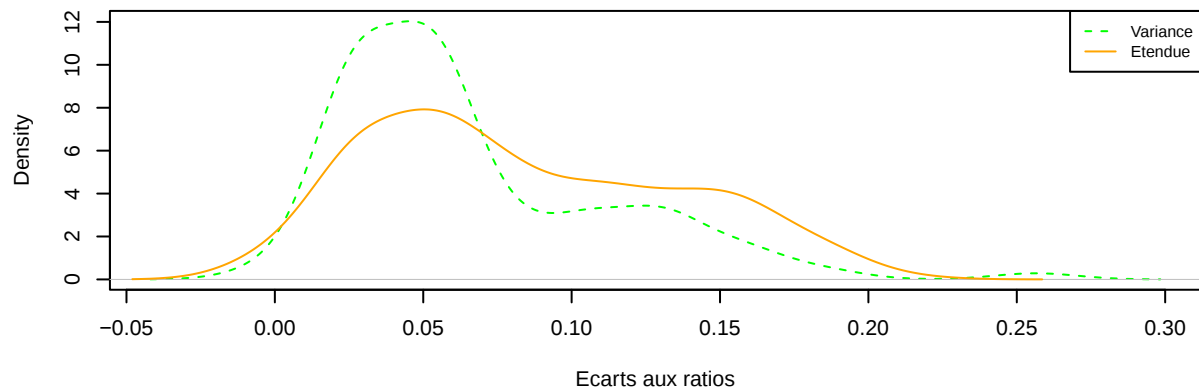
Distribution des écarts aux ratio modalité de niveau Bas



Distribution des écarts aux ratio modalité de niveau Intermédiaire



Distribution des écarts aux ratio modalité de niveau Elevé



L'étendue semble très très légèrement meilleure notamment pour la modalité de niveau élevé. Mais les deux métriques se valent, il n'y pas de vraie différence.

Consigne 2

On souhaite proposer un algorithme pour une randomisation dans deux bras de traitements D et E suivant le ratio 1:1. On prendra la variance comme métrique du score de déséquilibre et $p^H = 0.8$ comme probabilité d'allocation du bras préféré donné par les scores de déséquilibre.

On commence par construire une fonction calculant les probabilités d'allocation pour deux bras via l'approche BCM :

```
# Definition de la fonction

BCM_2bras_DE <- function(ratio, pH){
  # Principe : Calculer les probabilites d'allocation via BCM.
  #
  # Entrees :
  #   ratio : Ratio d'allocation sous la forme (1,1) pour 1:1.
  #   pH    : Probabilite du traitement prefere.
  #
  # Sortie :
  #   Probas_Allo : Matrice contenant les probabilites d'allocation pour les
  #                 bras D x E.

  # Calcul des probabilites d'allocations
  pD = pH
  pE = 1 - ( ( sum(ratio)-ratio[2] ) / ( sum(ratio)-ratio[1] ) ) * ( 1-pD )

  pD_E = ( ratio[1] / ( sum(ratio)-ratio[2] ) ) * ( 1-pE )
  pE_D = ( ratio[2] / ( sum(ratio)-ratio[1] ) ) * ( 1-pD )

  # Creation d'une matrice pour les stocker
  Probas_Allo = matrix(c(pD,pD_E,pE_D,pE), nrow = 2)

  # On renvoie la matrice
  return(Probas_Allo)
}

# Test d'appel pour verification
pH = 0.8
ratio = c(1,1)
Probas = BCM_2bras_DE(ratio, pH)

print(Probas)
```

```
##      [,1] [,2]
## [1,]  0.8  0.2
## [2,]  0.2  0.8
```

On souhaite maintenant calculer les scores de déséquilibre selon trois variables de stratification. Pour rappel, ces variables sont : « centre », « tum_prim » et « CK ». On va donc créer une nouvelle fonction, dont le but est de renvoyer le bras de traitement préféré en s'appuyant sur les scores de déséquilibre des bras D et E.

Pour construire cette fonction on souhaite se baser sur un tableau de traitement, en ligne on aura les deux traitements D et E et en colonne les modalités des variables de stratification. Pour chaque patient affecté à un des deux traitements, on remplira les modalités qui correspondent aux informations du patient.

On va donc construire une initialisation du tableau et une fonction permettant sa mise à jour au fur et à mesure que de nouveaux patients seront affectés à un traitement.

```
# Initialisation tableau a 2 bras D et E, selon les variables de stratification
```

```
Init_tab_traitement <- function() {
  # Principe : Initialiser le tableau de traitement.
  #
  # Sortie :
  #   Tableau_Traitement : Tableau de traitement vide.

  # Traitements
  Traitements <- c("D", "E")

  # Variables de Stratifications, initialisees à 0
  # (Initialisation = encore aucun patient en traitement)

  # Modalites des 5 centres
  C01 <- c(0, 0)
  C02 <- c(0, 0)
  C03 <- c(0, 0)
  C04 <- c(0, 0)
  C05 <- c(0, 0)

  # Modalites des trumeurs primitives 1 ou 2
  TP1 <- c(0, 0)
  TP2 <- c(0, 0)

  # Modalites du critere de Kohne
  Bas <- c(0, 0)
  Intermediaire <- c(0, 0)
  Eleve <- c(0, 0)

  # On combine les données
  Tableau_Traitement <- data.frame(Traitements,
                                   C01, C02, C03, C04, C05,
                                   TP1, TP2,
                                   Bas, Intermediaire, Eleve)

  return (Tableau_Traitement)
}

# Aperçu du tableau vide
Tab_vide = Init_tab_traitement()
print(Tab_vide)
```

```
##   Traitements C01 C02 C03 C04 C05 TP1 TP2 Bas Intermediaire Eleve
## 1           D   0   0   0   0   0   0   0   0           0     0
## 2           E   0   0   0   0   0   0   0   0           0     0
```

On introduit maintenant une fonction permettant la mise à jour du tableau au cours des différentes affectations des patients à un traitement. L'objectif est de remplir les modalités correspondant aux informations du patient.

C'est à dire que si un patient est, par exemple, dans le centre 5, qu'il a subi un acte de résection, que son critere de Köhne est de niveau Intermediaire et qu'il est affecté au traitement D, alors on ajoutera 1 dans chacune de ces modalités pour la ligne correspondant au traitement D.

Ainsi, on disposera d'un effectif par modalité des variables de stratification et par traitement.

```
# Mise a jour du tableau de traitement

MaJ_tab_traitement <- function (tab_actuel, patient, traitement) {
  # Principe : Mettre à jour le tableau de traitement
  #
  # Entrees :
  #   tab_traitement      : Tableau actuel de traitement.
  #   patients           : Informations du patient sous le format impose par le
  #                       jeu de donnees.
  #   traitement         : Traitement pour lequel le patient a ete affecte.
  #
  # Sortie :
  #   Tableau_Traitement : Tableau de traitement mis a jour.

  # On recupere les informations en entrees
  Tableau_Traitement = tab_actuel
  if ( traitement == "D" ) {
    k = 1
  }
  else {
    k = 2
  }

  # On procede a la mise a jour par modalites
  if ( patient$centre == "C01" ){
    Tableau_Traitement$C01[k] = Tableau_Traitement$C01[k] + 1
  }

  if ( patient$centre == "C02" ){
    Tableau_Traitement$C02[k] = Tableau_Traitement$C02[k] + 1
  }
  if ( patient$centre == "C03" ){
    Tableau_Traitement$C03[k] = Tableau_Traitement$C03[k] + 1
  }

  if ( patient$centre == "C04" ){
    Tableau_Traitement$C04[k] = Tableau_Traitement$C04[k] + 1
  }
  else{
    Tableau_Traitement$C05[k] = Tableau_Traitement$C05[k] + 1
  }

  if ( patient$tum_prim == 1 ){
    Tableau_Traitement$TP1[k] = Tableau_Traitement$TP1[k] + 1
  }
  else{
    Tableau_Traitement$TP2[k] = Tableau_Traitement$TP2[k] + 1
  }
}
```

```

if ( patient$CK == "Bas" ){
  Tableau_Traitement$Bas[k] = Tableau_Traitement$Bas[k] + 1
}
if ( patient$CK == "Intermediaire" ){
  Tableau_Traitement$Int[k] = Tableau_Traitement$Intermediaire[k] + 1
}
else{
  Tableau_Traitement$Eleve[k] = Tableau_Traitement$Eleve[k] + 1
}

# On renvoie le tableau mis à jour
return (Tableau_Traitement)
}

```

On va tester cette fonction sur le premier patient, qu'on affecte au traitement D à titre d'essai.

```

Tab_test = Init_tab_traitement()
Tab_test = MaJ_tab_traitement(Tab_test, Data[1,], "D")

# Informations du patient
print(Data[1,])

```

```

##   id_patient centre leuco nb_meta oms pal tum_prim CK
## 1           1    C05      1      1  1  1          2 Bas

```

```

# Mise a jour effectuee
print(Tab_test)

```

```

##   Traitements C01 C02 C03 C04 C05 TP1 TP2 Bas Intermediaire Eleve
## 1           D   0   0   0   0   1   0   1   1           0     1
## 2           E   0   0   0   0   0   0   0   0           0     0

```

On peut maintenant définir une fonction calculant les scores de déséquilibre, ici on choisit la variance comme métrique.

```

# Definition de la fonction renvoyant le bras prefere via les scores de
# desequilibre.

Scores_2bras_DE <- function (tab_actuel, patient, ratio){
  # Principe : Calcul de es scores de déséquilibres en fonction du tableau de
  # traitement et renvoie le traitement prefere
  #
  # Entrees :
  #   tab_actuel : Tableau de traitement actuel (k patients).
  #   patient    : Information du prochain patient a affecter (patient k+1).
  #   ratio      : Ratio d'allocation sous la forme (1,1) pour 1:1.
  #
  # Sortie : Traitement prefere

  # Pour calculer les scores on commence par creer deux nouveaux tableaux de

```

```
# traitement, un tableau ou on supposera le patient dans le groupe D et le  
# second ou on le supposera dans le groupe E.
```

```
Tab_traitement_D = tab_actuel  
Tab_traitement_E = tab_actuel
```

```
# On effectue les mises a jour dans chacun des tableaux en fonction des  
# informations du patient.
```

```
Tab_traitement_D = MaJ_tab_traitement(Tab_traitement_D, patient, "D")  
Tab_traitement_E = MaJ_tab_traitement(Tab_traitement_E, patient, "E")
```

```
# Division par les ratios
```

```
Tab_traitement_D[1,-1] = Tab_traitement_D[1,-1]/ratio[1]  
Tab_traitement_D[2,-1] = Tab_traitement_D[2,-1]/ratio[2]  
Tab_traitement_E[1,-1] = Tab_traitement_E[1,-1]/ratio[1]  
Tab_traitement_E[2,-1] = Tab_traitement_E[2,-1]/ratio[2]
```

```
# Calcul des Scores
```

```
# Par centre
```

```
if ( patient$centre == "C01" ){  
  Score_centre_D = var(Tab_traitement_D$C01)  
  Score_centre_E = var(Tab_traitement_E$C01)  
}
```

```
if ( patient$centre == "C02" ){  
  Score_centre_D = var(Tab_traitement_D$C02)  
  Score_centre_E = var(Tab_traitement_E$C02)  
}
```

```
if ( patient$centre == "C03" ){  
  Score_centre_D = var(Tab_traitement_D$C03)  
  Score_centre_E = var(Tab_traitement_E$C03)  
}
```

```
if ( patient$centre == "C04" ){  
  Score_centre_D = var(Tab_traitement_D$C04)  
  Score_centre_E = var(Tab_traitement_E$C04)  
}
```

```
if ( patient$centre == "C05" ){  
  Score_centre_D = var(Tab_traitement_D$C05)  
  Score_centre_E = var(Tab_traitement_E$C05)  
}
```

```
# Par tumeurs resequees ou non
```

```
if ( patient$tum_prim == 1 ){  
  Score_tp_D = var(Tab_traitement_D$TP1)  
  Score_tp_E = var(Tab_traitement_E$TP1)  
}
```

```
if ( patient$tum_prim == 2 ){
```

```

    Score_tp_D = var(Tab_traitement_D$TP2)
    Score_tp_E = var(Tab_traitement_E$TP2)
  }

  #Par critere de Kohne
  if ( patient$CK == "Bas" ){
    Score_ck_D = var(Tab_traitement_D$Bas)
    Score_ck_E = var(Tab_traitement_E$Bas)
  }

  if ( patient$CK == "Intermediaire" ){
    Score_ck_D = var(Tab_traitement_D$Intermediaire)
    Score_ck_E = var(Tab_traitement_E$Intermediaire)
  }

  if ( patient$CK == "Eleve" ){
    Score_ck_D = var(Tab_traitement_D$Eleve)
    Score_ck_E = var(Tab_traitement_E$Eleve)
  }

  Score_D = Score_centre_D + Score_tp_D + Score_ck_D
  Score_E = Score_centre_E + Score_tp_E + Score_ck_E

  # On cherche le score minimal
  Scores = c(Score_D,Score_E)
  Indice_Score_minimal = which(Scores==min(Scores))

  if ( length(Indice_Score_minimal) == 1 ) {
    Tr_Score_min = Indice_Score_minimal
  }

  # On tire de facon equiprobable si le score minimum n'est pas unique
  else {
    Tr_Score_min = sample(Indice_Score_minimal, size = 1)
  }

  # On renvoie le traitement d'affectation prefere
  if ( Tr_Score_min == 1 ){
    return ("D")
  }

  else {
    return ("E")
  }
}

```

On va essayer de tester cette fonction, avec notre tableau de test construit précédemment.

```

ratio = c(1,1)

Traitement_prefere = Scores_2bras_DE (Tab_test, Data[2,], ratio)
print(Traitement_prefere)

```

```
## [1] "E"
```

On a maintenant à disposition les fonctions nécessaire à la création d'un algorithme de randomisation. On va procéder à son implémentation.

```
# Algorithme de randomisation a 2 bras

Algorithme_Radom_2bras_DE <- function (Data, pH, ratio){
  # Principe : Proposer une randomisation des patients en deux bras de
  #           traitement D et E, selon les variables de stratification.
  #
  # Entrees :
  #   Data   : Jeu de données.
  #   pH     : Probabilite du traitement prefere.
  #   ratio  : Ratio d'allocation sous la forme (1,1) pour 1:1.

  # On definit n, le nombre de patients du jeu de donnees
  n = nrow(Data)

  # On commence par calculer les probabilites d'allocation via BCM
  Probas_Allo = BCM_2bras_DE(ratio, pH)

  # On initialise le tableau de traitement
  Tab_Traitement = Init_tab_traitement()

  # On tire aleatoirement le traitement du premier patient
  Random_Traitement = sample(c("D","E"), size = 1)

  # On intialise un vecteur qui contiendra les traitements affectes
  Traitement = c(Random_Traitement)

  # On met a jour le tableau de traitement avec le premier patient
  Tab_Traitement = MaJ_tab_traitement(Tab_Traitement,
                                       Data[1,],
                                       Random_Traitement)

  # On parcours le jeu de donnees pour affecter un traitement a chaque patient
  for ( i in 2:n ) {

    # On cherche le traitement prefere
    Traitement_prefere = Scores_2bras_DE (Tab_Traitement, Data[i,], ratio)

    # On retrouve les probabilites d'allocation associees a ce traitement
    if ( Traitement_prefere == "D" ) {
      indic_traitement_prefere = 1
    }
    else {
      indic_traitement_prefere = 2
    }

    Probas_traitement_pref = Probas_Allo[indic_traitement_prefere,]
    if ( which ( Probas_traitement_pref==max(Probas_traitement_pref) ) == 1 ) {
      Tr_prefere = "D"
    }
  }
}
```

```

else {
  Tr_prefere = "E"
}

# On met à jour le tableau
Tab_Traitement = MaJ_tab_traitement(Tab_Traitement,
                                     Data[i,],
                                     Tr_prefere)

# On met a jour notre variable Traitement
Traitement = append(Traitement, Tr_prefere, after = length(Traitement))
}

return (Traitement)
}

```

On va maintenant tester l'algorithme une première fois sur notre jeu de données.

```

pH = 0.8
ratio = c(1,1)

head(Algorithme_Radom_2bras_DE(Data, pH, ratio))

```

```
## [1] "E" "D" "D" "E" "E" "E"
```

A partir de cet algorithme on peut simuler autant de randomisation que souhaité. Ici, on souhaite simuler 100 randomisations sur notre jeu de données et décrire les écarts aux ratios attendus pour l'ensemble des données, puis pour les modalités de chaque variable de stratification.

On va donc effectuer N=100 simulations, et calculer la moyenne des ratios obtenus via la randomisation.

```

# On definit les nombre de simulations
# Attention N = 100 ~ 2 minutes de temps de calcul
N = 100

# On definit les parametres des simulations
pH = 0.8
ratio = c(1,1)

Data_ratio = c()
Data_ratio_Bas = c(); Data_ratio_Int = c(); Data_ratio_Elv = c();
Data_ratio_C01 = c(); Data_ratio_C02 = c(); Data_ratio_C03 = c();
Data_ratio_C04 = c(); Data_ratio_C05 = c()
Data_ratio_TP1 = c(); Data_ratio_TP2 = c()

for (i in 1:N) {
  tr = Algorithme_Radom_2bras_DE(Data, pH, ratio)

  Data_simul = cbind(Data,tr)

  # Ratios obtenus par traitement par rapport au Bras de traitement D

```

```

Data_ratio = rbind( Data_ratio,
                    prop.table(table(Data_simul$tr))/prop.table(table(Data_simul$tr))[1] )

# Ratios par rapport au Bras de traitement D, par modalités de la variable CK
Data_ratio_Bas = rbind( Data_ratio_Bas,
                        prop.table(table(Data_simul$tr, Data$CK))[1,1]
                        /prop.table(table(Data_simul$tr, Data$CK))[1,1] )
Data_ratio_Int = rbind( Data_ratio_Int,
                        prop.table(table(Data_simul$tr, Data$CK))[1,2]
                        /prop.table(table(Data_simul$tr, Data$CK))[1,2] )
Data_ratio_Elv = rbind( Data_ratio_Elv,
                        prop.table(table(Data_simul$tr, Data$CK))[1,3]
                        /prop.table(table(Data_simul$tr, Data$CK))[1,3] )
Data_ratio_C01 = rbind( Data_ratio_C01,
                        prop.table(table(Data_simul$tr, Data$centre))[1,1]
                        /prop.table(table(Data_simul$tr, Data$centre))[1,1] )
Data_ratio_C02 = rbind( Data_ratio_C02,
                        prop.table(table(Data_simul$tr, Data$centre))[1,2]
                        /prop.table(table(Data_simul$tr, Data$centre))[1,2] )
Data_ratio_C03 = rbind( Data_ratio_C03,
                        prop.table(table(Data_simul$tr, Data$centre))[1,3]
                        /prop.table(table(Data_simul$tr, Data$centre))[1,3] )
Data_ratio_C04 = rbind( Data_ratio_C04,
                        prop.table(table(Data_simul$tr, Data$centre))[1,4]
                        /prop.table(table(Data_simul$tr, Data$centre))[1,4] )
Data_ratio_C05 = rbind( Data_ratio_C05,
                        prop.table(table(Data_simul$tr, Data$centre))[1,5]
                        /prop.table(table(Data_simul$tr, Data$centre))[1,5] )

Data_ratio_TP1 = rbind( Data_ratio_TP1,
                        prop.table(table(Data_simul$tr, Data$tum_prim))[1,1]
                        /prop.table(table(Data_simul$tr, Data$tum_prim))[1,1] )
Data_ratio_TP2 = rbind( Data_ratio_TP2,
                        prop.table(table(Data_simul$tr, Data$tum_prim))[1,2]
                        /prop.table(table(Data_simul$tr, Data$tum_prim))[1,2] )
}

```

On construit alors un petit Data frame pour stocker les écarts aux ratio comme suit.

```

Data_ecarts_ratios_DE <- data.frame(General = abs(ratio[2] - Data_ratio[,2]),
                                   Bas = abs(ratio[2] - Data_ratio_Bas[,2]),
                                   Int = abs(ratio[2] - Data_ratio_Int[,2]),
                                   Elv = abs(ratio[2] - Data_ratio_Elv[,2]),
                                   C01 = abs(ratio[2] - Data_ratio_C01[,2]),
                                   C02 = abs(ratio[2] - Data_ratio_C02[,2]),
                                   C03 = abs(ratio[2] - Data_ratio_C03[,2]),
                                   C04 = abs(ratio[2] - Data_ratio_C04[,2]),
                                   C05 = abs(ratio[2] - Data_ratio_C05[,2]),
                                   TP1 = abs(ratio[2] - Data_ratio_TP1[,2]),
                                   TP2 = abs(ratio[2] - Data_ratio_TP2[,2])
                                   )

head(Data_ecarts_ratios_DE)

```

```
##      General      Bas      Int      Elv      C01      C02 C03 C04
## 1 0.000000000 0.01111111 0.04651163 0.009090909 0.05555556 0.00000000 0 0
## 2 0.008264463 0.01111111 0.04651163 0.009174312 0.05555556 0.00000000 0 0
## 3 0.000000000 0.01123596 0.00000000 0.009090909 0.00000000 0.00000000 0 0
## 4 0.008196721 0.01111111 0.00000000 0.009090909 0.00000000 0.02857143 0 0
## 5 0.000000000 0.01123596 0.00000000 0.009090909 0.00000000 0.02941176 0 0
## 6 0.008196721 0.01123596 0.04444444 0.009090909 0.05263158 0.00000000 0 0
##      C05      TP1      TP2
## 1 0.05 0.007194245 0.009615385
## 2 0.00 0.007194245 0.009708738
## 3 0.00 0.007142857 0.009708738
## 4 0.00 0.007142857 0.009615385
## 5 0.05 0.007194245 0.009615385
## 6 0.00 0.007142857 0.009615385
```

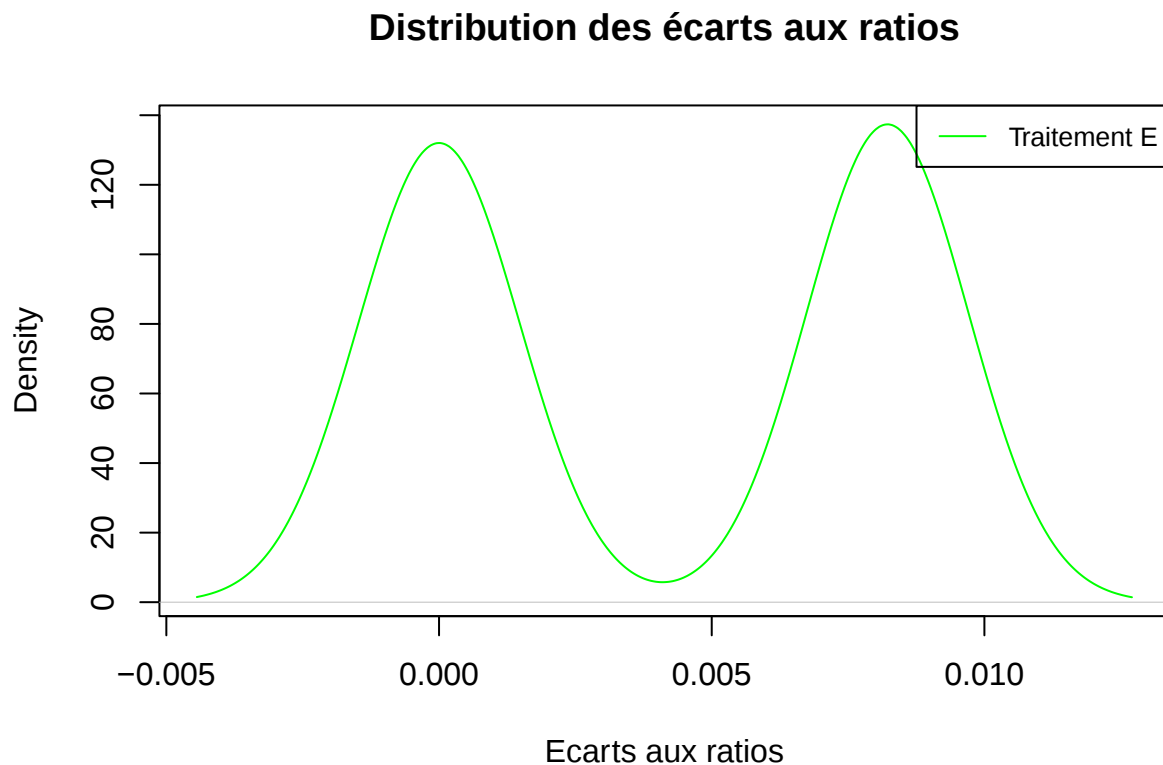
```
par(mfrow=c(1,1))

d <- density(Data_ecarts_ratios_DE$General)

plot(d,type = "l", col = "green", pch = 4, lty = 1,
     xlab = "Ecart aux ratios",
     main = " ")

legend("topright", legend = c("Traitement E"), col = c("green"), lty = 1, cex=0.8)

title(main = "Distribution des écarts aux ratios")
```



```

# On représente les modalités de CK sur 3 graphiques pour plus de lisibilité
par(mfrow=c(3,1))

d_bas <- density(Data_ecarts_ratios_DE$Bas)
d_int <- density(Data_ecarts_ratios_DE$Int)
d_elv <- density(Data_ecarts_ratios_DE$Elv)

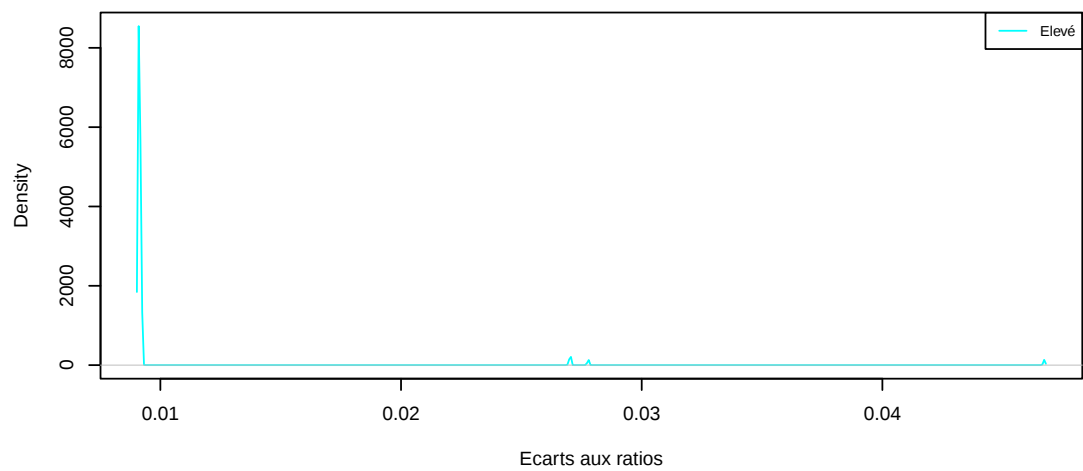
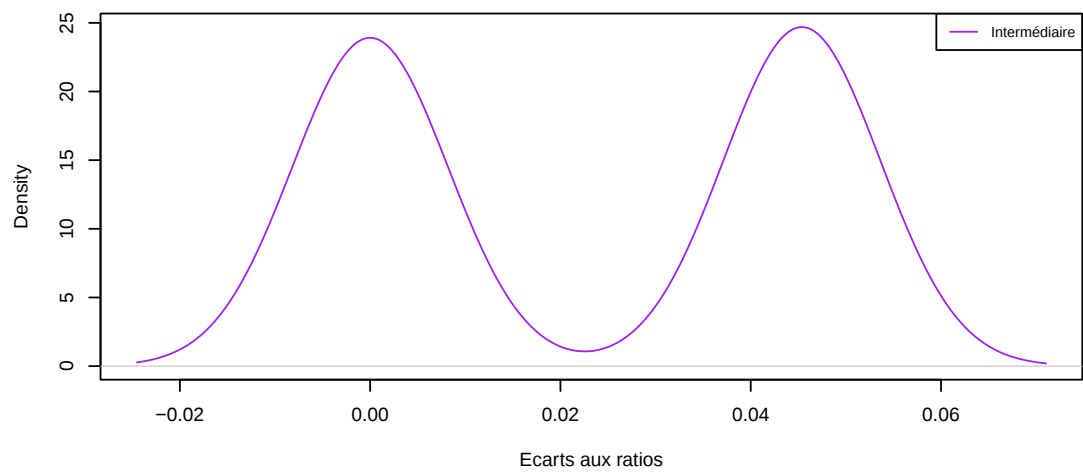
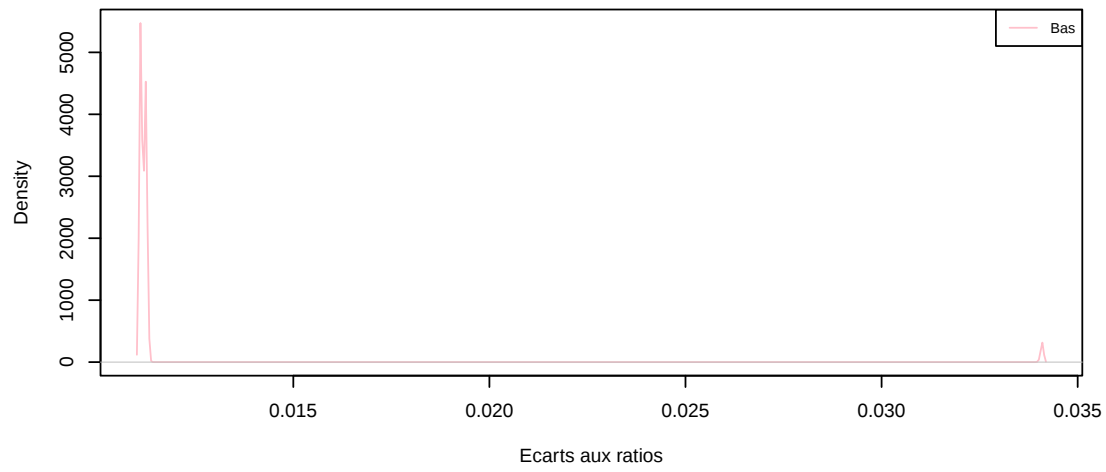
plot(d_bas, type = "l", col = "pink", pch = 4, lty = 1,
     ylim = c(0, max(d_bas$y)),
     xlab = "Ecart aux ratios",
     main = "Distribution des écarts aux ratios")
legend("topright", legend = c("Bas"), col = c("pink"), lty = 1, cex=0.8)

plot(d_int, type = "l", col = "purple", pch = 4, lty = 1,
     ylim = c(0, max(d_int$y)),
     xlab = "Ecart aux ratios",
     main = " ")
legend("topright", legend = c("Intermédiaire"), col = c("purple"), lty = 1, cex=0.8)

plot(d_elv, type = "l", col = "cyan", pch = 4, lty = 1,
     ylim = c(0, max(d_elv$y)),
     xlab = "Ecart aux ratios",
     main = " ")
legend("topright", legend = c("Elevé"), col = c("cyan"), lty = 1, cex=0.8)

```

Distribution des écarts aux ratios



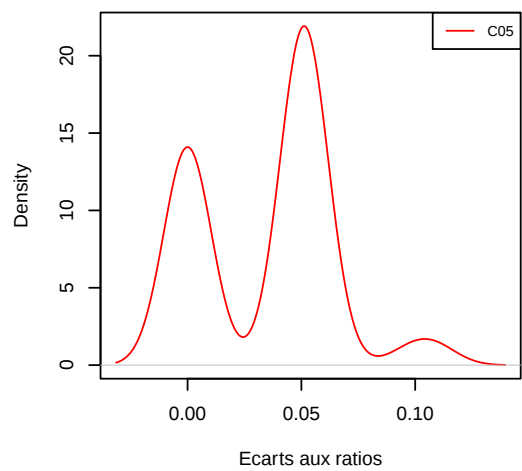
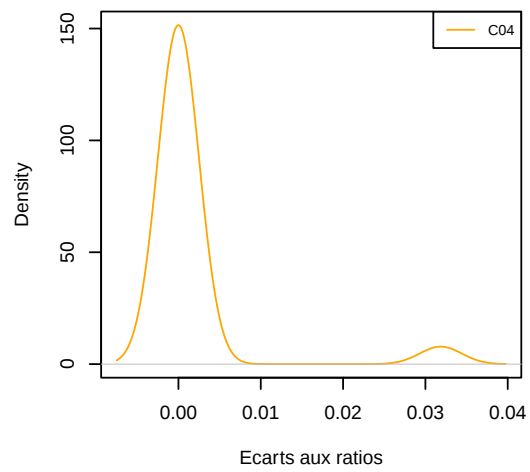
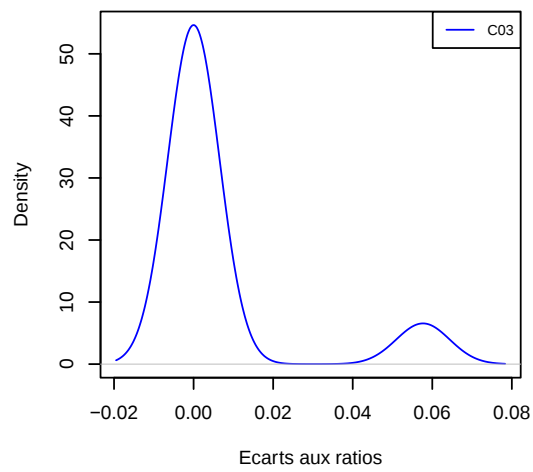
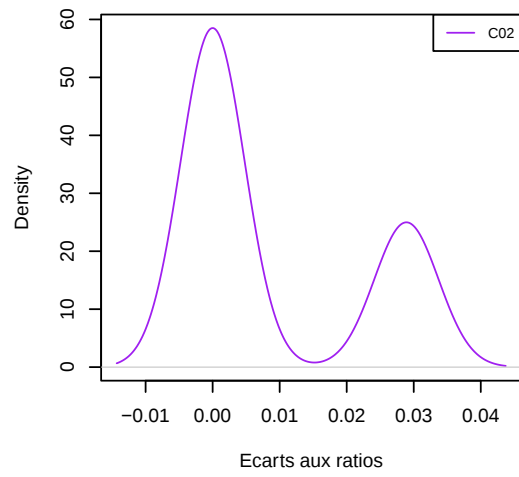
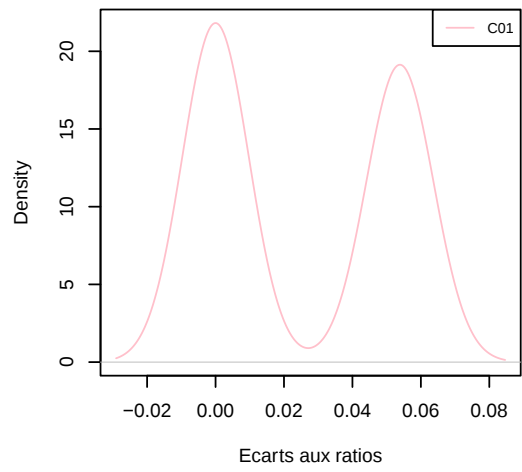
Les écarts

sont extrêmement faibles à l'exception du niveau Intermédiaire, où ils restent correct. Il n'y a pas eu trop de compromis à faire quant à l'attribution du traitement, pour cette variable et le respect, des ratios.

```
# On représente les modalités du centre sur 5 graphiques pour plus de lisibilité  
par(mfrow=c(3,2))
```

```
d_C01 <- density(Data_ecarts_ratios_DE$C01)  
d_C02 <- density(Data_ecarts_ratios_DE$C02)  
d_C03 <- density(Data_ecarts_ratios_DE$C03)  
d_C04 <- density(Data_ecarts_ratios_DE$C04)  
d_C05 <- density(Data_ecarts_ratios_DE$C05)  
  
plot(d_C01, type = "l", col = "pink", pch = 4, lty = 1,  
     ylim = c(0, max(d_C01$y)),  
     xlab = "Ecart aux ratios",  
     main = "Distribution des écarts aux ratios")  
legend("topright", legend = c("C01"), col = c("pink"), lty = 1, cex=0.8)  
  
plot(d_C02, type = "l", col = "purple", pch = 4, lty = 1,  
     ylim = c(0, max(d_C02$y)),  
     xlab = "Ecart aux ratios",  
     main = " ")  
legend("topright", legend = c("C02"), col = c("purple"), lty = 1, cex=0.8)  
  
plot(d_C03, type = "l", col = "blue", pch = 4, lty = 1,  
     ylim = c(0, max(d_C03$y)),  
     xlab = "Ecart aux ratios",  
     main = " ")  
legend("topright", legend = c("C03"), col = c("blue"), lty = 1, cex=0.8)  
  
plot(d_C04, type = "l", col = "orange", pch = 4, lty = 1,  
     ylim = c(0, max(d_C04$y)),  
     xlab = "Ecart aux ratios",  
     main = " ")  
legend("topright", legend = c("C04"), col = c("orange"), lty = 1, cex=0.8)  
  
plot(d_C05, type = "l", col = "red", pch = 4, lty = 1,  
     ylim = c(0, max(d_C05$y)),  
     xlab = "Ecart aux ratios",  
     main = " ")  
legend("topright", legend = c("C05"), col = c("red"), lty = 1, cex=0.8)
```

Distribution des écarts aux ratios



Les écarts aux ratios semblent moins bons pour le centre 5 que les autres. Cela signifie qu'on a eu un peu plus de mal à attribuer un traitement selon les ratios demandés pour le centre 5. Cependant, cela reste très correct.

Dans le centre 4, les écarts aux ratios sont très faibles et concentrés autour de 0. On a un très bon respect des ratios.

Enfin, pour terminer cette étude on s'intéresse aux modalités de la variable `tum_prim`.

On représente les modalités sur 2 graphiques

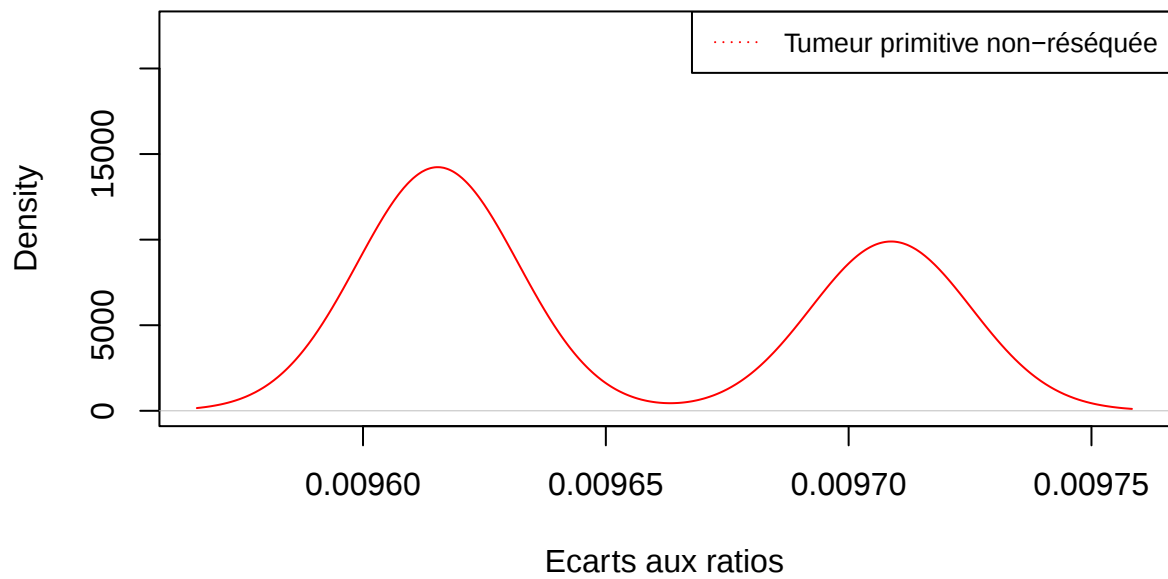
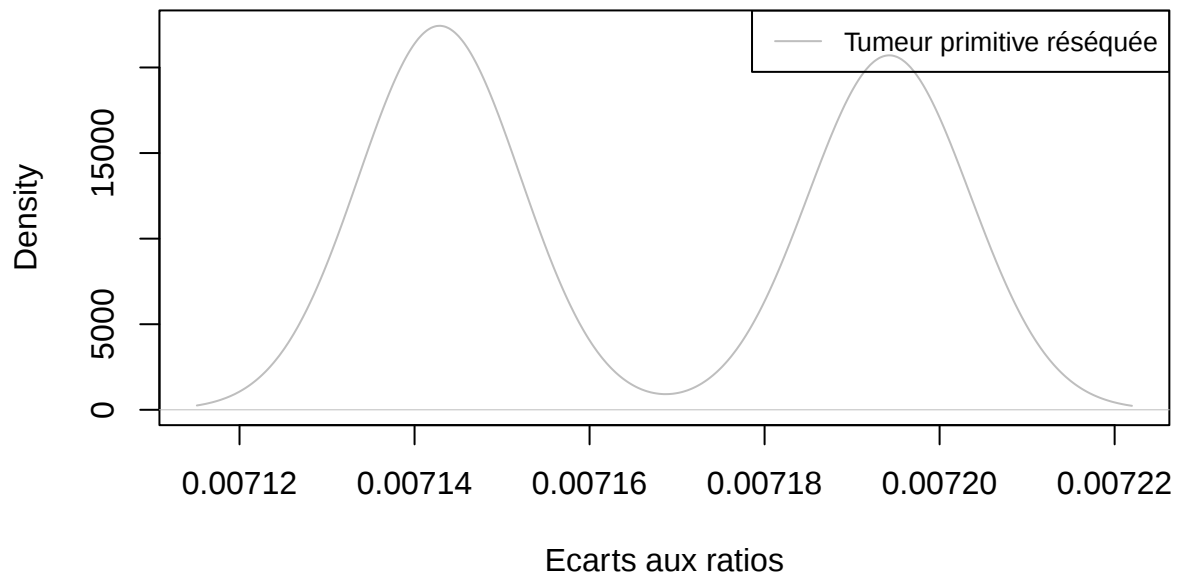
```
par(mfrow=c(2,1))

d_TP1 <- density(Data_ecarts_ratios_DE$TP1)
d_TP2 <- density(Data_ecarts_ratios_DE$TP2)

plot(d_TP1, type = "l", col = "gray", pch = 4, lty = 1,
     ylim = c(0, max(max(d_TP1$y),max(d_TP2$y))),
     xlab = "Ecart aux ratios",
     main = " Distribution des écarts aux ratios ")
legend("topright", legend = c("Tumeur primitive réséquée"), col = c("gray"),
     lty = 1, cex=0.8)

plot(d_TP2, type = "l", col = "red", pch = 4, lty = 1,
     ylim = c(0, max(max(d_TP1$y),max(d_TP2$y))),
     xlab = "Ecart aux ratios",
     main = " ")
legend("topright", legend = c("Tumeur primitive non-réséquée"), col = c("red"),
     lty = 3, cex=0.8)
```

Distribution des écarts aux ratios



Les écarts aux ratios sont extrêmement faibles et concentrés autour de 0, les ratios sont très bien respectés selon cette variable.