S3 Data Vizualisation

1 Introduction **1**

2 Représentation variable quantitative **1**
2.A Histogramme . . . 1
2.A.1 Avec ajout de courbe de densité . . . 2
2.A.2 Superposition . . . 3
2.B Boxplot . . . 6
2.B.1 Classique . . . 6
2.B.2 Avec jitter . . . 6
2.C Violin plots . . . 7
2.D Bar plot . . . 9
2.D.1 Bar plot ordonnée : variable ordonnée . . . 9
2.E Pie plot (camembert) . . . 11
2.F Alternative au pie plot : Tree map . . . 11
2.G Diagramme cartésien : x - y . . . 12

3 Croisement entre deux variables catégorielles **14**
3.A Treemap à 2 variables catégorielles . . . 14

4 Données répétées : données mesurées au cours du temps **16**
4.A Représentation répétée . . . 16
4.B Diagramme en fagots . . . 17

5 Courbes de survie **20**
5.A Courbe de Kaplan-Meier . . . 20

6 Méthodes multidimensionnelles **22**
6.A Représentation sphérique . . . 22
6.B Heatmap . . . 24
6.C Lasagne plot . . . 25

7 Textométrie : traitement automatique du langage **26**

1 Introduction

Data Vizualisation : nécessaire pour **communication**

2 Représentation variable quantitative

2.A Histogramme

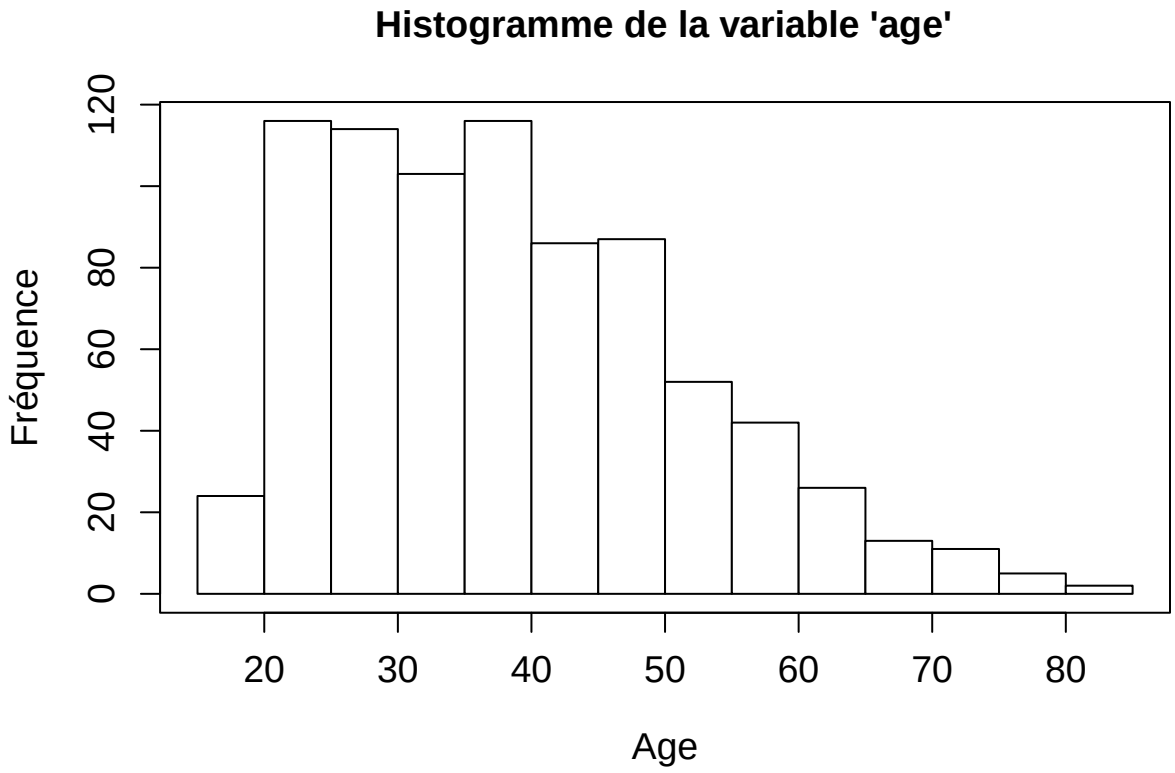
Représentation de la distribution d'une variable **quantitiative**

Syntaxe : fonction `hist()` : `hist(x, xlab="", ylab="", col="", cex.axis=1, cex.lab=1, main="")`

- `x` : variable numérique et `xlab`: étiquette de l'axe des x

- `ylab` : étiquette de l'axe des y
- `col` : couleur des barres
- `cex.axis` : taille des textes des axes
- `cex.lab` : taille des étiquettes des axes
- `main` : titre du graphique
- `box()`: ajoute une bordure autour du graphique

```
hist(smp$age, xlab="Age", ylab="Fréquence", col="white", cex.axis=1.2, cex.lab=1.2,
↪  main="Histogramme de la variable 'age'")
box()
```

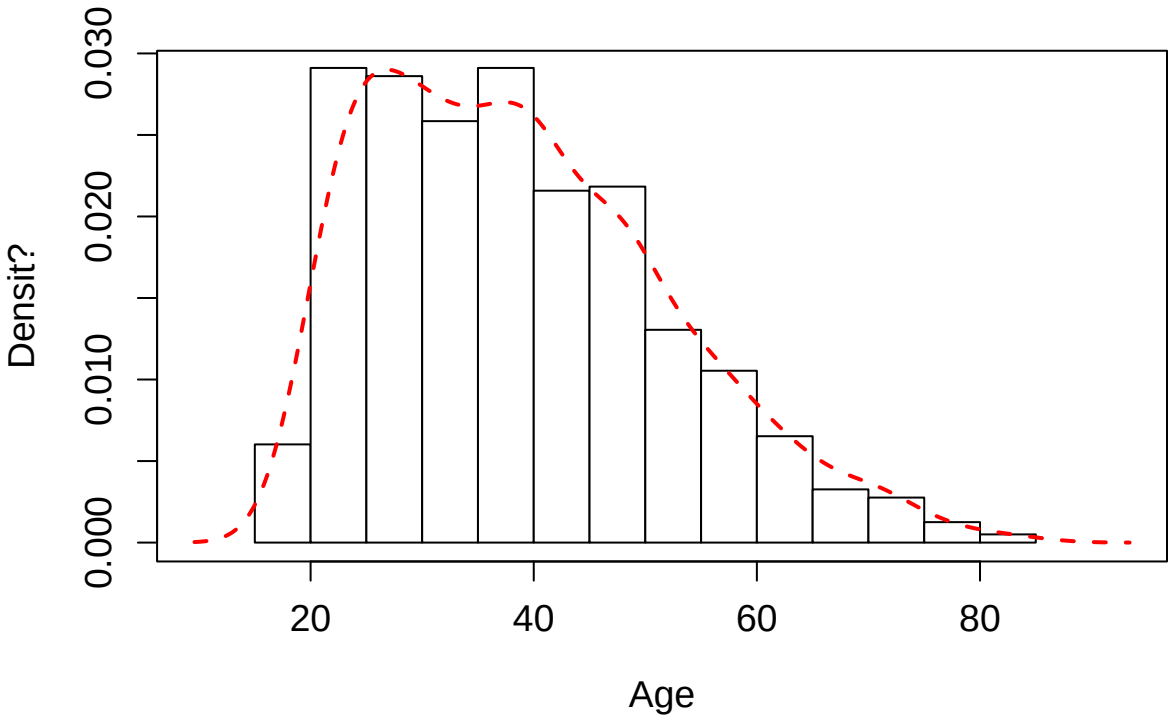


2.A.1 Avec ajout de courbe de densité

Densité = estimation lissée de la distribution des valeurs
Correspond à des **densités de probabilité**.

Pour Falissard : bien pour communiquer avec des gens qui connaissent les statistiques, sinon garder uniquement histogramme

```
dens <- density(smp$age, na.rm=TRUE)
hist(smp$age, xlim=range(dens$x), ylim=range(dens$y), xlab="Age", ylab="Densit?", freq=FALSE,
↪  col="white", cex.axis=1.2, cex.lab=1.2, main="")
lines(dens, col="red", lty=2, lwd=2)
box()
```



ou

```
hist(smp$age, xlab="Age", ylab="Densité", freq=FALSE, col="white", cex.axis=1.2, cex.lab=1.2,
↪  main="")
lines(density(smp$age, na.rm=TRUE), col="red", lty=2, lwd=2)
#calcul direct de la densité dans lines() (plutôt que de passer par un objet `dens`)
box()
```

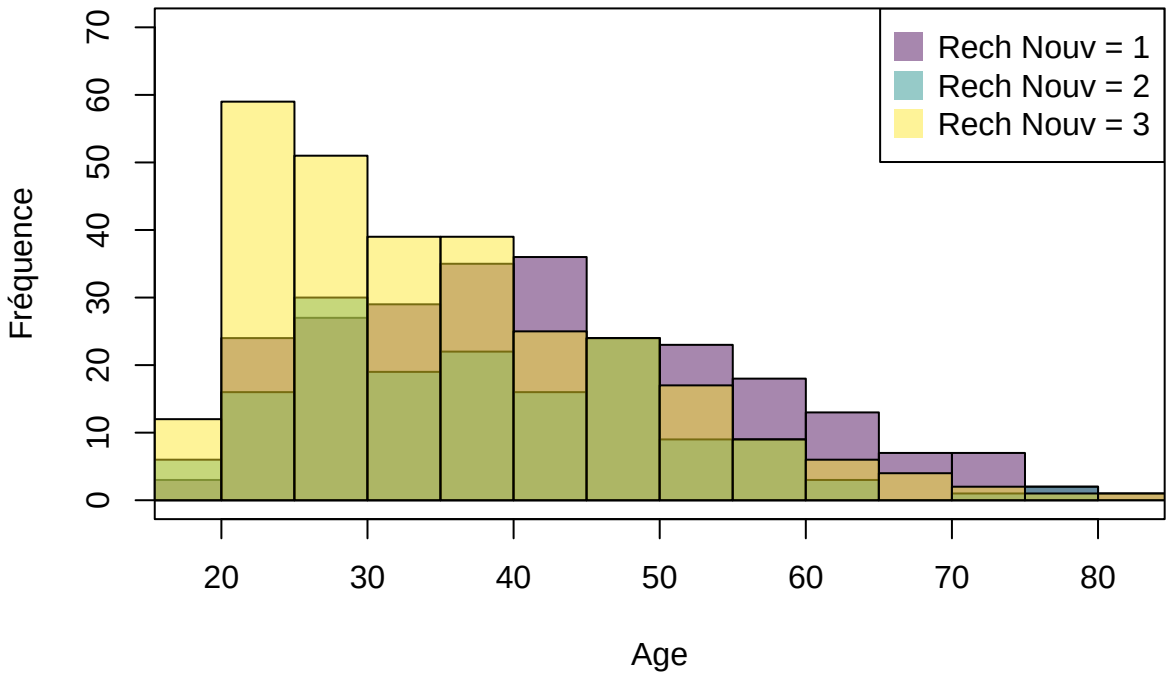
2.A.2 Superposition

En histogramme

Superposition des histogrammes de la variable `age` selon les valeurs de la variable catégorielle ordinale `recherche.nouv` (1, 2 ou 3)

```
vir.3 <- viridis(n = 3) #sert à générer 3 couleurs distinctes

couleurs <- col2rgb(vir.3) # matrice 3x3 (R, G, B) des couleurs

# Génération des couleurs avec transparence alpha
col1 <- rgb(red=couleurs[1,1], green=couleurs[2,1], blue=couleurs[3,1], alpha=120, maxColorValue =
↪  255)
col2 <- rgb(red=couleurs[1,2], green=couleurs[2,2], blue=couleurs[3,2], alpha=120, maxColorValue =
↪  255)
col3 <- rgb(red=couleurs[1,3], green=couleurs[2,3], blue=couleurs[3,3], alpha=120, maxColorValue =
↪  255)

# Premier histogramme
hist(smp$age[smp$recherche.nouv == 1],
     breaks = 15, xlim = c(18, 82), ylim = c(0, 70),
     col = col1, xlab = "Age", ylab = "Fréquence", main = "")

# Ajout des autres histogrammes
hist(smp$age[smp$recherche.nouv == 2],
     breaks = 15, xlim = c(18, 82), ylim = c(0, 70),
     col = col2, add = TRUE)
```

```r
hist(smp$age[smp$recherche.nouv == 3],
     breaks = 15, xlim = c(18, 82), ylim = c(0, 70),
     col = col3, add = TRUE)

# Ajout de la légende
legend("topright",
       legend = c("Rech Nouv = 1", "Rech Nouv = 2", "Rech Nouv = 3"),
       col = c(col1, col2, col3), pt.cex = 2, pch = 15)

box()
```

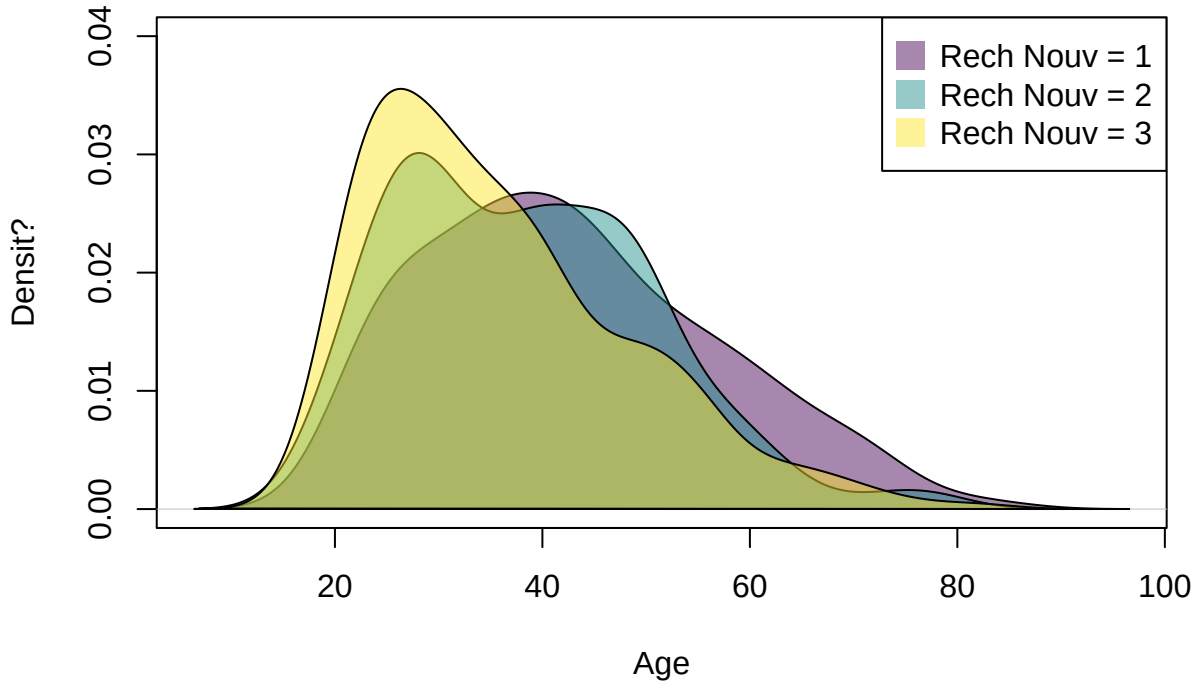


En courbes de densité

Syntaxe :

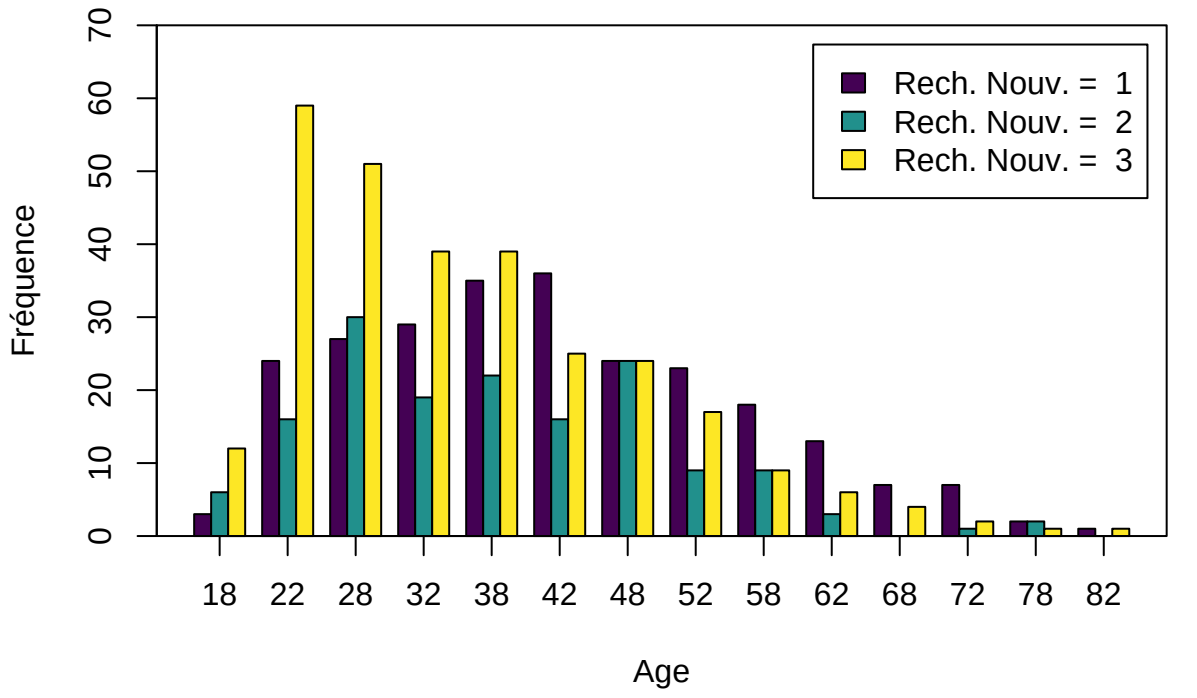
1. Définir les vecteurs de densité avec la fonction `density()`
2. Tracer les courbes avec la fonction `plot()` et `lines()`
3. Ajouter des polygones avec la fonction `polygon()`
    - Nécessaire sinon ça n'affiche rien
4. Ajouter une légende avec la fonction `legend()`

```r
#définition des trois vecteurs pour courbes de desnité
dens1 <- density(smp$age[smp$recherche.nouv==1], na.rm=TRUE)
dens2 <- density(smp$age[smp$recherche.nouv==2], na.rm=TRUE)
dens3 <- density(smp$age[smp$recherche.nouv==3], na.rm=TRUE)

#création des trois courbes de densité
plot(dens1, main="", ylim=c(0,0.04), xlab="Age", ylab="Densit?", type="n")
polygon(dens1, col=col1) #polygon()
lines(dens2, main="", type="n", add=TRUE)
polygon(dens2, col=col2)
lines(dens3, main="", type="n", add=TRUE)
polygon(dens3, col=col3)
legend("topright", legend=c("Rech Nouv = 1","Rech Nouv = 2","Rech Nouv = 3"),
↪  col=c(col1,col2,col3), pt.cex=2, pch=15 )
```



Ou superposition des histogrammes

```
vir_3 <- viridis(n = 3)
x <- list(smp$age[smp$recherche.nouv==1],smp$age[smp$recherche.nouv==2],smp$age[smp$recherche.nouv
 ↪  ==3])
multhist(x, ylab="Fréquence", xlab="Age", axis.lty=1, ylim=c(0,70), col=vir_3,
 ↪  legend.text=paste("Rech. Nouv. = ",1:3))
box()
```



2.B Boxplot

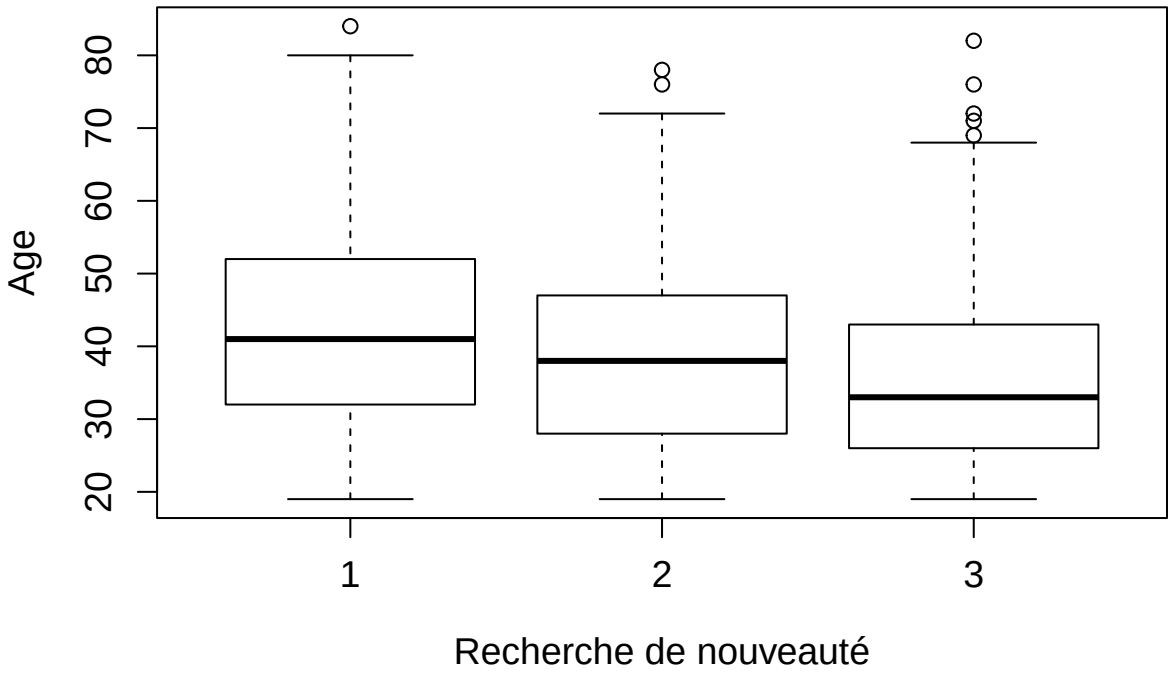
2.B.1 Classique

Représentation de la distribution d'une variable **quantitiative** selon les modalités d'une variable **catégorielle** ordinale ou nominale

Syntaxe : fonction `boxplot()` : `boxplot(y~x, data=, ylab="", xlab="", col="", cex.axis=1, cex.lab=1, main="")`

- `y~x` : variable numérique en fonction de la variable catégorielle
- `data=` : data.frame contenant les variables
- `ylab` : étiquette de l'axe des y
- `xlab` : étiquette de l'axe des x
- `col` : couleur des boîtes
- `cex.axis` : taille des textes des axes
- `cex.lab` : taille des étiquettes des axes

```
boxplot(age~recherche.nouv, data=smp, ylab="Age", xlab="Recherche de nouveauté", cex.axis=1.2,
↪  cex.lab=1.2, col="white", main="")
```



2.B.2 Avec jitter

Ajout des points de données individuels avec la fonction `jitter()`
Problème des boxplots : points superposés **simplifient** la représentation graphique

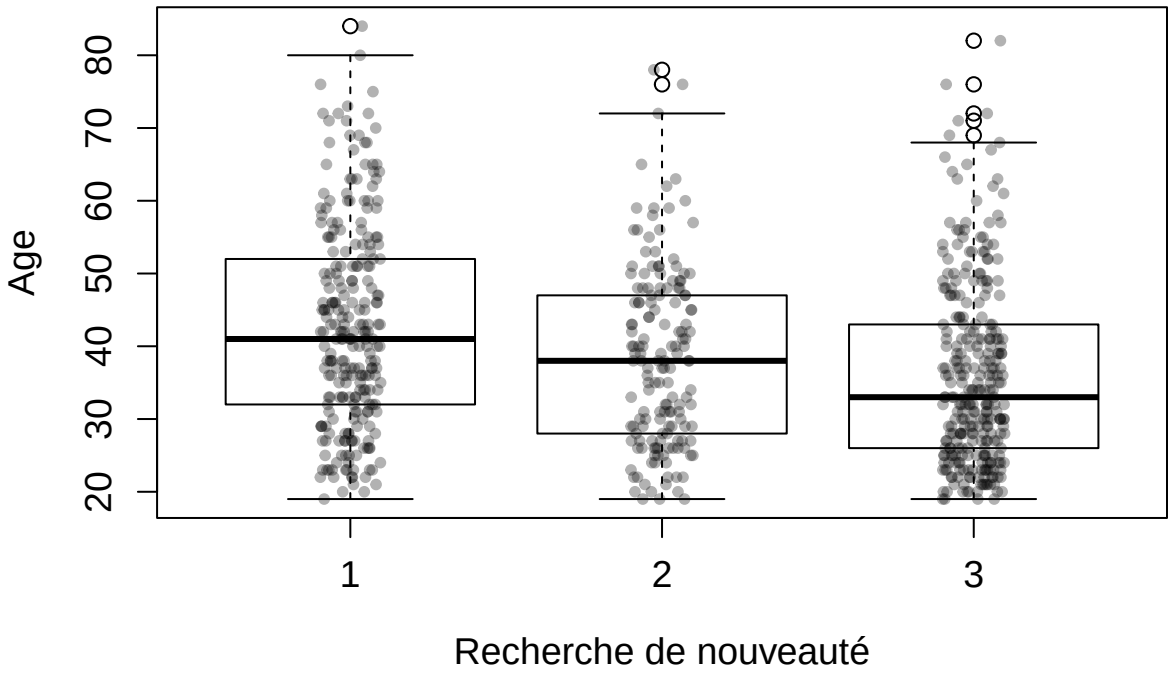
Syntaxe :

1. Boxplot classique
2. Ajout des points individuels avec jitter
   - `jitter(rep(x, n), amount=)` : génère des points autour de la valeur x, n fois, avec une dispersion définie par amount
   - `points(xjit, yvalues, pch=20, col=rgb(0,0,0,.3))` : ajoute les points jitter aux coordon-

nées xjit et yvalues, avec une transparence alpha de 0.3

- Répéter pour chaque modalité de la variable catégorielle

```
#boxplot classique
boxplot(age~recherche.nouv, data=smp, ylab="Age", xlab="Recherche de nouveauté", cex.axis=1.2,
↪  cex.lab=1.2, col="white", main="")
#ajout des points individuels avec jitter
xjit1 <- jitter(rep(1, table(smp$recherche.nouv)[1]),amount=0.1)
points(xjit1, na.omit(smp$age[smp$recherche.nouv==1]), pch=20, col=rgb(0,0,0,.3))
xjit2 <- jitter(rep(2, table(smp$recherche.nouv)[2]),amount=0.1)
points(xjit2, na.omit(smp$age[smp$recherche.nouv==2]), pch=20, col=rgb(0,0,0,.3))
xjit3 <- jitter(rep(3, table(smp$recherche.nouv)[3]),amount=0.1)
points(xjit3, na.omit(smp$age[smp$recherche.nouv==3]), pch=20, col=rgb(0,0,0,.3))
```



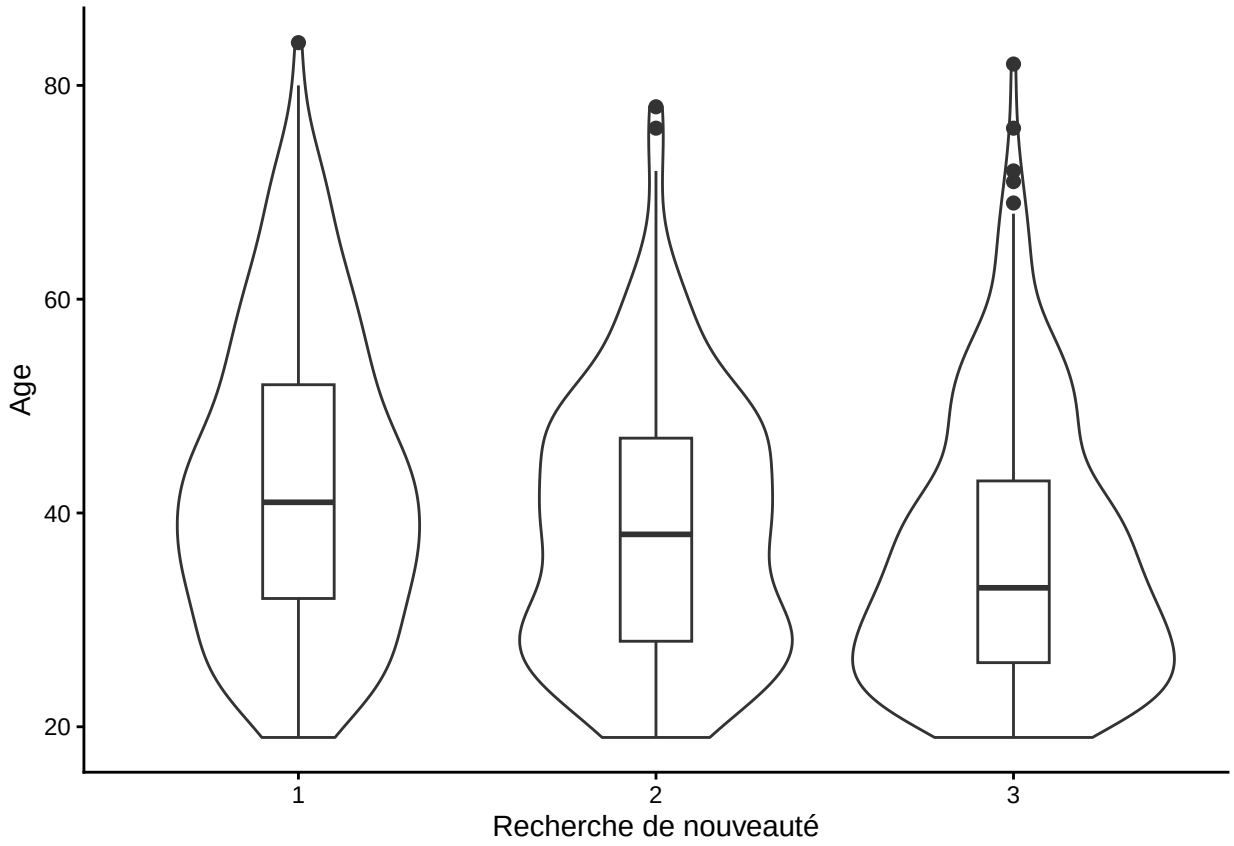
2.C Violin plots

Représentation de la distribution d'une variable **quantitiative** selon les modalités d'une variable **catégorielle** ordinale ou nominale, avec une estimation de la densité
Bruno Falissard n'est pas un grand fan.

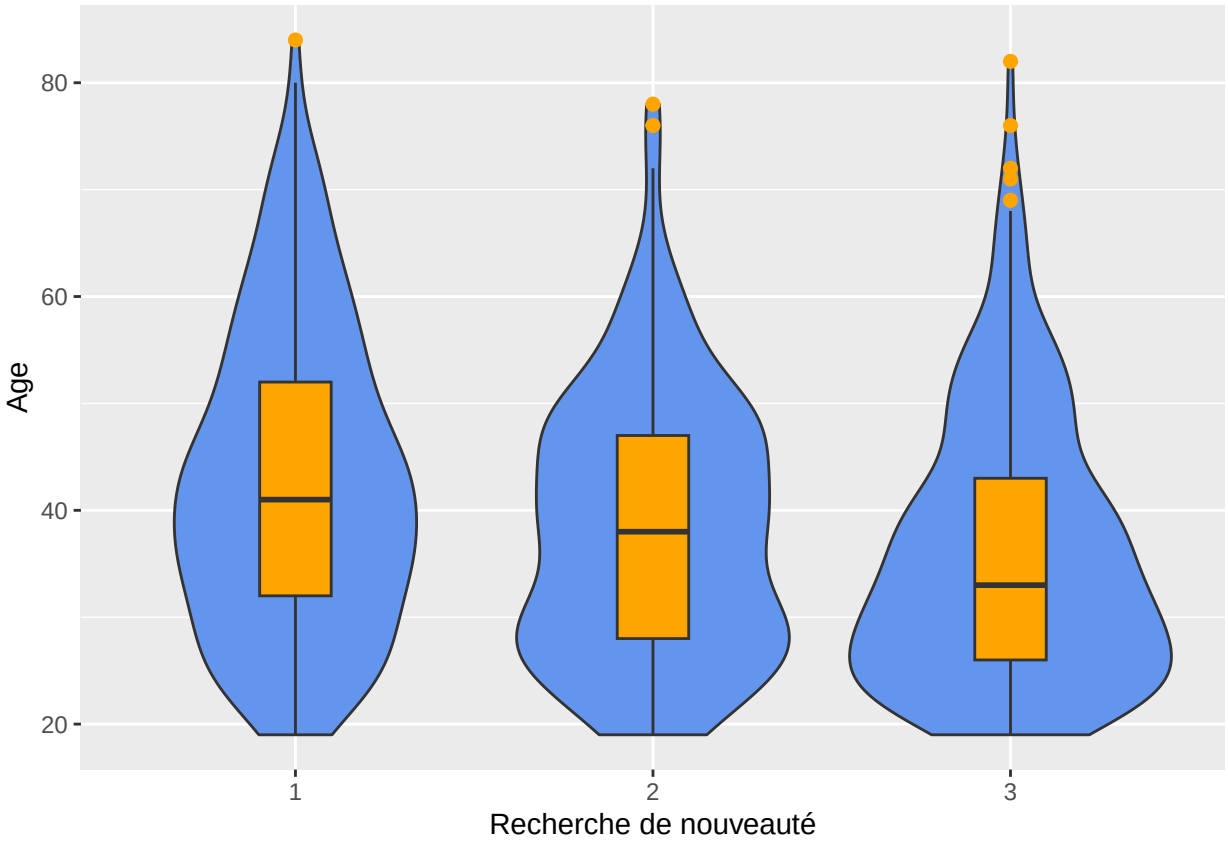
Syntaxe : utilisation du package `ggplot2` avec les fonctions `geom_violin()` et `geom_boxplot()`

1. Définir l'objet ggplot avec les données et les esthétiques
2. Ajouter la géométrie du violin plot avec `geom_violin()`
3. Ajouter la géométrie du boxplot avec `geom_boxplot()`

```
#1ère étape:
p <- ggplot(na.omit(smp[,c("recherche.nouv","age")]), aes(factor(recherche.nouv), age))
p <- p + geom_violin()
p <- p + geom_boxplot(width = .2, outlier.size = 2)
p <- p + theme_classic() + labs(y = "Age", x = "Recherche de nouveauté")
p
```




```
#2e étape : ajout de couleurs
p <- ggplot(na.omit(smp[,c("recherche.nouv","age")]), aes(factor(recherche.nouv), age))
p <- p + geom_violin(fill = "cornflowerblue")
p <- p + geom_boxplot(width = .2, outlier.size = 2, fill = "orange", outlier.color = "orange")
p <- p + labs(y = "Age", x = "Recherche de nouveauté")
p
```



2.D Bar plot

2.D.1 Bar plot ordonnée : variable ordonnée

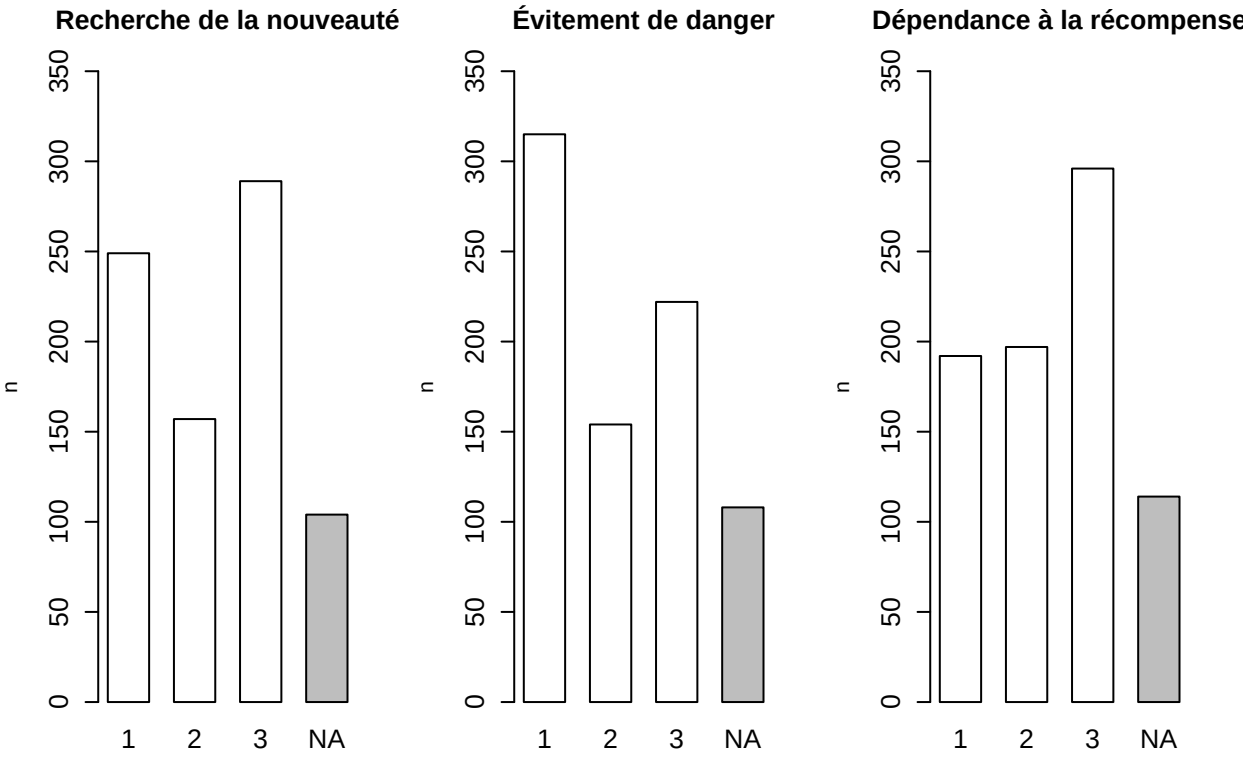
Représentation de la distribution d'une variable **catégorielle** nominale ou ordinale

Syntaxe : fonction `barplot()` : `barplot(height, names.arg="", xlab="", ylab="", col="", cex.axis=1, cex.lab=1, main="")`

- `height` : vecteur des hauteurs des barres (fréquences ou pourcentages)
- `names.arg` : étiquette des barres

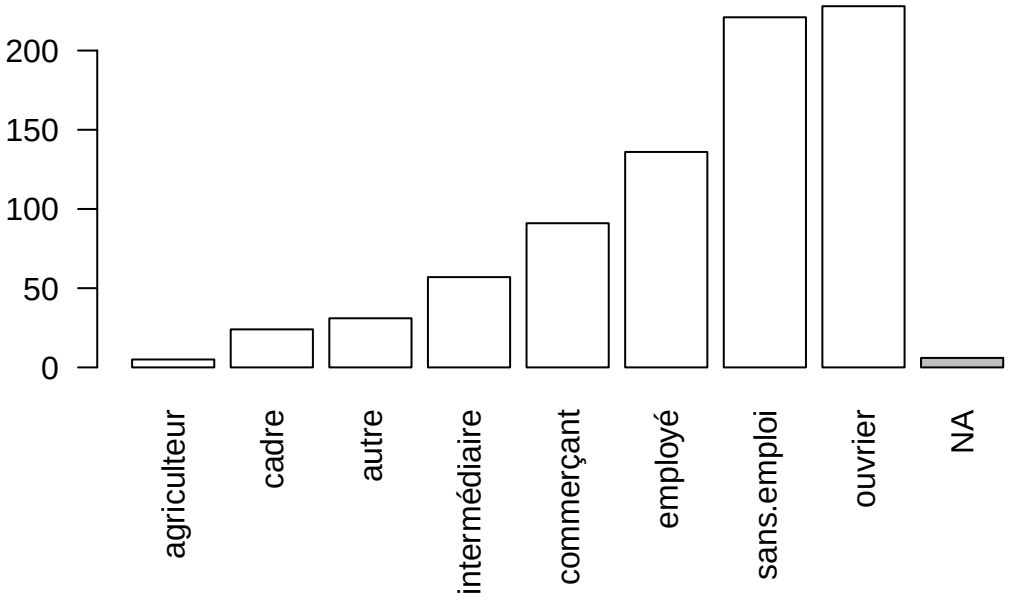
NB : `par()` sert à

```
par(mfrow=c(1,3)) #sert à afficher 3 graphiques sur la même ligne
barplot(table(smp$recherche.nouv,useNA="ifany"), main="Recherche de la nouveauté", ylab="n",
 ↪  ylim=c(0,350), space=0.6, names.arg=c("1","2","3","NA"), cex.axis=1.2, cex.names=1.2,
 ↪  col=c("white","white","white","grey"))
barplot(table(smp$evit.danger,useNA="ifany"), main="Évitement de danger", ylab="n", ylim=c(0,350),
 ↪  space=0.6, names.arg=c("1","2","3","NA"), cex.axis=1.2, cex.names=1.2,
 ↪  col=c("white","white","white","grey"))
barplot(table(smp$dep.recompense,useNA="ifany"), main="Dépendance à la récompense", ylim=c(0,350),
 ↪  ylab="n", names.arg=c("1","2","3","NA"), space=0.6, cex.axis=1.2, cex.names=1.2,
 ↪  col=c("white","white","white","grey"))
```



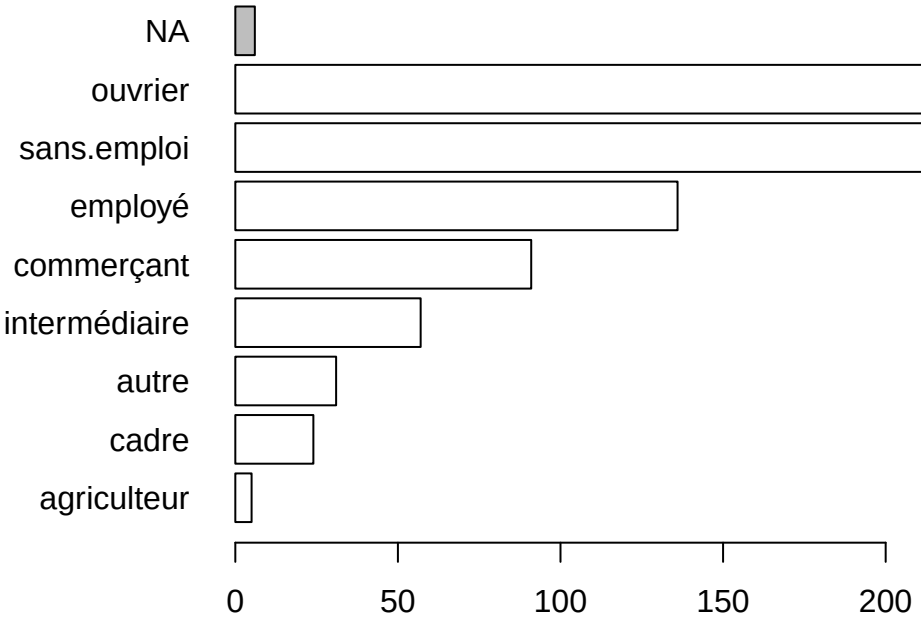
Variable non ordonnée :

```
dt <- data.frame(table(smp$profession, useNA="ifany"))
dt$Var1 <- as.character(dt$Var1)
dt.tri <- dt[order(dt$Freq[1:(nrow(dt)-1)]),]
dt.tri[9,2] <- sum(is.na(smp$profession))
dt.tri[9,1] <- "NA"
par(mar=c(9,4,4,4))
barplot(dt.tri$Freq, las=2, names.arg=dt.tri$Var1, col=c(rep("white",8),"grey"))
```



Représentation horizontale :

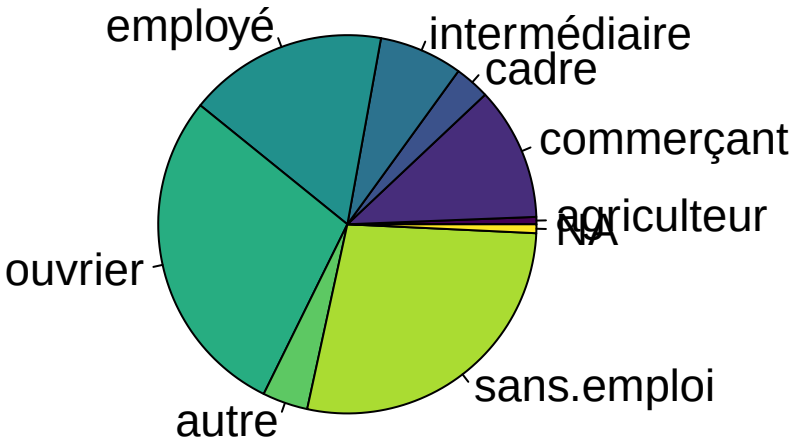
```r
par(mar=c(4,9,4,4))
barplot(dt.tri$Freq, las=1, names.arg=dt.tri$Var1, col=c(rep("white",8),"grey"),horiz=TRUE)
```



2.E Pie plot (camembert)

Les puristes n'aiment pas trop les camemberts car difficile de comparer entre 2 catégories sur l'ensemble. Mais l'avantage du camembert : permet de voir la proportion d'une catégorie sur l'ensemble (sur les 100%)

```r
vir_9 <- viridis(n = 9)
dt$Var1[9] <- "NA"
pie(dt$Freq,col = vir_9,labels=dt$Var1,cex=1.4)
```

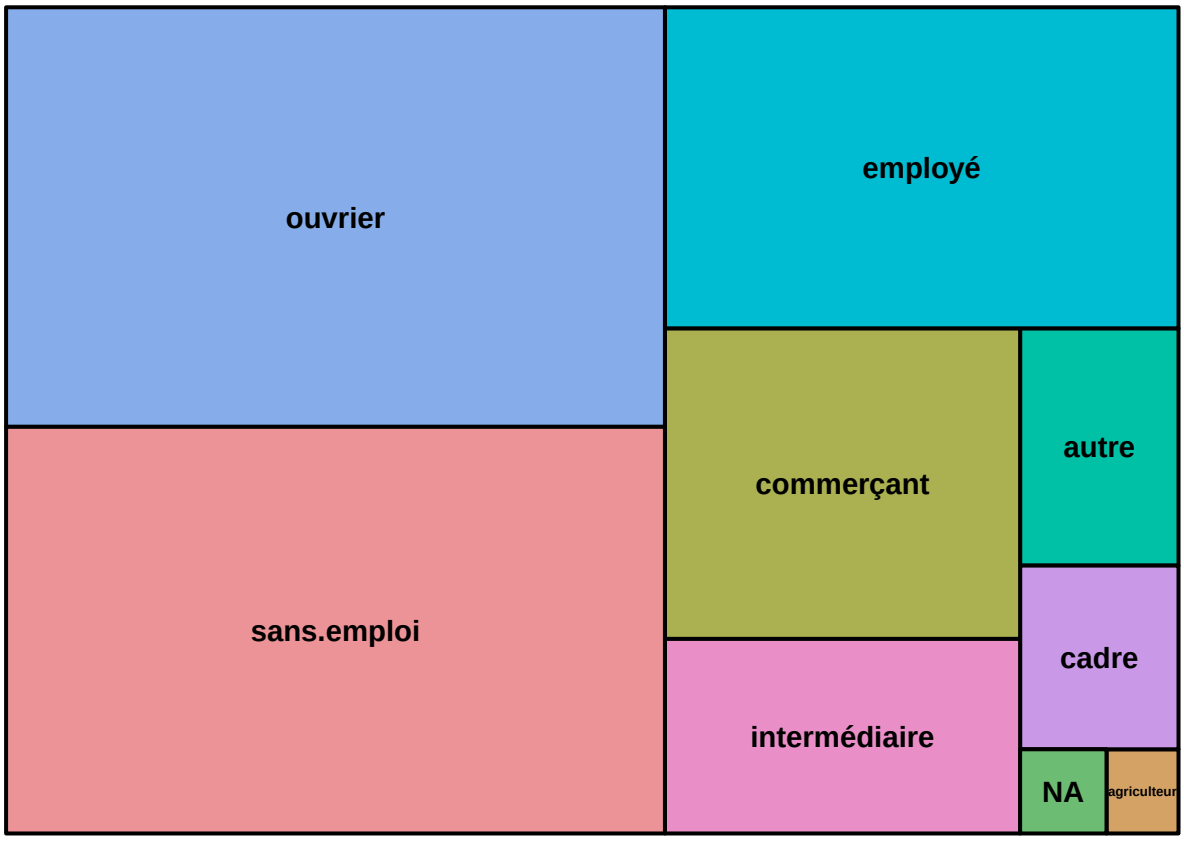


2.F Alternative au pie plot : Tree map

À la mode !

Représenter en largeur les effectifs, + sur l'effectif global

```
treemap(dt, index="Var1", vSize="Freq", type="index", title="")
```

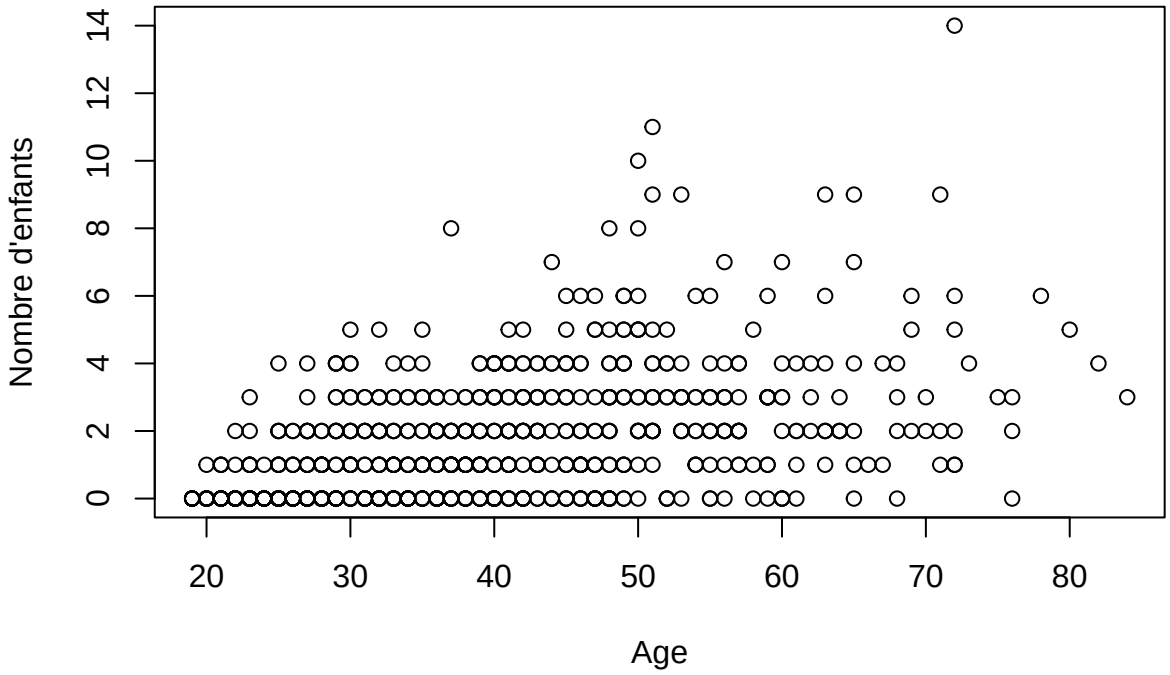


2.G Diagramme cartésien : x - y

- On représente ici le nombre d'enfants en fonction de l'âge.
- Le pb est que le nombre d'enfants est discret : donc superposition des points

Points superposés :

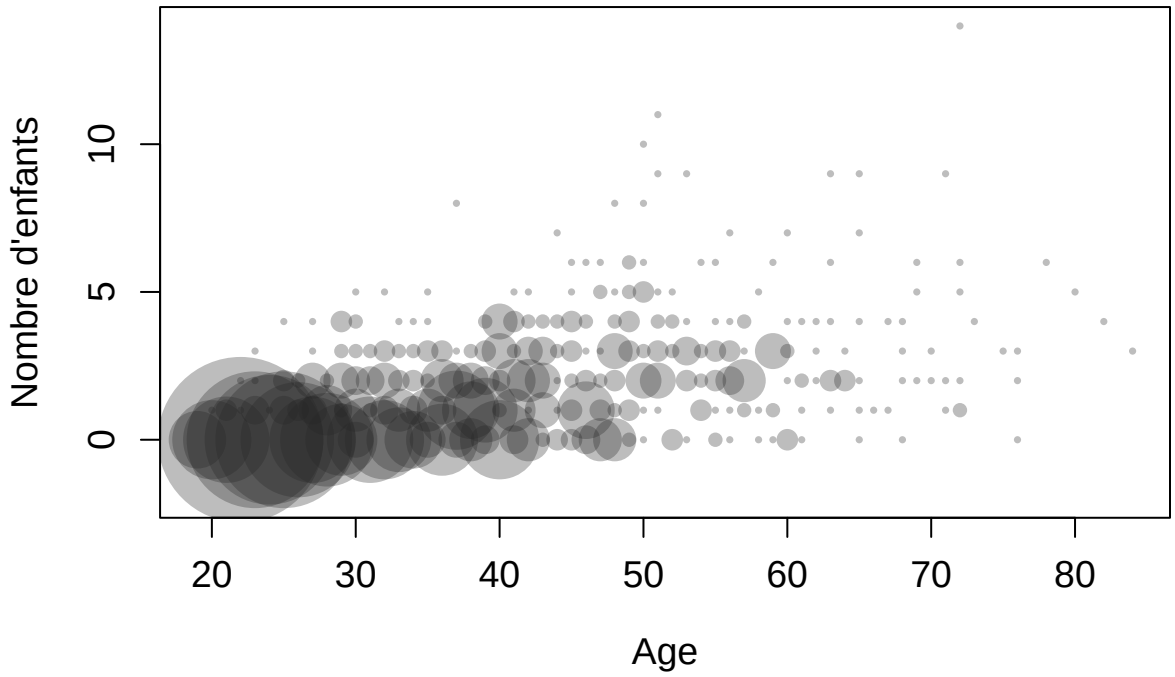
```
plot(smp$age,smp$nb.enfants, xlab="Age", ylab="Nombre d'enfants", main="")
```



1ère option : taille des points

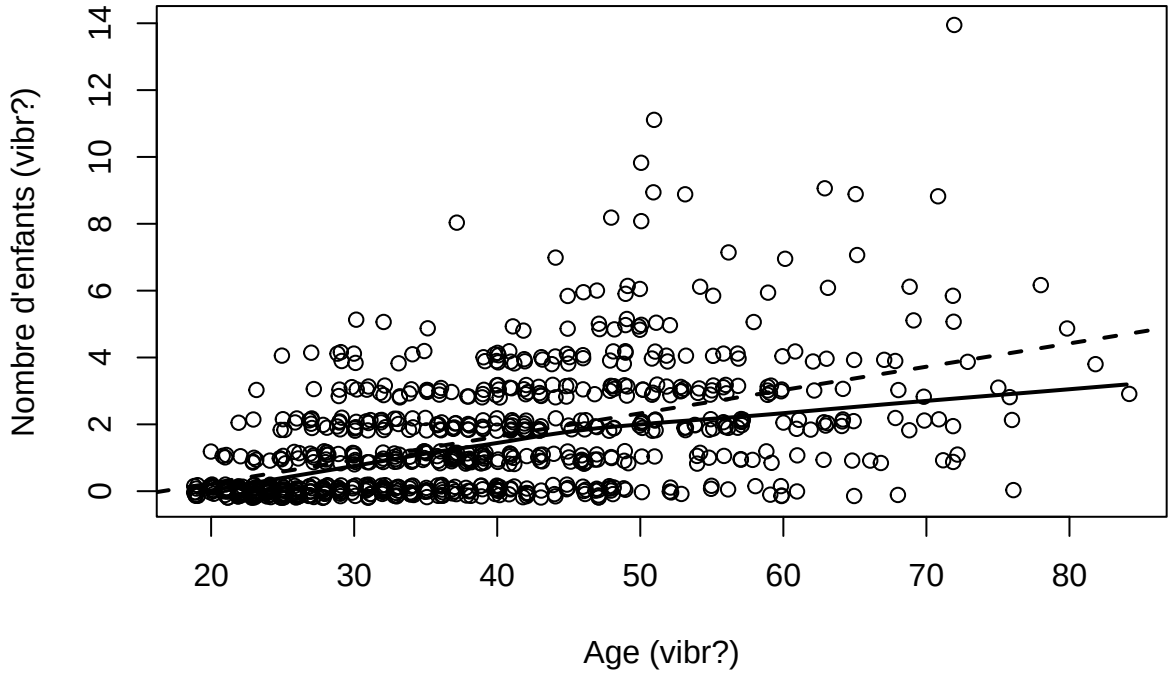
C'est la classe selon Bruno Falissard

```
xy <- xyTable(smp$age,smp$nb.enfants)
coeff.reduc <- 0.5

plot(xy$x , xy$y , cex=xy$number*coeff.reduc , pch=16 , col=rgb(0.15,0.15,0.15,0.3) ,
 ↪  ylim=c(-2,14), cex.axis=1.2, cex.lab=1.2, xlab= "Age" , ylab="Nombre d'enfants")
```



2ème option : `jitter()`

```r
plot(jitter(smp$age),jitter(smp$nb.enfants), xlab="Age (vibr?)", ylab="Nombre d'enfants (vibr?)",
↪  main="")
abline(lm(nb.enfants ~ age, data=smp), lwd=2, lty=2)
nona <- na.omit(smp[,c("nb.enfants","age")])
lines(lowess(nona$age,nona$nb.enfants), lwd=2, cex.axis=1.2, cex.lab=1.2)
```



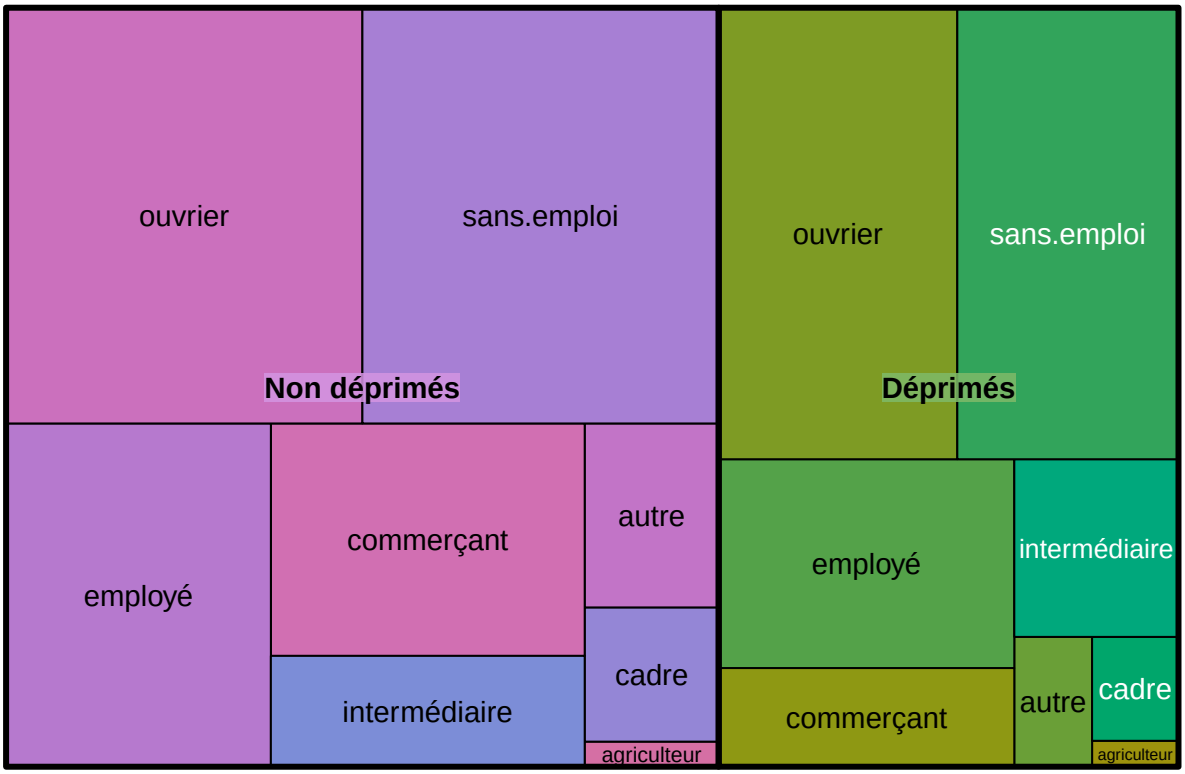
3 Croisement entre deux variables catégorielles

3.A Treemap à 2 variables catégorielles

Ici : croiser déprimés OUI/NON avec catégories socio-professionnelles.

```r
library(treemap)
tbl <- table(smp$profession,smp$depression)
groupe <- c(rep("Déprimés",8),rep("Non déprimés",8))
sousgroupe <- rep(rownames(tbl),2)
valeurs <- c(tbl[,2],tbl[,1])
dt <- data.frame(groupe,sousgroupe,valeurs)
treemap(dt, index=c("groupe","sousgroupe"), vSize="valeurs",type="index")
```

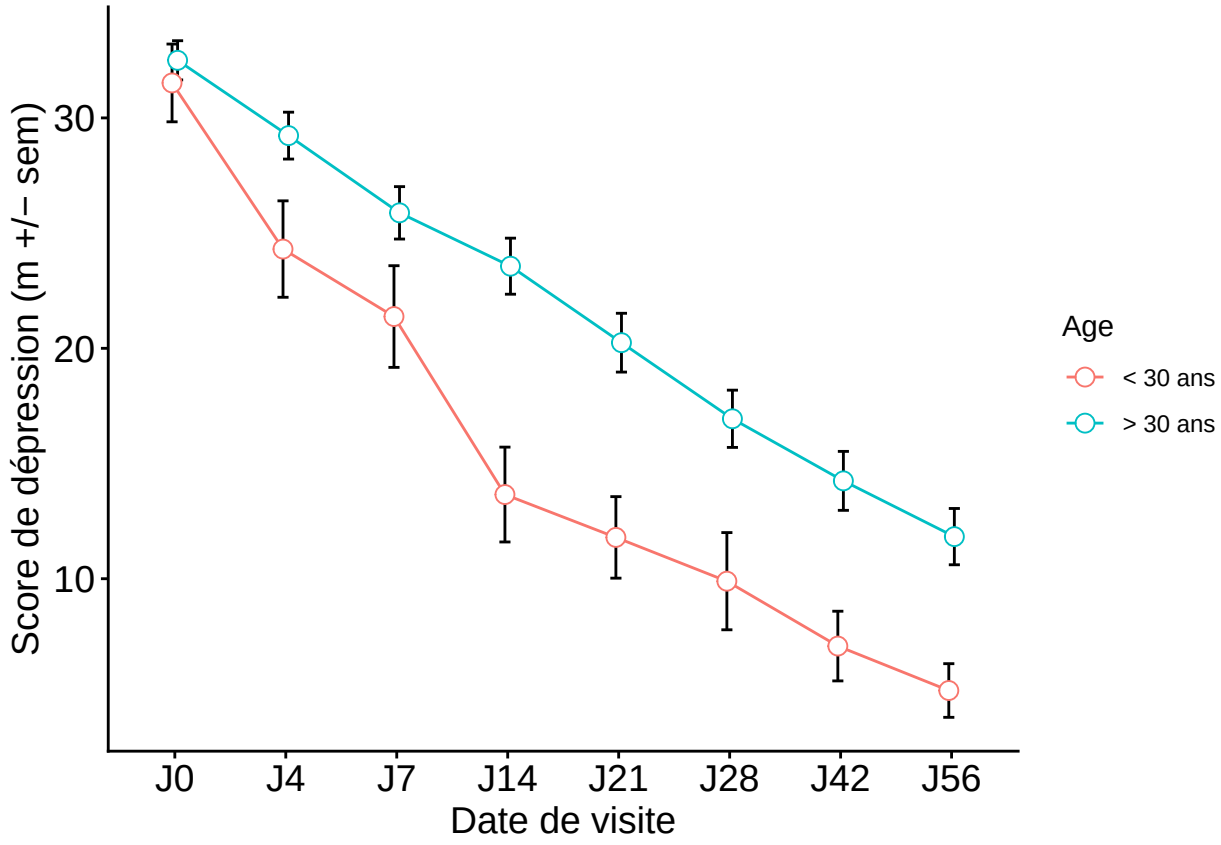
valeurs


ouvrier
sans.emploi
Non déprimés
employé
commerçant
autre
intermédiaire
cadre
agriculteur
ouvrier
sans.emploi
Déprimés
employé
intermédiaire
commerçant
autre
cadre
agriculteur

4 Données répétées : données mesurées au cours du temps

4.A Représentation répétée

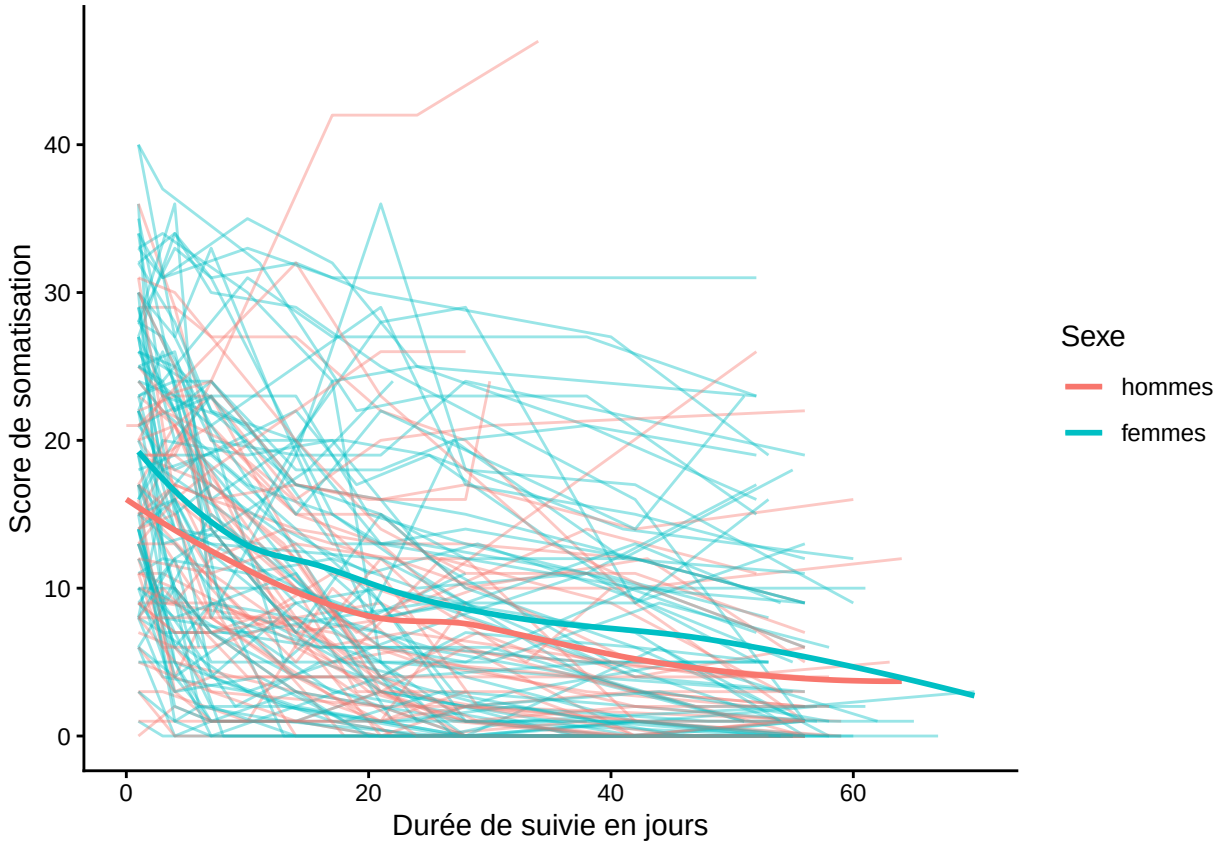
```
d1 <- aggregate(scl$depression,by=list(scl$VISIT,scl$age>30),FUN=mean)
names(d1) <- c("date","age","mean")
d2 <- aggregate(scl$depression,by=list(scl$VISIT,scl$age>30),FUN=sd)
d3 <- aggregate(scl$depression,by=list(scl$VISIT,scl$age>30),FUN=length)
d1$sem <- d2$x/sqrt(d3$x)
pd <- position_dodge(0.1)

p <- ggplot(d1, aes(x=date, y=mean, group = age, colour= age))
p <- p + geom_errorbar(aes(ymin=mean-sem, ymax=mean+sem), colour="black", width=.2, position=pd)
p <- p + geom_line(position=pd)
p <- p + geom_point(position=pd, size=3, shape=21, fill="white")
p <- p + theme_classic() + labs(y = "Score de dépression (m +/- sem)", x = "Date de visite")
p <- p + theme(axis.title.x = element_text(size=14),axis.title.y = element_text(size=14))
p <- p + theme(axis.text.x = element_text(size=14),axis.text.y = element_text(size=14))
p <- p + scale_color_discrete(name="Age", labels=c("< 30 ans", "> 30 ans"))
p
```



4.B Diagramme en fagots

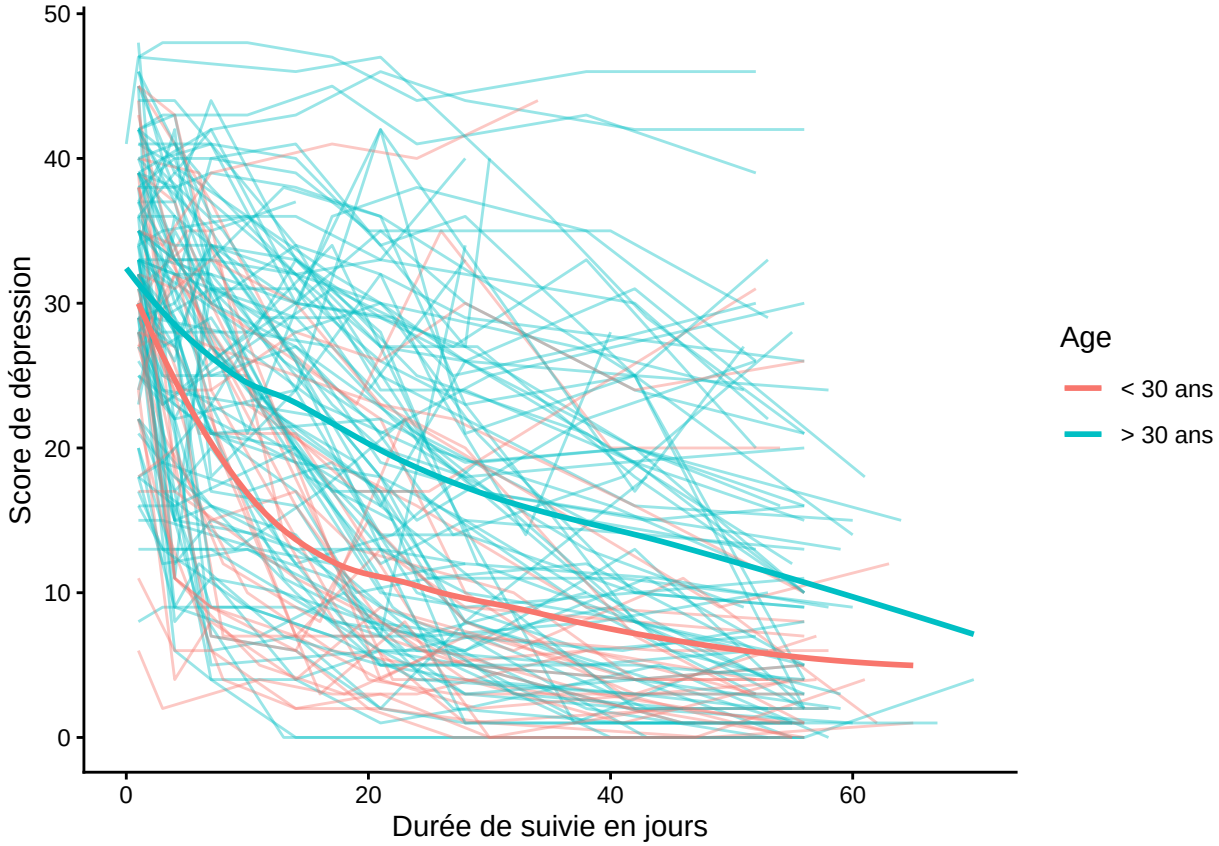
1. Représentation du score de somatisation (`smp$somatisation`) selon le sexe

```
p <- ggplot(na.omit(scl), aes(JOUR.VISIT, somatisation, group = NUMERO, color = as.factor(SEXE)))
p <- p + geom_line(alpha = 0.4) + geom_smooth(aes(group = as.factor(SEXE)), se = FALSE)
p <- p + theme_classic() + labs(y = "Score de somatisation", x = "Durée de suivie en jours")
p <- p + scale_color_discrete(name="Sexe", labels=c("hommes", "femmes"))
p
```



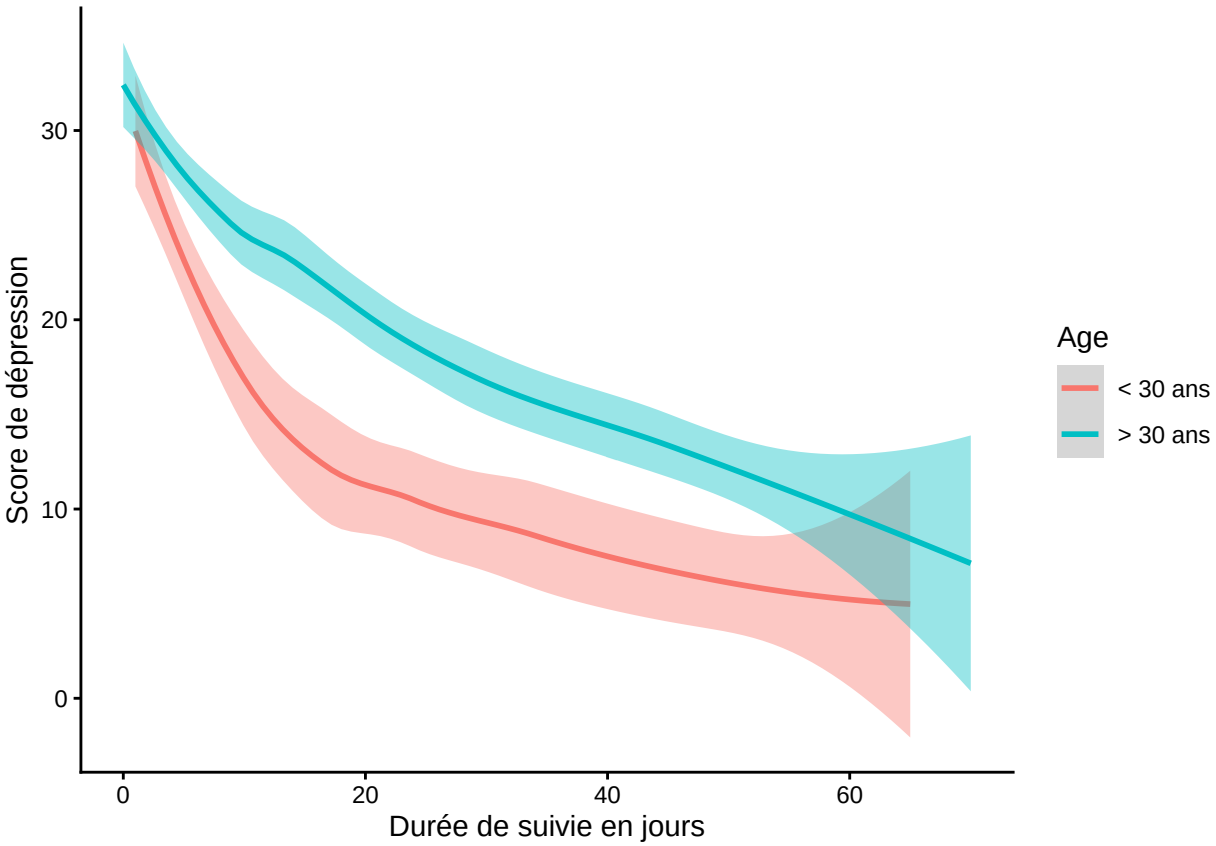
2. Représentation du score de dépression selon l'âge (> ou < 30 ans)

```
p <- ggplot(na.omit(scl), aes(JOUR.VISIT, depression, group = NUMERO, color = age>30))
p <- p + geom_line(alpha = 0.4) + geom_smooth(aes(group = age>30), se = FALSE)
p <- p + theme_classic() + labs(y = "Score de dépression", x = "Durée de suivie en jours")
p <- p + scale_color_discrete(name="Age", labels=c("< 30 ans", "> 30 ans"))
p
```



3. Lissage

```
p <- ggplot(na.omit(scl), aes(JOUR.VISIT, depression, group = NUMERO, color = age>30))
p <- p + geom_smooth(aes(group = age>30, fill = age>30))
p <- p + guides( fill = FALSE)     #To remove legend for fill aesthetic
p <- p + theme_classic() + labs(y = "Score de dépression", x = "Durée de suivie en jours")
p <- p + scale_color_discrete(name="Age", labels=c("< 30 ans", "> 30 ans"))
p
```

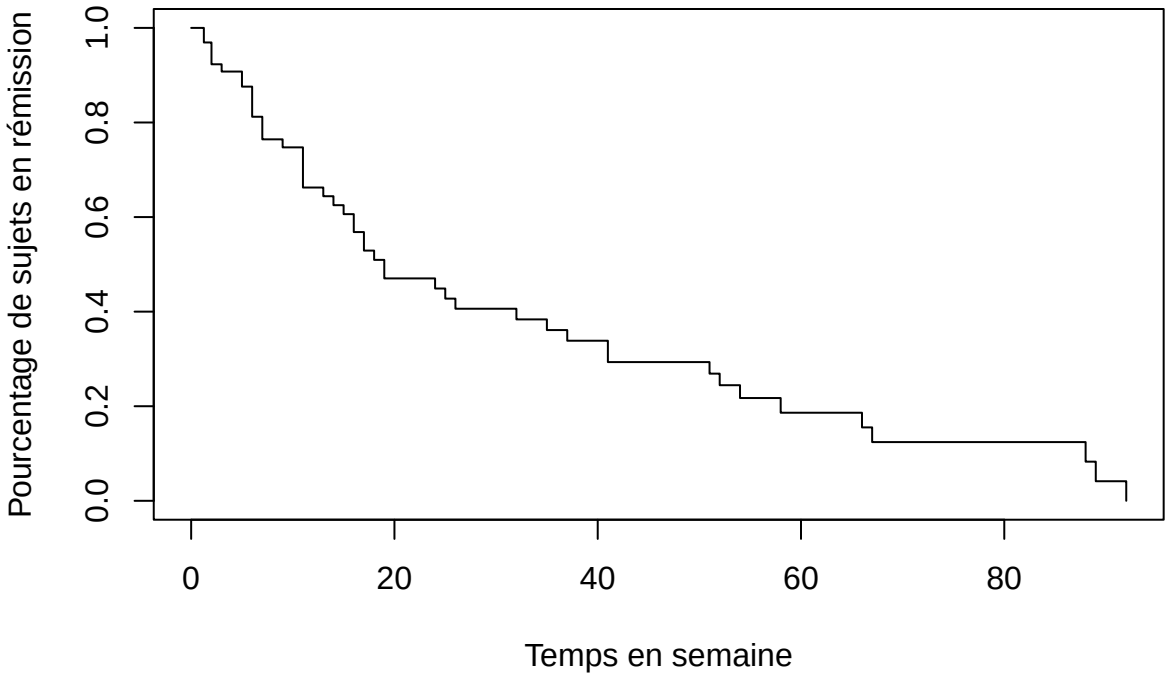

Score de dépression
30
20
10
0
0
20
40
60
Durée de suivie en jours
Age
< 30 ans
> 30 ans

5 Courbes de survie

5.A Courbe de Kaplan-Meier

Syntaxe : fonction `survfit()` du package `survival`
Ici représentation du pourcentage de sujets en rémission en fonction du temps (en semaines) pour le dataset `myelome`.

```
library(survival)
plot(survfit(Surv(T,DECES)~1,data=myelome),xlab="Temps en semaine",ylab="Pourcentage de sujets en
↪  rémission",conf.int=FALSE)
```



ou autre option : Utilisation du package `survminer` et `ggplot2` pour une meilleure présentation

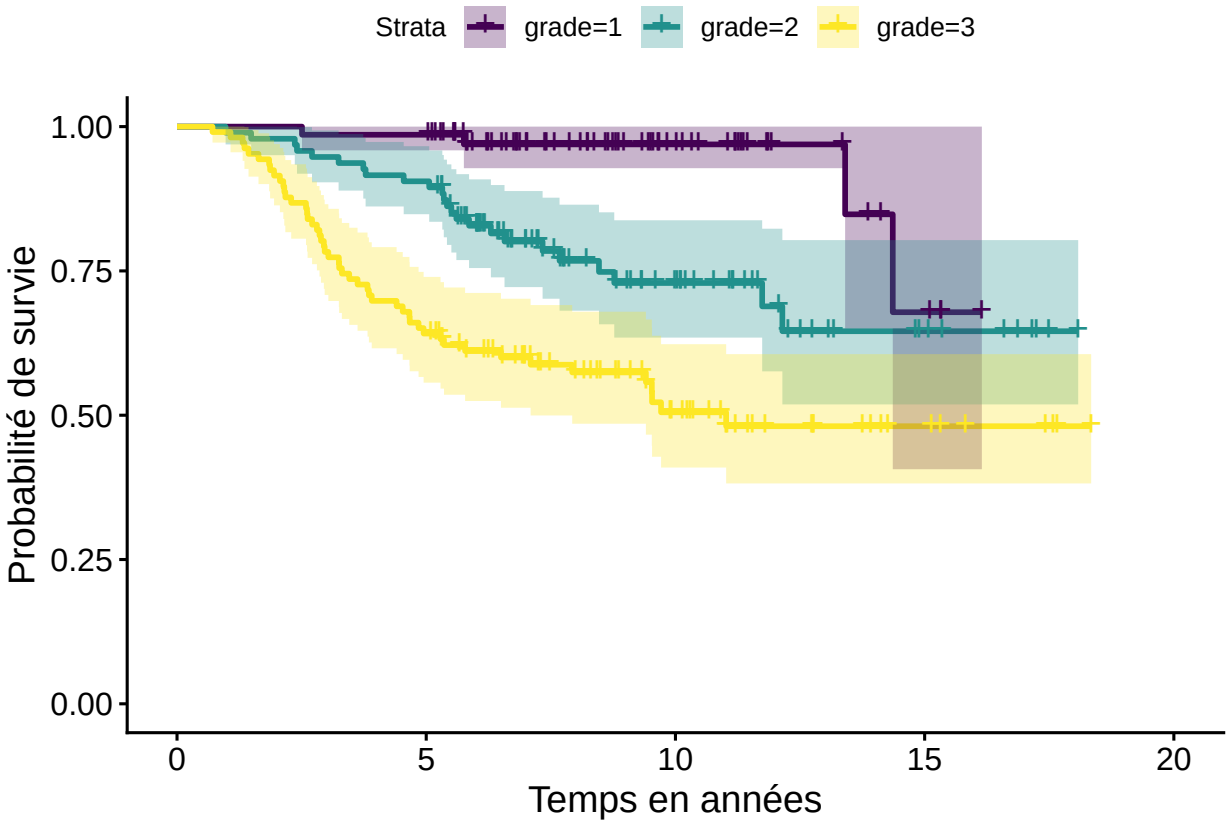
Ici représentation de la probabilité de survie en fonction du temps (en années) pour le dataset `ks` (patients atteints de cancer du rein).

1. Définir un objet `fit` avec la fonction `survfit()` contenant la formule de survie

```
fit <- survfit(Surv(survival, eventdeath) ~ grade, data = ks)
```

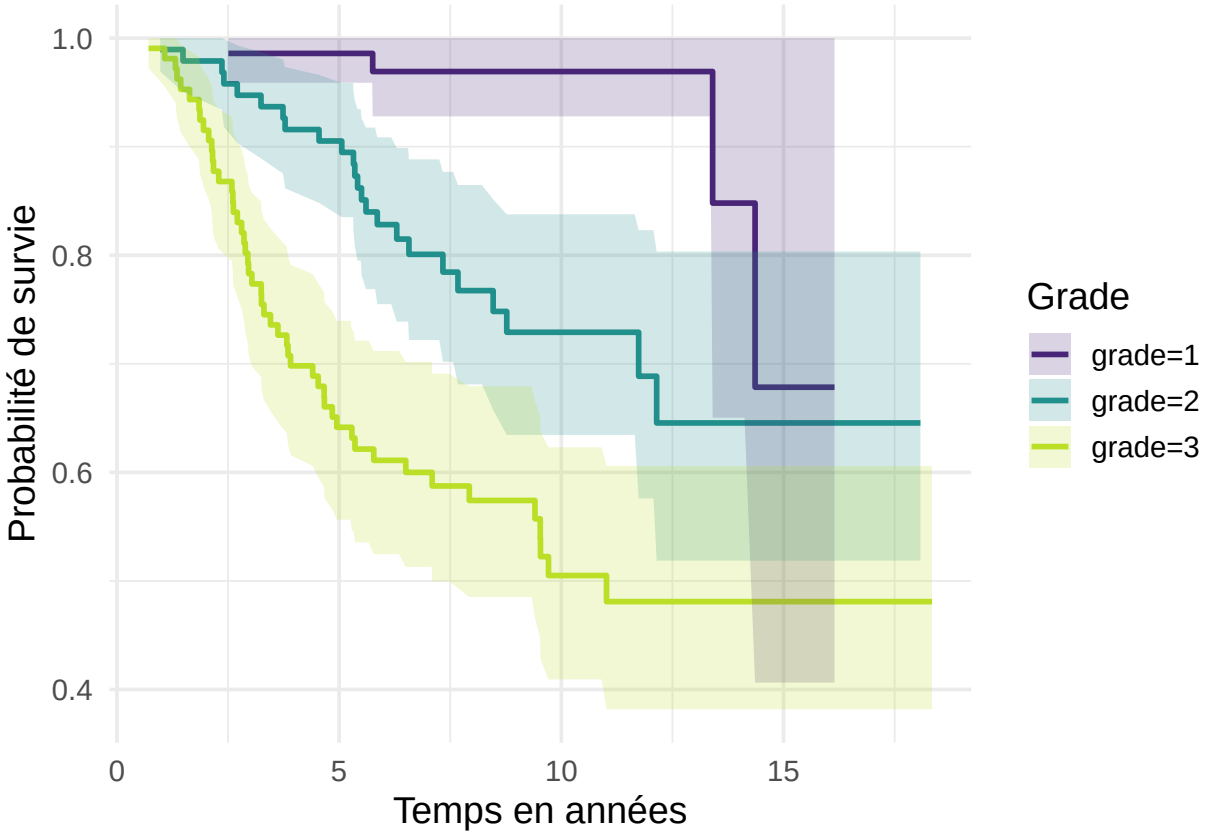
2. Représentation avec `survminer` ou avec `ggplot2`
    - avec `survminer` : fonction `ggsurvplot()`

```
p <- ggsurvplot(
  fit,
  data = ks,
  palette = viridis(3),
  conf.int = TRUE,
  xlab = "Temps en années",
  ylab = "Probabilité de survie"
)

print(p)
```



- avec `ggplot2` : package `broom` nécessaire pour convertir l'objet `fit` en data.frame utilisable pa

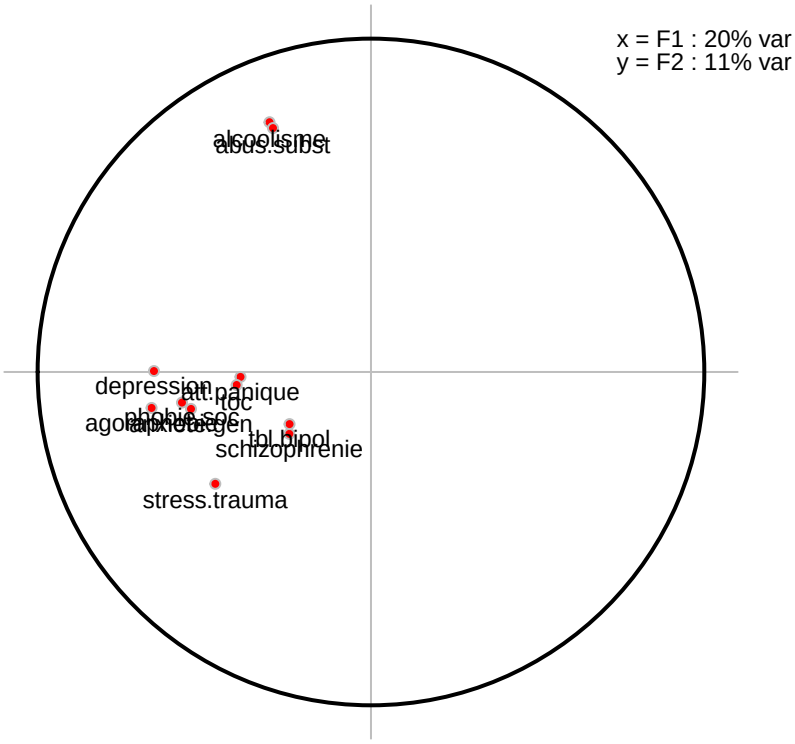
```r
library(survival)
library(broom)
library(ggplot2)
library(viridis)

# 1. Modele de survie
fit <- survfit(Surv(survival, eventdeath) ~ grade, data = ks)

# 2. Conversion en data.frame utilisable par ggplot
df <- tidy(fit)

# 3. Plot ggplot2 avec intervalle de confiance
ggplot(df, aes(x = time, y = estimate, color = strata)) +
  geom_step(linewidth = 1) +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high, fill = strata), alpha = 0.2, color = NA) +
  scale_color_viridis(discrete = TRUE, option = "D", begin = 0.1, end = 0.9) +
  scale_fill_viridis(discrete = TRUE, option = "D", begin = 0.1, end = 0.9) +
  labs(
    x = "Temps en années",
    y = "Probabilité de survie",
    color = "Grade",
    fill = "Grade"
  ) +
  theme_minimal(base_size = 14)
```



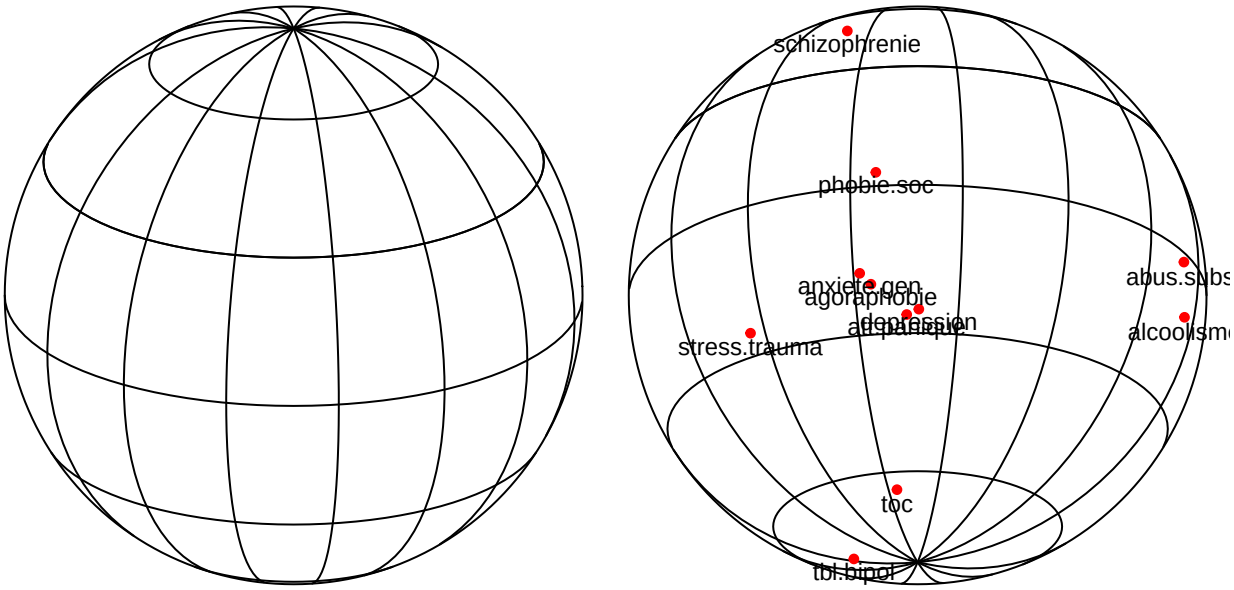
6 Méthodes multidimensionnelles

6.A Représentation sphérique

```
acp <- smp[,c("schizophrenie","depression","tbl.bipol","anxiete.gen","stress.trauma",
"att.panique","agoraphobie","toc","phobie.soc","alcoolisme","abus.subst")]
mdspca(acp)
```




```
sphpca(acp)
```



Représentation en réseau

À la mode car le mot réseau est à la mode

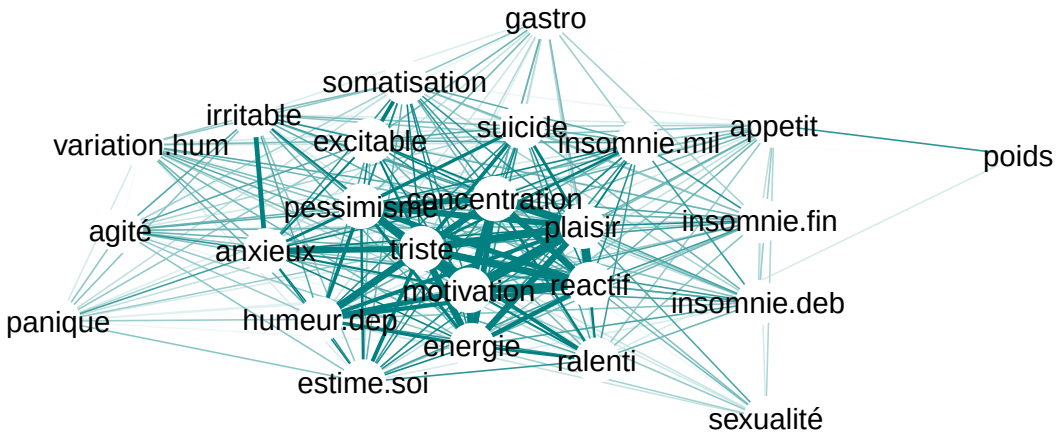
Représente graphiquement des associations entre variables :

- Plus la corrélation est forte entre les variables
- Plus les variables sont proches et reliées par des traits épais

Pas scientifiquement très rigoureux (+ c'est proche, + c'est gros : on *ne quantifie pas*).

Par ex ici : `hypersomnie` : pas terrible !

```
noms <- c("insomnie.deb","insomnie.mil", "insomnie.fin",
"hypersomnie","triste","irritable","anxieux","reactif","variation.hum","humeur.dep",
"appetit","poids","concentration","estime.soi","pessimisme","suicide","motivation",
"energie","plaisir","sexualité","ralenti","agité","somatisation","excitable","panique","gastro")
qgraph(cor(rush[,3:28], use="complete.obs"), minimum = 0.25, borders = FALSE, vsize = 4,
layout="spring", labels=noms, label.cex=5, label.scale.equal=TRUE,
 ↪  edge.color=rgb(0,0.5,0.5),edge.width=0.5)
```



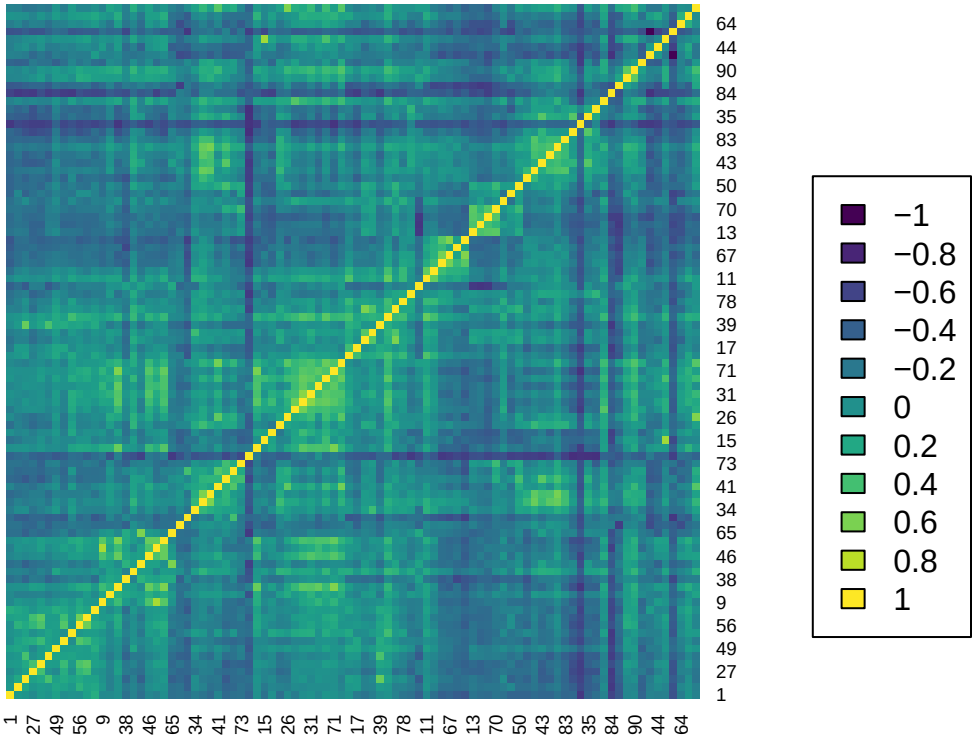
hypersomnie

6.B Heatmap

- Permet de représenter **énormément de variables** ensemble
- Ici : 90 variables = 8100 coefficients de corrélations

```
#définition des variables
somatisation <- c(1, 4, 12, 27, 42, 48, 49, 52, 53, 56, 58, 40)
obsession <- c(9, 10, 28, 38, 3, 45, 46, 51, 55, 65)
sensitivite <- c(6, 21, 34, 36, 37, 41, 61, 69, 73)
depression <- c(5, 14, 15, 20, 22, 26, 29, 30, 31, 32, 54, 71, 79)
anxiete <- c(2, 17, 23, 33, 39, 57, 72, 78, 80, 86)
hostilite <- c(11, 24, 63, 67, 74, 81)
phobie <- c(13, 25, 47, 70, 75, 82, 50)
paranoia <- c(8, 18, 43, 68, 76, 83)
psychose <- c(7, 16, 35, 62, 77, 84, 85, 87, 90, 88)
divers <- c(19, 44, 59, 60, 64, 66, 89)

#création de la matrice de corrélation
scl.dim <- scl[,5+c
 ↪  (somatisation,obsession,sensitivite,depression,anxiete,hostilite,phobie,paranoia,psychose,divers)]
library(viridisLite)
dsph <- function(x) {as.dist(0.5*(1-x)^2)}

#représentation
heatmap(cor(na.omit(scl.dim)), col=viridis(255), scale="none", distfun=dsph, Rowv=NA, Colv=NA,
```

```
cexRow=0.7, cexCol=0.7, labRow=substring(names(scl.dim), 2, 20), labCol=substring(names(scl.dim),
↪  2, 20))
legend(x="right", legend=(-5:5)/5, fill=viridis(11))
```



6.C Lasagne plot

Représente des **données répétées** en Heatmap.

Chaque ligne représente un patient déprimé suivi à differents moments du temps.

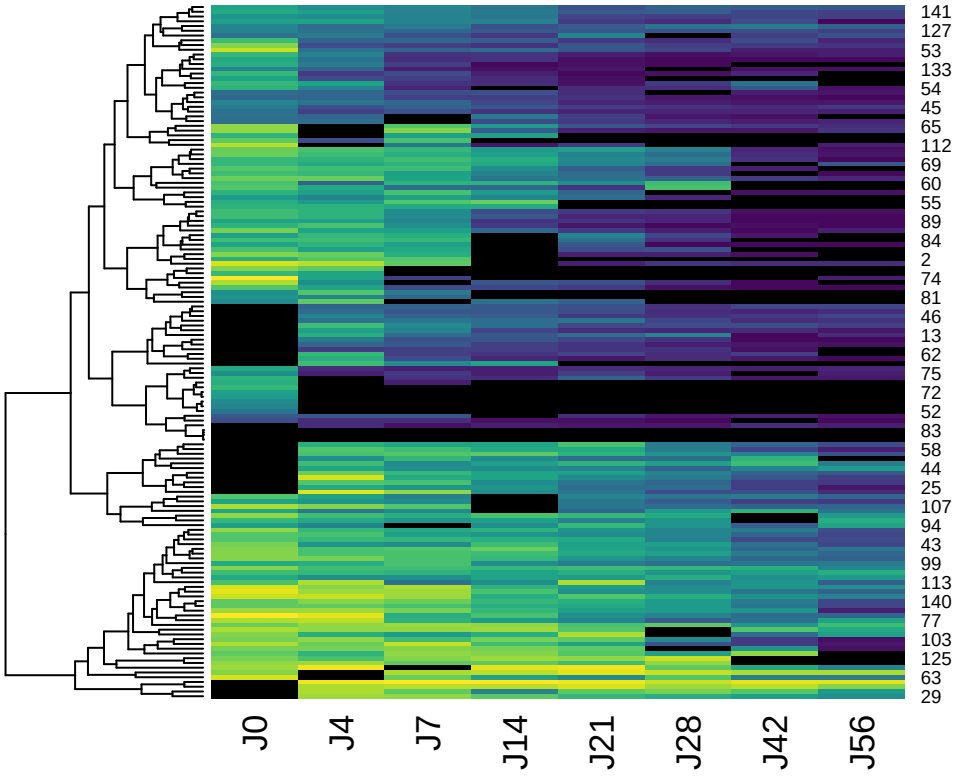
- – ils sont jaunes, + ils sont déprimés
- – ils sont bleus, - ils sont déprimés
- noir = donnée manquante.

Chaque colonne représente les variables.

Représente **l'évolution du score de dépression au cours du temps** de **tous les sujets**.

Permet de regrouper les sujets en **pattern d'évolution similaires**

```
library(reshape2)
scl.w <- dcast(scl[,c("NUMERO","VISIT","depression")], NUMERO~VISIT, value.var="depression")
scl.w <- scl.w[,-c(1,10)]
int <- scl.w
int[is.na(int)] <- 0
couleurs <- viridis(255)
couleurs[1] <- "#000000FF"
heatmap(as.matrix(int), col=couleurs, scale="none", Rowv=NULL, Colv=NA)
```



7 Textométrie : traitement automatique du langage

Ici : on s'intéresse à l'association de mots dans des articles traitant de santé mentale des médecins