Suivi du pâturage avec un appareil photo grand public et des logiciels libres : méthode et validation

Maurice MAHIEU[1]*, Rémy ARQUET[2], Alexandre TRICHEUR[3],
Claire COLLAS[4], Stefan JURJANZ[4]

Annexes

ANNEXE 1 : **R script pour obtenir un vecteur standard de points-limites pour la classification des pixels en fonction de $VARI_{green}$**

ANNEXE 2 : **Image aérienne** (drone) **utilisée comme référence pour le calcul des limites de classe de l'indice VARIgreen. a) Couleurs RVB, b) Fausses couleurs, 15 classes VARIgreen, avec le pourcentage de pixels correspondant à chaque classe.**

ANNEXE 3 : **R script pour interpréter les classes VARIgreen en comparant les organes de plantes clairement identifiables sur les images originales et sur les images codées en fausses couleurs**

ANNEXE 4 : **Exemple de codage d'images de feuilles, tiges et matériaux morts. a) image originale RVB avec une zone pâturée** (sous la ligne pointillée)**, b) codage VARIgreen à 15 classes et c) codage VARIgreen à 3 classes** (pourcentages de pixels de chaque classe).

ANNEXE 5 : **Exemple de codage d'image de végétaux souillés par de la boue. a) image RVB originale avec à gauche salissure modérée et au centre salissure intense par le passage de véhicules en conditions humides, b) codage VARIgreen à 15 classes et c) codage VARIgreen à 3 classes.**

ANNEXE 6 : **Exemple de traitement d'image : a) image originale de la zone pâturée la veille** (les bouses ont déjà été collectées et les animaux déplacés à la position suivante)**, b) sélection des pixels correspondant à la zone pâturée et images codées selon la valeur VARIgreen de chaque pixel c) codage en 15 classes et d) codage en 3 classes** (et pourcentages de pixels correspondants).

ANNEXE 7 : **R script de traitement automatique de plusieurs images, quantification et comparaison des classes VARIgreen.**

ANNEXE 1 : R script pour obtenir un vecteur standard de points-limites pour la classification des pixels en fonction de $VARI_{green}$.

IMPORTANT.
La ligne de texte en vert n'est pas une commande exécutable pour R, mais du commentaire.
Seules les lignes en noir sont des lignes de commandes (script) R
Il est possible de copier-coller ce script dans n'importe quel éditeur de script R
Les noms de variable, fichier ou dossier sont modifiables au gré de l'utilisateur
Exécuter les lignes de script une à une peut aider à mieux comprendre la logique
du script et à l'adapter aux besoins de l'utilisateur

```
require(jpeg)
```
Appelle le package "jpeg", qui doit avoir été installé sur la machine au préalable

```
img=readJPEG(file.choose(),native = F)
```
Ouvre l'explorateur pour sélectionner et importer un fichier image.jpg

```
imgDm <- dim(img)
```
Extrait les dimensions de la matrice image

Transforme la matrice image en tableau avec, pour chaque pixel, les coordonnées x, y et les valeurs des canaux Rouge, Vert et Bleu
```
imgRGB <- data.frame(
 x = rep(1:imgDm[2], each = imgDm[1]),
 y = rep(imgDm[1]:1, imgDm[2]),
 R = as.vector(img[,,1]),
 G = as.vector(img[,,2]),
 B = as.vector(img[,,3])
 )

imgRGB$VARI <- (imgRGB$G - imgRGB$R)/(imgRGB$G + imgRGB$R - imgRGB$B)
```
Calcul de l'index $VARI_{green}$ pour chaque pixel, d'après Gitelson et al. (2002)

```
imgRGB$VARI <- ifelse(imgRGB$VARI > 10,10,ifelse(imgRGB$VARI < -10, -10, imgRGB$VARI))
```
Remplace les valeurs VARI ± inf, resultant de R+G-B = 0

```
imgRGB <- na.omit(imgRGB)
```
Conserve les pixels de la zone d'intérêt, en supprimant les pixels noirs (R=G=B=0), calcul $VARI_{green}$ impossible (0/0) donnant des NAs

```
require(cluster)

avari <- clara(imgRGB[,"VARI"], 15, metric = "euclidean", stand = FALSE,
 samples = 5, sampsize = min(nrow(imgRGB), 1000 + 2 * 15), trace = 0,
 medoids.x = TRUE, keep.data = T, rngR = FALSE, pamLike = FALSE)
```
Classification des pixels d'après les valeurs $VARI_{green}$

```
avari$medoids
```
Affiche les 15 valeurs des médoïdes

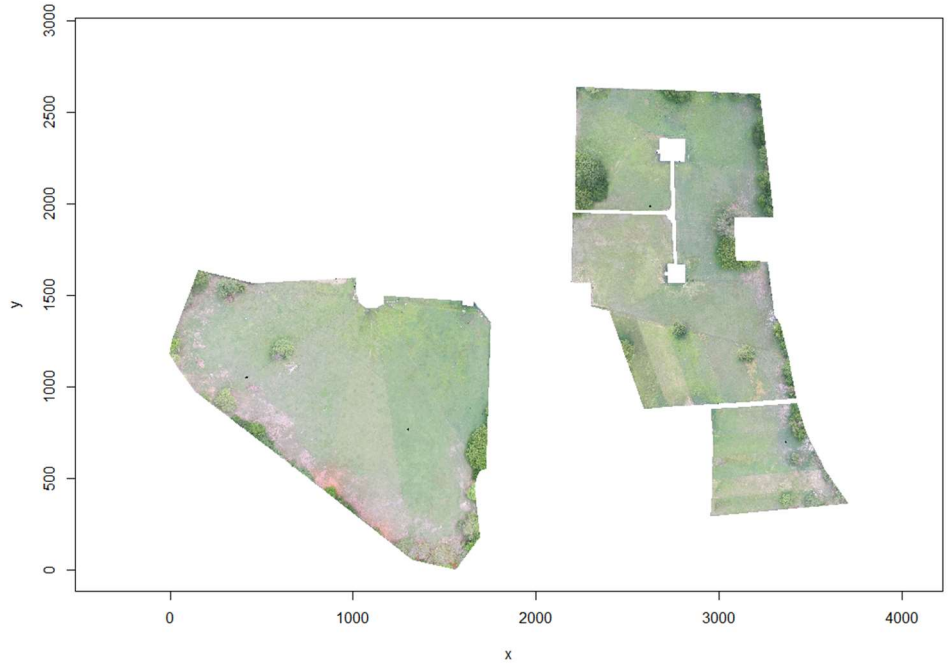
```
b <- as.vector(avari$medoids[order(avari$medoids[,1]),1]))
```
Trie les valeurs de médoïdes en ordre ascendant

```
Bki <- as.vector(NA,mode="numeric")
 for (k in 1:(length(b) -1))
 {
 bki[k] <- (b[k + 1] + b[k])/2
 }
```
Calcul des limites de classe à utiliser comme standard
```
bk15 <- c(-10, bki, 10)
```
Construit le vecteur des limites de classes, 15 classes VARIgreen
```
dput(bk15, file = paste("~/xx/bk15-",Sys.Date(), sep = ""))
```
Enregistre le vecteur des limites de classes dans le dossier xx

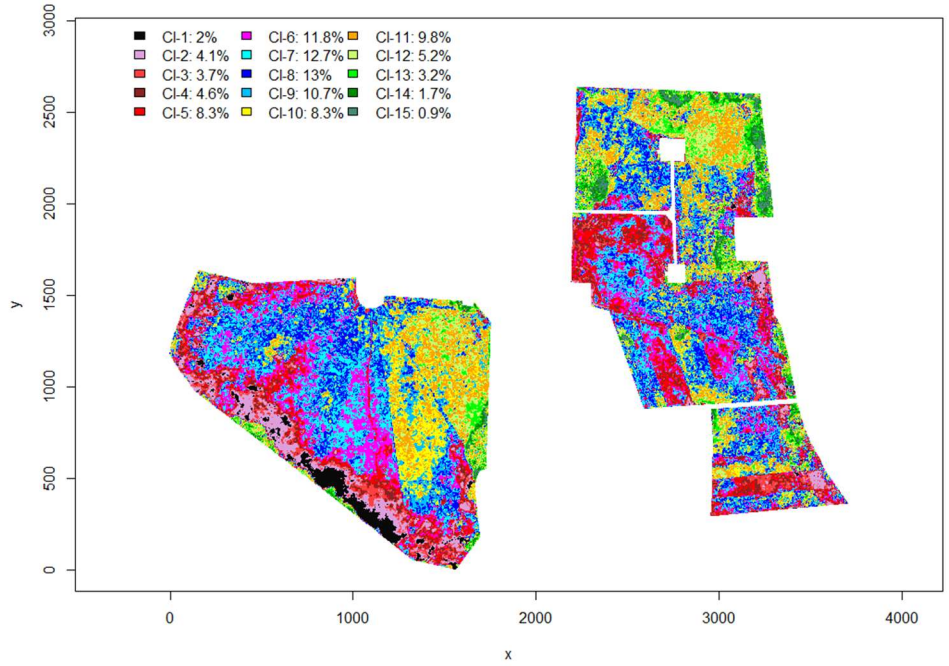
ANNEXE 2 : **Image aérienne** (drone) **utilisée comme référence pour le calcul des limites de classe de l'indice VARIgreen. a) Couleurs RVB, b) Fausses couleurs, 15 classes VARIgreen, avec le pourcentage de pixels correspondant à chaque classe.**

APPENDIX 2 : **Drone reference image used as standard for $VARI_{green}$ breakpoint calculation. 1) Top: RGB colours; 2) bottom: 15-class $VARI_{green}$ false colours.**

a) Couleurs RVB



b) Fausses couleurs (15 classes VARIgreen, % par classe)



ANNEXE 3 : **R script pour interpréter les classes VARI$_{green}$ en comparant les organes de plantes clairement identifiables sur les images originales et sur les images codées en fausses couleurs**

ANNEXE 3 : **R script pour interpréter les classes VARI$_{green}$ en comparant les organes de plantes clairement identifiables sur les images originales et sur les images codées en fausses couleurs**

```
require(jpeg)

img <- readJPEG(file.choose(),native = F)
```
Ouvre l'explorateur pour sélectionner et importer un fichier image.jpg

```
imgDm <- dim(img)
```
Extrait les dimensions de la matrice image
```
imgRGB <- data.frame(x = rep(1:imgDm[2],each = imgDm[1]),y = rep(imgDm[1]:1, imgDm[2]), R
= as.vector(img[,,1]),G = as.vector(img[,,2]),B = as.vector(img[,,3]))
```
Transforme la matrice image en tableau avec, pour chaque pixel, les coordonnées x, y et les valeurs des canaux Rouge, Vert et Bleu
```
imgRGB$VARI <- (imgRGB$G - imgRGB$R)/(imgRGB$G + imgRGB$R - imgRGB$B)
```
Calcul de l'index VARI$_{green}$ pour chaque pixel, d'après GITELSON *et al.* (2002)

```
imgRGB$VARI <- ifelse(imgRGB$VARI > 10,10,ifelse(imgRGB$VARI < -10, -10, imgRGB$VARI))
```
Remplace les valeurs VARI ± inf, resultant de R+G-B = 0
```
imgRGB <- na.omit(imgRGB)
```
Conserve les pixels de la zone d'intérêt, en supprimant les pixels noirs (R=G=B=0), calcul VARI$_{green}$ impossible (0/0) donnant des NAs

```
bk15 <- dget(file.choose())
```
Récupère le vecteur des limites de classes dans le dossier xx "~/xx/bk15-yyyy-mm-dd"

```
imgRGB$kclv <- cut(imgRGB$VARI,breaks = bk15,labels = F)
```
Encode les pixels dans une des 15 classes VARIgreen prédéfinies
```
coulv <- c("grey4", "plum", "brown1", "brown4", "red", "magenta", "cyan", "blue",
"deepskyblue", "yellow", "orange", "darkolivegreen1", "green", "green4", "aquamarine4")
```
Définit 15 couleurs pour coder les 15 classes VARIgreen avec des fausses couleurs (peuvent être changées au goût des utilisateurs)

```
coulxv <- col2rgb(coulv, alpha = FALSE)
```
Codage hexagesimal des couleurs pour un affichage accéléré
```
colxv <- data.frame(t(coulxv))
names(colxv) <- c("R","G","B")
row.names(colxv) <- 1:nrow(colxv)
kFColv <- rgb(colxv[imgRGB$kclv,],maxColorValue = 255)
```
Construction du vecteur des fausses couleurs pour l'image graphique

```
statisv <- data.frame(NA)
for (i in 1:(length(bk15) - 1))
{
 Kcli <- imgRGB$kclv[imgRGB$kclv == i]
 statisv[i,1] <- coulv[i]
 statisv[i,2] <- length(kcli)
}
statisv[,3] <- sum(statisv[,2])
```
Calcul du nombre de pixels de chaque classe
```
statisv[,4] <- statisv[,2]/statisv[,3]
statisv[,5] <- round(100*statisv[,4],1)

names(statisv) <- c("colcode","pixCount","pixtot","ratio","pcpx")
sum(statisv$ratio,na.rm=T)
```
Pour vérification, doit être égal à 1

```
plot(x = imgRGB$x,y = imgRGB$y,col = kFColv,pch = ".",
ylim = c(0,max(imgRGB$y)*1.1), asp = 1,xlab = "x",ylab = "y",
 main = expression(paste("k -Medoids Clustering of 15 ", VARI[green], " classes")), cex =
.01)
```

Graphe en fausses couleurs, avec codage en 15 classes VARIgreen

```
legend("top",ncol = 15,
paste("Cl-",1:15,"\n",as.character(statisv$pcpx),"%",sep=""),
 fill = statisv$colcode,cex = .7, bty = "n")
```

Ajoute une légende avec le pourcentage de pixels dans chaque classe

```
plot(x = imgRGB$x,y = imgRGB$y,col = rgb(imgRGB[,3:5]) ",type = "p", pch = ".", cex =
.01, ylim = c(0,max(imgRGB$y)*1.1),xlab = "x",ylab = "y", asp = 1,
 main = "RGB colors)
```

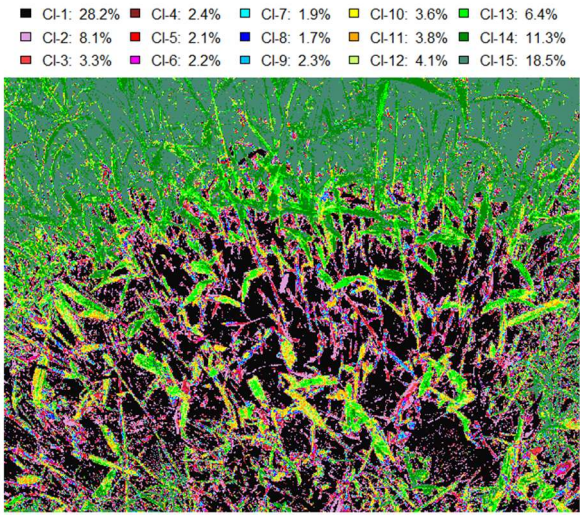
Graphe en couleurs naturelles (RVB)

ANNEXE 4 : **Exemple de codage d'images de feuilles, tiges et matériaux morts. a) image originale RVB avec une zone pâturée** (sous la ligne pointillée)**, b) codage VARIgreen à 15 classes et c) codage VARIgreen à 3 classes** (pourcentages de pixels de chaque classe).

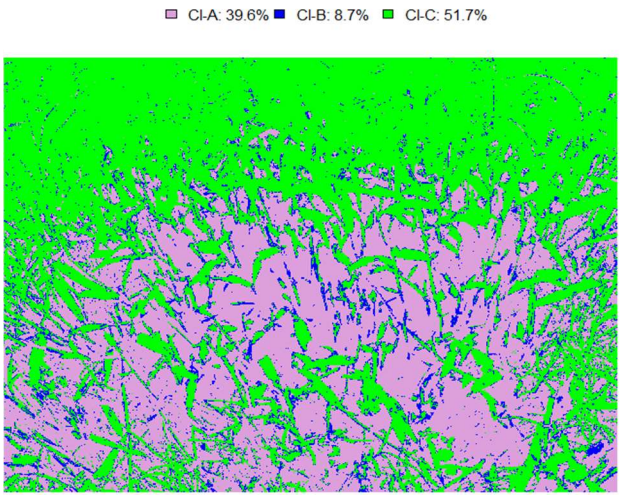
APPENDIX 4 : E**xample of image coding of leaves, stems and dead materials. 1) Top: original RGB image: grazed area** (below the dotted line) ***vs.* non grazed grass** (above the dotted line)**. 2) Bottom left: VARIgreen coding with 15 classes and 3) bottom right: VARIgreen coding with 3 classes.** The pixel percentages of each class are indicated.

a) Image originale, couleurs RVB

b) codage VARIgreen à 15 classes



c) codage VARIgreen à 3 classes

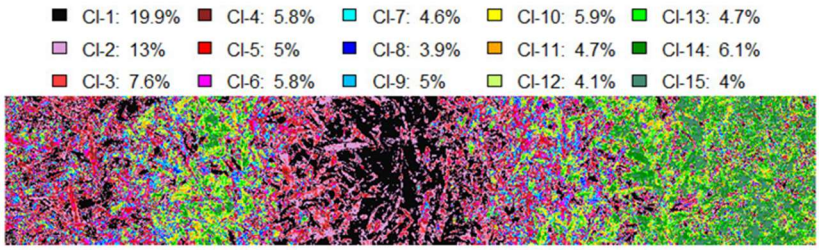


ANNEXE 5 : **Exemple de codage d'image de végétaux souillés par de la boue. a) image RVB originale avec à gauche salissure modérée et au centre salissure intense par le passage de véhicules en conditions humides, b) codage $VARI_{green}$ à 15 classes et c) codage $VARI_{green}$ à 3 classes.**

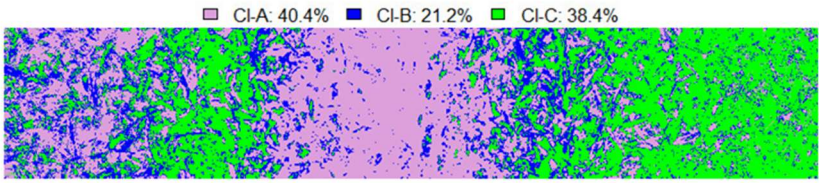
APPENDIX 5 : **Example of image coding of mud-soiled grass. From top to bottom a) RGB original image with medium trampling (left) and heavy trampling by vehicle wheel in humid conditions** (centre) **b) 15-class $VARI_{green}$ and c) 3-class $VARI_{green}$ coding.**

a) Image originale, couleurs RVB

b) codage VARIgreen à 15 classes



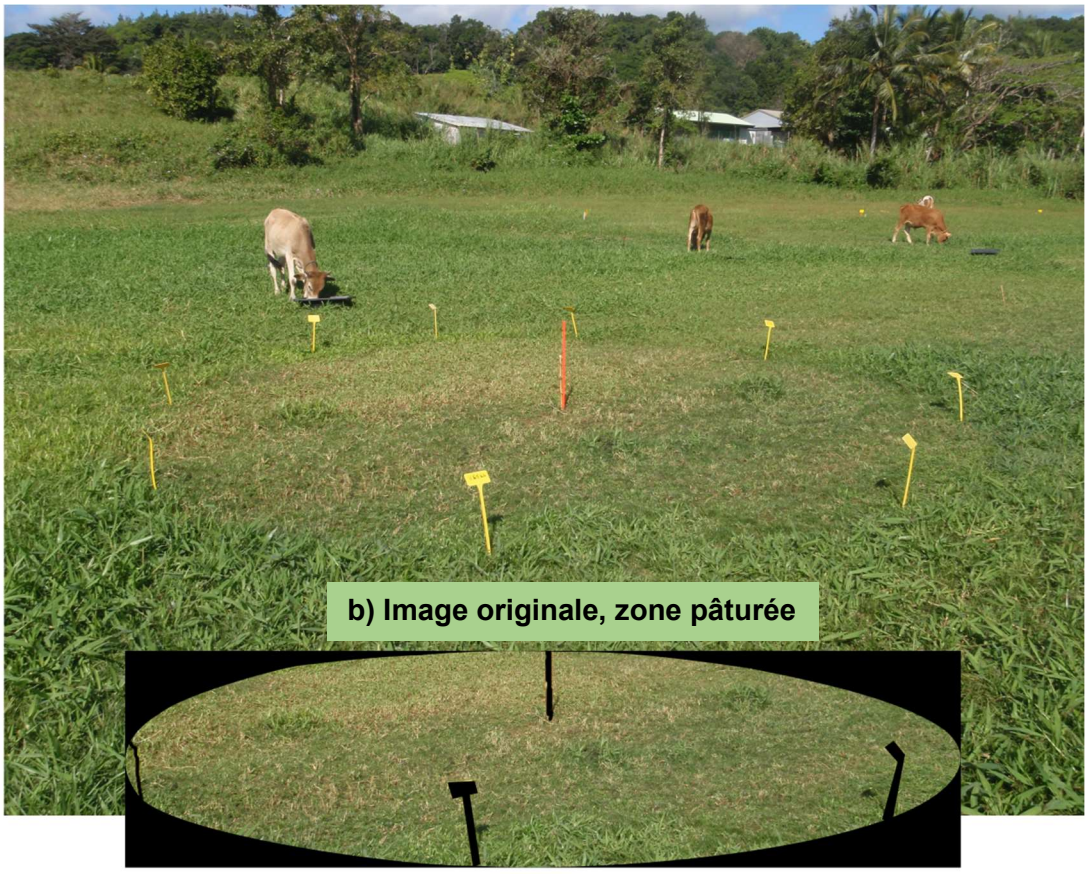
c) codage VARIgreen à 3 classes



ANNEXE 6 : **Exemple de traitement d'image : a) image originale de la zone pâturée la veille** (les bouses ont déjà été collectées et les animaux déplacés à la position suivante)**, b) sélection des pixels correspondant à la zone pâturée et images codées selon la valeur $VARI_{green}$ de chaque pixel c) codage en 15 classes et d) codage en 3 classes** (et pourcentages de pixels correspondants).

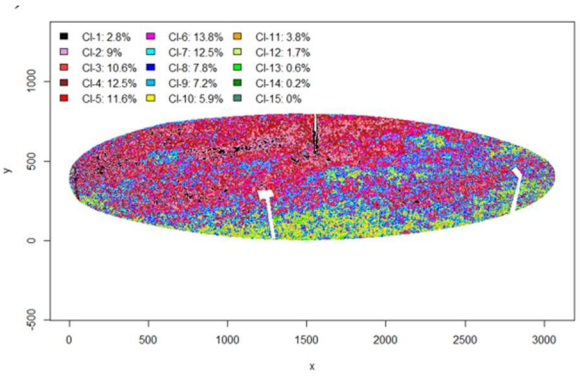
APPENDIX 6 : **Example of image processing. a) image of the area grazed the day before** (the dung pats have already been collected and the animal moved to the next area)**, b) selection of pixels corresponding to the grazed area and images coded according to the $VARI_{green}$ value of each pixel: c) 15-class clustering, d) 3-class clustering).**

a) Image originale

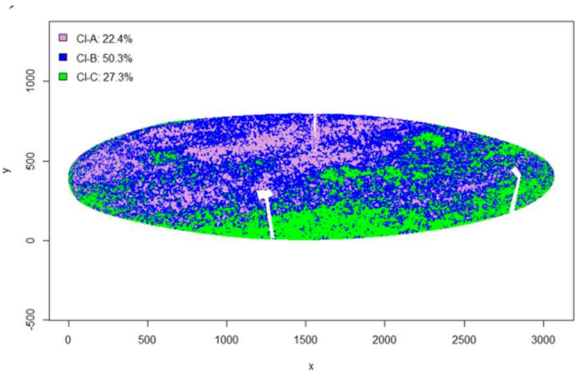


b) Image originale, zone pâturée

c) codage VARIgreen à 15 classes



d) codage VARIgreen à 3 classes



ANNEXE 7 : **R script de traitement automatique de plusieurs images, quantification et comparaison des classes VARI$_{green}$**

```
require(jpeg)

 bk15 <- dget(file.choose())
```
Récupère le vecteur des limites de classes dans le dossier xx "~/xx/bk15-yyyy-mm-dd"

```
files <- choose.files()
```
Ouvre l'explorateur pour sélectionner tous les fichiers image.jpg à analyser, et enregistre les chemins d'accès dans files

```
resu <- data.frame(NA)
for (i in 1:length(files))
```
Lit et analyse séquentiellement toutes les images
```
{
 Img <- readJPEG(files[i],native = F)
imgDm <- dim(img)

imgRGB <- data.frame(x = rep(1:imgDm[2], each = imgDm[1]),y = rep(imgDm[1]:1, imgDm[2]),R
= as.vector(img[,,1]),G = as.vector(img[,,2]),B = as.vector(img[,,3]))

imgRGB$VARI <- (imgRGB$G - imgRGB$R)/(imgRGB$G + imgRGB$R - imgRGB$B)
imRGB <- na.omit(imgRGB)
imRGB$VARI <- ifelse(imRGB$VARI > 10,10,ifelse(imRGB$VARI < -10,-10,imRGB$VARI))

imRGB$kclv <- cut(imRGB$VARI,breaks = bk15,labels = F)
```
Pixel coding

```
save(imRGB,file = paste("~/zz/",substring(files[i], nchar(files[i]) - 15), nchar(files[i])
- 4),"code.RData",sep = ""))
```
Sauvegarde les fichiers de résultat dans un dossier dédié zz, avec le même nom de code "code.RData" que l'image d'origine, sans l'extension ".jpg". Adaptez le nom de fichier à votre propre codification.
```
for (j in 1:(length(bk15)-1))
{
 Kcli <- imRGB$kclv[imRGB$kclv==j]
 resu[i,1] <- tablexif[i,1]
 resu[i,j+1] <- length(kcli)
}
}
```
Pour chaque image,et pour chaque classe VARIgreen, enregistre le nombre de pixels.
```
resu[ ,17] <- resu[ ,2] + resu[ ,3] + resu[ ,4] + resu[ ,5] + resu[ ,6] + resu[ ,7] +
resu[ ,8] + resu[ ,9] + resu[ ,10] + resu[ ,11] + resu[ ,12] + resu[ ,13] + resu[ ,14] +
resu[ ,15] + resu[ ,16]
```
Calcul du pourcentage de pixel de chaque classe VARIgreen class pour chaque image

```
for (l in 1:(length(bk15)-1))
{
 resu[,17 + l]=round(resu[,l + 1]/resu[,17],4)
}

names(resu) <- c("file", "cl01", "cl02", "cl03", "cl04", "cl05", "cl06", "cl07", "cl08",
"cl09", "cl10", "cl11", "cl12", "cl13", "cl14", "cl15", "sumPx", "ratiocl01", "ratiocl02",
"ratiocl03", "ratiocl04", "ratiocl05", "ratiocl06", "ratiocl07", "ratiocl08", "ratiocl09",
"ratiocl10", "ratiocl11", "ratiocl12", "ratiocl13", "ratiocl14", "ratiocl15")
resu$dphot <- as.Date(paste(substring(resu$file,nchar(resu$file) - 5, nchar(resu$file) -
4),"/", substring(resu$file,nchar(resu$file) - 7, nchar(resu$file) - 6),"/",
substring(resu$file,nchar(resu$file) - 9, nchar(resu$file) - 8),sep=""),"%y/%m/%d")
```
Récupère la date de prise de vue du nom du fichier image. DEPEND de la CODIFICATION du nom de fichier!!!
```
resu$dpat <- as.Date(ifelse(nchar(resu$file) < 20,resu$dphot + 2,
resu$dphot - 1),origin = "1970-01-01")
```
Récupère la date de pâturage en fonction de la date de prise de vue et du codage "avant" ou "après" pâturage.
DEPEND de la CODIFICATION du nom de fichier!!!
```
resu$offer <- ifelse(nchar(resu$file) < 20,"AV", substring(resu$file,4,6))
```
Récupère le niveau d'offre (traitement) en fonction du codage du nom de fichier.
DEPEND de la CODIFICATION du nom de fichier!!!

```
resu$anim <- ifelse(nchar(resu$file) < 20,"av", substring(resu$file,9,12))
```
Récupère l'identification de l'animal du nom de fichier, ou l'indication de "avant pâturage".
DEPEND de la CODIFICATION du nom de fichier!!!

```
resu$ratioCl_A <- round((resu[,2] + resu[,3] + resu[,4])/resu[,17],4)
resu$ratioCl_B <- round((resu[,5] + resu[,6] + resu[,7] + resu[,8])/resu[,17],4)
resu$ratioCl_C <- round((resu[,9] + resu[,10] + resu[,11] + resu[,12] + resu[,13] +
resu[,14] + resu[,15] + resu[,16])/resu[,17],4)
```
Regroupement des classes cl01 à cl03, cl04 à cl07 et cl08 à cl15 (Cl-A, Cl-B et Cl-C)

```
save.image("~/yy/imageresults.RData")
```
Sauvegarde de l'espace de travail dans le dossier YY

##################

POUR LA VISUALISATION DES DONNÉES, EXÉCUTER LES COMMANDES SUIVANTES

################

```
bagay15 <- resu[,c(18:36)]
```
Extrait les variables d'intérêt pour analyses (vérifier et adapter la liste des variables si nécessaire)

```
cl15 <- c("01","02","03","04","05","06","07","08","09","10","11","12","13","14","15")
```
Nomme les 15 classes

```
VARI15asin <- data.frame(NA)
J <- 0
for (i in 1:nrow(bagay15))
{
 for (k in 1:15)
 {
 J <- j+1
 bik <- bagay15[i,1:15]
 VARI15asin[j,1] <- bagay15[i,18]
 VARI15asin[j,2] <- bagay15[i,19]
 VARI15asin[j,3] <- bagay15[i,20]
 VARI15asin[j,4] <- cl15[k]
 VARI15asin[j,5] <- bik[,k]

 }
}
names(VARI15asin) <- c("dpat","offer","anim","class","pixratio")
VARI15asin[,1] <- as.Date(VARI15asin[,1],origin = "1970-01-01")
VARI15asin[,2] <- as.factor(VARI15asin[,2])
VARI15asin[,3] <- as.factor(VARI15asin[,3])
VARI15asin[,4] <- as.factor(VARI15asin[,4])

VARI15asin$pcpx <- asin(sqrt(VARI15asin$pixratio))


summary(model15 <- aov(pcpx ~ offer/class,VARI15asin))
```
Lance une Analyse de Variance
```
require(agricolae)

(comp15 <- HSD.test(model15,c("offer","class"), group = T))
```
Lance un test de Tukey pour comparer les moyennes

```
backtrans15 <- comp15$means
```
Récupère les estimations des moyennes et écart-types transformés

```
backtrans15$moy <- (sin(comp15$means$pcpx))^2
```
Transformation inverse des moyennes

```
backtrans15$hi <- (sin(comp15$means$pcpx + (comp15$means$std)/sqrt(comp15$means$r)))^2
```
Calcule transformation inverse moyenne + erreur-standard
```
backtrans15$mi <- (sin(comp15$means$pcpx - (comp15$means$std)/sqrt(comp15$means$r)))^2
```
Calcule transformation inverse moyenne - erreur-standard

```
backtrans15$Min <- (sin(comp15$means$Min))^2
backtrans15$Max <- (sin(comp15$means$Max))^2

plot(1:15,backtrans15$moy[1:15],type = "l",col = "red",
 xlab = expression(paste(VARI[green], " classs 1 to 15: from 'soil' to 'chlorophyll'")),
ylab = "pixels %", ylim = c(0,max(backtrans15$hi)), lty = 1,
las = 1, lwd = 4, axes = F)
```

Graphique des profils $VARI_{green}$

```
axis(1,at = 1:15,labels = 1:15)
axis(2,at = seq(0,.25,by = .05),labels = seq(0,25,by = 5),las = 1)
lines(1:15,backtrans15$moy[16:30], col = "black",lty = 2,lwd = 4)
lines(1:15,backtrans15$moy[31:45], col = "blue",lty = 3,lwd = 4)
lines(1:15,backtrans15$moy[46:60], col = "green4",lty = 4,lwd = 4)

segments((1:15)-0.03,backtrans15$hi[1:15],
 (1:15) - 0.03,backtrans15$mi[1:15],col = "red",lwd = 2,lty = 1)
segments((1:15) - 0.01,backtrans15$hi[16:30],
 (1:15) - 0.01,backtrans15$mi[16:30],col = "black",lwd = 2,lty = 1)
segments((1:15) + 0.01,backtrans15$hi[31:45],
 (1:15) + 0.01,backtrans15$mi[31:45],col = "blue",lwd = 2,lty = 1)
segments((1:15) + 0.03,backtrans15$hi[46:60],
 (1:15) + 0.03,backtrans15$mi[46:60],col = "green4",lwd = 2,lty = 1)

legend("top",lwd = 4,ncol = 2,col = c("red","black","blue","green4"),
legend = paste(c("100, ", "150, ","300, ","before "), "n = ", comp15$means$r[seq(2,47,by =
15)]), bty = "n",lty = 1:4)
```

##################

Regroupement en 3 classes $VARI_{green}$: A = sol + organes végétaux morts ; B = tiges + organes sénescents ; C = organes verts (chlorophylliens).

```
##################
cl3 <- c("A","B","C")

bagay3 <- resu[,c(34:39)]
```

Récupère les variables d'intérêt pour analyse (vérifier et adapter la liste des variables si nécessaire

```
VARI3asin = data.frame(NA)
J = 0
for (i in 1:nrow(bagay3))
{
 for (k in 1:3)
 {
 J = j + 1
 bik = bagay3[i,5:7]
 VARI3asin[j,1] <- bagay3[i,1]
 VARI3asin[j,2] <- bagay3[i,2]
 VARI3asin[j,3] <- bagay3[i,3]
 VARI3asin[j,4] <- bagay3[i,4]
 VARI3asin[j,5] <- cl3[k]
 VARI3asin[j,6] <- bik[,k]
 }
}
names(VARI3asin) <- c("dpat","offer","anim","class","pixratio")
VARI3asin[,1] <- as.Date(VARI3asin[,2],origin = "1970-01-01")
VARI3asin[,2] <- as.factor(VARI3asin[,1])
VARI3asin[,3] <- as.factor(VARI3asin[,3])
VARI3asin[,4] <- as.factor(VARI3asin[,4])
VARI3asin $pcpx <- asin(sqrt(VARI3asin$pixratio))

summary(model3 <- aov(pcpx ~ offer/class,VARI3asin))

summary(lm(fitted(model3)~VARI3asin$pcpx))
```

Renvoie le R^2 du modèle

```
require(agricolae)
```

```
(comp3 <- HSD.test(model3,c("offer","class"), group = T))

backtrans3 <- comp3$means

backtrans3$moy <- (sin(comp3$means$pcpx))^2

backtrans3$hi <- (sin(comp3$means$pcpx +(comp3$means$std)/sqrt(comp3$means$r)))^2
```
Calcule moyenne + erreur standard (transformation inverse)
```
backtrans3$mi <- (sin(comp3$means$pcpx -(comp3$means$std)/sqrt(comp3$means$r)))^2
```
Calcule moyenne - erreur standard (transformation inverse)
```
backtrans3$Min <- (sin(comp3$means$Min))^2
backtrans3$Max <- (sin(comp3$means$Max))^2
```
Boîte à moustache des valeurs prises par les différentes classes/traitements
```
boxplot(pixratio ~ class + offer, data = VARI3asin,horizontal = F, axes = F,
 ylab = "pixels %", xlab = expression(paste(VARI[green]," class Cl-A to Cl-C")),
       border = rep(c("red","black","blue","green4"),each=3), boxwex = 0.5,
      ylim = c(0,1.2))

axis(2,at = seq(0,1,by = 0.10),labels = seq(0,100,by = 10),cex.axis = 1,las = 1)

axis(1,at = 1:3,labels = c("Cl-A","Cl-B","Cl-C"),cex.axis = 1, col.axis = "red")
axis(1,at = 4:6,labels = c("Cl-A","Cl-B","Cl-C"),cex.axis = 1, col.axis = "black")
axis(1,at = 7:9,labels = c("Cl-A","Cl-B","Cl-C"),cex.axis = 1, col.axis = "blue")
axis(1,at = 10:12,labels = c("Cl-A","Cl-B","Cl-C"),cex.axis = 1,
col.axis = "green4")

abline(v = c(3.5,6.5,9.5),col = 1)
text(c(2,5,8,11,14,17,20,23),rep(1.18,4),
 c(paste("offer =",c("100","150","300")),"before"),
 col = c("red","black","blue","green4"))
text(c(2,5,8,11,14,17,20,23),rep(1.12,8),
 paste("n =",backtrans3$r[c(2,5,8,11)]),
 col = c("red","black","blue","green4"),cex = 0.8)

lines(1:3,backtrans3$moy[1:3],col = "red",lwd=3)
segments(1:3,backtrans3$mi[1:3],1:3,backtrans3$hi[1:3],col = "red")
lines(4:6,backtrans3$moy[4:6],col = "black",lwd = 3)
segments(4:6,backtrans3$mi[4:6],4:6,backtrans3$hi[4:6],col = "black")
lines(7:9,backtrans3$moy[7:9],col = "blue",lwd = 3)
segments(7:9,backtrans3$mi[7:9],7:9,backtrans3$hi[7:9],col = "blue")
lines(10:12,backtrans3$moy[10:12],col = "green4",lwd = 3)
segments(10:12,backtrans3$mi[10:12],10:12,backtrans3$hi[10:12],col = "green4")

text(1:12,rep(1.04,12),c("a","$","A","b","$","B","c","$","C","d","£","D"),
cex = 0.8,col = "blue")
```
Ajoute des symboles pour signifier les différences, DEPEND DES DONNÉES et des résultats des analyses statistiques. À adapter à chaque situation