

# Projet Stats

## Analyse des données

### Statistiques descriptives

Le jeu de données est composé de 768 bâtiments. Pour chaque bâtiment, nous possédons les 10 variables suivantes :

- Quantitatives :
  - `Relative.compactness` : compacité relative à un cube de même volume ( $RC = 6 \times V^{0.66} \times A^{-1}$  avec  $V$  et  $A$  respectivement le volume et l'aire totale du bâtiment (murs, toit, sol))
  - `Surface.area` : surface totale (sol + murs + toit)
  - `Wall.area` : surface de mur extérieurs
  - `Roof.area` : surface de toit
  - `Glazing.area` : Surface vitrée (en pourcentage de la surface au sol)
  - `Energy` : Énergie consommée
- Catégorielles :
  - `Overall.height` : hauteur du bâtiment (soit 3.5 soit 7 m)
  - `orientation` : orientation de la maison (Nord, Sud, Est, Ouest)
  - `Glazing.area.distr` : Distribution des vitres (uniforme, ou plus d'un coté)
  - `Energy.efficiency` : Indicateur de l'énergie consommée (de A (meilleur) à G (pire))

On cherchera dans les prochaines parties à prédire les variables `Energy` et `Energy.efficiency` en fonction des autres. Nous allons dans un premier temps réaliser quelques statistiques exploratoires sur toutes nos variables.

```
# Lecture jeu de données, mise sous forme de facteur et corrections
data = read.table("DataEnergy-Student.csv", header = TRUE, sep = ",")
data$Glazing.area.distr = as.factor(factor(data$Glazing.area.distr))
data$Energy.efficiency = as.factor(data$Energy.efficiency)
data$orientation = as.factor(data$orientation)
data$Overall.height = factor(data$Overall.height, ordered = TRUE)
data$Glazing.area[which(data$Glazing.area.distr == 0)] = 0

# Informations sur le jeu de données
summary(data)

# Séparation des variables quantitatives et qualitatives
quanti = c(1, 2, 3, 4, 7, 9)
quali = c(5, 6, 8, 10)
allvariables = 1:10

##  Relative compactness  Surface.area      Wall.area      Roof.area
##  Min.   :0.6125      Min.   :501.4       Min.   :234.3     Min.   :105.3
##  1st Qu.:0.6779      1st Qu.:598.7       1st Qu.:291.8     1st Qu.:137.4
##  Median :0.7517      Median :673.1        Median :315.8     Median :183.3
##  Mean   :0.7645      Mean   :671.3        Mean   :318.3      Mean   :176.5
##  3rd Qu.:0.8350      3rd Qu.:744.6        3rd Qu.:343.0     3rd Qu.:220.5
##  Max.   :0.9912      Max.   :826.0        Max.   :425.8      Max.   :225.8
##
##  Overall.height orientation  Glazing.area      Glazing.area.distr      Energy
##  Min.   :3.500      Min.   :1.000      Min.   :0.0000      Min.   :0.0000
##  1st Qu.:3.500      1st Qu.:1.000      1st Qu.:0.0000      1st Qu.:0.0000
##  Median :7.000      Median :2.000      Median :0.0000      Median :0.0000
##  Mean   :6.991      Mean   :2.000      Mean   :0.0000      Mean   :0.0000
##  3rd Qu.:7.000      3rd Qu.:2.000      3rd Qu.:0.0000      3rd Qu.:0.0000
##  Max.   :7.000      Max.   :2.000      Max.   :1.0000      Max.   :1.0000
```

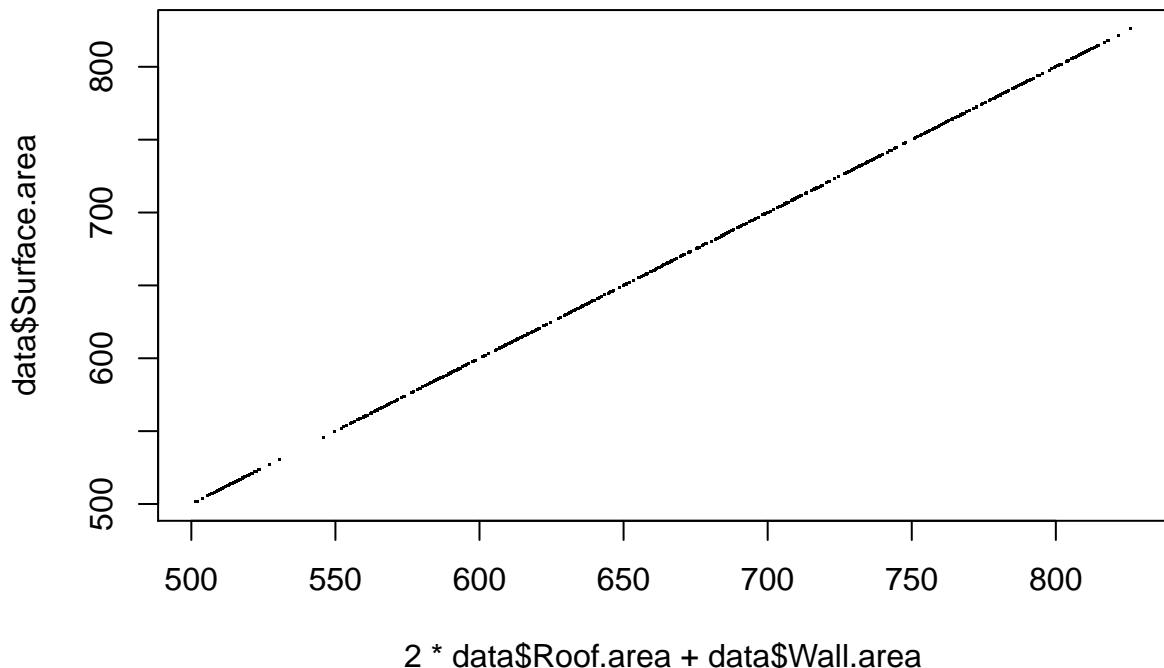
```

## 3.5:384      East :192   Min.    :0.0000  0: 48      Min.    :10.21
## 7  :384      North:192   1st Qu.:0.1031  1:144   1st Qu.:29.36
##                   South:192   Median  :0.2475  2:144   Median  :41.76
##                   West :192   Mean    :0.2344  3:144   Mean    :46.92
##                           3rd Qu.:0.3912  4:144   3rd Qu.:64.33
##                           Max.    :0.4270  5:144   Max.    :94.84
##
## Energy.efficiency
## A:208
## B:109
## C: 80
## D: 79
## E:109
## F:102
## G: 81

```

La commande summary nous permet de noter qu'il y a le même nombre de bâtiments pour chaque orientation, pour chaque hauteur. La distribution des fenêtres est aussi répartie équitablement entre les bâtiments (sauf pour ceux sans fenêtres (modalité 0), qui sont moins nombreux).

```
plot(2*data$Roof.area + data$Wall.area, data$Surface.area, pch=1)
```



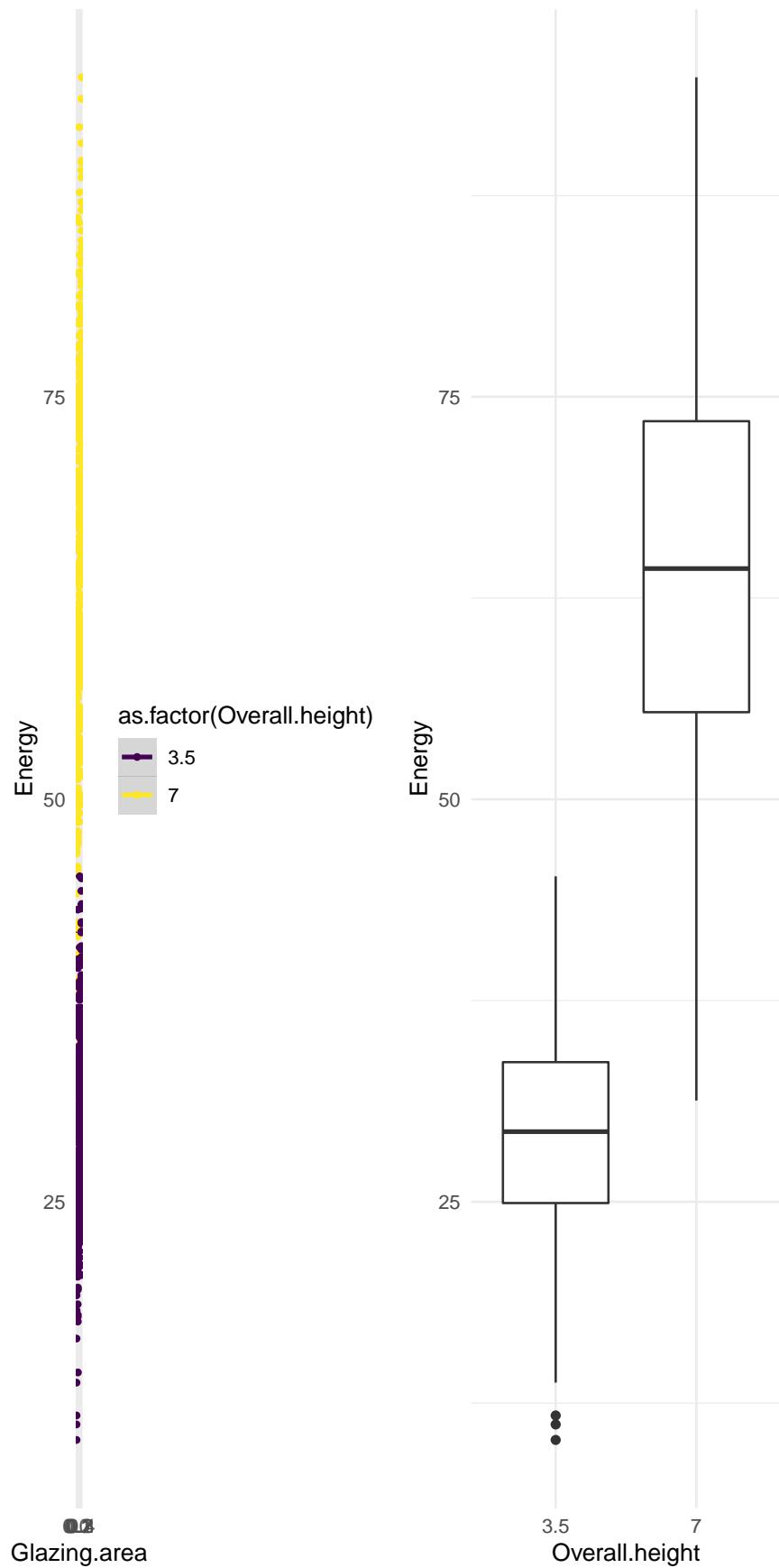
La courbe précédente nous montre que les variables Roof.area, Wall.area et Surface.area sont liées par la relation :  $\text{Surface.area} = 2 \times \text{Roof.area} + \text{Wall.area}$ .

```

gg1 = ggplot(data) +
  aes(x = Glazing.area, y = Energy, colour = as.factor(Overall.height)) +
  geom_smooth(method = "lm") +
  geom_point(size = 1L) +
  theme_minimal()
gg2 = ggplot(data) + geom_boxplot(data=data, aes(x=Overall.height, y=Energy)) +
  theme_minimal()
grid.arrange(gg1,gg2,ncol=2)

```

```
## `geom_smooth()` using formula 'y ~ x'
```

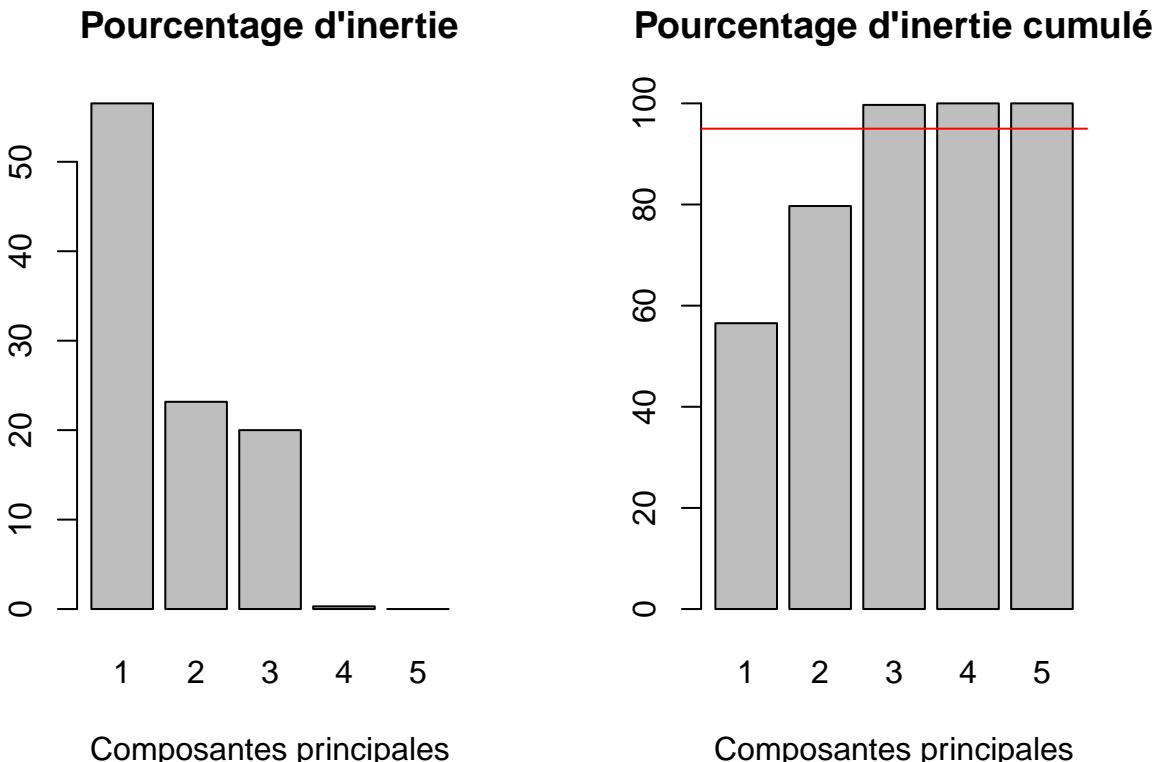


On voit que la surface vitrée à une influence claire sur l'énergie consommée. La hauteur du bâtiment joue aussi un rôle important.

## Analyse en Composantes Principales (ACP)

On souhaite réaliser une ACP afin de mener une analyse en dimension plus faible. Nous allons pour cela utiliser le package **FactoMineR** qui contient toutes les fonctions nécessaires à la réalisation de notre ACP. On cherche d'abord combien d'axes principaux nous allons avoir besoin.

```
library("FactoMineR")
par(mfrow=c(1,2))
res.acp <- PCA(data,scale.unit=T,quali.sup=quali,quanti.sup=9,ncp=8, graph=F)
barplot(res.acp$eig[, "percentage of variance"], main="Pourcentage d'inertie",
        names.arg = seq(1,5), xlab = "Composantes principales")
barplot(res.acp$eig[, "cumulative percentage of variance"],
        main="Pourcentage d'inertie cumulé",
        names.arg = seq(1,5), xlab = "Composantes principales")
abline(h = 95, col = "red")
```



```
print(paste("Pourcentage d'inertie expliquée par les trois premiers axes :",
            res.acp$eig[, "cumulative percentage of variance"] [3]))
```

```
## [1] "Pourcentage d'inertie expliquée par les trois premiers axes : 99.6979839436333"
```

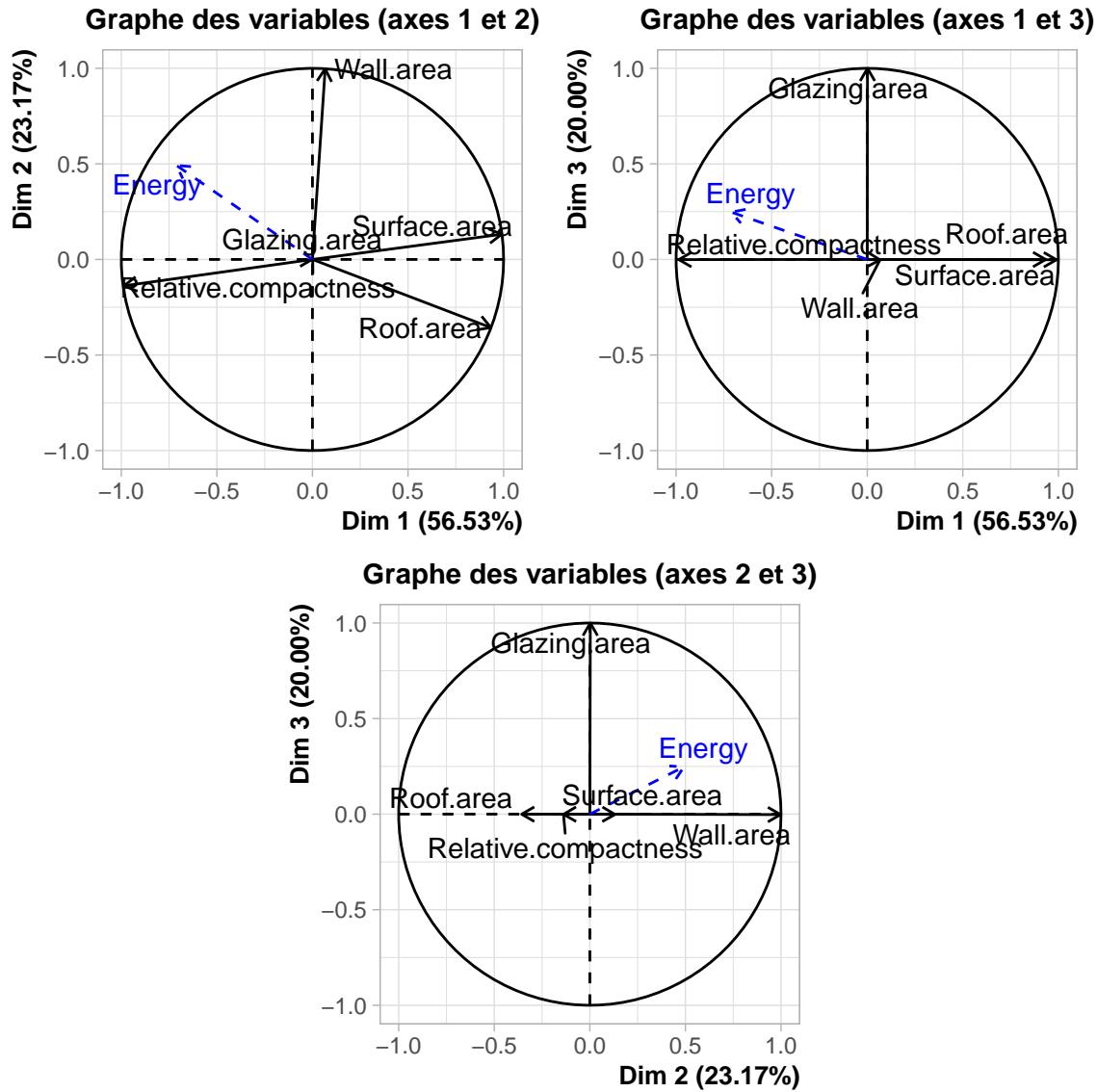
On voit sur le premier graphe que l'inertie portée par les trois premiers axes principaux est prépondérante par rapport aux autres. D'après le graphe de pourcentage d'inertie cumulée, ils expliquent presque 99% de l'inertie. Les deux premiers axes ne portent eux que 82% de l'inertie. Ne choisir que deux axes effacerait trop d'informations. Nous allons donc poursuivre notre analyse sur les trois premiers axes principaux.

Les graphes des variables nous donnent des informations très intéressantes sur les variables corrélées.

```

gg1 = plot.PCA(res.acp, choix="var", axes = c(1,2), new.plot = FALSE,
               title = "Graphe des variables (axes 1 et 2)")
gg2 = plot.PCA(res.acp, choix="var", axes = c(1,3), new.plot = FALSE,
               title = "Graphe des variables (axes 1 et 3)")
gg3 = plot.PCA(res.acp, choix="var", axes = c(2,3), new.plot = FALSE,
               title = "Graphe des variables (axes 2 et 3)")
layout_matrix <- matrix(c(1, 1, 2, 2, 4, 3, 3, 4), nrow = 2, byrow = TRUE)
grid.arrange(gg1, gg2, gg3, layout_matrix = layout_matrix)

```



Les variables `Relative.compactness`, `Surface.area` et `Roof.area` semblent être plutôt portées par le premier axe principal (en positif pour les deux premières, négatif pour les deux autres).

Le second axe porte principalement la variable `Wall.area`, et le troisième la variable `Glazing.area`.

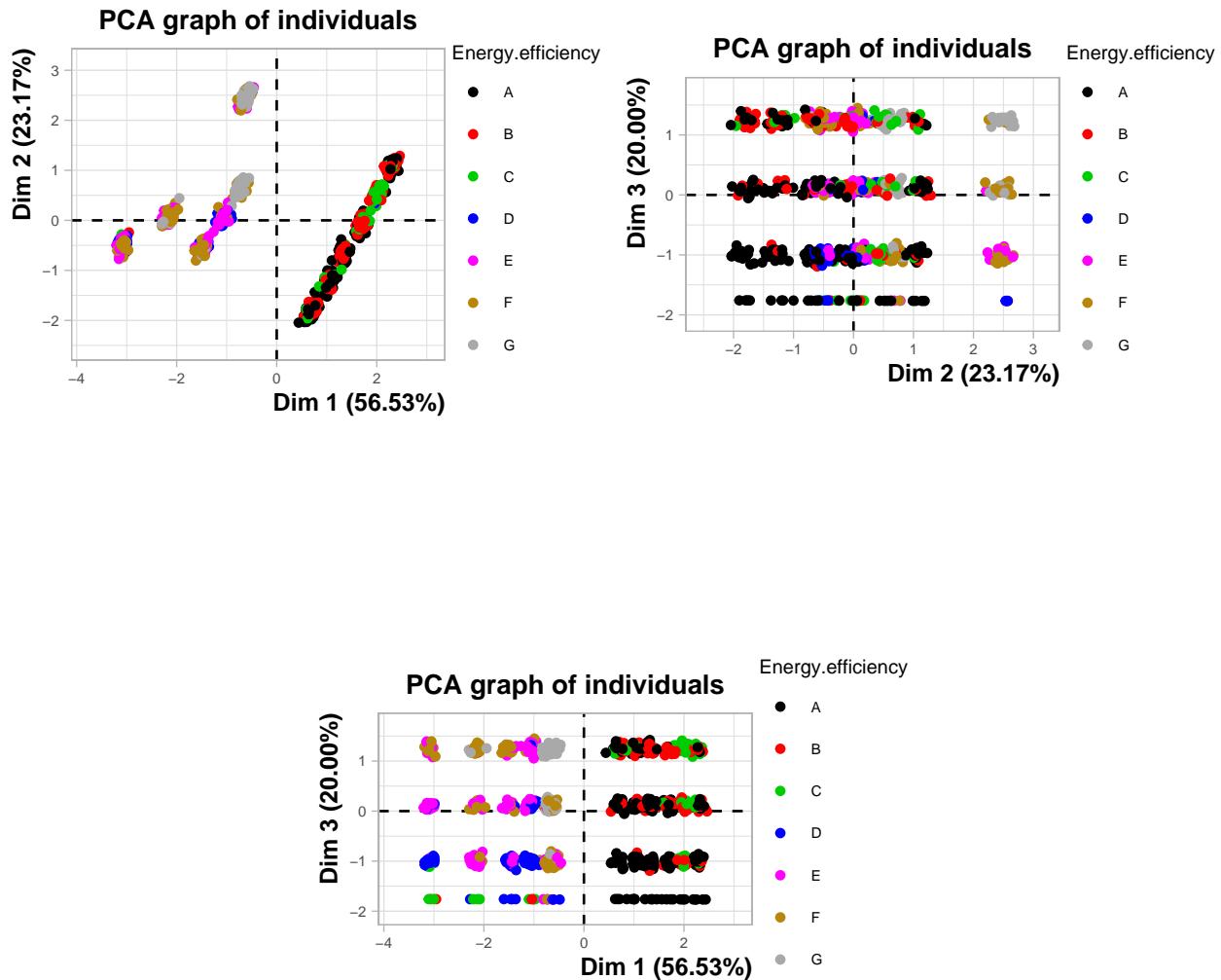
Finalement, les deux premiers axes ont plutôt trait à la forme du bâtiment (surface au sol pour le premier et surface murée pour le second), tandis que le troisième axe correspond à la surface vitrée.

Ainsi, l'énergie dépensée dépend d'abord de la forme du bâtiment, et ensuite de la surface vitrée.

Nous pouvons également observer les graphes des individus. Nous représentons les différentes classes

énergétiques par couleur.

```
gg1 = plot(res.acp, choix="ind", axes = c(1,2), invisible="quali",
           habillage="Energy.efficiency", label = "none", new.plot = FALSE) +
  theme(text = element_text(size=8))
gg2 = plot(res.acp, choix="ind", axes = c(2,3), invisible="quali",
           habillage="Energy.efficiency", label = "none", new.plot = FALSE) +
  theme(text = element_text(size=8))
gg3 = plot(res.acp, choix="ind", axes = c(1,3), invisible="quali",
           habillage="Energy.efficiency", label = "none", new.plot = FALSE) +
  theme(text = element_text(size=8))
grid.arrange(gg1, gg2, gg3, layout_matrix = layout_matrix)
```



On peut voir sur le premier et le troisième graphique que l'axe 1 marque une claire séparation entre les classes A, B, C et les classes D, E, F, G.

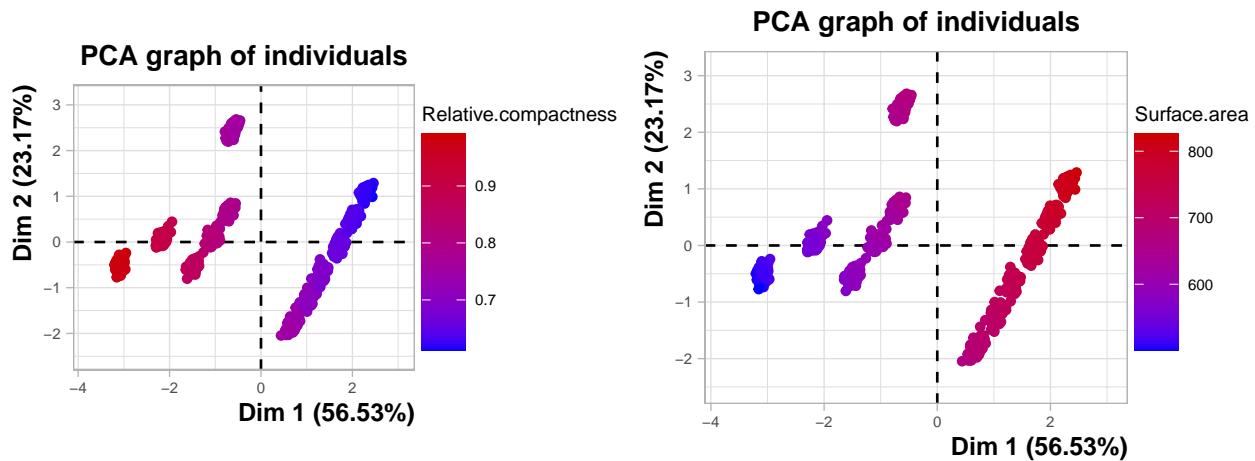
Les maisons dont la consommation énergétique la plus faible possèdent des coordonnées négatives pour la première composante. Ces maisons sont celles qui ont une compacité relative plus faible (plus basses et avec une surface au sol plus élevée), comme on peut le voir sur les graphiques suivants :

```
gg1 = plot(res.acp, choix="ind", axes = c(1,2), invisible="quali",
           habillage="Relative.compactness", label = "none", new.plot = FALSE) +
```

```

theme(text = element_text(size=8))
gg2 = plot(res.acp, choix="ind", axes = c(1,2), invisible="quali",
           habillage="Surface.area", label = "none", new.plot = FALSE) +
  theme(text = element_text(size=8))
gg3 = plot(res.acp, choix="ind", axes = c(1,2), invisible="quali",
           habillage="Overall.height", label = "none") +
  theme(text = element_text(size=8))
grid.arrange(gg1,gg2,gg3,layout_matrix = layout_matrix)

```



Ainsi, les maisons plus étalées sur le sol possèdent les meilleures performances énergétiques.

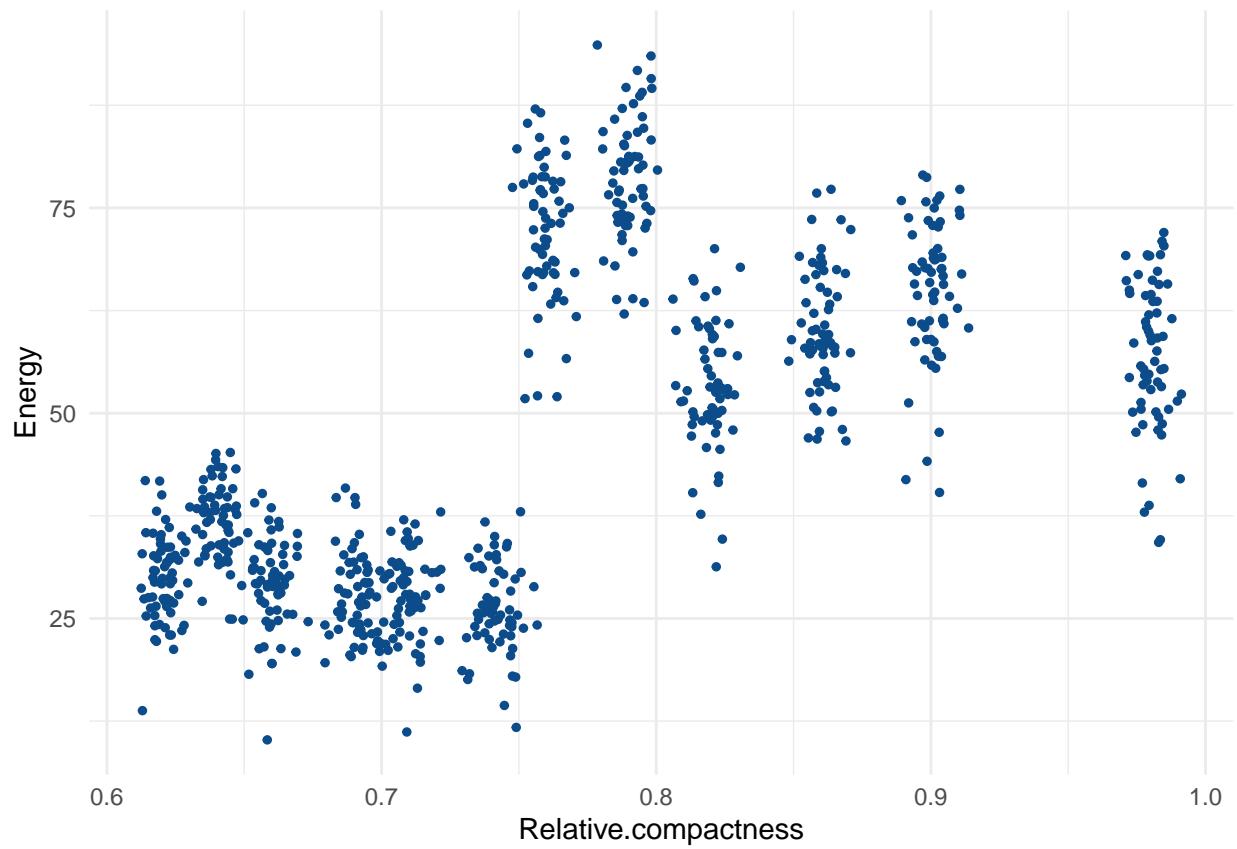
## Clustering de variables

### Analyse visuelle

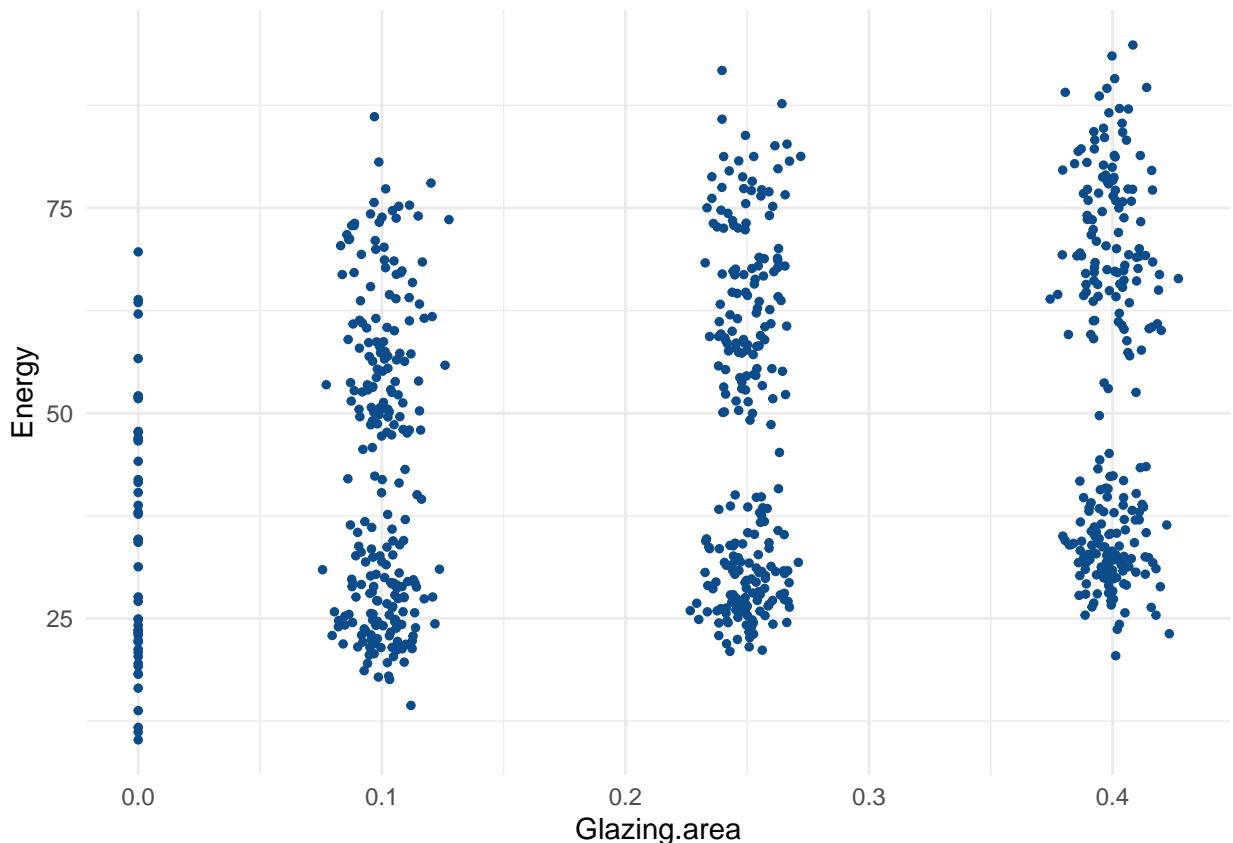
```

ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = "#0c4c8a") +
  theme_minimal()

```



```
ggplot(data) +  
  aes(x = Glazing.area, y = Energy) +  
  geom_point(size = 1L, colour = "#0c4c8a") +  
  theme_minimal()
```



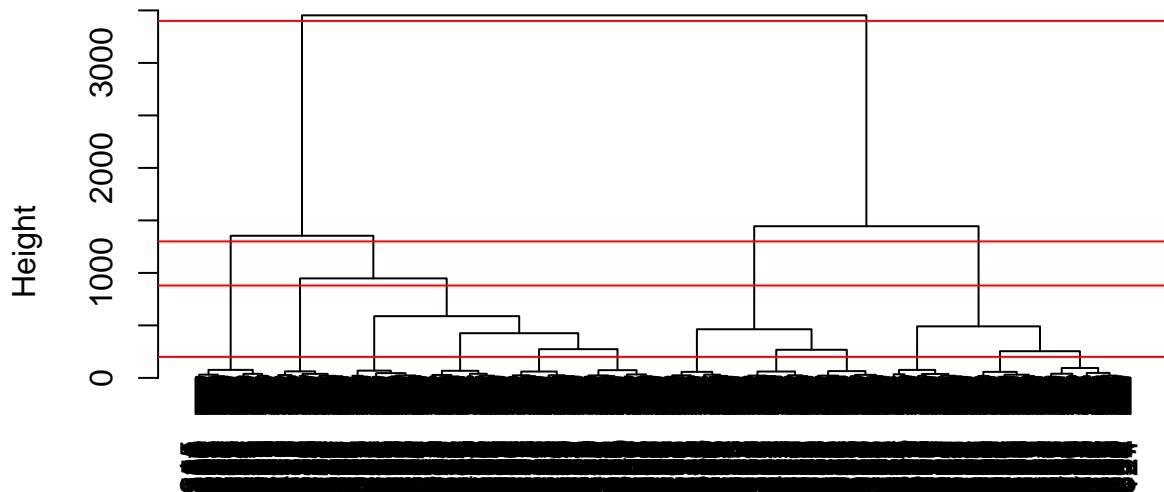
Visuellement, c'est selon la compacité relative et la surface vitrée que le distingue le mieux des groupes. Selon la compacité relative, on constate principalement deux groupes. On peut également voir 12 groupes mais ceux-ci sont moins évidents. Selon la surface vitrée, on distingue principalement 4 groupes.

### Clustering hiérarchique

Après cette analyse visuelle, nous allons utiliser des méthodes de clustering pour confirmer ou infirmer nos observations. Commençons par le clustering hiérarchique.

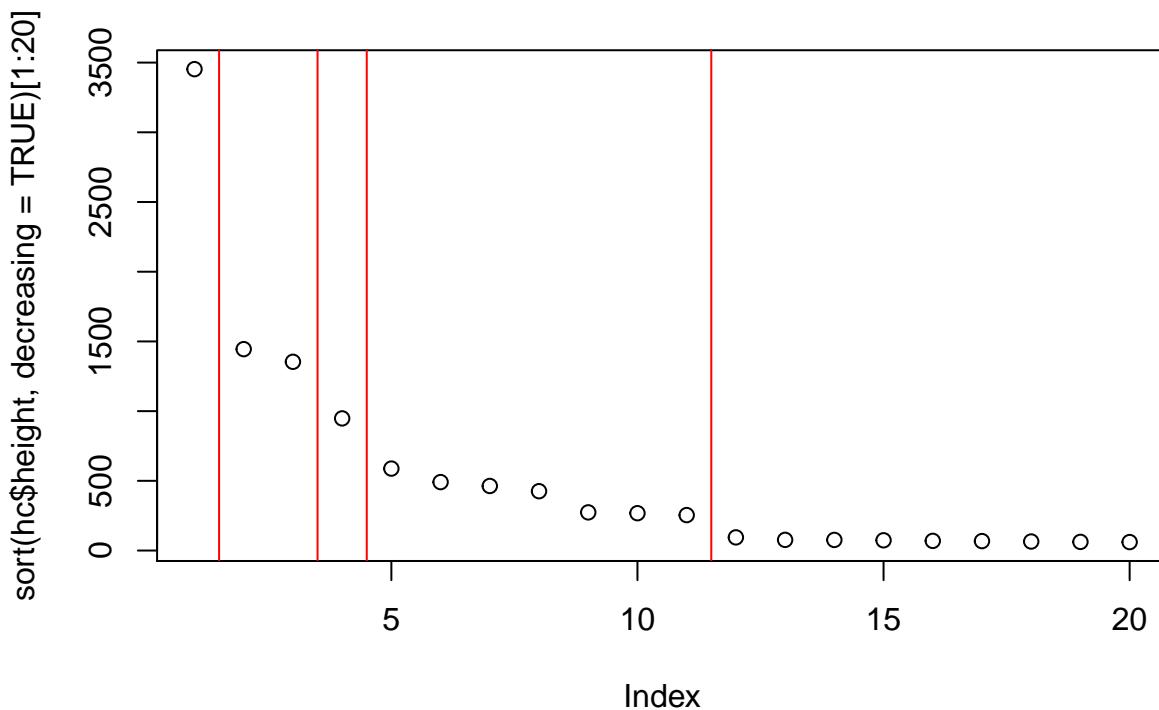
```
hc = hclust(dist(data[c(1,2,3,4,5,7)])), method = "ward.D2")
plot(hc)
abline(h=3400,col="red")
abline(h=1300,col="red")
abline(h=880,col="red")
abline(h=200,col="red")
```

## Cluster Dendrogram



```
dist(data[c(1, 2, 3, 4, 5, 7)])
hclust (*, "ward.D2")
```

```
plot(sort(hc$height, decreasing = TRUE)[1:20])
abline(v=1.5,col="red")
abline(v=3.5,col="red")
abline(v=4.5,col="red")
abline(v=11.5,col="red")
```

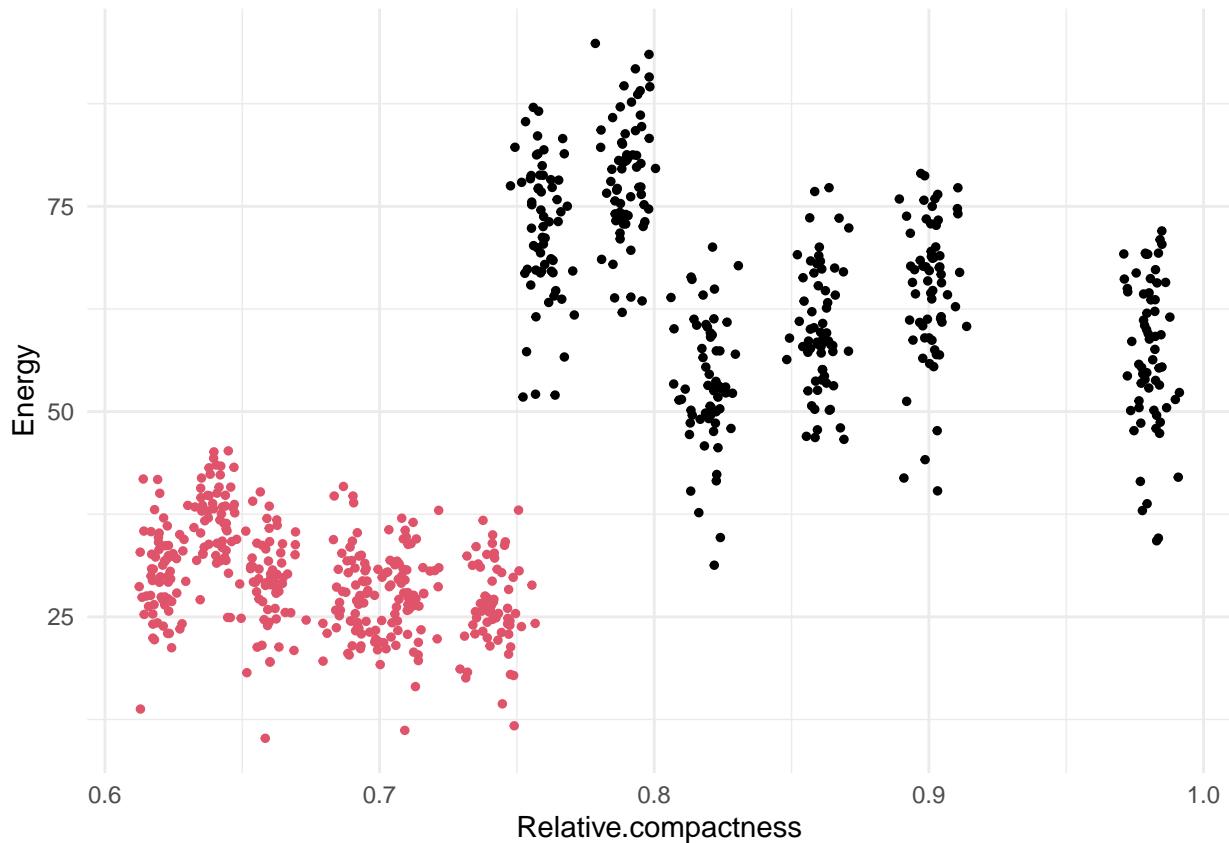


Sur le dendrogramme, deux classes apparaissent clairement. On peut aussi distinguer 4,5 ou 12 classes. Sur le graphique de la variance inter-classe, on observe un saut important au passage à 2 classes. On retrouve également les résultats que l'on a obtenu avec le dendrogramme en observant des sauts aux passages à 4, 5 et 12 classes.

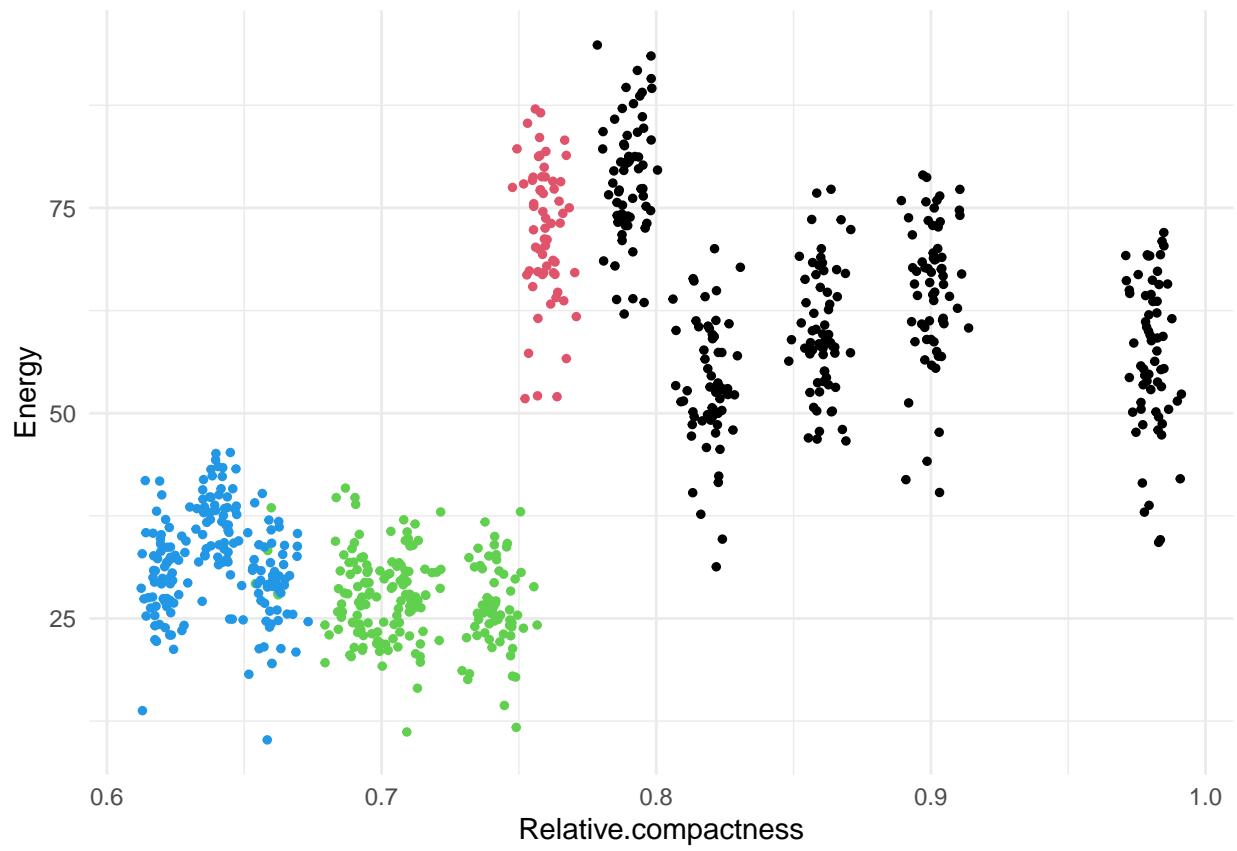
Ces premiers outils semblent confirmer les nombres de classes qui semblent pertinents : 2, 4 et 12.

On réalise donc dans un premier temps un découpage en 2, 4 et 12 classes selon le clustering hiérarchique. On compare ces résultats avec nos observations selon plusieurs variables explicatives.

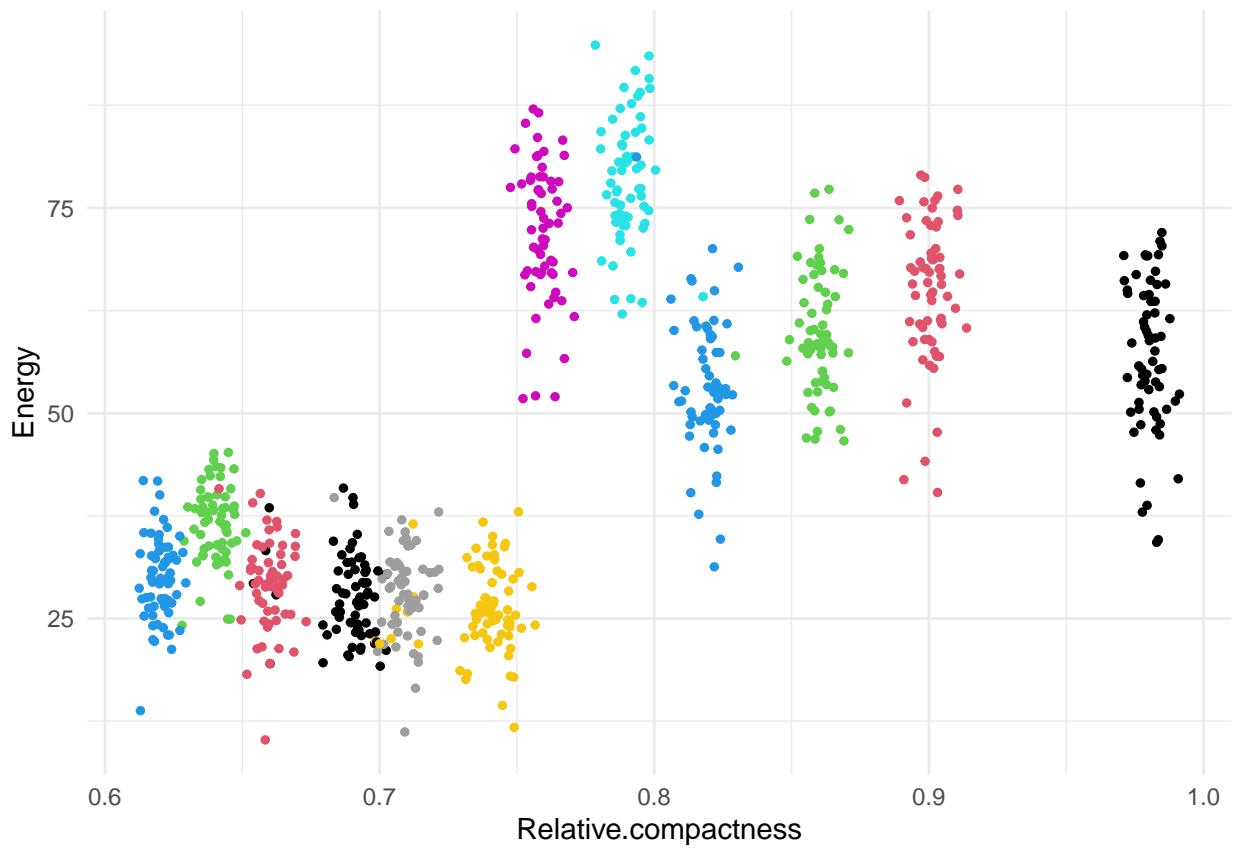
```
class2 = cutree(hc, k = 2)
ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = class2) +
  theme_minimal()
```



```
class4 = cutree(hc, k = 4)
ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = class4) +
  theme_minimal()
```

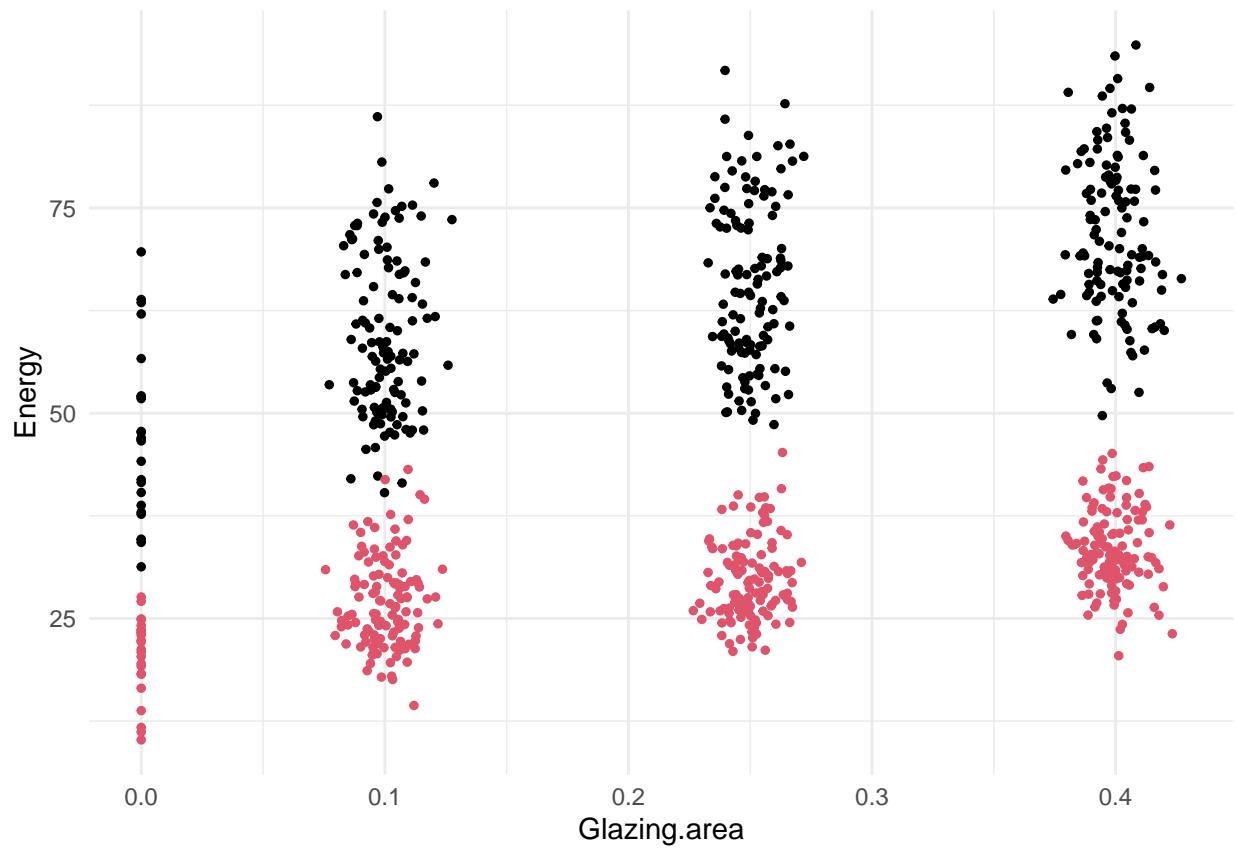


```
class12 = cutree(hc, k = 12)
ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = class12) +
  theme_minimal()
```

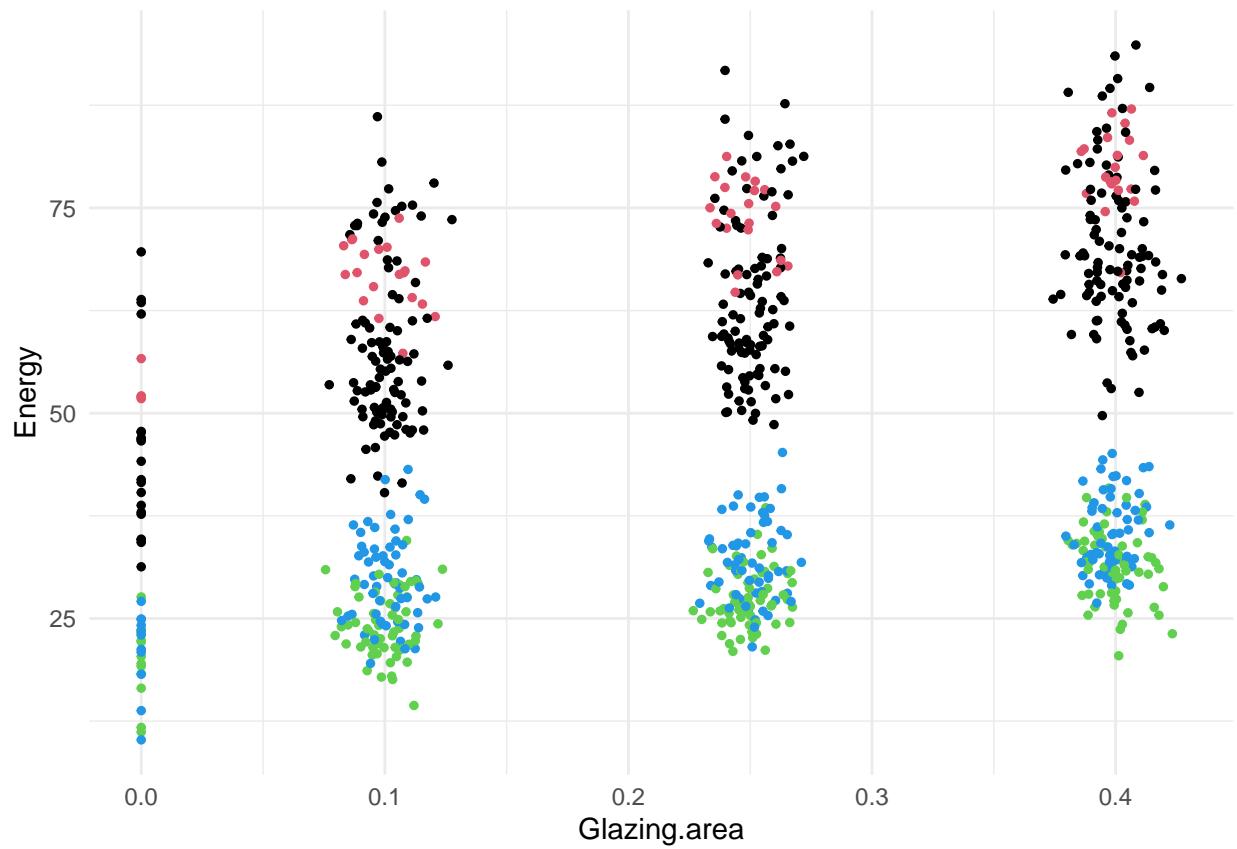


Visuellement selon la compacité relative, le découpage en 2 groupes et celui en 12 groupes semblent très pertinents. Celui en 4 groupes ne semble pas s'expliquer par la compacité relative.

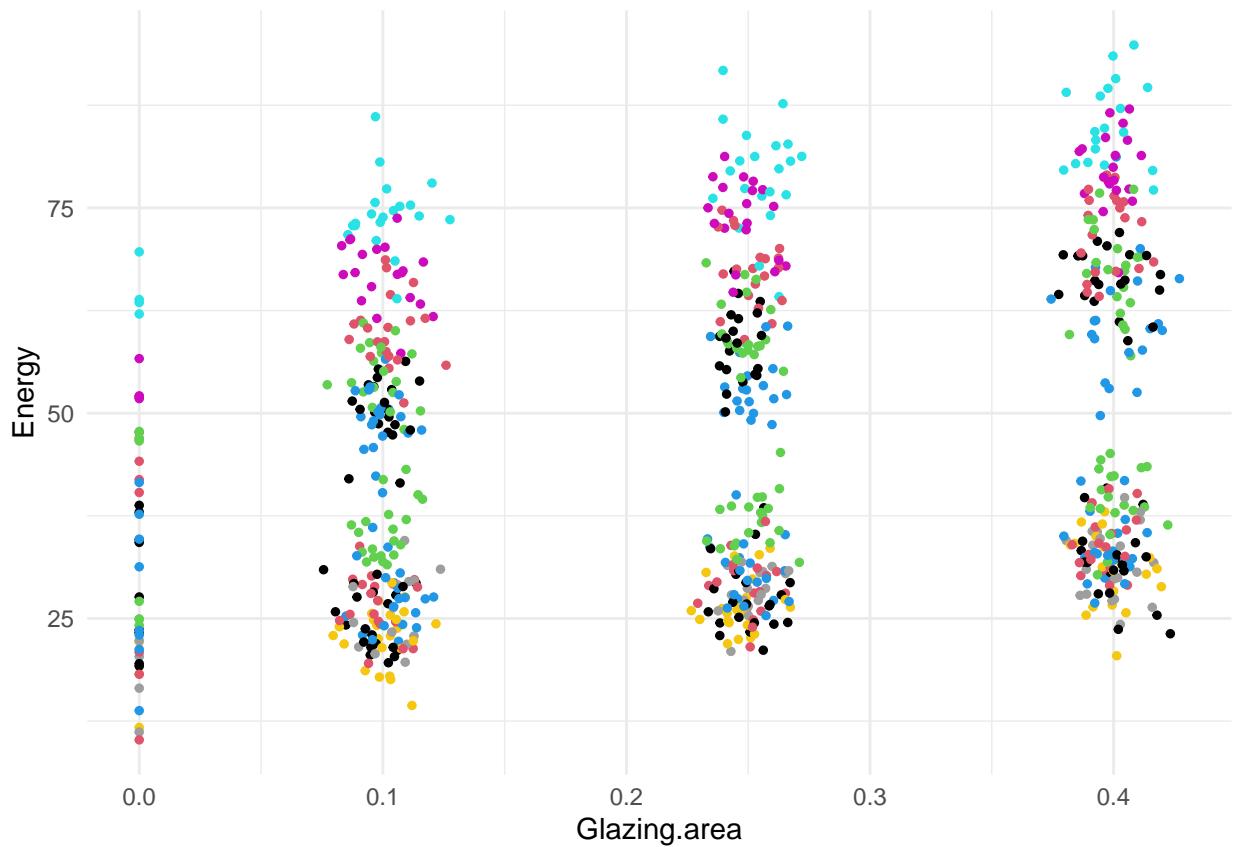
```
class2 = cutree(hc, k = 2)
ggplot(data) +
  aes(x = Glazing.area, y = Energy) +
  geom_point(size = 1L, colour = class2) +
  theme_minimal()
```



```
class4 = cutree(hc, k = 4)
ggplot(data) +
  aes(x = Glazing.area, y = Energy) +
  geom_point(size = 1L, colour = class4) +
  theme_minimal()
```

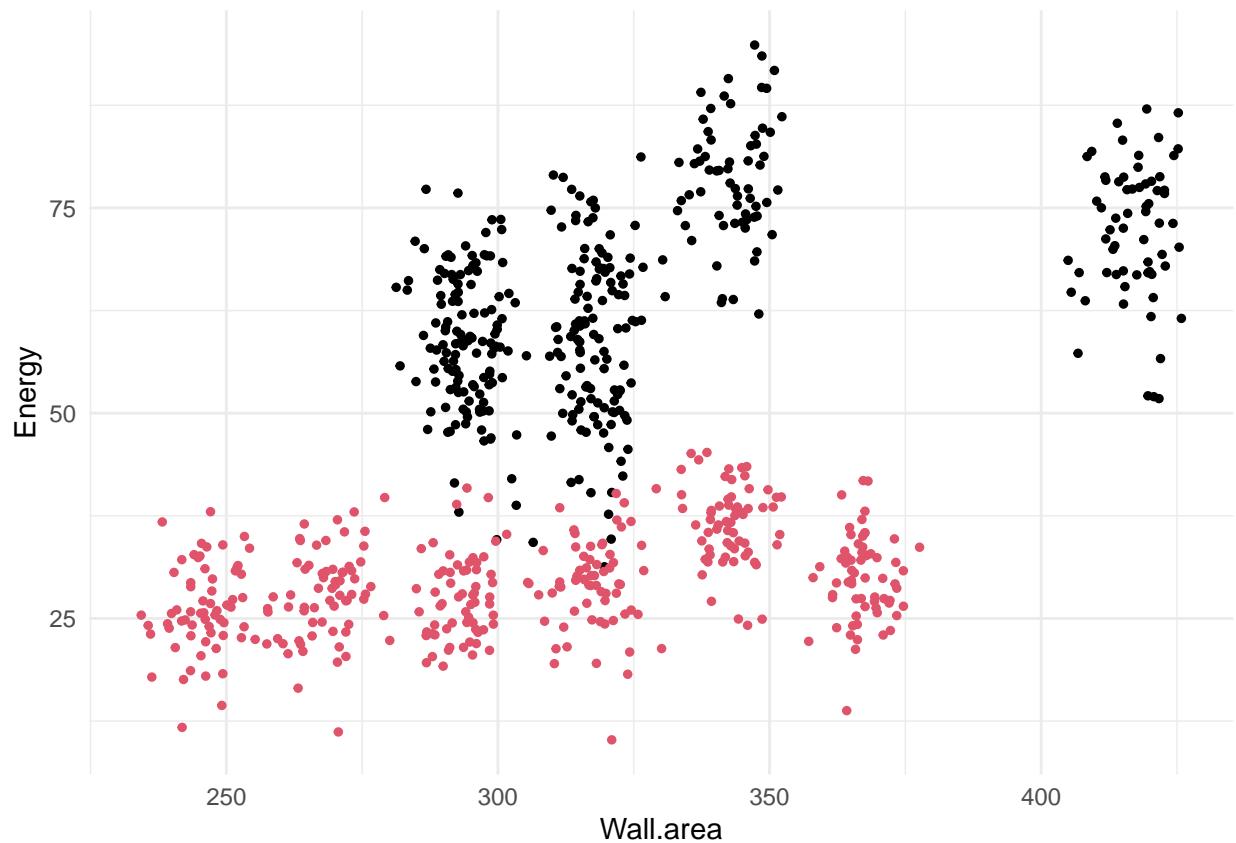


```
class12 = cutree(hc, k = 12)
ggplot(data) +
  aes(x = Glazing.area, y = Energy) +
  geom_point(size = 1L, colour = class12) +
  theme_minimal()
```

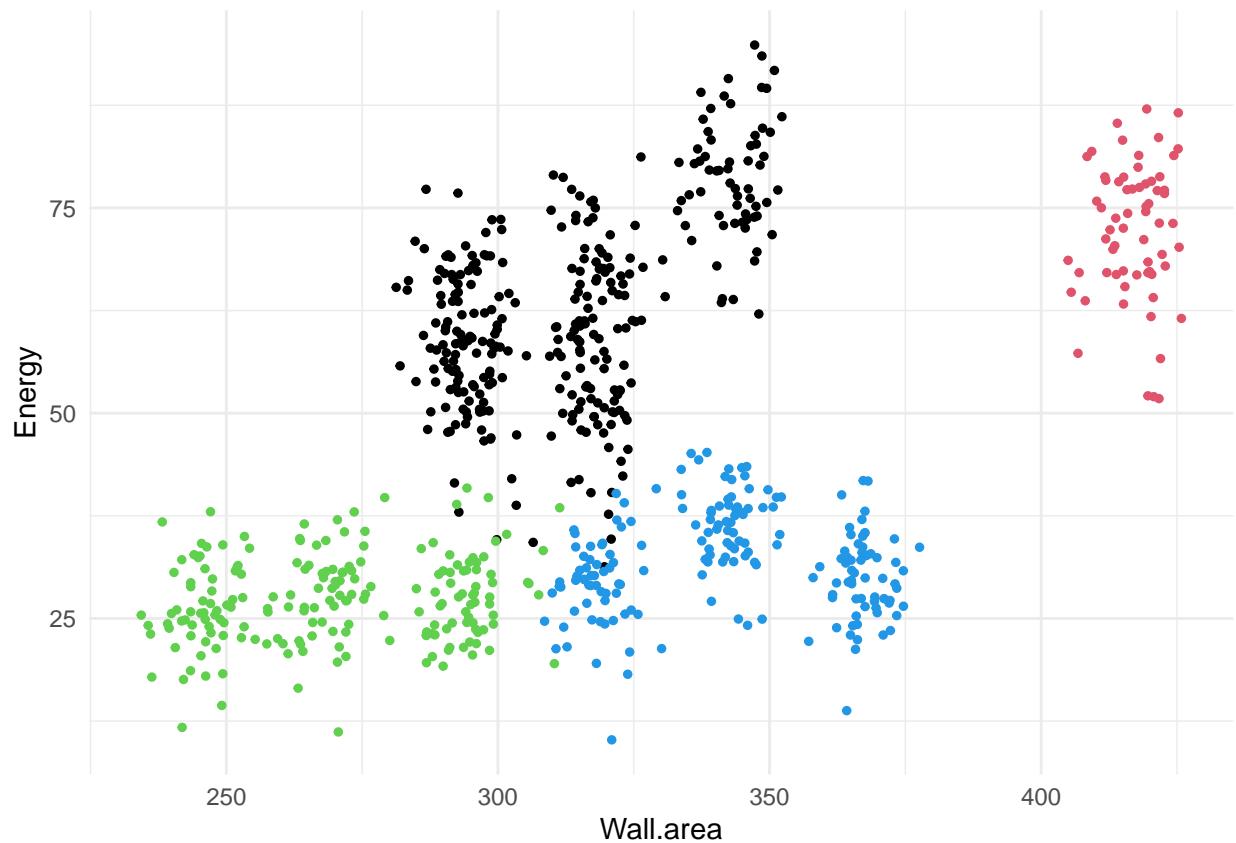


Selon la surface vitrée, seul le découpage en 2 classes semble pertinent. On aurait pu s'attendre à un découpage en 4 classes pertinents sur ce graphique mais ce n'est pas le cas.

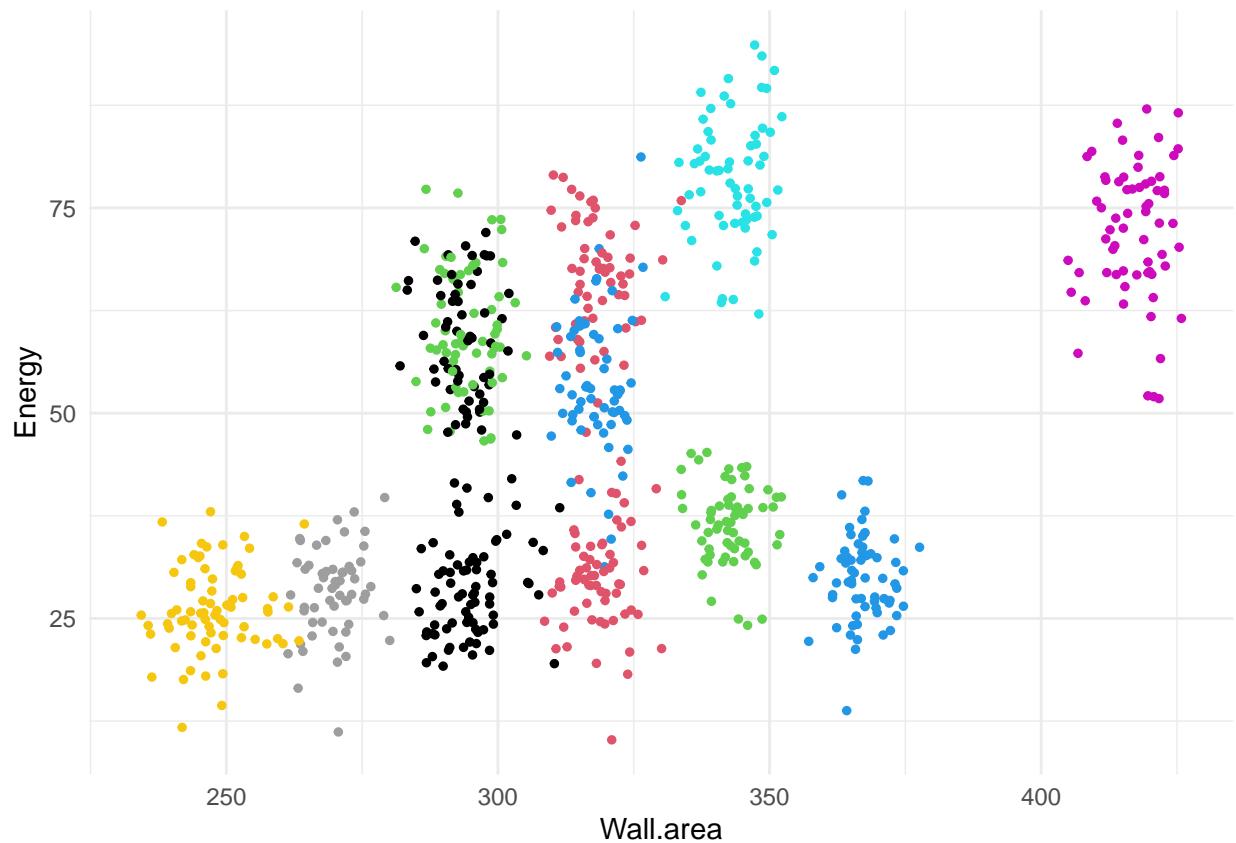
```
class2 = cutree(hc, k = 2)
ggplot(data) +
  aes(x = Wall.area, y = Energy) +
  geom_point(size = 1L, colour = class2) +
  theme_minimal()
```



```
class4 = cutree(hc, k = 4)
ggplot(data) +
  aes(x = Wall.area, y = Energy) +
  geom_point(size = 1L, colour = class4) +
  theme_minimal()
```



```
class12 = cutree(hc, k = 12)
ggplot(data) +
  aes(x = Wall.area, y = Energy) +
  geom_point(size = 1L, colour = class12) +
  theme_minimal()
```



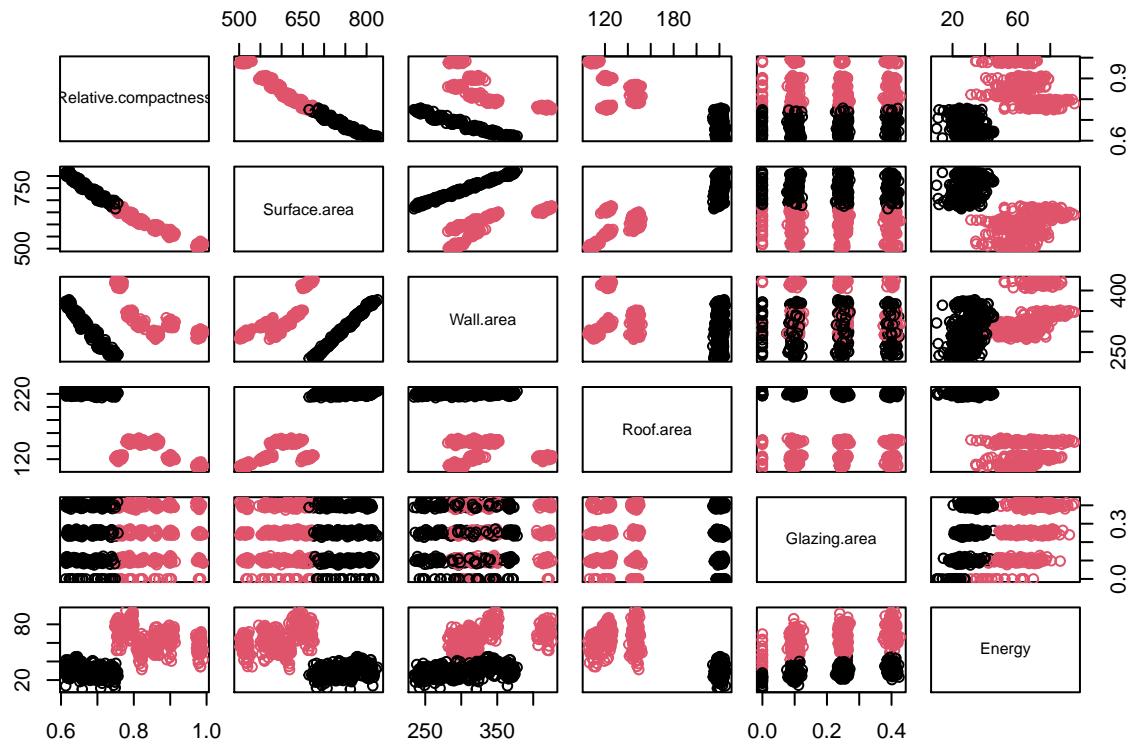
Finalement, le découpage en 4 classes semble pertinent selon la surface des murs.

### k-means

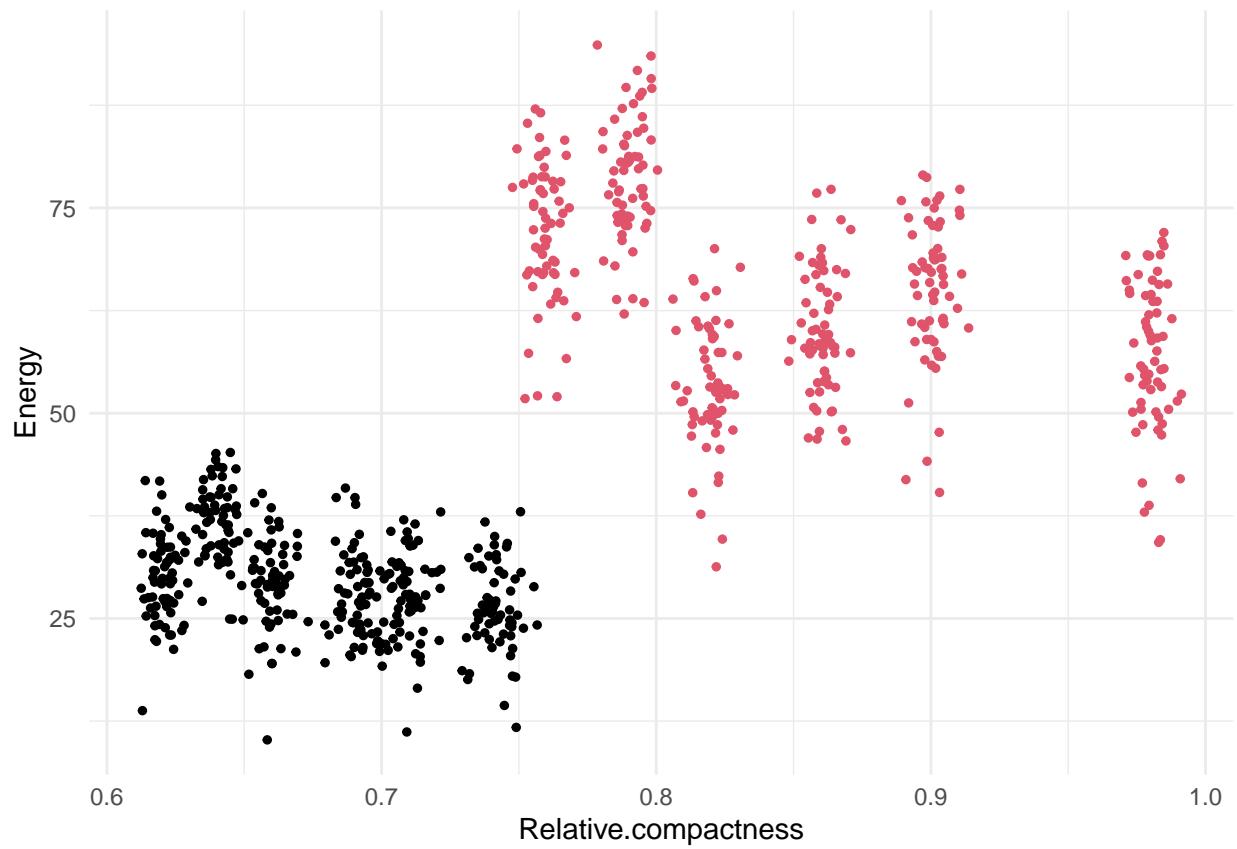
On utilise à présent un autre outil : k-means. Comparons les résultats avec ceux du clustering hiérarchique.

```
# k-means à 2 classes
kmres2 = kmeans(data[,c(1:5,7)], centers = 2)
kmclus2 = kmres2$cluster

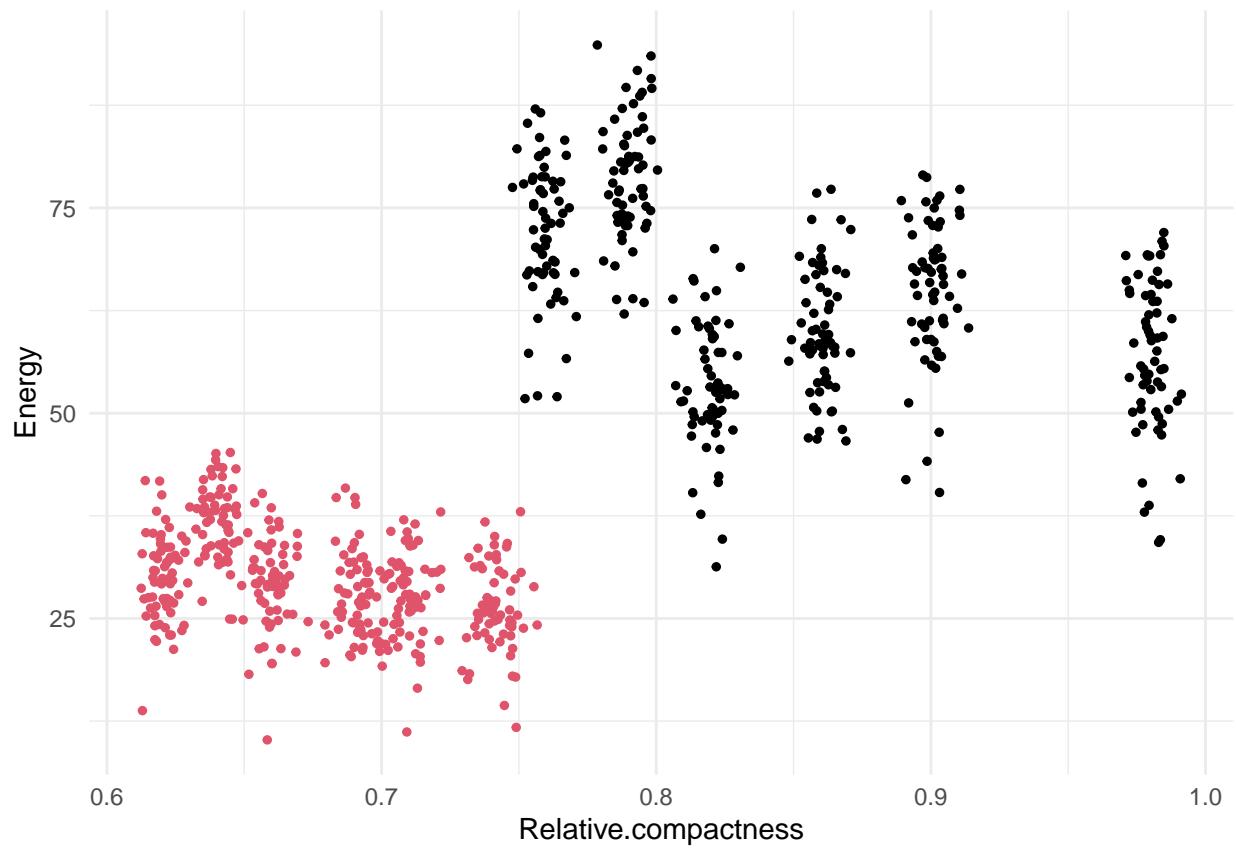
pairs(data[,quanti], col = kmclus2)
```



```
ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = kmclus2) +
  theme_minimal()
```



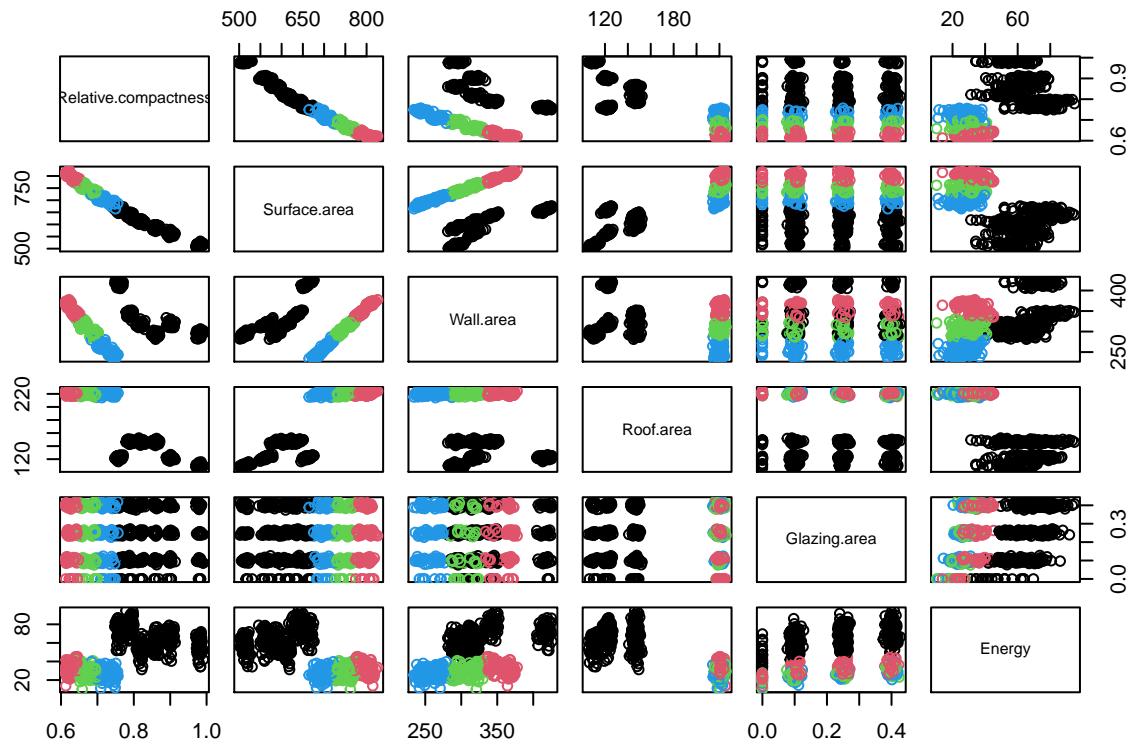
```
# clustering hierarchique à 2 classes
ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = class2) +
  theme_minimal()
```



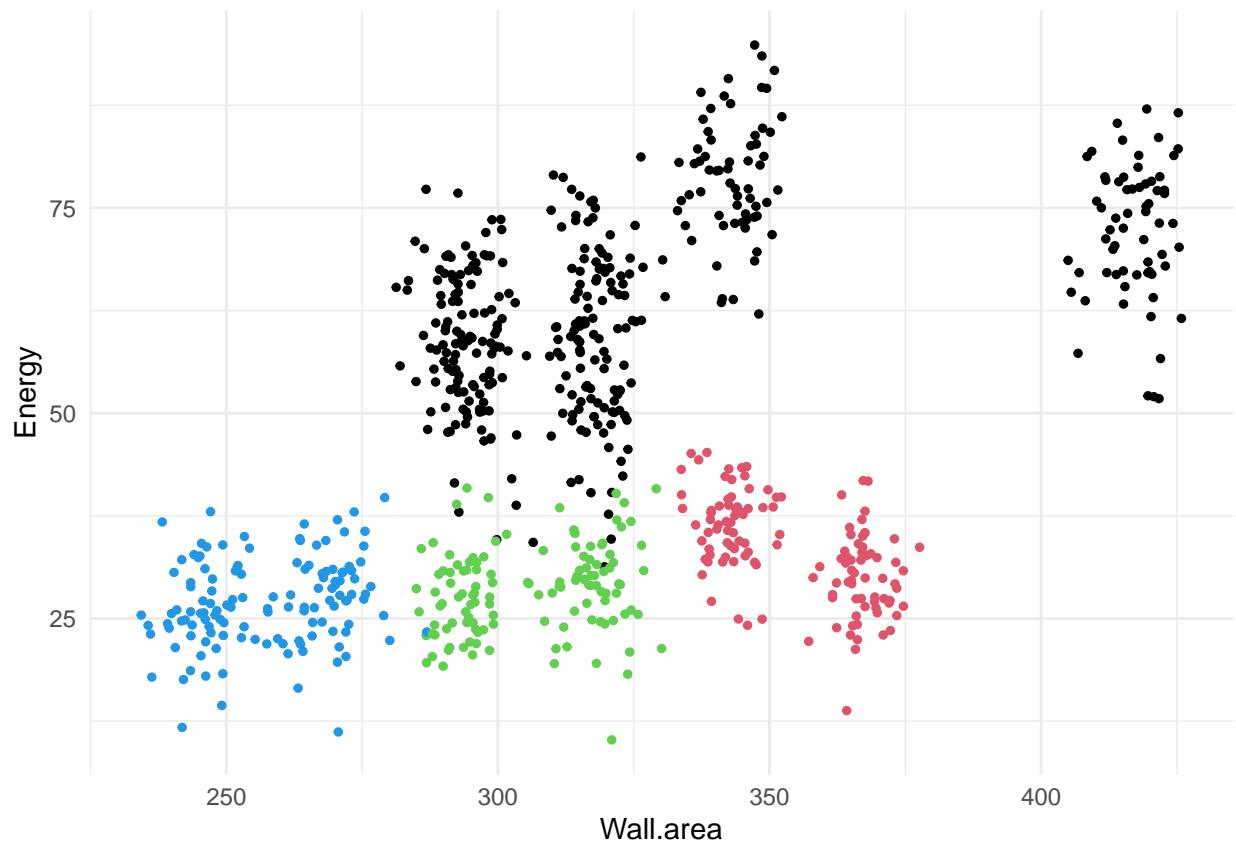
Pour le découpage à deux classes, les deux méthodes fournissent exactement les mêmes résultats.

```
# k-means à 4 classes
kmres4 = kmeans(data[,c(1:5,7)], centers = 4)
kmclus4 = kmres4$cluster

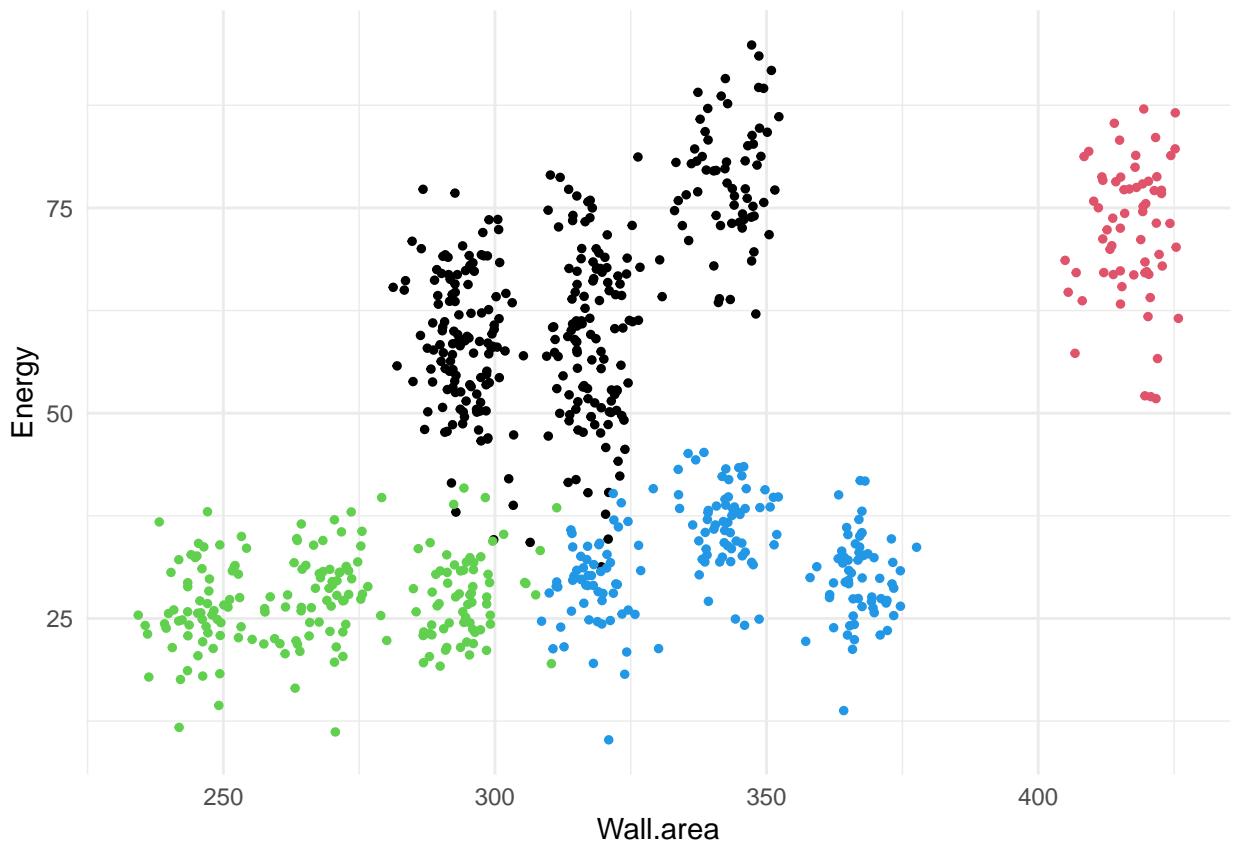
pairs(data[,quanti], col = kmclus4)
```



```
ggplot(data) +
  aes(x = Wall.area, y = Energy) +
  geom_point(size = 1L, colour = kmclust4) +
  theme_minimal()
```



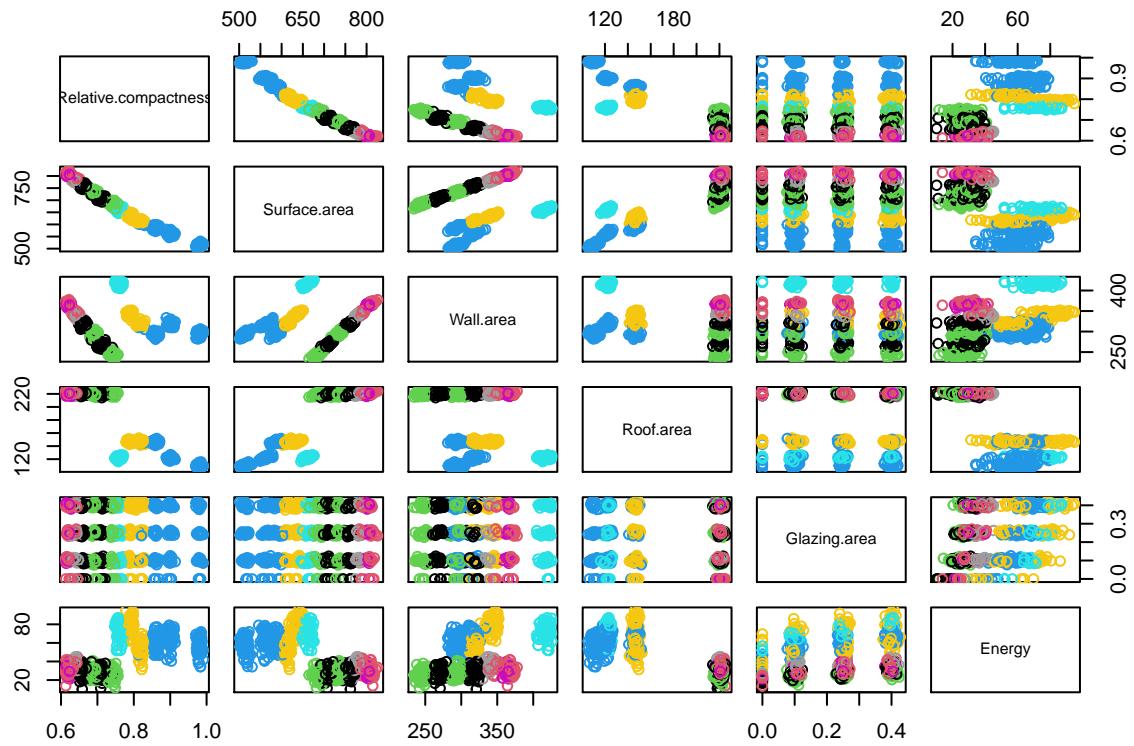
```
# clustering hierarchique à 4 classes
ggplot(data) +
  aes(x = Wall.area, y = Energy) +
  geom_point(size = 1L, colour = class4) +
  theme_minimal()
```



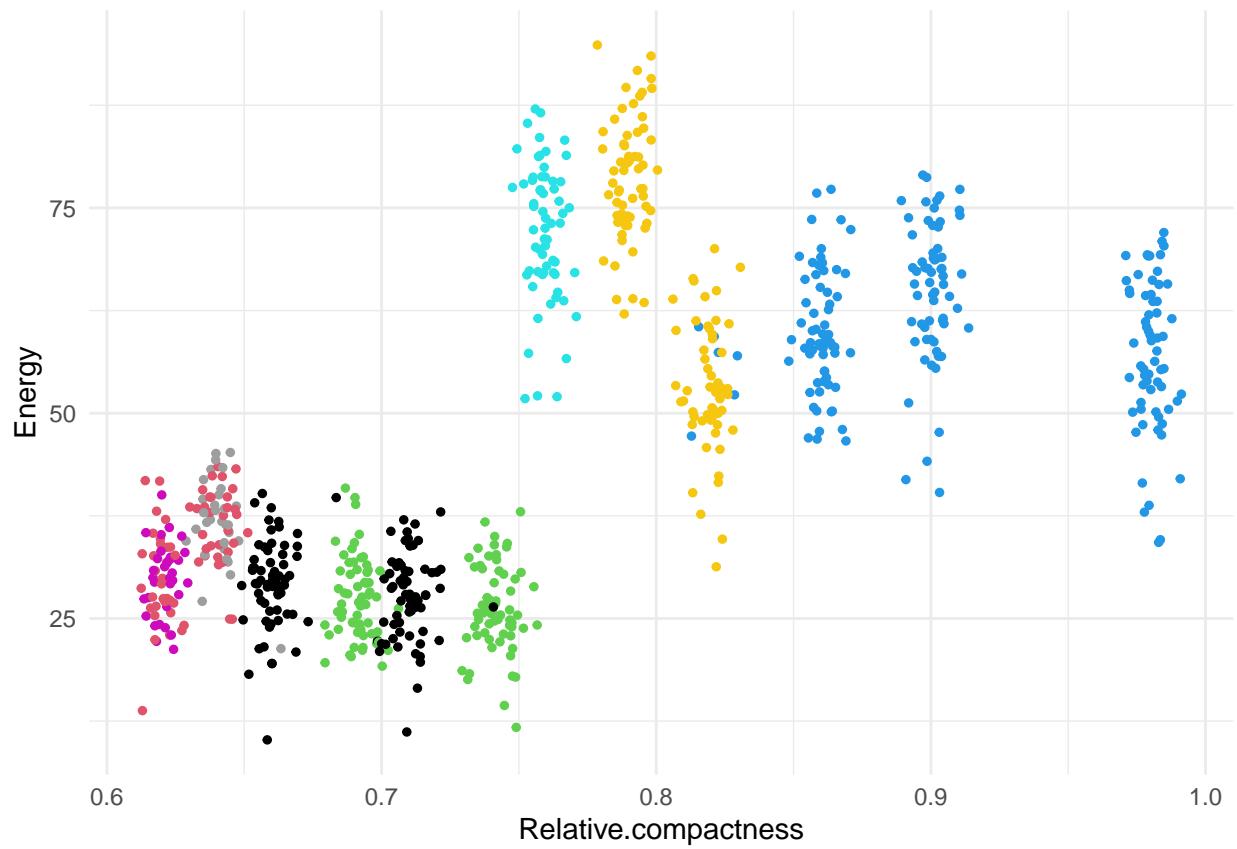
Après plusieurs exécutions, k-means ne donne presque jamais le même résultat que le clustering hiérarchiques. Alors que le clustering hiérarchiques semble créer des classes pertinentes selon la surface des murs, le k-means réalise souvent un découpage moins pertinent. En effet, le découpage issu de k-means consiste souvent à séparer en deux les données, puis à séparer en trois un des deux groupes obtenus de façon assez arbitraire. Le découpage en 4 classes avec k-means est donc peu pertinent.

```
# k-means à 12 classes
kmres12 = kmeans(data[,c(1:5,7)], centers = 12)
kmclus12 = kmres12$cluster

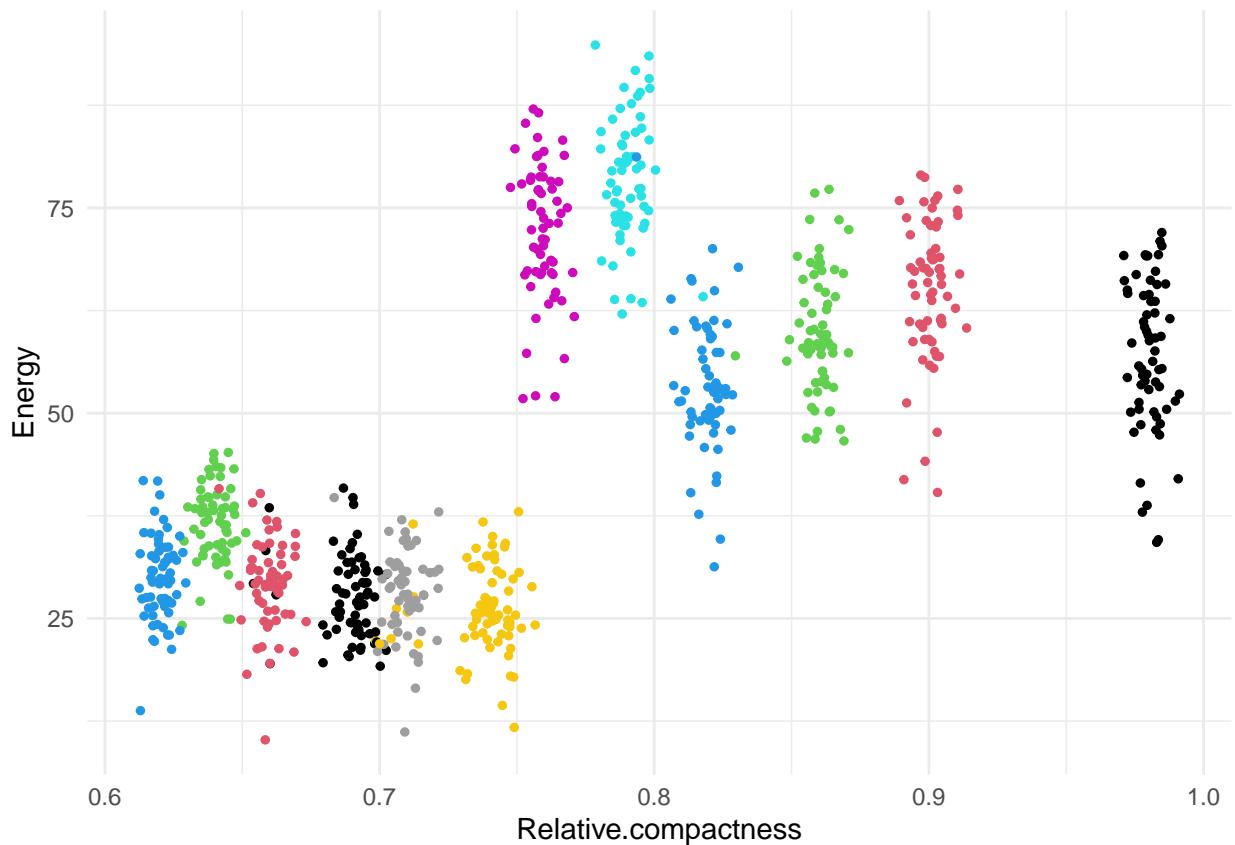
pairs(data[,quanti], col = kmclus12)
```



```
ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = kmclus12) +
  theme_minimal()
```



```
# clustering hierarchique à 12 classes
ggplot(data) +
  aes(x = Relative.compactness, y = Energy) +
  geom_point(size = 1L, colour = class12) +
  theme_minimal()
```



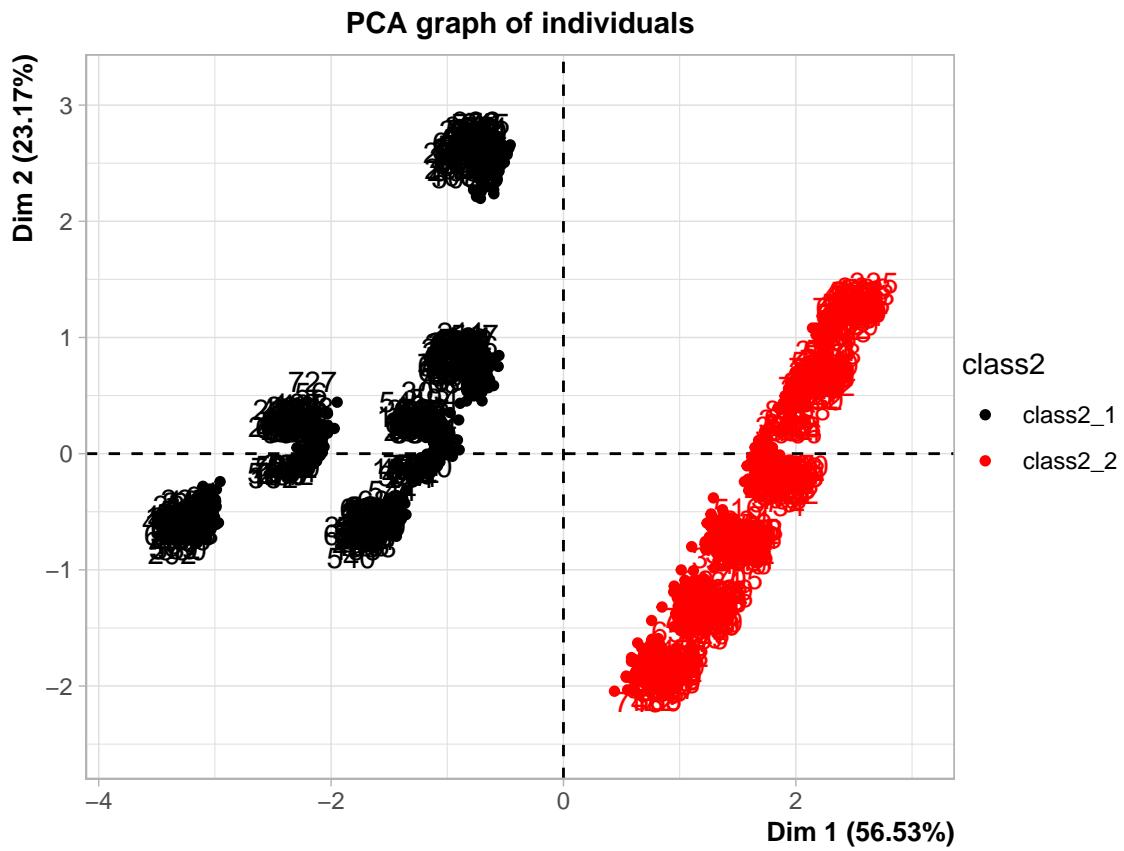
Selon les exécutions, k-means donne ici des résultats assez proches du clustering hiérarchique.

#### Visualisation des classes sur les axes de l'ACP

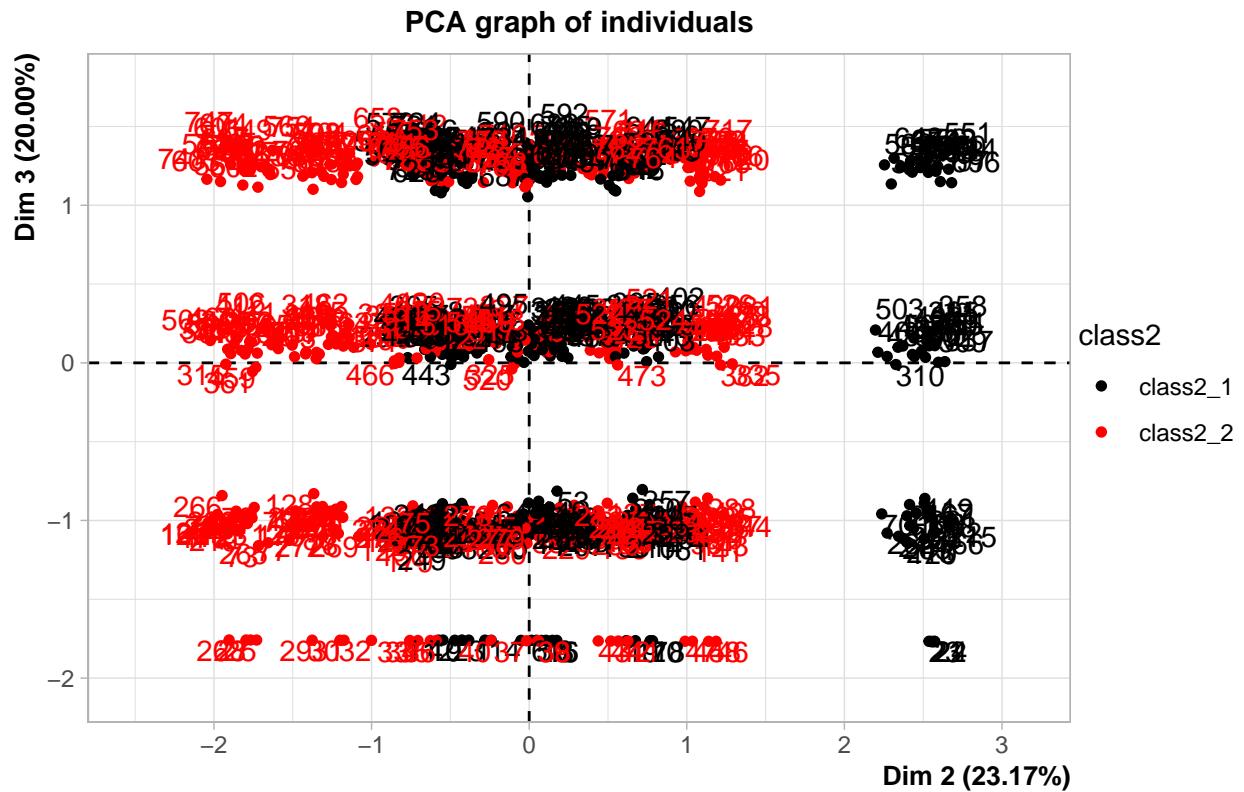
On utilise ici les classes obtenues par clustering hiérarchiques car celles-ci semblaient plus naturelles visuellement.

```
data$class2 = as.factor(class2)
data$class4 = as.factor(class4)
data$class12 = as.factor(class12)
res.acp <- PCA(data,scale.unit=T,quali.sup=c(quali,11,12,13),quanti.sup=9,ncp=8, graph=F)

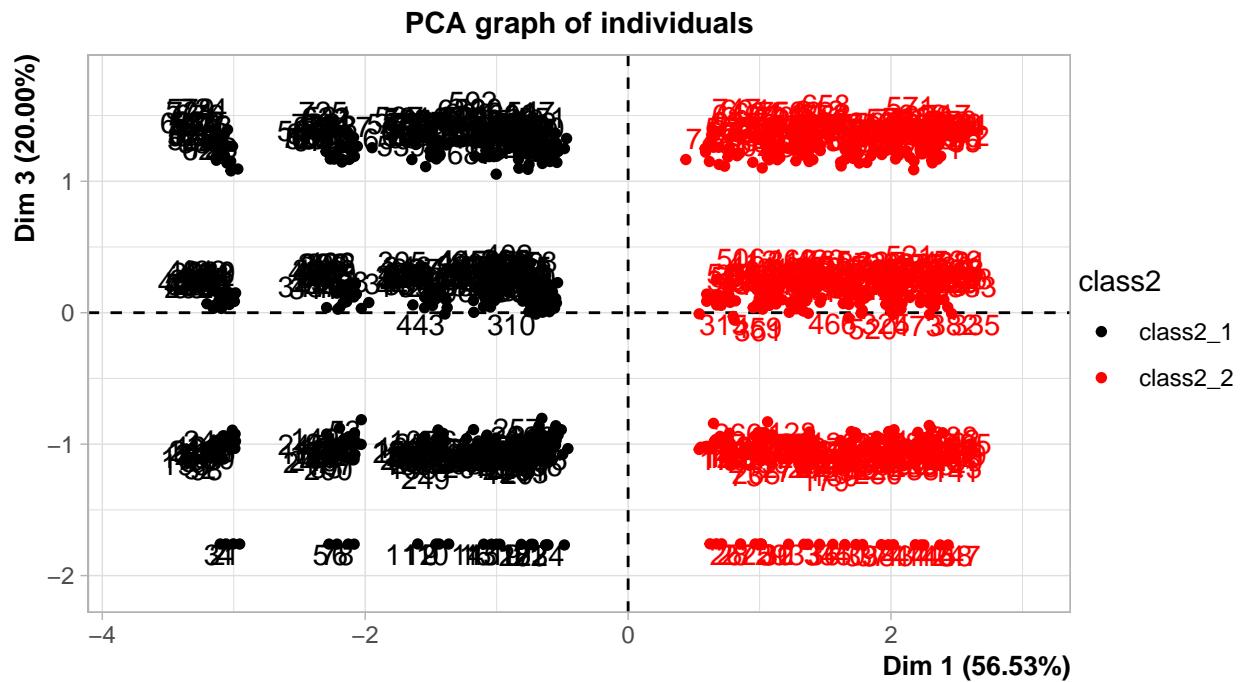
plot(res.acp, choix="ind", axes = c(1,2), invisible="quali", habillage="class2")
```



```
plot(res.acp, choix="ind", axes = c(2,3), invisible="quali", habillage="class2")
```

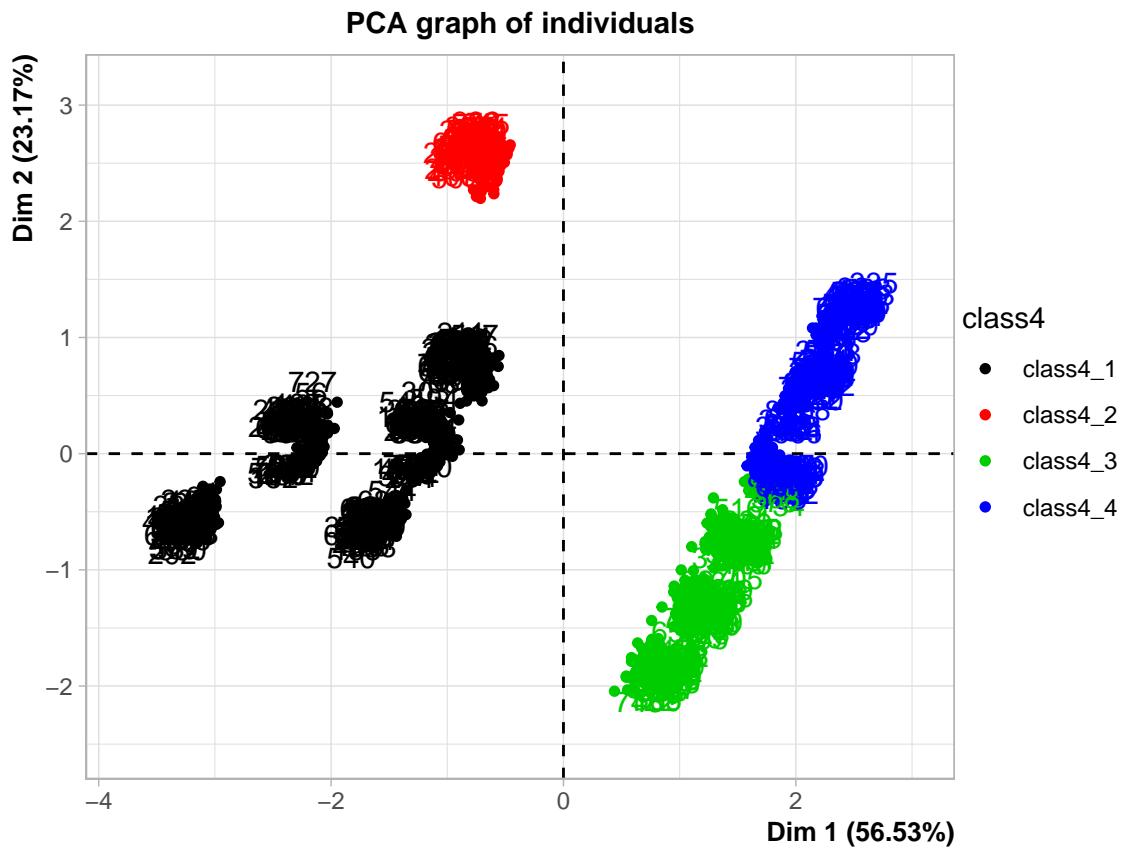


```
plot(res.acp, choix="ind", axes = c(1,3), invisible="quali", habillage="class2")
```

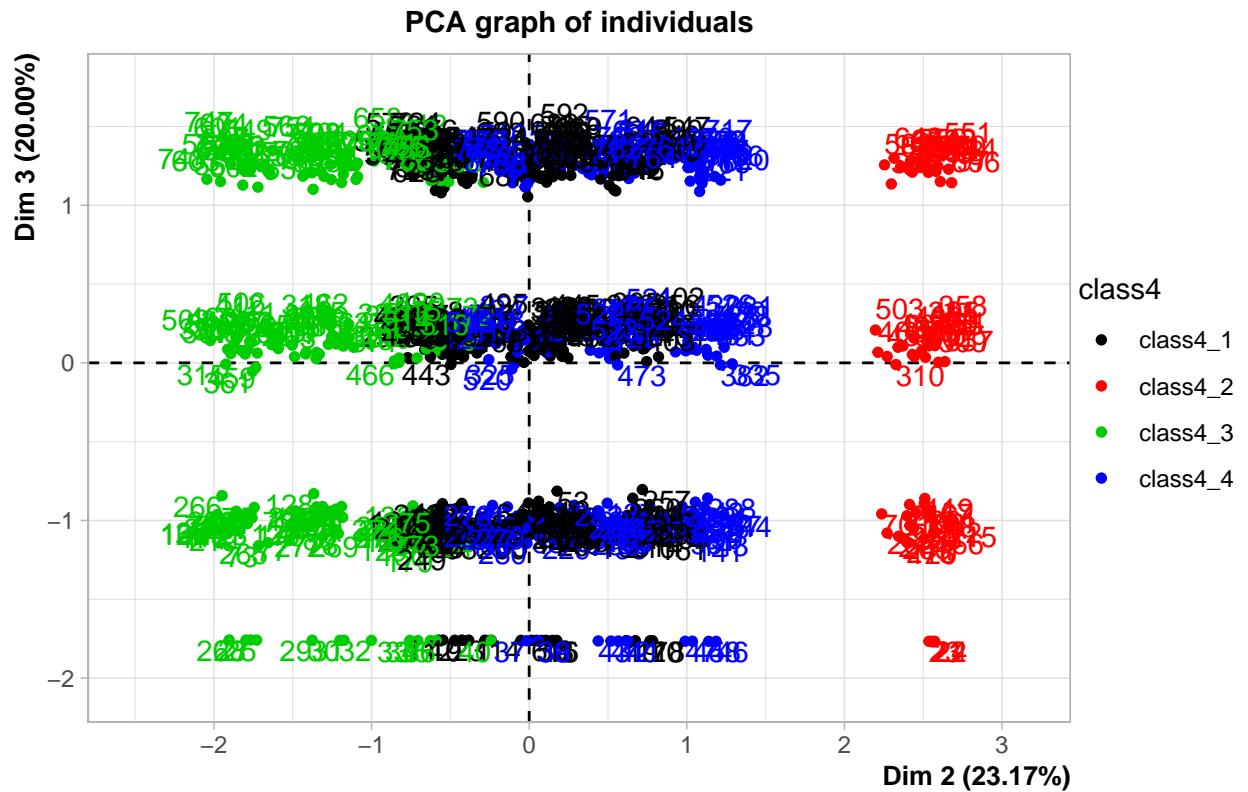


On voit ici que les données sont bien découpées en 2 classes selon le premier axe. Cela donne une bonne confiance en le découpage en 2 classes.

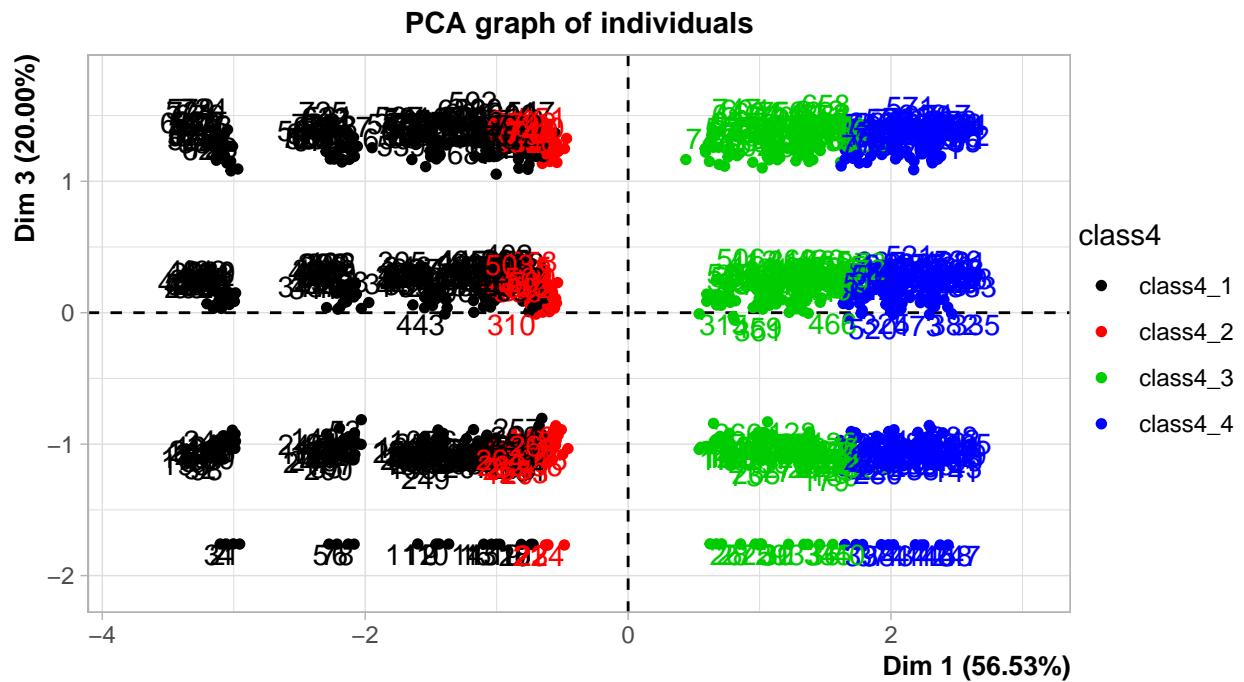
```
plot(res.acp, choix="ind", axes = c(1,2), invisible="quali", habillage="class4")
```



```
plot(res.acp, choix="ind", axes = c(2,3), invisible="quali", habillage="class4")
```

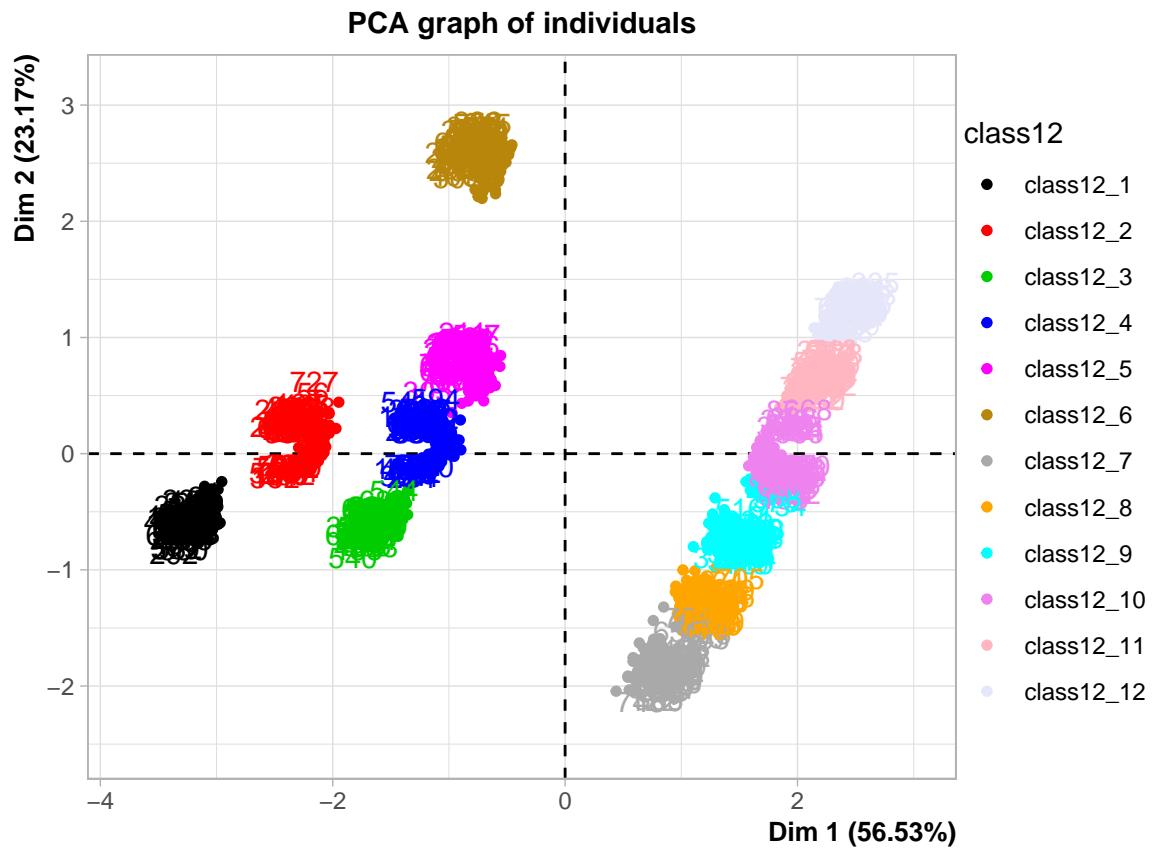


```
plot(res.acp, choix="ind", axes = c(1,3), invisible="quali", habillage="class4")
```

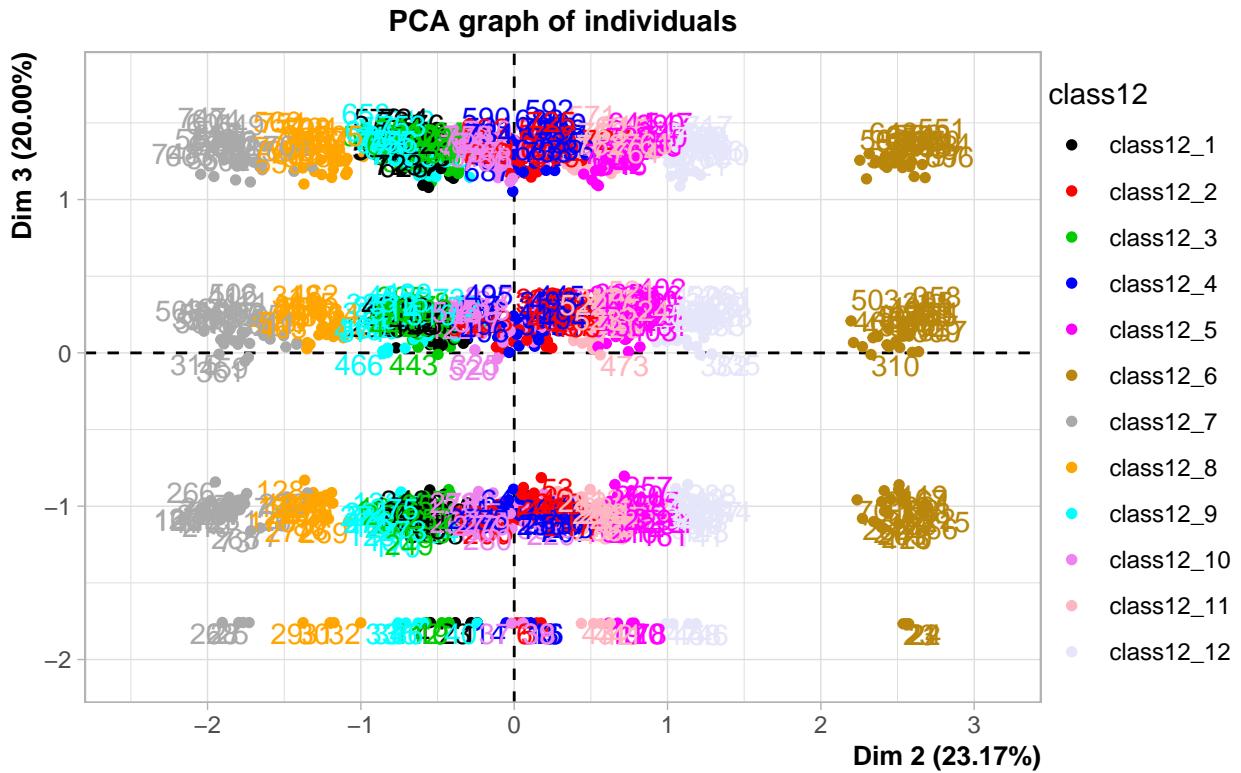


Le découpage en 4 classes se fait selon les axes 1 et 2, même si il est moins évident que celui en 2 classes. (notamment sur les classes 3 et 4 ci-dessus).

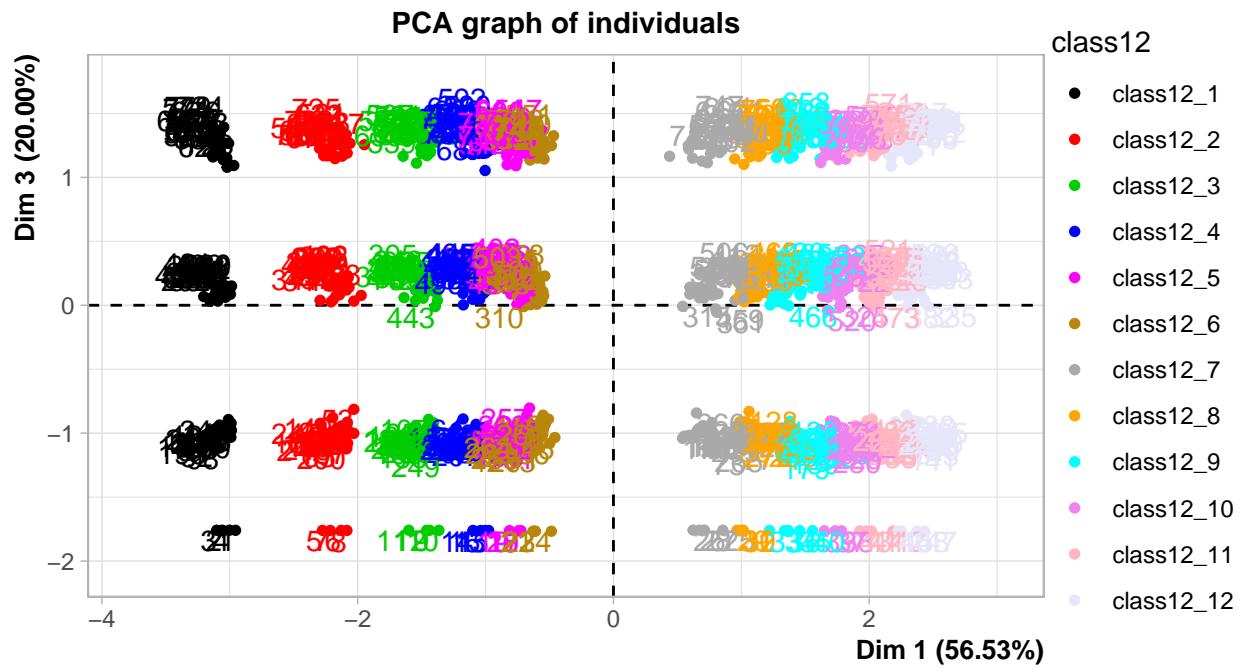
```
plot(res.acp, choix="ind", axes = c(1,2), invisible="quali", habillage="class12")
```



```
plot(res.acp, choix="ind", axes = c(2,3), invisible="quali", habillage="class12")
```



```
plot(res.acp, choix="ind", axes = c(1,3), invisible="quali", habillage="class12")
```



Ici, on peut avoir la même observation que pour le découpage en 4 classes : le découpage est moins net que celui en 2 classes et se fait selon les axes 1 et 2.

## Modèles linéaires

### Modèle fonction des variables quantitatives

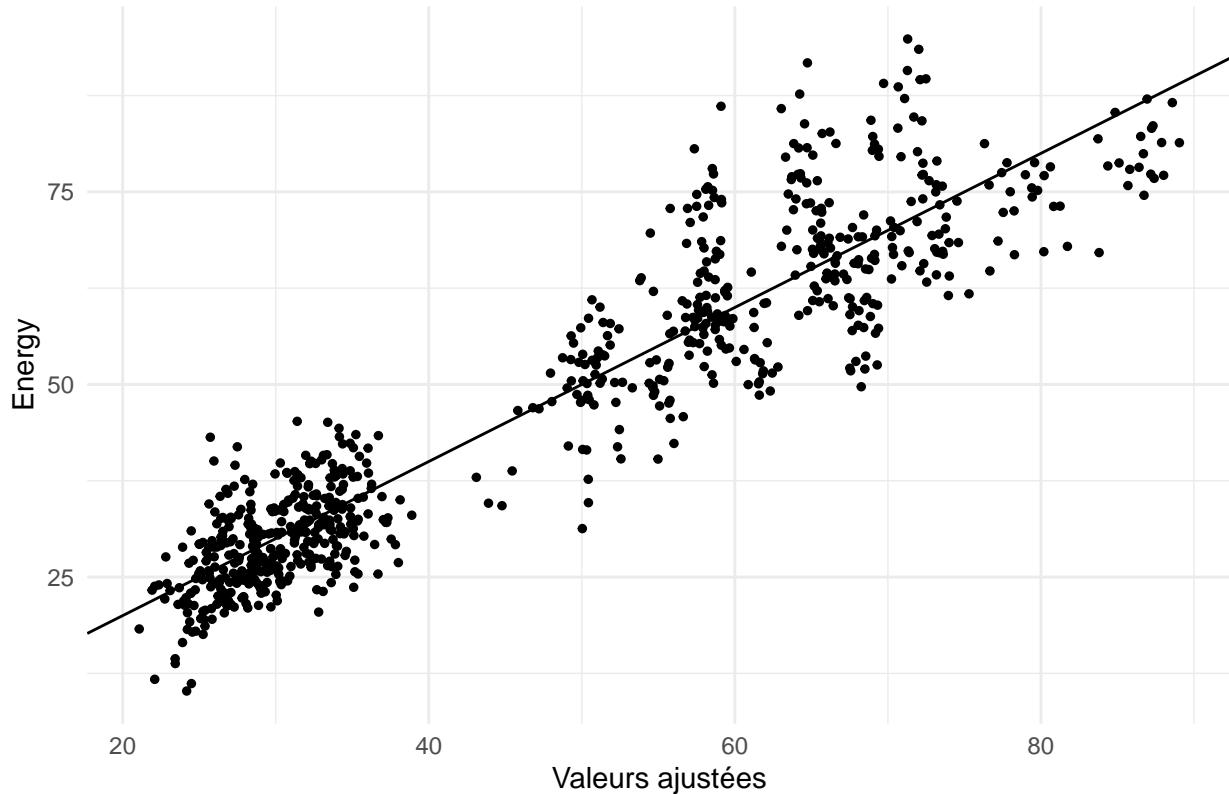
Nous étudierons dans un premier temps des modèles linéaires expliquant la variable quantitative `Energy` en fonction des variables *quantitatives* uniquement. Nous écarterons cependant la variable `Roof.area`, car nous avons vu qu'elle était combinaison linéaire des variables `Wall.area` et `Surface.area`. La conserver rendrait notre modèle singulier.

### Modèle avec interactions

Nous essayons un premier modèle contenant les variables explicatives et leurs interactions.

```
model_quanti_complet = lm(Energy ~ . - Roof.area)^2, data = data[,quanti])
r.sq = summary(model_quanti_complet)$r.squared ; paste("R^2 =", r.sq)
data$fitted_quanti_complet = model_quanti_complet$fitted.values
ggplot(data) +
  aes(x = fitted_quanti_complet, y = Energy) +
  geom_point(size = 1L) +
  geom_abline(slope = 1) +
  labs(title = "Régression linéaire de `Energy` en fonction des variables quantitatives",
       x = "Valeurs ajustées") +
  theme_minimal()
```

## Regression linéaire de 'Energy' en fonction des variables quantitatives



```
## [1] "R² = 0.869233997503484"
```

Le modèle obtenu semble déjà bien passer au sein des données : on trouve un  $R^2$  de 0.8952.

Cependant, ce modèle conserve beaucoup de variables, on peut donc se demander si elles sont toutes pertinentes. Nous allons donc essayer de n'en conserver que certaines.

### Selection de variables par critère BIC

Nous allons réaliser dans un premier temps une sélection de variables par critère BIC.

```
modselect_quanti_bic_back = stepAIC(model_quanti_complet, trace=FALSE, direction=c("backward"), k=log(nrow
paste("Energy ~", paste(dimnames(modselect_quanti_bic_back$qr$qr)[[2]][-1], collapse=" + ")))
```

```
## [1] "Energy ~ Relative.compactness + Surface.area + Wall.area + Glazing.area + Relative.compactness:"
```

Le modèle sélectionné conserve nos 5 des variables quantitatives, mais supprime certaines de leurs interactions. Afin de s'assurer que l'on peut simplifier notre modèle, on réalise un test de sous-modèle entre le modèle complet et le modèle sélectionné.

```
anova(modselect_quanti_bic_back, model_quanti_complet)
r.sq = summary(modselect_quanti_bic_back)$r.squared ; paste("R² =", r.sq)
```

```
## Analysis of Variance Table
##
## Model 1: Energy ~ Relative.compactness + Surface.area + Wall.area + Glazing.area +
##           Relative.compactness:Surface.area + Relative.compactness:Glazing.area +
##           Surface.area:Wall.area + Wall.area:Glazing.area
## Model 2: Energy ~ ((Relative.compactness + Surface.area + Wall.area +
##           Roof.area + Glazing.area)^2 - Roof.area)^2
```

```

##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1    759 39772
## 2    757 39571  2     200.47 1.9175 0.1477
## [1] "R2 = 0.868571517617869"

```

On obtient une p-valeur de 0.2864. On ne rejette donc pas notre modèle sélectionné au seuil de 5 %. Ainsi, notre modèle possède un  $R^2$  de 0.8943, ce qui est à peine plus faible que ce que nous avions avec toutes nos variables.

### Selection de variable par régression régularisée

On peut également tenter de sélectionner nos variables par régression régularisée. Nous nous contenterons d'étudier le modèle sans interactions car la fonction que nous utiliserons ne traite pas de priorisation entre les effets principaux et les interactions.

```

# centrage et réduction des données
energy = scale(data["Energy"], center=T, scale=T)
# f = as.formula(paste("Energy ~ ", paste(dimnames(data[, c(1, 2, 3, 7)])[[2]], collapse=" + "), " )~2"))
# expli = scale(model.matrix(f, data)[-1], center=T, scale=T)
expli = scale(data[quanti[-6]], center=T, scale=T)

# création du tableau de tau
tau_seq <- seq(0, 0.1, by = 0.0001)

fitridge <- glmnet(x = expli, y = energy, family = "gaussian", alpha = 0, lambda = tau_seq, intercept = TRUE)

# récupération du tau minimum par validation croisée
ridge_cv = cv.glmnet(x = expli, y = energy, family = "gaussian", alpha = 0, lambda = tau_seq, intercept = TRUE)
tau_min_ridge = ridge_cv$lambda.min
print(tau_min_ridge)

# affichage des estimations de tau
dfridge = data.frame(tau = rep(fitridge$lambda, ncol(expli)),
                      theta = as.vector(t(fitridge$beta)),
                      variable = rep(colnames(expli), each = length(fitridge$lambda)))
ggplot(dfridge, aes(x = tau, y = theta, col = variable)) +
  geom_line() +
  geom_vline(xintercept = tau_min_ridge, linetype = "dotted", color = "red") +
  theme(legend.position = "right")

# illustration de la meilleure valeur de tau
df2 = data.frame(tau = ridge_cv$lambda, MSE = ridge_cv$cvm, cvup = ridge_cv$cvup, cvlo = ridge_cv$cvlo)
ggplot(df2) +
  geom_line(aes(x = tau, y = MSE)) +
  geom_vline(xintercept = tau_min_ridge, col = "red", linetype = "dotted") +
  geom_line(aes(x = tau, y = cvup), col = "blue", linetype = "dotted") +
  geom_line(aes(x = tau, y = cvlo), col = "blue", linetype = "dotted")

```

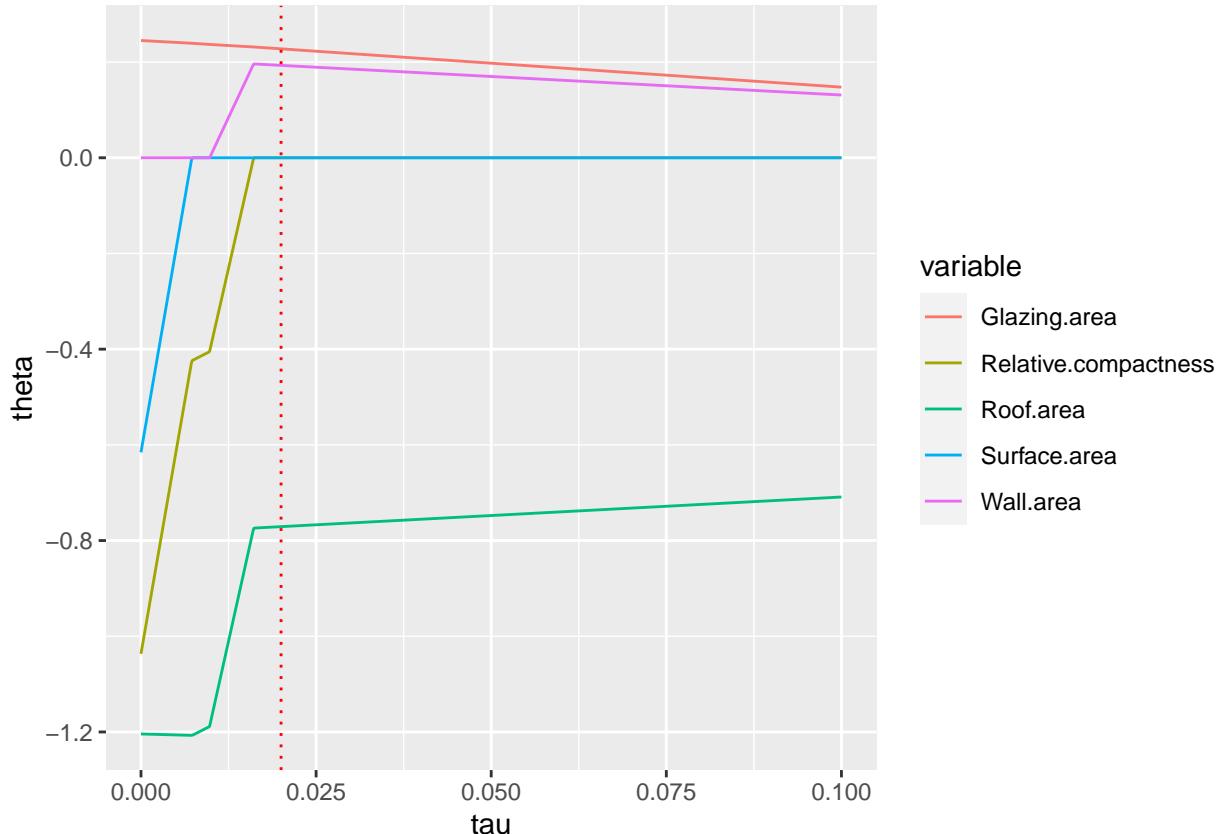
**Ridge** La régression Ridge ne nous donne pas vraiment de résultats intéressant, il est difficile de voir clairement quelles variables écarter. Nous allons donc essayer une régression lasso.

```

fitlasso <- glmnet(x = expli, y = energy, family = "gaussian", alpha = 1, lambda = tau_seq, intercept =
dlasso=data.frame(tau = rep(fitlasso$lambda,ncol(expli)),
theta=as.vector(t(fitlasso$beta)),
variable=rep(colnames(expli),each=length(fitlasso$lambda)))
ggplot(dlasso,aes(x=tau,y=theta,col=variable)) +
geom_line() +
geom_vline(xintercept = 0.02, linetype = "dotted", color = "red") +
theme(legend.position="right")

```

## Lasso



```

# récupération du tau minimum par validation croisée
lasso_cv = cv.glmnet(x = expli, y = energy, family = "gaussian", alpha = 1, lambda = tau_seq, intercept =
tau_min_lasso = lasso_cv$lambda.min
print(tau_min_lasso)
# illustration de la meilleure valeur de tau
df3=data.frame(tau=lasso_cv$lambda,MSE=lasso_cv$cvm,cvup=lasso_cv$cvup,cvlo=lasso_cv$cvlo)
ggplot(df3)+ 
  geom_line(aes(x=tau,y=MSE))+ 
  geom_vline(xintercept = tau_min_lasso,col="red",linetype="dotted")+
  geom_line(aes(x=tau,y=cvup),col="blue",linetype="dotted")+
  geom_line(aes(x=tau,y=cvlo),col="blue",linetype="dotted")

```

Certains variables s'écrasent bien à 0. En choisissant un seuil de 0.02, la régularisation écarterait les variables `Surface.area` et `Relative.compactness`. Cependant ce seuil est arbitraire. En essayant de le choisir par validation croisée pour minimiser le critère MSE, nous trouvons 0. Cela signifierait qu'il faudrait conserver toutes les variables. Cela serait cohérent avec les résultats de la sélection par critère BIC, qui conservait toutes les variables (plus certaines interactions). Le fait de ne pas pouvoir tester les interactions nous limite

sur les conclusions que nous pouvons tirer des régression régularisées.

Nous n'avons jusqu'à présent considéré que les variables quantitatives. Nous allons maintenant essayer de voir si les variables qualitatives de notre jeu de données ne pourraient pas nous apporter des informations supplémentaires.

## Modèle dépendant de toutes les variables

On construit un modèle complet avec interactions, et on teste un sous-modèle sans interactions pour voir si on pourrait s'en passer.

```
model_complet = lm(Energy ~ . - Roof.area)^2, data = data[,c(1:9)])
data$fitted_complet = model_complet$fitted.values
model_no_interactions = lm(Energy ~ . - Roof.area, data = data[,c(1:9)])
anova(model_complet, model_no_interactions)

## Analysis of Variance Table
##
## Model 1: Energy ~ ((Relative.compactness + Surface.area + Wall.area +
##     Roof.area + Overall.height + orientation + Glazing.area +
##     Glazing.area.distr) - Roof.area)^2
## Model 2: Energy ~ (Relative.compactness + Surface.area + Wall.area + Roof.area +
##     Overall.height + orientation + Glazing.area + Glazing.area.distr) -
##     Roof.area
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     690 28596
## 2     754 35755 -64   -7158.6 2.6989 2.653e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La p-valeur est très faible : on doit conserver des interactions. Nous allons essayer de voir si nous ne pourrions pas nous en passer de certaines grâce à un algorithme de sélection de variables.

```
modselect_bic_back = stepAIC(model_complet,trace=FALSE,direction=c("backward"),k=log(nrow(data)))
formula = paste("Energy ~", paste(dimnames(modselect_bic_back$qr$qr)[[2]][-1], collapse=" + "))
## [1] "Energy ~ Relative.compactness + Surface.area + Wall.area + Overall.height.L + Glazing.area + Glazin
```

Ainsi la sélection laisse complètement de côté la variable orientation, et supprime également certaines interactions. On réalise un test de sous-modèle afin de savoir si nous pouvons simplifier le modèle complet.

```
anova(modselect_bic_back, model_complet)
summary(modselect_bic_back)
```

```
## Analysis of Variance Table
##
## Model 1: Energy ~ Relative.compactness + Surface.area + Wall.area + Overall.height +
##     Glazing.area + Glazing.area.distr + Relative.compactness:Surface.area +
##     Relative.compactness:Wall.area + Relative.compactness:Overall.height +
##     Relative.compactness:Glazing.area + Surface.area:Overall.height +
##     Wall.area:Overall.height + Wall.area:Glazing.area
## Model 2: Energy ~ ((Relative.compactness + Surface.area + Wall.area +
##     Roof.area + Overall.height + orientation + Glazing.area +
##     Glazing.area.distr) - Roof.area)^2
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     750 30060
## 2     690 28596 60   1464.2 0.5888 0.9943
##
```

```

## Call:
## lm(formula = Energy ~ Relative.compactness + Surface.area + Wall.area +
##     Overall.height + Glazing.area + Glazing.area.distr + Relative.compactness:Surface.area +
##     Relative.compactness:Wall.area + Relative.compactness:Overall.height +
##     Relative.compactness:Glazing.area + Surface.area:Overall.height +
##     Wall.area:Overall.height + Wall.area:Glazing.area, data = data[, ,
##     c(1:9)])
##
## Residuals:
##      Min    1Q Median    3Q   Max 
## -23.7266 -3.6705  0.0643  4.1012 19.3183 
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                -187.78909   72.68098 -2.584  0.00996  
## Relative.compactness        225.65319   88.82546  2.540  0.01127  
## Surface.area                  1.05531   0.17546  6.014 2.82e-09  
## Wall.area                   -1.48506   0.21641 -6.862 1.42e-11  
## Overall.height.L              110.25251   57.94602  1.903  0.05747  
## Glazing.area                 -106.60437   19.86074 -5.368 1.06e-07  
## Glazing.area.distr1           7.13980   1.16327  6.138 1.36e-09  
## Glazing.area.distr2           6.74169   1.16733  5.775 1.13e-08  
## Glazing.area.distr3           5.83976   1.16502  5.013 6.71e-07  
## Glazing.area.distr4           6.87640   1.16413  5.907 5.28e-09  
## Glazing.area.distr5           6.01828   1.16678  5.158 3.20e-07  
## Relative.compactness:Surface.area  -1.31266   0.22295 -5.888 5.91e-09  
## Relative.compactness:Wall.area      2.05508   0.28853  7.123 2.49e-12  
## Relative.compactness:Overall.height.L -199.92145   49.51111 -4.038 5.95e-05  
## Relative.compactness:Glazing.area      130.28044   16.68716  7.807 1.97e-14  
## Surface.area:Overall.height.L          0.18401   0.07073  2.602  0.00946  
## Wall.area:Overall.height.L            -0.17965   0.07009 -2.563  0.01057  
## Wall.area:Glazing.area                 0.12224   0.04007  3.050  0.00237  
##
## (Intercept)                    **  
## Relative.compactness                  *  
## Surface.area                      ***  
## Wall.area                         ***  
## Overall.height.L                   .  
## Glazing.area                      ***  
## Glazing.area.distr1                  ***  
## Glazing.area.distr2                  ***  
## Glazing.area.distr3                  ***  
## Glazing.area.distr4                  ***  
## Glazing.area.distr5                  ***  
## Relative.compactness:Surface.area    ***  
## Relative.compactness:Wall.area       ***  
## Relative.compactness:Overall.height.L ***  
## Relative.compactness:Glazing.area    ***  
## Surface.area:Overall.height.L        **  
## Wall.area:Overall.height.L          *  
## Wall.area:Glazing.area                 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##

```

```

## Residual standard error: 6.331 on 750 degrees of freedom
## Multiple R-squared:  0.9007, Adjusted R-squared:  0.8984
## F-statistic:  400 on 17 and 750 DF,  p-value: < 2.2e-16

```

On obtient une p-valeur supérieure à 99%, on ne rejette donc pas le modèle réduit, et on peut simplifier le modèle.

## Pouvoir de prédiction du modèle

Maintenant qu'on a déterminé les variables d'intérêt, nous allons tester le pouvoir de prédiction de notre modèle. Pour cela, nous allons faire de la validation croisée.

```

set.seed(42)
train.control = trainControl(method = "cv", number = 10, p = 0.8)
model_cv = train(modselect_bic_back$terms, data = data, method = "lm", trControl = train.control)
print(model_cv)

## Linear Regression
##
## 768 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 692, 691, 692, 692, 691, 691, ...
## Resampling results:
##
##   RMSE      Rsquared     MAE
##   6.381876  0.8979905  4.956418
##
## Tuning parameter 'intercept' was held constant at a value of TRUE

```

On obtient un  $R^2$  de presque 90%, le modèle passe donc bien par les données. De plus, l'erreur moyenne absolue de moins de 5. En sachant que la variable Energy s'étale de 10 à 95, notre erreur est assez faible : elle représente moins de 6% de l'étendue de nos données.

## Modèle linéaire généralisée

### Mise en forme de la data Frame

```

# Modèle linéaire généralisée
# Définition de Energy.efficiencyBIs
new_data <- data[,c(-11,-12,-13,-14,-15)]
data.mlg = data.frame(new_data, Energy.efficiency.bis = rep(0, nrow(new_data)))
data.mlg$Energy.efficiency.bis[which(data.mlg$Energy.efficiency == "A" | data.mlg$Energy.efficiency == "B")]
data.mlg$Energy.efficiency.bis[which(data.mlg$Energy.efficiency != "A" & data.mlg$Energy.efficiency != "B")]
data.mlg$Energy.efficiency.bis = as.factor(data.mlg$Energy.efficiency.bis)
data.mlg<-data.mlg[,c(-4,-9,-10)]

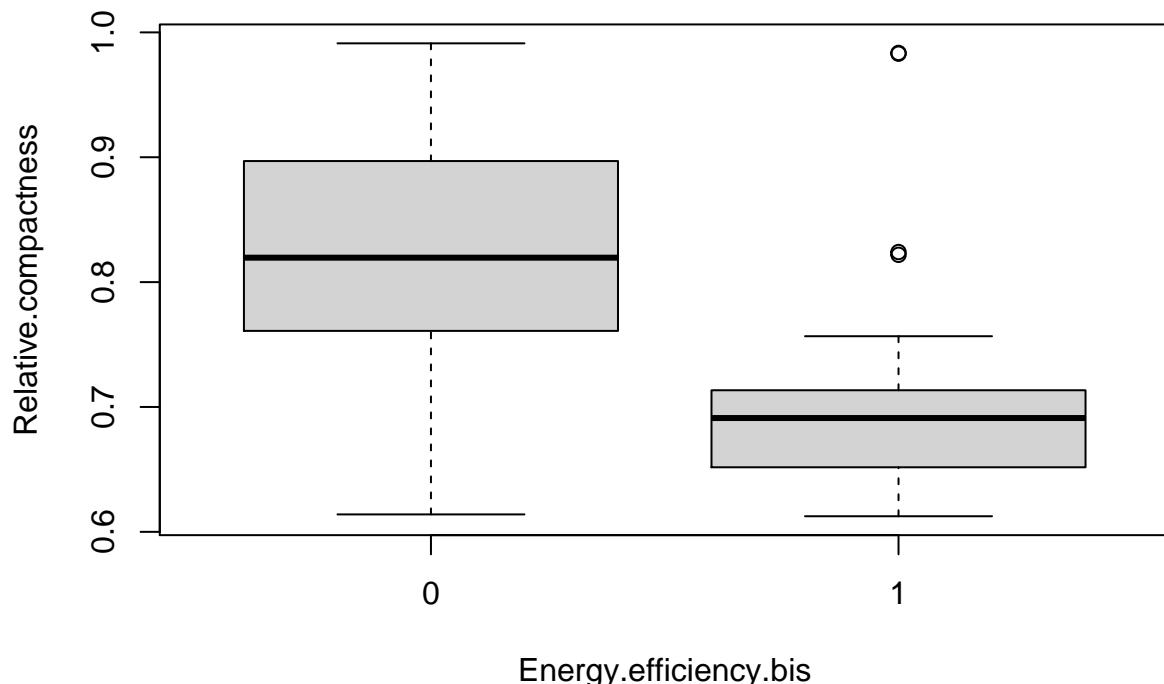
```

### Statistique descriptive

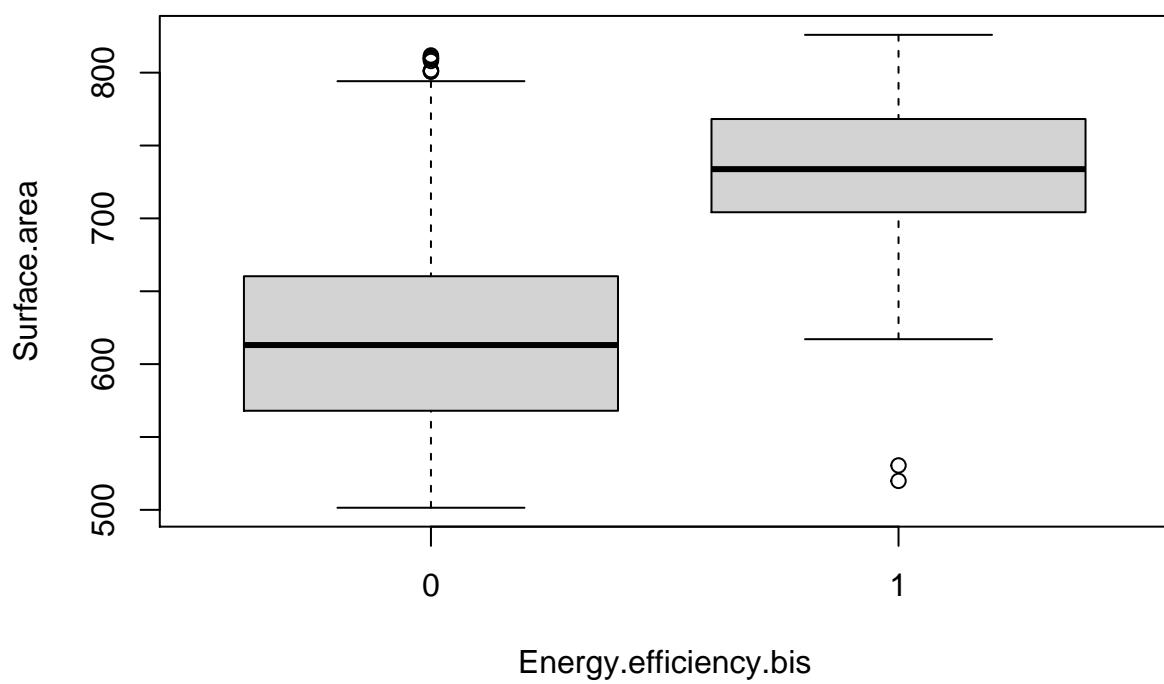
```

#Statistique Descriptive
boxplot(Relative.compactness~ Energy.efficiency.bis,data=data.mlg)

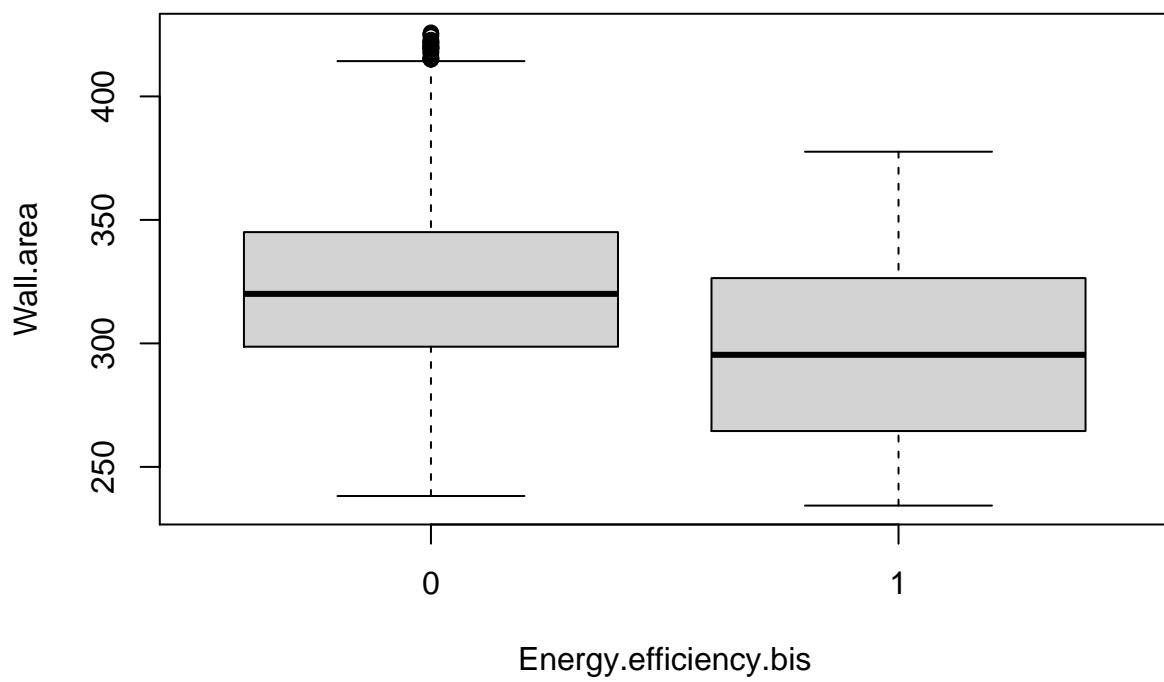
```



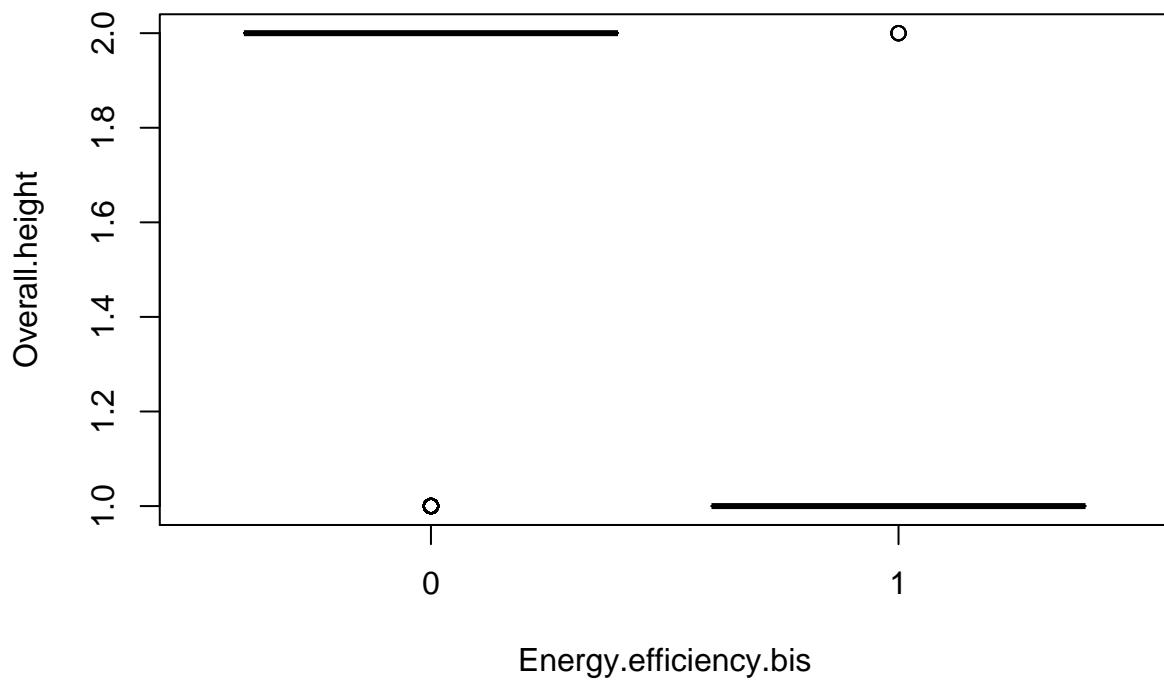
```
boxplot(Surface.area~ Energy.efficiency.bis,data=data.mlg)
```



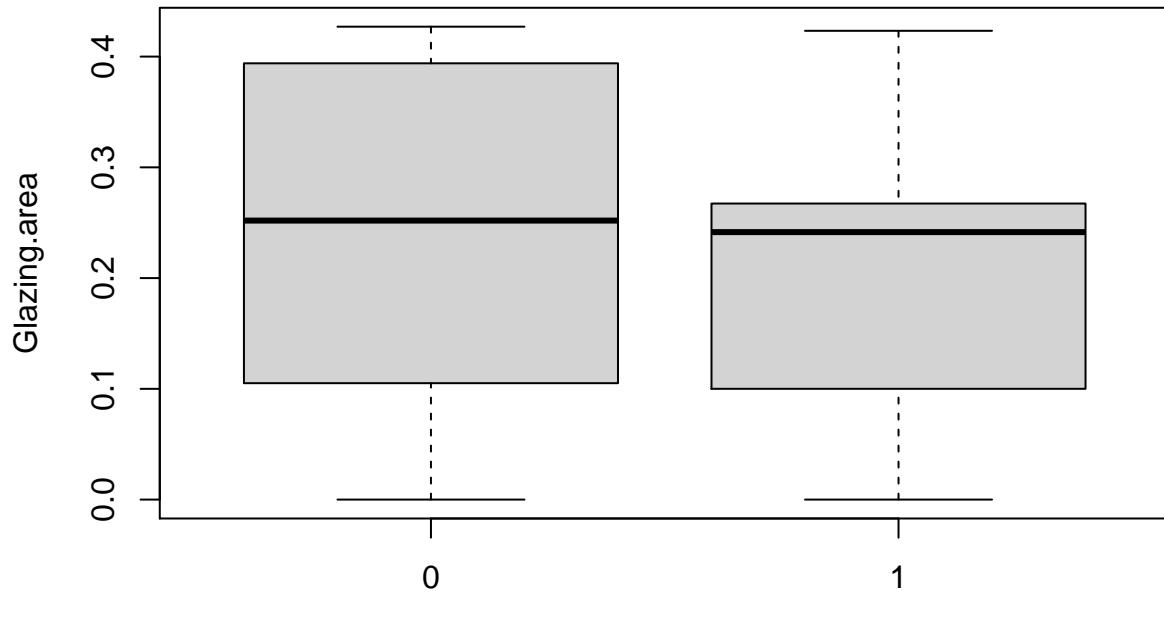
```
boxplot(Wall.area~ Energy.efficiency.bis,data=data.mlg)
```



```
boxplot(Overall.height~ Energy.efficiency.bis,data=data.mlg)
```

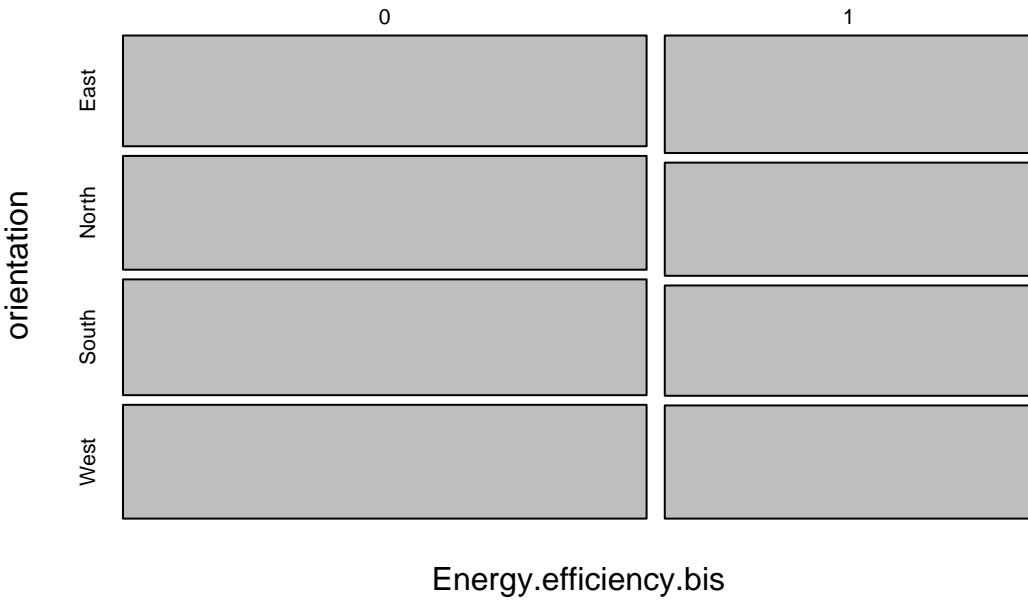


```
boxplot(Glazing.area~ Energy.efficiency.bis,data=data.mlg)
```



```
mosaicplot(table(data.mlg[,c("Energy.efficiency.bis", "orientation"))))

table(data.mlg[, c("Energy.efficiency.bis", "orientation")])
```



L'analyse descriptive du modèle linéaire généralisé par rapport à la variable Energy.efficiency.bis nous permet de noter les points suivant :

- Les individus qui ont une meilleure Energy.efficiency.bis ont une Relative compactness faible.
- Les individus qui ont une meilleure Energy.efficiency.bis ont une Surface.area plus grande.
- Les individus qui ont une meilleure Energy.efficiency.bis ont des surface de Wall.area faible.
- Les batiments ayant une overall height faible ont une meilleur Energy.efficiency.bis

## Modèle linéaire généralisé additif

Puisque, les variables Surface.area, Wall.area et la Roof.area sont liées, nous allons l'enlever du jeu de données. En outre, l'analyse statistique nous a permis d'identifier que toutes les variables avaient des effets sur l'Energy.efficiency.bis donc nous allons considérer le modèle additif ### Ajustement du modèle

```
# Ajustement du modèle linéaire généralisé additif
mlg_rg <- glm(Energy.efficiency.bis ~ ., data=data.mlg, family=binomial(link=logit))
summary(mlg_rg)

##
## Call:
## glm(formula = Energy.efficiency.bis ~ ., family = binomial(link = logit),
##      data = data.mlg)
##
## Deviance Residuals:
##    Min      1Q   Median      3Q     Max
## -2.49777 -0.00005 -0.00001  0.34972  2.07133
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -25.39565  397.93404 -0.064  0.9491
## Relative.compactness 28.47395  15.40742  1.848  0.0646 .
## Surface.area    0.03440   0.02723  1.263  0.2066
## Wall.area       -0.03065   0.01834 -1.671  0.0946 .
## Overall.height.L -15.90933  561.65975 -0.028  0.9774
## orientationNorth -0.20383   0.42163 -0.483  0.6288
## orientationSouth -0.44211   0.41629 -1.062  0.2882
## orientationWest  -0.23270   0.41912 -0.555  0.5788
## Glazing.area     -7.00598   1.35893 -5.156 2.53e-07 ***
## Glazing.area.distr1 -17.93512  794.30110 -0.023  0.9820
## Glazing.area.distr2 -17.87646  794.30110 -0.023  0.9820
## Glazing.area.distr3 -17.68639  794.30110 -0.022  0.9822
## Glazing.area.distr4 -17.53386  794.30110 -0.022  0.9824
## Glazing.area.distr5 -16.78565  794.30109 -0.021  0.9831
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1041.17 on 767 degrees of freedom
## Residual deviance: 299.25 on 754 degrees of freedom
## AIC: 327.25
##
## Number of Fisher Scoring iterations: 19
```

## Calcul du pseudo R2

```
#Calcul de pseudo R2 de mlg_rg
pseudoR2 = 1 - ( mlg_rg$deviance/mlg_rg>null.deviance)
print("Pseudo R2 ")
print(pseudoR2)

## [1] "Pseudo R2 "
## [1] 0.712583
```

Le Pseudo R2 trouvé est de 0.712, alors on peut modéliser ce problème par un MLG. ### Recherche de sous modèle

```

print("##### AIC #####")
mlg.BIC <- bestglm(data.mlg, family = binomial, IC = "BIC")

## Morgan-Tatar search since family is non-gaussian.

## Note: factors present with more than 2 levels.

mlg.BIC$BestModel
print('')
print("##### BIC #####")
mlg.AIC <- bestglm(data.mlg, family = binomial, IC = "AIC")

## Morgan-Tatar search since family is non-gaussian.

## Note: factors present with more than 2 levels.

mlg.AIC$BestModel

## [1] "##### AIC #####
##
## Call: glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
## Coefficients:
##             (Intercept) Relative.compactness     Overall.height.L
##                   -16.562                  21.333                  -8.467
##             Glazing.area
##                   -7.980
##
## Degrees of Freedom: 767 Total (i.e. Null); 764 Residual
## Null Deviance: 1041
## Residual Deviance: 325.7      AIC: 333.7
## [1] ""
## [1] "##### BIC #####
##
## Call: glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
## Coefficients:
##             (Intercept) Relative.compactness     Overall.height.L
##                   -7.553                  22.308                 -19.591
##             Glazing.area  Glazing.area.distr1  Glazing.area.distr2
##                   -6.930                  -18.161                 -18.122
##   Glazing.area.distr3  Glazing.area.distr4  Glazing.area.distr5
##                   -17.934                  -17.823                 -17.048
##
## Degrees of Freedom: 767 Total (i.e. Null); 759 Residual
## Null Deviance: 1041
## Residual Deviance: 303.2      AIC: 321.2

### Choix du sous modèle. ### Comparaison du modèle complet avec le modèle AIC
#
mlg_rg_best_aic <- glm(Energy.efficiency.bis~Relative.compactness + Overall.height+Glazing.area,family=anova(mlg_rg_best_aic,mlg_rg,test="Chisq")

## Analysis of Deviance Table
##
```

```

## Model 1: Energy.efficiency.bis ~ Relative.compactness + Overall.height +
##      Glazing.area
## Model 2: Energy.efficiency.bis ~ Relative.compactness + Surface.area +
##      Wall.area + Overall.height + orientation + Glazing.area +
##      Glazing.area.distr
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      764    325.66
## 2      754  299.25 10   26.407 0.003229 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

La valeur obtenue étant de 0.003229 donc on rejette le modèle AIC au risque de 5%.

```

# On note que le sous modèle bic est un sous modèle du modèle AIC
##### Comparaison du modèle AIC avec le sous modèle BIC
mlg_rg_best_bic <- glm(Energy.efficiency.bis~Relative.compactness + Glazing.area +
                         Overall.height + Glazing.area.distr,family=binomial(link=logit),data=data.mlg)
summary(mlg_rg_best_bic)
anova(mlg_rg_best_bic,mlg_rg,test="Chisq")

```

### Comparaison du modèle complet avec le modèle BIC

```

##
## Call:
## glm(formula = Energy.efficiency.bis ~ Relative.compactness +
##      Glazing.area + Overall.height + Glazing.area.distr, family = binomial(link = logit),
##      data = data.mlg)
##
## Deviance Residuals:
##       Min        1Q        Median         3Q        Max
## -2.37276 -0.00004 -0.00001  0.37367  2.49707
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.553     384.029 -0.020   0.984
## Relative.compactness    22.308      3.859  5.780 7.47e-09 ***
## Glazing.area             -6.930      1.340 -5.171 2.33e-07 ***
## Overall.height.L        -19.591     543.083 -0.036   0.971
## Glazing.area.distr1    -18.161     768.035 -0.024   0.981
## Glazing.area.distr2    -18.122     768.035 -0.024   0.981
## Glazing.area.distr3    -17.934     768.035 -0.023   0.981
## Glazing.area.distr4    -17.823     768.035 -0.023   0.981
## Glazing.area.distr5    -17.048     768.035 -0.022   0.982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1041.17  on 767  degrees of freedom
## Residual deviance: 303.22  on 759  degrees of freedom
## AIC: 321.22
##
## Number of Fisher Scoring iterations: 19
##
```

```

## Analysis of Deviance Table
##
## Model 1: Energy.efficiency.bis ~ Relative.compactness + Glazing.area +
##           Overall.height + Glazing.area.distr
## Model 2: Energy.efficiency.bis ~ Relative.compactness + Surface.area +
##           Wall.area + Overall.height + orientation + Glazing.area +
##           Glazing.area.distr
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      759    303.22
## 2      754    299.25  5    3.9659  0.5543

```

La p-valeur obtenue étant de 0.5543 on retient le modèle BIC au risque 5%

## Modèle avec intéraction

Regardons si le modèle avec intéraction définit mieux le problème de régression.

*#Ajustement du modèle complet avec intéraction*

```
mlg_rg_inter <- glm(Energy.efficiency.bis~Relative.compactness +
```

```
Surface.area + Wall.area + Overall.height + orientation +
```

```
Glazing.area + Glazing.area.distr + Surface.area:Relative.compactness+Wall.area:Glazing.area +Surfa
```

```
summary(mlg_rg_inter)
```

```
##
```

```
## Call:
```

```
## glm(formula = Energy.efficiency.bis ~ Relative.compactness +
##       Surface.area + Wall.area + Overall.height + orientation +
##       Glazing.area + Glazing.area.distr + Surface.area:Relative.compactness +
##       Wall.area:Glazing.area + Surface.area:Wall.area, family = binomial(link = "logit"),
##       data = data.mlg)
```

```
##
```

```
## Deviance Residuals:
```

```
##      Min        1Q     Median        3Q       Max
## -2.455560 -0.000005 -0.000001  0.30123  2.19527
```

```
##
```

```
## Coefficients:
```

	Estimate	Std. Error	z value	Pr(> z )
## (Intercept)	1.153e+02	3.972e+02	0.290	0.77154
## Relative.compactness	-3.415e+01	5.269e+01	-0.648	0.51696
## Surface.area	-1.616e-01	1.118e-01	-1.445	0.14836
## Wall.area	-2.731e-01	1.010e-01	-2.705	0.00684 **
## Overall.height.L	-1.794e+01	5.528e+02	-0.032	0.97411
## orientationNorth	-2.444e-01	4.234e-01	-0.577	0.56375
## orientationSouth	-4.593e-01	4.166e-01	-1.102	0.27033
## orientationWest	-2.629e-01	4.190e-01	-0.628	0.53031
## Glazing.area	-2.440e+01	1.435e+01	-1.700	0.08911 .
## Glazing.area.distr1	-1.794e+01	7.817e+02	-0.023	0.98169
## Glazing.area.distr2	-1.790e+01	7.817e+02	-0.023	0.98173
## Glazing.area.distr3	-1.768e+01	7.817e+02	-0.023	0.98195
## Glazing.area.distr4	-1.757e+01	7.817e+02	-0.022	0.98206
## Glazing.area.distr5	-1.683e+01	7.817e+02	-0.022	0.98283
## Relative.compactness:Surface.area	8.481e-02	8.549e-02	0.992	0.32120
## Wall.area:Glazing.area	5.248e-02	4.285e-02	1.225	0.22061
## Surface.area:Wall.area	3.417e-04	1.435e-04	2.381	0.01725 *
## ---				

```

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1041.17  on 767  degrees of freedom
## Residual deviance: 292.06  on 751  degrees of freedom
## AIC: 326.06
##
## Number of Fisher Scoring iterations: 19

```

### Calcul du pseudo R2

```

#Calcul de pseudo R2
pseudoR2 = 1 - (mlg_rg_inter$deviance/mlg_rg_inter>null.deviance)
print("Pseudo R2 ")
print(pseudoR2)

```

```

## [1] "Pseudo R2 "
## [1] 0.7194924

```

Le pseudo R2 obtenu est de 0.7796646 (contre 0.712583 pour le modèle additif).

### Recherche de sous modèle

```

modelbestinter = step(mlg_rg_inter,trace = FALSE)
summary(modelbestinter)

```

```

##
## Call:
## glm(formula = Energy.efficiency.bis ~ Surface.area + Wall.area +
##     Overall.height + Glazing.area + Glazing.area.distr + Surface.area:Wall.area,
##     family = binomial(link = "logit"), data = data.mlg)
##
## Deviance Residuals:
##      Min        1Q        Median         3Q        Max 
## -2.70377  -0.00005  -0.00001   0.33791   2.17080 
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)    
## (Intercept)          8.736e+01  3.823e+02   0.228  0.81927    
## Surface.area       -9.341e-02  3.785e-02  -2.468  0.01359 *  
## Wall.area          -2.459e-01  9.097e-02  -2.703  0.00687 ** 
## Overall.height.L  -1.617e+01  5.393e+02  -0.030  0.97607    
## Glazing.area      -6.927e+00  1.350e+00  -5.131 2.89e-07 *** 
## Glazing.area.distr1 -1.825e+01  7.627e+02  -0.024  0.98091    
## Glazing.area.distr2 -1.825e+01  7.627e+02  -0.024  0.98091    
## Glazing.area.distr3 -1.804e+01  7.627e+02  -0.024  0.98113    
## Glazing.area.distr4 -1.792e+01  7.627e+02  -0.023  0.98125    
## Glazing.area.distr5 -1.714e+01  7.627e+02  -0.022  0.98207    
## Surface.area:Wall.area 2.948e-04  1.148e-04   2.568  0.01022 * 
## ---                
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
## 
```

```

##      Null deviance: 1041.17  on 767  degrees of freedom
## Residual deviance:  296.45  on 757  degrees of freedom
## AIC: 318.45
##
## Number of Fisher Scoring iterations: 19

```

### Comparaison du modèle complet avec le sous modèle trouvé par la méthode du step

```

mlg_inter_best_aic <- glm(Energy.efficiency.bis ~ Relative.compactness + Surface.area +
                           Wall.area + Overall.height + orientation + Glazing.area +
                           Glazing.area.distr + Surface.area:Wall.area,family=binomial(link=logit),data=data.mlg)
summary(mlg_inter_best_aic)
anova(mlg_inter_best_aic,mlg_rg_inter,test="Chisq")

##
## Call:
## glm(formula = Energy.efficiency.bis ~ Relative.compactness +
##       Surface.area + Wall.area + Overall.height + orientation +
##       Glazing.area + Glazing.area.distr + Surface.area:Wall.area,
##       family = binomial(link = logit), data = data.mlg)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q        Max 
## -2.51473 -0.00005 -0.00001  0.34056  2.22727 
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)    
## (Intercept)             5.509e+01  3.797e+02   0.145   0.8846    
## Relative.compactness    1.544e+01  1.684e+01   0.917   0.3593    
## Surface.area            -6.112e-02 5.222e-02  -1.170   0.2418    
## Wall.area               -2.252e-01 9.399e-02  -2.396   0.0166 *  
## Overall.height.L        -1.590e+01 5.332e+02  -0.030   0.9762    
## orientationNorth        -2.377e-01 4.235e-01  -0.561   0.5746    
## orientationSouth        -4.574e-01 4.171e-01  -1.097   0.2728    
## orientationWest         -2.566e-01 4.194e-01  -0.612   0.5406    
## Glazing.area            -6.938e+00 1.353e+00  -5.128 2.93e-07 *** 
## Glazing.area.distr1    -1.823e+01 7.541e+02  -0.024   0.9807    
## Glazing.area.distr2    -1.820e+01 7.541e+02  -0.024   0.9807    
## Glazing.area.distr3    -1.799e+01 7.541e+02  -0.024   0.9810    
## Glazing.area.distr4    -1.785e+01 7.541e+02  -0.024   0.9811    
## Glazing.area.distr5    -1.709e+01 7.541e+02  -0.023   0.9819    
## Surface.area:Wall.area 2.589e-04 1.219e-04   2.124   0.0336 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1041.17  on 767  degrees of freedom
## Residual deviance:  294.33  on 753  degrees of freedom
## AIC: 324.33
##
## Number of Fisher Scoring iterations: 19
##
## Analysis of Deviance Table

```

```

## 
## Model 1: Energy.efficiency.bis ~ Relative.compactness + Surface.area +
##           Wall.area + Overall.height + orientation + Glazing.area +
##           Glazing.area.distr + Surface.area:Wall.area
## Model 2: Energy.efficiency.bis ~ Relative.compactness + Surface.area +
##           Wall.area + Overall.height + orientation + Glazing.area +
##           Glazing.area.distr + Surface.area:Relative.compactness +
##           Wall.area:Glazing.area + Surface.area:Wall.area
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      753    294.33
## 2      751    292.06  2   2.2716  0.3212

```

La p-valeur étant de 0.9577 on ne rejette donc pas H<sub>0</sub> ON conserve le sous modèle AIC ## Choix du meilleur sous modèle Comparaison du modèle AIC sans interaction avec le modèle AIC avec interaction Dans ce cas précis les modèle sans interaction est un sous modèle de celui avec interaction donc on peut réaliser un test de sous modèle

```
anova(mlg_rg_best_aic, mlg_inter_best_aic, test="Chisq")
```

```

## Analysis of Deviance Table
##
## Model 1: Energy.efficiency.bis ~ Relative.compactness + Overall.height +
##           Glazing.area
## Model 2: Energy.efficiency.bis ~ Relative.compactness + Surface.area +
##           Wall.area + Overall.height + orientation + Glazing.area +
##           Glazing.area.distr + Surface.area:Wall.area
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      764    325.66
## 2      753    294.33 11    31.33 0.0009762 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

La p-valeur étant de 0.006016 on rejette H<sub>0</sub> Donc on conserve mlg\_inter\_best\_aic.

## Regression polytomique pour la variable Energy.efficiency

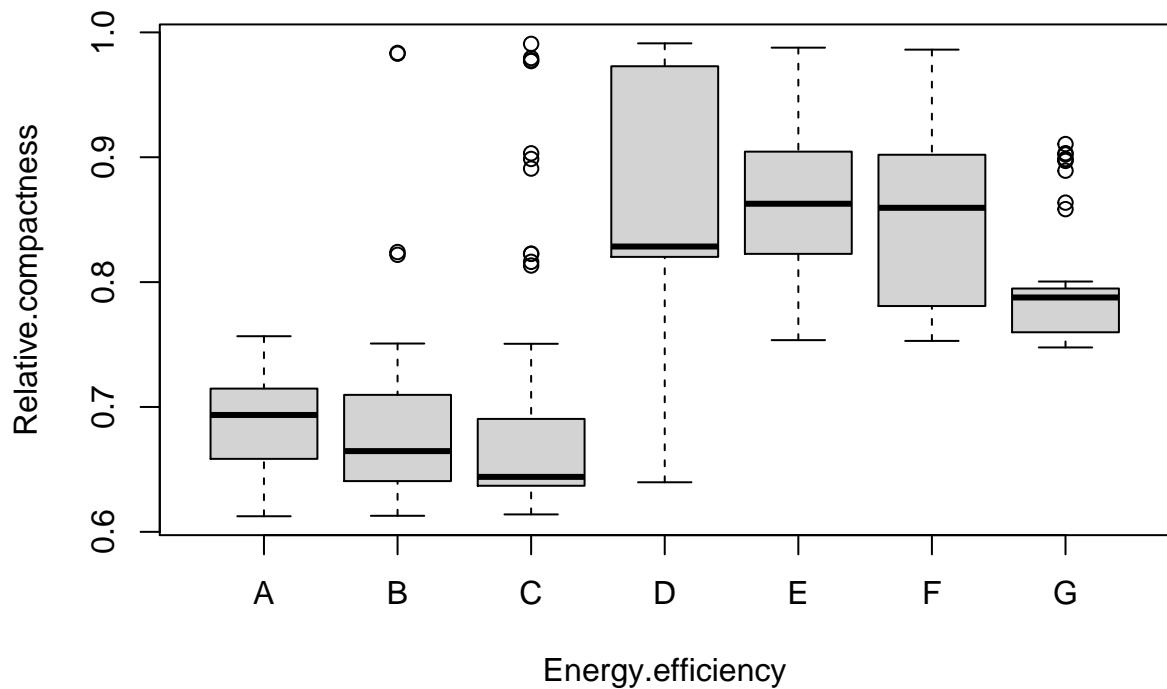
### Mise en forme du jeu de données utilisées

Dans la mise en forme du jeu de données, la variable Energie et Roof.Area ont été supprimées. En raison du fait que, d'une part les variables Surface.area, Wall.area et la Roof.area sont liées, et d'autre part la variable de sortie ici considérée est Energy.efficiency.

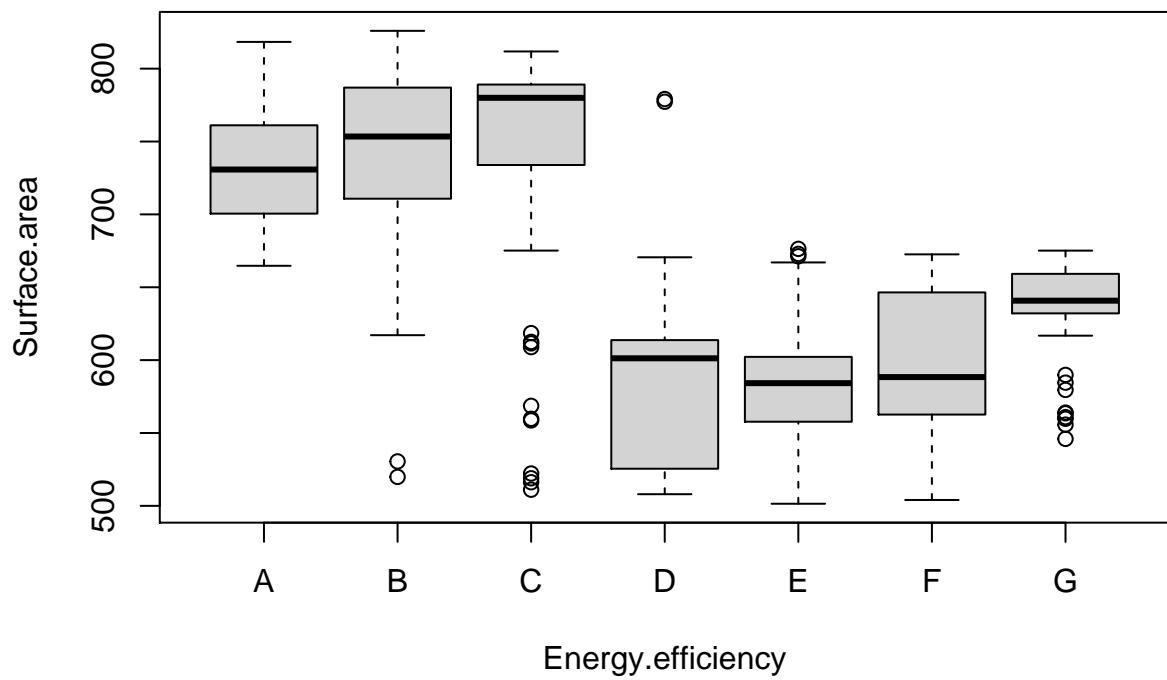
```
data.mlg.p<-new_data[,c(-4,-9)]
```

### Statistique descriptive

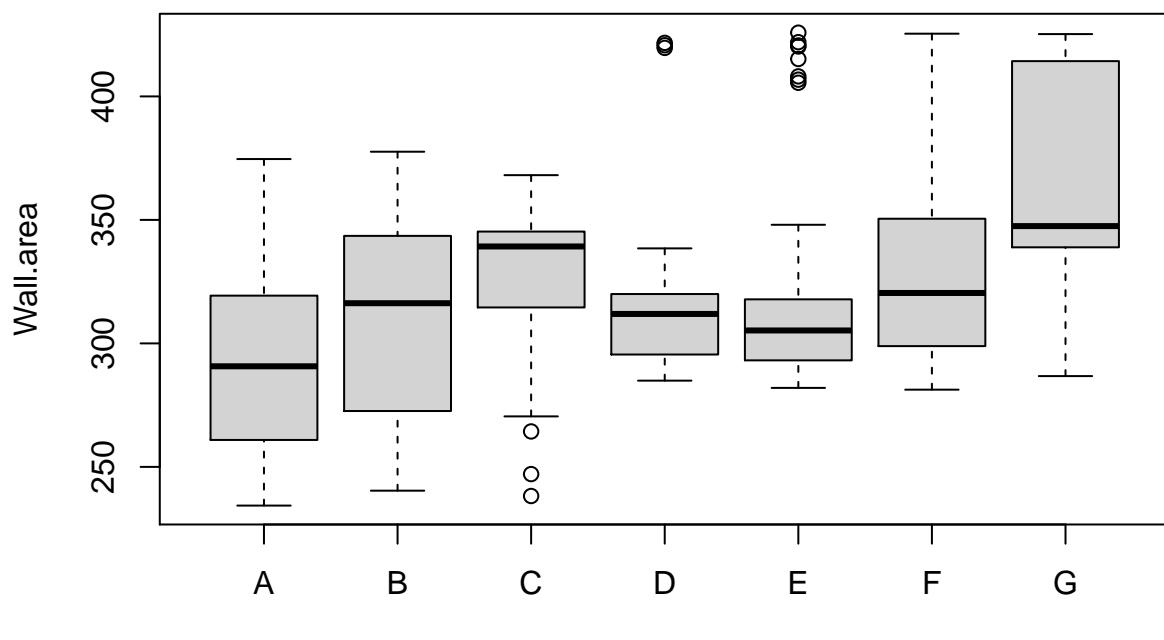
```
#Statistique Descriptive
boxplot(Relative.compactness~ Energy.efficiency, data=data.mlg.p)
```



```
boxplot(Surface.area~ Energy.efficiency,data=data.mlg.p)
```

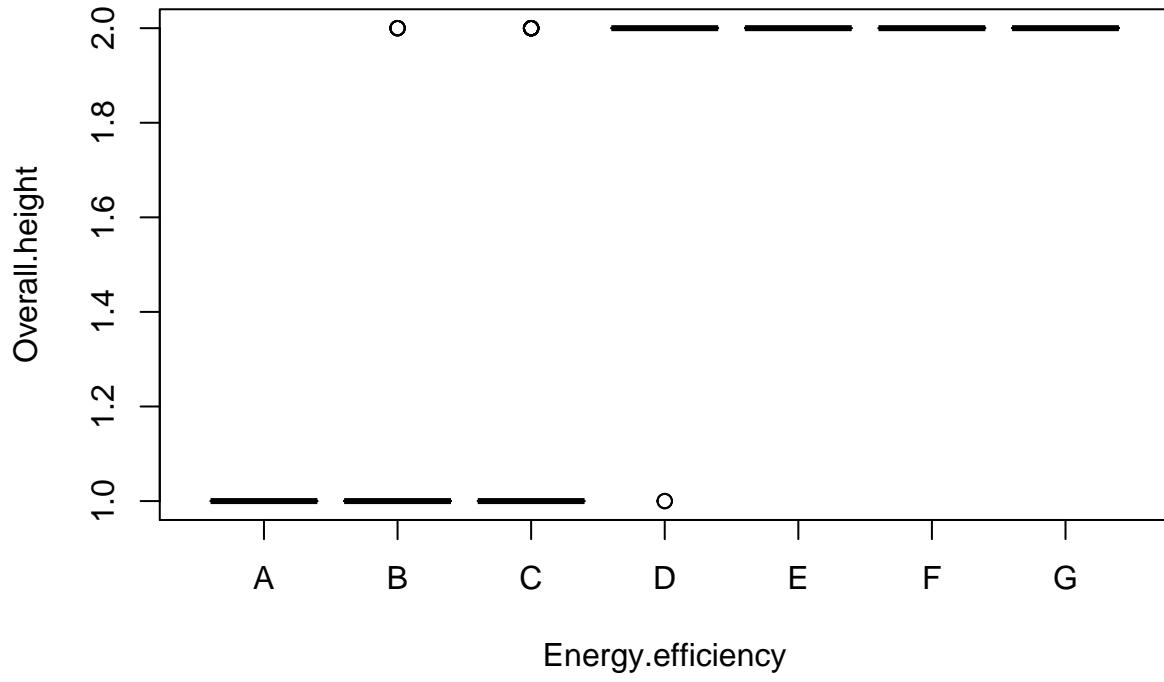


```
boxplot(Wall.area~ Energy.efficiency,data=data.mlg.p)
```



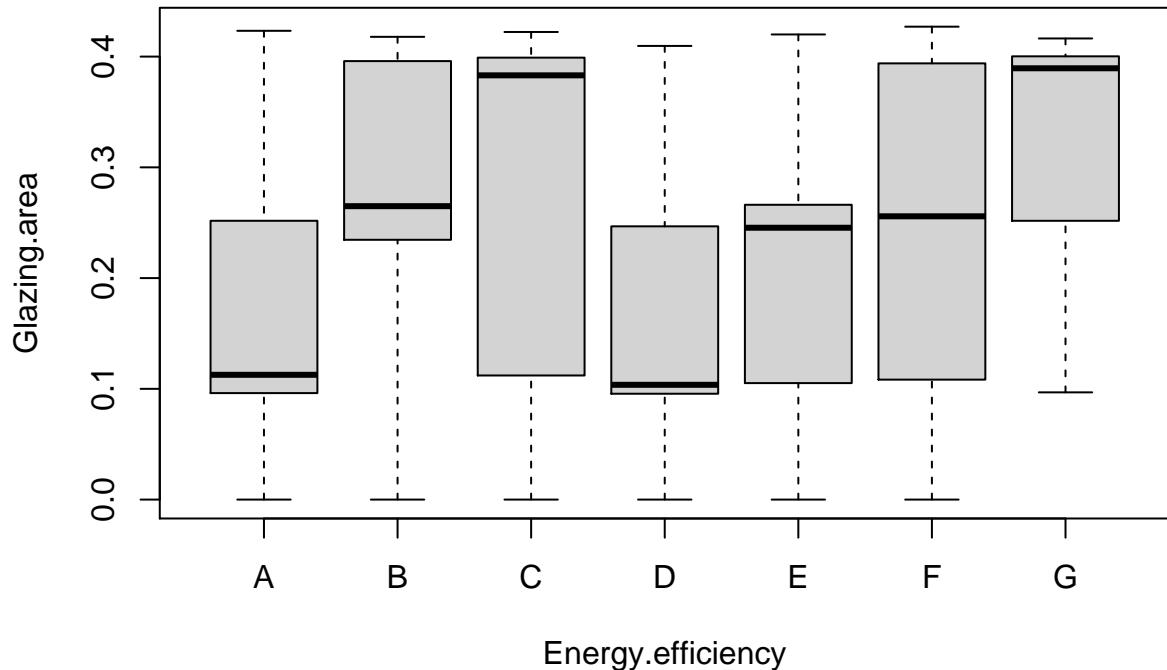
Energy.efficiency

```
boxplot(Overall.height~ Energy.efficiency,data=data.mlg.p)
```

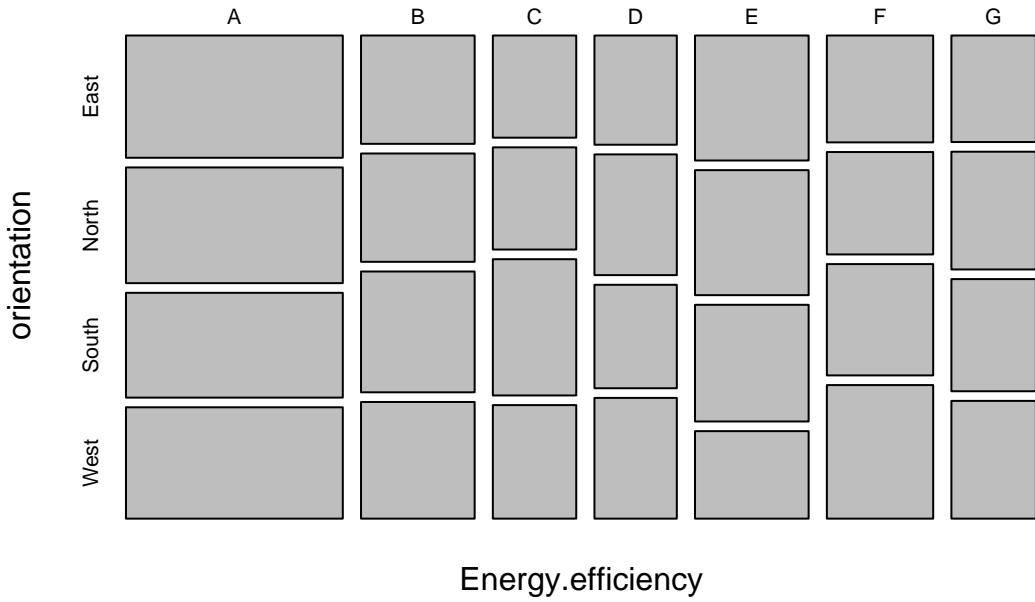


Energy.efficiency

```
boxplot(Glazing.area~ Energy.efficiency,data=data.mlg.p)
```



```
mosaicplot(table(data.mlg.p[,c("Energy.efficiency", "orientation")]))  
  
table(data.mlg.p[, c("Energy.efficiency", "orientation")])
```



L'analyse des figures obtenues nous permet de noter que :

- \* Les classes d'energies A,B,C ont une faible relative compactness par rapport aux classes D,E,F,G
- \* Les classes d'energies A,B,C ont une grande Surface.area par rapport aux classes D,E,F,G
- \* Les classes d'energies B,D,E,F ont en moyenne la même surface de Wall.area, mais pas la même distribution
- La classe A a une faible surface de Wall.area et la classe G en a une grande
- \* Les classes d'energies A,B,C ont une faible Overall.height par rapport aux classes D,E,F,G

## Modèle linéaire généralisé polytomique additif

L'analyse statistique nous a permis d'identifier que toutes les variables avaient des effets sur l'Energy.efficiency donc nous allons considérer le modèle additif. Nous allons considérer les niveaux comme ordonnées pour notre analyse.

```
# transformation en variable ordinaire
data.mlg.p$Energy.efficiency = factor(data.mlg.p$Energy.efficiency, order = TRUE, levels = c("A", "B",
    "C", "D", "E", "F", "G"))
```

### Ajustement du modèle.

```
modelord <- vglm(Energy.efficiency ~ Relative.compactness, data = data.mlg.p, family = acat())

## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, :
## iterations terminated because half-step sizes are very small

## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, : some
## quantities such as z, residuals, SEs may be inaccurate due to convergence at a
## half-step
```

## Non-linear models

### Classification

```
prct_bien_classe=function(table){
    a =(table[1]+table[4])/sum(table)
    return(a)
}

data.nlm.class = data.mlg

# On sépare les données en un ensemble d'apprentissage et un ensemble de test
nb_train <- floor(0.75 * nrow(data.nlm.class))
train_ind <- sample(seq_len(nrow(data.nlm.class)), size = nb_train)

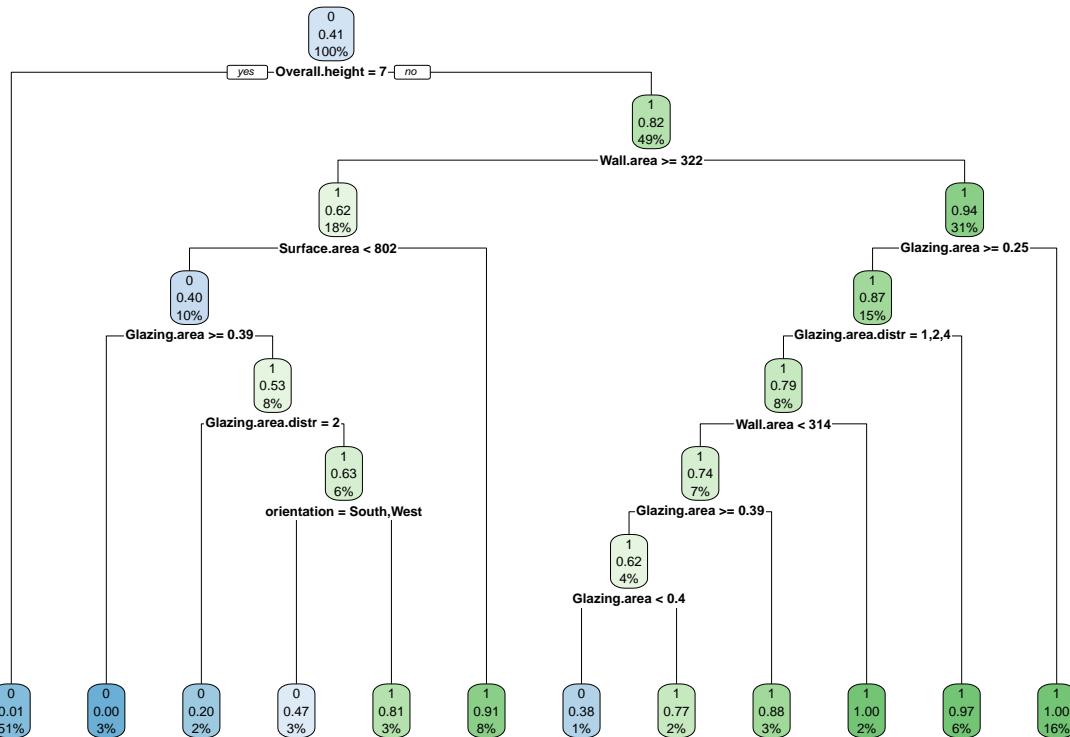
train <- data.nlm.class[train_ind, ]
test <- data.nlm.class[-train_ind, ]

hatY = (mlg_rg$fitted.values > 0.5)
# table de contingences pour la régression logistique
t_reg_log = table(data.mlg[-train_ind,]$Energy.efficiency.bis, hatY[-train_ind])
t_reg_log

##
##      FALSE TRUE
##      0     97   13
##      1      5   77
pct_reg_log = prct_bien_classe(t_reg_log)
pct_reg_log

## [1] 0.90625
library(rpart) # chargement de la librairie
library(rpart.plot)
```

```
tree.dis=rpart(Energy.efficiency.bis~,data=train, parms = list(split = "information"),cp=0.001)
rpart.plot(tree.dis)
```



```

pred.tree <- predict(tree.dis,newdata=test,type="class")
t_class_dis = table(pred.tree, test[, "Energy.efficiency.bis"])
t_class_dis

```

```
##  
## pred.tree 0 1  
##          0 103 14  
##          1    7 68
```

```
pct_class_dis = prct_bien_classe(t_class_dis)  
pct_class_dis
```

```
## [1] 0.890625
```

```
xmat <- xpred.rpart(tree.dis)
```

# Comparaison des valeurs prédictes et observées

```
xerr <- (train$Energy.efficiency.bis != 0) != (xmat > 1.5)
```

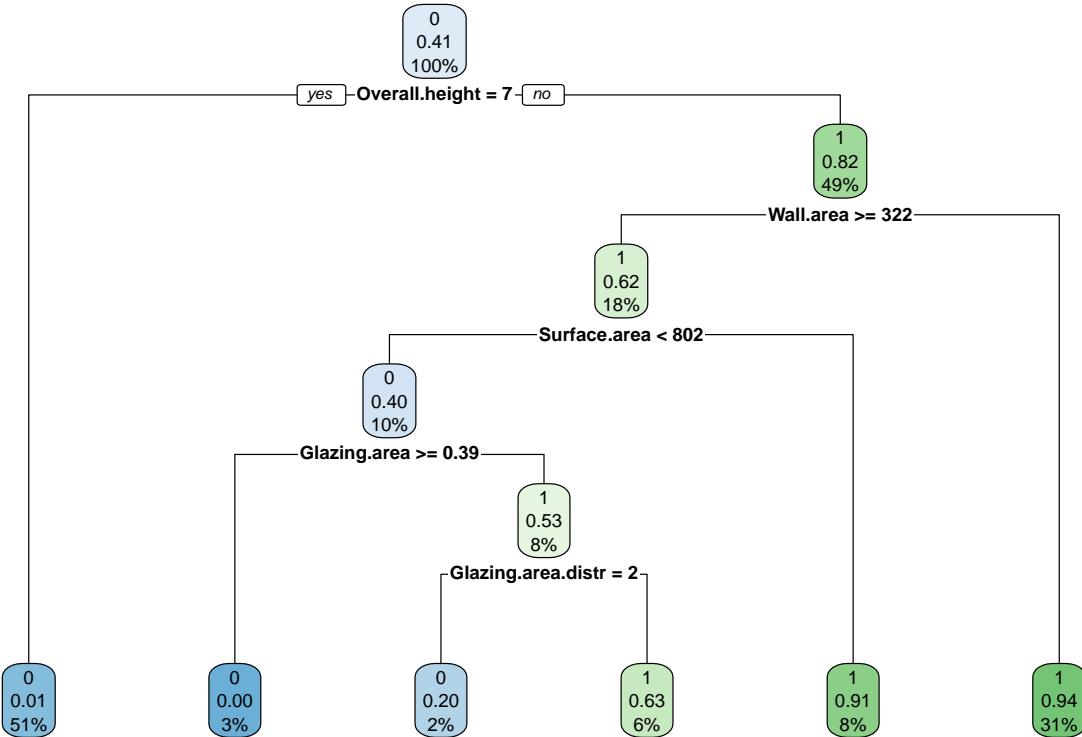
# Calcul des estimations des taux d'erreur

```
CVerr <- apply(xerr, 2, sum)/nrow(xerr)
```

```
cpMin <- as.numeric(attributes(which_min(CVerr))$names)
```

```
tree dis opti <- rpart(Energy_efficiency ~ ., data =
```

```
tree.dis.opti <- rpart(EM  
rpart.plot(tree.dis.opti)
```



```

pred.tree.opti <- predict(tree.dis,newdata=test,type="class")
t_class_dis = table(pred.tree.opti, test[, "Energy.efficiency.bis"])

pct_class_dis = prct_bien_classe(t_class_dis)
pct_class_dis

## [1] 0.890625

library(randomForest)

## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
## The following object is masked from 'package:gridExtra':
##       combine
## The following object is masked from 'package:dplyr':
##       combine
## The following object is masked from 'package:ggplot2':
##       margin

rf.dis <- randomForest(Energy.efficiency.bis ~ ., data = train[,c(-4,-7)],
                        xtest = test[, c(-4,-7,-8)], ytest = test[, "Energy.efficiency.bis"],
                        ntree = 500, do.trace = 50, importance = TRUE)

## ntree      OOB      1      2 |      Test      1      2
## 50: 8.33% 9.09% 7.23% | 6.77% 6.36% 7.32%

```

```

##   100:  7.64%  7.62%  7.66%|  7.29%  6.36%  8.54%
##   150:  7.12%  7.33%  6.81%|  7.81%  7.27%  8.54%
##   200:  7.64%  7.92%  7.23%|  7.81%  7.27%  8.54%
##   250:  7.81%  8.50%  6.81%|  7.81%  7.27%  8.54%
##   300:  7.64%  8.21%  6.81%|  7.81%  7.27%  8.54%
##   350:  7.99%  8.50%  7.23%|  7.81%  7.27%  8.54%
##   400:  7.99%  8.21%  7.66%|  7.81%  7.27%  8.54%
##   450:  7.99%  8.50%  7.23%|  7.81%  7.27%  8.54%
##   500:  7.99%  8.50%  7.23%|  7.81%  7.27%  8.54%

pred.rf <- rf.dis$test$predicted
table(pred.rf, test[, "Energy.efficiency.bis"])

##
## pred.rf    0    1
##      0 102    7
##      1    8  75

library(caret)
cvControl <- trainControl(method = "cv", number = 10)
rfFit <- train(train[, -8], train[, 8], method = "rf", tuneLength = 8,
               trControl = cvControl, trace = FALSE)

rf.dis.opti <- randomForest(Energy.efficiency.bis ~ ., data = train[,c(-4,-7)],
                             xtest = test[, c(-4,-7,-8)], ytest = test[, "Energy.efficiency.bis"],
                             ntree = 500, do.trace = 50, importance = TRUE, mtry=as.integer(rfFit$bestTune))
pred.rf.opti <- rf.dis.opti$test$predicted
t_arbre = table(pred.rf.opti, test[, "Energy.efficiency.bis"])
pct_arbre = prct_bien_classe(t_arbre)
t_arbre

## note: only 6 unique complexity parameters in default grid. Truncating the grid to 6 .
##
## ntree      OOB      1      2|      Test      1      2
##   50:  7.29%  7.62%  6.81%|  8.33%  8.18%  8.54%
##   100:  8.16%  8.21%  8.09%|  8.33%  8.18%  8.54%
##   150:  7.81%  8.50%  6.81%|  8.33%  8.18%  8.54%
##   200:  7.64%  8.50%  6.38%|  8.33%  8.18%  8.54%
##   250:  7.64%  8.50%  6.38%|  7.81%  7.27%  8.54%
##   300:  7.81%  8.80%  6.38%|  7.81%  7.27%  8.54%
##   350:  7.81%  8.80%  6.38%|  7.81%  7.27%  8.54%
##   400:  7.64%  8.21%  6.81%|  8.33%  8.18%  8.54%
##   450:  7.81%  8.21%  7.23%|  8.33%  8.18%  8.54%
##   500:  7.81%  8.21%  7.23%|  8.33%  8.18%  8.54%

##
## pred.rf.opti    0    1
##      0 101    7
##      1    9  75

```

## Régression

Mise en forme des données tests et apprentissage.

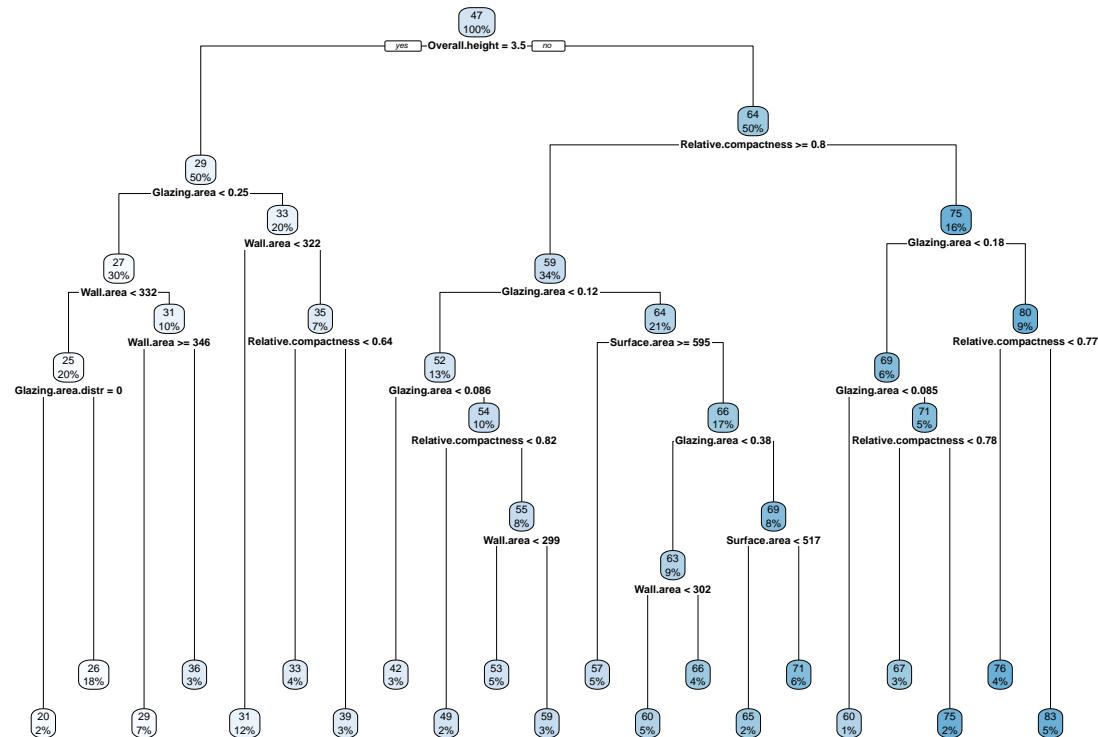
```
data.nlm.reg = new_data[,c(-4,-10)]
```

```
# On sépare les données en un ensemble d'apprentissage et un ensemble de test
nb_train <- floor(0.75 * nrow(data.nlm.reg))
train_ind <- sample(seq_len(nrow(data.nlm.reg)), size = nb_train)

train <- data.nlm.reg[train_ind, ]
test <- data.nlm.reg[-train_ind, ]
```

Estimation de l'arbre de régression.

```
library(rpart) # chargement de la librairie
library(rpart.plot)
tree.reg=rpart(Energy ~ ., data=train, cp=0.001)
rpart.plot(tree.reg)
```



```
plot(tree.reg)
text(tree.reg)
```



Elagage de l'arbre de régression.

```
xmat <- xpred.rpart(tree.reg)
xerr <- (xmat-train[, "Energy"])^2
CVerr <- apply(xerr, 2, sum)
```

Construction de l'arbre avec cp minimisant l'erreur

Pour cela, nous allons rechercher le cp qui minimiser l'erreur.

```
cpMin <- as.numeric(attributes(which.min(CVerr))$names)
tree.reg <- rpart(Energy~, data = train, control = rpart.control(cp = cpMin))
#library(partykit)
#plot(as.party(tree.dis.reg), type = "simple")
```

Construction de l'arbre prédit par régression sur l'ensemble test

```
pred.treer <- predict(tree.reg,newdata=test)
```

Table de contingences

```
t_reg = table(pred.treer > 35, test[, "Energy"] > 35)

pct_reg = prct_bien_classe(t_reg)
pct_reg

## [1] 0.9010417
```