

> Importer des données et lire les données

`read.csv(file, header = {TRUE ou FALSE, sep = "x", dec = ".", ...})`

1^{re} ligne
= non des colonnes

espaces par défaut

`read.table(file.txt, header = {TRUE ou FALSE, sep = "x", ...})`

`names(file)`: renvoie noms des colonnes d'un dataframe

→ selection d'un sous-ensemble de données:

`File[...]` : `File[File$Colonne 1 == "x"]` : donne toutes les lignes dont la donnée de la colonne 1 est égale à x

`File[File$Colonne 1 == "x", 2]` : donne tous les éléments de la colonne 2 qui correspondent à une ligne dont la donnée de la colonne 1 est x

`File[File$Colonne 1 == "x", c(1,4)]`

> Représentations

→ HISTOGRAMME : `hist()`

`hist(vecteur, main = "...", col = "...", border = "...", xlab = "...", ylab = "...", xlim = ..., ylim = ..., breaks)`

↑
range of values des axes

↑
largeur des barres
le découpage des valeurs

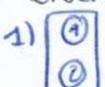
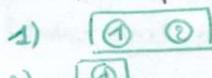
→ GRAPHE DE CLEVELAND : `dotchart()`

`dotchart(vecteur, main = "...", pch = 20)`

`dotchart(sort(vecteur), main = "...", pch = 20)`

↳ ordonné

☞ pour afficher plusieurs plot ensemble : `par(...)`



par(mfrow)

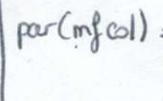
1 2
3 4

par(mfcol)

1 2
3 4



c



c



c



c



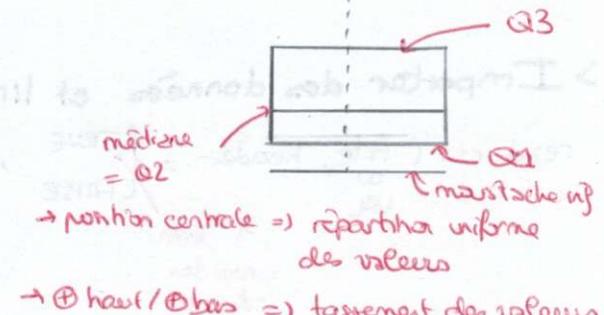
c

→ BOÎTE À MOUSTACHE : boxplot()

points abscissas → Théo Jalabert 

boxplot (vecteur, col = "...", main = "...", horizontal = TRUE)

par placer la boîte à moustache horizontalement
Là respecte son sens de lecture de gauche à droite



→ PRÉSENTATION EN CAMEMBERT pre()

pre(summary(vecteur), main = "...", col = "..."/c = ("...", "...", "...",))

→ PRÉSENTATION EN BÂTONS barplot()

Fréquences absolues ou relatives

barplot (summary(vecteur)
summary(vecteur)/length(vecteur)
vecteur)

, main = "...", col = "...", las = {1, 2})

label horizontaux
↓
label verticaux

> RELATION ENTRE DEUX VARIABLES

Croisement	Paramètre	Graphique
Quantitatif x Quantitatif	Covariance coefficient de corrélation coeff de déterminant	nuage de points
Qualitatif x Qualitatif	Chi - Deux Coefficient de Cramer	mosaïque représentation en ballons
Quantitatif x Qualitatif	Rapport de corrélation	reprz inter et intragroupes boîtes à moustaches

$$\text{cov}(X, Y) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

$$r = \frac{\text{cov}(X, Y)}{S_x S_y} \text{ et } r^2 = \frac{\text{cor}(v_1, v_2)}{S_x S_y}$$

→ NUAGE DE POINTS plot()

plot (vecteur abs, vecteur ord, pch = 20, main = "...", xlab, ylab, type = "n")

text (vect abs, vect ord, vecteur nom)

Là donne le cadre du graphique main vide (graphique à ajouter à la ligne suivante)

attach (dataframe) : permet de récupérer directement une variable d'un dataframe en

→ Valeur du Chi-Deux de contingence

On construit une table de contingence (table théorique) par définition le lien entre deux variables : hypothèse d'indépendance entre les deux variables i.e. équation : $\frac{n_{ij}}{n}$

Chi-Deux de contingence : compare $E_0 = n_{ij}$ avec $E_T = \frac{n_{ij}}{n}$

effectifs
observés



effectifs théoriques

On a $\chi^2 = \sum \frac{(E_0 - E_T)^2}{E_T} \Rightarrow$

- > si $\chi^2 = 0$: indépendance des deux variables
- > si χ^2 est petit : E_0 et E_T presque identiques
↳ variables peu liées entre elles
- > si χ^2 est grand : E_0 et E_T différents
↳ variables très liées entre elles

Code R : `chisq.test(v1, v2)`

→ REPRÉSENTATION EN BALLONS `balloonplot()`

`library(gplots)`

`balloonplot(table(v1, v2), main = "...")`

→ REPRÉSENTATION EN MOSAÏQUE `mosaicplot()`

se base sur les écarts entre E_0 et E_T : $\frac{(E_0 - E_T)}{\sqrt{E_T}}$

`mosaicplot(table(v1, v2), main = "...", shade = TRUE)`

↳ permet de mettre en relief les différences

• bleu : les E_0 sont + grands que E_T
sur l'absence de lien entre les deux variables

• rouge : les E_0 sont + petits que les E_T
sur l'absence de lien entre les deux variables

→ Quantitatif x Qualitatif

> Variation : • variance descriptive mesurée sur un groupe de n individus

$$S^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}$$

• variance estimée de la population à partir d'un échantillon de n individus

$$\hat{\sigma}^2 = \frac{n}{n-1} S^2 \quad \text{ou encore} \quad \hat{\sigma}^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}$$

On préfère travailler sur la variation totale, i.e sur la somme des carrés des écarts à la moyenne (SCE)

$$SCE_T = \sum_{i=1}^n (x_i - \bar{x})^2$$

SCE \leftarrow function (x) sum ((x - mean(x)) 2)

Variation inter-groupe : corré des écarts entre la moyenne du groupe et la moyenne globale

$$SCE_B = \sum_{k=1}^n n_k (\bar{x}_k - \bar{x})^2$$

Exemple :

crimes dans les Etats Américains

nbr d'Etats ayant été échantillonnes l'année k

nbr moyen de crimes pour l'année k

SCEB \leftarrow function (x , gpe) {

 moyenne \leftarrow tapply (x , gpe, mean)

 effectifs \leftarrow tapply (x , gpe, length)

 res \leftarrow (sum (effectifs * (moyenne - mean(x)) 2))

 return (res)

}

Rapport de corrélation

$$\eta^2 = \frac{SCE_B}{SCE_T} \Rightarrow \begin{array}{l} \bullet \text{proche de } 0 : \text{variables pas liées} \\ \bullet \text{proche de } 1 : \text{variables liées} \end{array}$$

eta2 \leftarrow function (x , gpe) { res \leftarrow SCEB (x , gpe) / SCE_T (x) ; return (res) }

(à finir)

(Voir le cas des graphes avec les modèles d'une variables qualitatives : boîte à moustache, graph de Cleveland)



→ NUAGE DE POINTS EN 3D : library(rgl) plot3d()

plot3d (dataframe, type = "s", col = vecteur - couleur)

↑ vecteur ↑

créé à l'aide de rgl
apparaissant

💡 colMeans(dataframe) : permet de calculer les moyennes des colonnes d'un dataframe

💡 Si on veut faire apparaître l'ellipsoïde dans le NCP d'une ACP :

plot3d (ellipso3d(cov(dataframe)), col = "grey", alpha = 0.5, add = TRUE)

> Centrage et réduction

Quand trop de différence dans le rôle des données : on opte pour un centrage et une réduction par donner la même importance à chaque variable

→ Effectuer centrage et réduction : library(ade4) scalewt()

v1 ← scalewt(dataframe, center = TRUE, scale = TRUE)

↳ renvoie une matrice

alors on a : v2 ← as.data.frame(v1)

💡 fonction pour arrondir : round(dataframe, a)

↑ précision de l'arrondi

💡 par avoir un graph (2D ou 3D) pertinent quand valeur des axes limitées :

On pose un vecteur lims ← c(min(vecteur_dataframe), max(vecteur_dataframe))

et dans la fonction plot / plot3d (..., xlim = lims, ylim = lims, zlim = lims)

Si on réalise une ACP normée : les données sont centrées et réduites

💡 COURS : Forme générale du nuage : drayé (ou ellipsoïde)

↳ définie par ses 3 axes

- 1) longueur de la drayé : plus grand diamètre
- 2) largeur de la drayé : diamètre moyen
- 3) épaisseur de la drayé : plus petit diamètre

> ACP centrée-réduite dans ade4 : library(ade4) dedri.pca()

dedri.pca(dataframe, center = TRUE, scale = TRUE), scannf = FALSE, nf = 3)

Q? : Select the number of axes?

permet de conserver automatiquement 3 axes

- `dataframe acp$tob`: contient les données du tableau initial après centrage et réduction
- `acp$cw`: poids des colonnes : par défaut chaque variable a un poids de 1 → canonique
- `acp$lw`: poids des lignes : par défaut chaque individu a un poids de $\frac{1}{n}$ → uniforme
- `acp$eig`: valeurs propres (eigen values) de la plus petite des matrices à diagonaliser

↳ nous renseigne sur la fraction de l'énergie totale prise en charge par chaque axe

• `summary(acp)` - Total inertia :

Eigenvalues :

Ax1 Ax2 Ax3

Projected inertia (%)

Cumulative projected inertia (%)

• `acp$rank` : donne le rang de la matrice diagonalisée : ici le nombre de composantes principales

• `acp$nf` : nombre de facteurs conservés dans l'analyse

• `acp$c1` : coordonnées des variables (colonnes) : norme unité

• `acp$li` : coordonnées des individus (lignes) : norme unité

• `acp$co` : coordonnées des variables (colonnes) : normés à la $\sqrt{\text{valeur propre correspondante}}$

• `acp$li` : coordonnées des individus (lignes) : normés à la $\sqrt{\text{valeur propre correspondante}}$

↳ lien entre `acp$c1` et `acp$co`

	acp\$c1			acp\$co		
	CS1	CS2	CS3	V1	Comp 1	Comp 2
V1	a	:	:	V1	x	
V2	b	:	:	V2		
V3	c	:	:	V3	y	

$$\alpha = a \times \sqrt{\text{valeur propre de axe 1}}$$

Code R

$$acpcoComp1$$

$$= acp$c1$CS1 *$$

$$\sqrt{acp$eig[1]}$$

Par les axes

$$t(t(acp$c1) + \sqrt{acp$eig})$$

donne `acp$co`

↳ lien entre `acp$li` et `acp$li` : idem

$$acpliRS1 * \sqrt{acp$eig[1]}$$

↳ donne 1re colonne de `acp$li`

$$\text{et } t(t(acp$li) * \sqrt{acp$eig})$$

↳ donne `acp$li`

• `acp$call` : trace des calculs lors de l'appel de la fonction `prcomp()`

• `acp$cent` : donne les moyennes des variables analysées

• `acp$norm` : donne les écarts-types (\sqrt{n}) des variables analysées

$$\frac{x - m}{\sqrt{n}}$$

> Représentation graphiques dans ade4

© Théo Jalabert

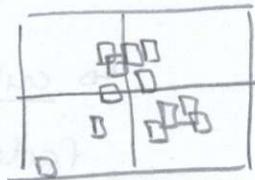
→ REPRÉSENTATION DES INDIVIS

↳ sur les différents plans factoriels

slabel(acp\$li, xax=a, yax=b, clabel=1, S)

↳ (a,b) s]

Le plan factoriel sur lequel on veut projeter



⊕ rajouter en information supplémentaire une variable qualitative définissant des groupes d'individus : s.class()

s.class(dfxy=acp\$li, fac=vector_grapes, col=vector_color, xax=a, yax=b)

→ REPRÉSENTATION DES VARIABLES

Le cercle des corrélations

s.corcircle()

s.corcircle(acp\$co, xax=1, yax=2)

→ REPRÉSENTATION SIMULTANÉE INDIVIS / VARIABLES

scatter(acp, posneig = "bottomright")
{"none"}

> Changement de pondération

ACP classique : pondération associée aux individus est uniforme

acp\$lw : tous les individus ont la même pondération (TD: 0,05)

poidsU ← acp\$lw # pondération uniforme

poidsD ← depente\$effetf # on pond le vecteur qui contient les effets des individus (ici par âge moyen)

poidsD ← poidsD/nm(poidsD)

round(poidsD, 3)

↳ nouvelles pondérations

→ CRITÈRE DE KAISER

© Théo Jalabert

↳ critère strict : on ne retient que les axes dont les valeurs propres sont > 1

↳ critère dans le cas des données non réduites : on garde les facteurs dont on a :

$$\lambda > \frac{\text{Tr}(V)}{P}$$

avec V matrice de covariance

P nombre de facteurs

$$(I-1)(J-1)$$

$$4 \times 8 = 32$$

→ INERTIE PROJETÉE SUR LES AXES

↳ summary(zep)

Projected inertia (%)

↳ à partir des valeurs propres :

$$\text{pve} \leftarrow 100 * \text{acp$eig} / \text{sum(acp$eig)}$$

→ EFFET DE TAILLE

On observe un effet de taille dans une ACP sur les variables lorsque les points sont regroupés du même côté d'un point factoriel.

TD 1105 : AFC

© Théo Jalabert

→ TABLE DE CONTINGENCE : `table()`

`c ← table(v1, v2)`

↳ donne un tableau croisé qui rentre les individus entre les modèles des deux variables qualitatives

↳ on le transforme en dataframe :

`dfc ← data.frame(unclass(c))`

> Score

→ SCORE A PRIORI : on se base souvent sur une opposition naturelle des modèles

↳ Ex des yeux :

`scorecheveux ← c(1, -1, -1, 1)` opposition forte/clair

`names(scorecheveux) ← colnames(couleurs)`

`scorecheveux`

Blond	Naran	Noir	Rou
1	-1	-1	1

Par chaque ligne de la tabl des contingence (couleur des yeux), on obtient la fréquence observée

`dfcouleurens ← data.frame(unclass(couleurs))`

`dfcouleurens[["Bleu",]] / sum(dfcouleurens[["Bleu",]])` fréquence des yeux Bleu par couleur de cheveux

Blond	Naran	Noir	Rou	
Bleu	0,46	0,39	0,09	0,08

Puis on calcule le score moyen par le modèle yeux Bleu

`yeux.Bleu ← dfcouleurens[["Bleu",]] / sum(dfcouleurens[["Bleu",]])` fréquences observées des yeux Bleu

Bleu	Blond	Naran	Noir	Rou
	0,44	-0,39	-0,09	0,08

`sum(yeux.Bleu * scorecheveux)` ← score moyen par une modèle (ici yeux Bleu)
0,0326

On peut calculer le score moyen par tout les couleurs des yeux

`freqyeux ← apply(dfcouleurens, 1, function(x) x / sum(x))`

`t(freqyeux)` on prend la transposée pour avoir les couleurs des yeux sur les lignes

`Scoreyeux ← apply(t(freqyeux), 1, function(x) sum(x * scorecheveux))`

Bleu	Naran	Noisette	Vert	Score moyen de -0,7 : <0 donc cheveux foncés
------	-------	----------	------	--

Pour obtenir le score moyen d'un modèle :

© Théo Jalabert

THJ

1) on attribue un score à priori

2) par chaque ligne de la table de contingence, on calcule fréquence observée : $\frac{n_i}{\text{sum}(n)}$

3) puis on fait le score moyen du modèle en faisant :

$$\text{sum}(\text{freq observées} * \text{score à priori})$$

(f_{ij}, n_{ij}) etat \rightarrow s_{ij}

→ SCORE OPTIMUM = dudi.coa(1)

AFC = méthode qui donne un score sur les colonnes tq les scores moyens des lignes soient les plus séparés possibles (et vice versa)

library (ade4)

ac \leftarrow dudi.coa(dfcouleur, scannf = F, nf = 3)

ac\$c1[, 1] : scores optimaux par modèles des couleurs des cheveux
(plus généralement : les scores optimaux des modèles en colonne)

Scores qui sont par les modèles

rownames(ac\$c1) : donne les noms des modèles qui sont en colonne dans la table de contingence (ici Bleu Marron Noir Roux)

ac\$M[, 1] : scores optimaux par les modèles des couleurs des yeux
(plus généralement : les scores optimaux des modèles en ligne)

rownames(ac\$M[, 1]) : donne le nom des modèles qui sont en ligne
score par l'interprétation dans la table de contingence (ici Bleu Marron Noir Vert)

• On peut trouver les scores moyens des modèles des couleurs de yeux (resp. des couleurs de cheveux) à partir des scores optimaux des modèles des couleurs de cheveux en ac\$c1[, 1] (resp. des modèles des couleurs des yeux en ac\$M[, 1]).

• On peut directement trouver les scores moyens dans :

ac\$li[, 1] par les modèles en ligne

ac\$co[, 1] par les modèles en colonne

DONC

Score optimum	Score moyen	Table de contingence
ac\$c1[, 1] : score optimal par les modèles de V ₁	ac\$co[, 1] : score moyen par les modèles de V ₁	$\begin{array}{c cccc} V_1 & \Pi_1^1 & \Pi_1^2 & \Pi_1^3 \\ \hline V_2 & \Pi_2^1 \\ & \Pi_2^2 \\ & \Pi_2^3 \end{array}$
ac\$M[, 1] : score optimal par les modèles de V ₂	ac\$li[, 1] : score moyen par les modèles de V ₂	

> TABLE DE CONTINGENCE

© Théo Jalabert

- Soit on a déjà le tableau / les vecteurs à partir duquel/ desquel on veut faire la table :

$$c \leftarrow \text{table}(v_1, v_2)$$

- Soit on a les données : ex :

→ mode de règlement : annuel, mensuel, semestriel, biméthuel

→ situation maritale : célibataire, concubin, divorcé, marié ou veuf
+ les chiffres

et on construit nous même :

$$\text{table} \leftarrow \text{matrix}(c(\dots), \text{byrow} = T, \text{ncol} = \dots)$$

$$\text{colnames}(\text{table}) \leftarrow c(\dots)$$

$$\text{rownames}(\text{table}) \leftarrow c(\dots)$$

↳ informations de base :

$n \leftarrow \text{sum}(\text{table})$: nombre total d'individus

$I \leftarrow \text{nrow}(\text{table})$: nombre de modalités par la variable en ligne

$J = \text{ncol}(\text{table})$: nombre de modalités par la variable en colonne

↳ on peut construire le tableau des fréquences relatives : $f_{ij} = \frac{n_{ij}}{n}$

$$\text{freqtable} \leftarrow \text{table}/n$$

$$\text{round}(\text{freqtable}, \text{digts} = a)$$

\hookrightarrow nombre après la virgule

↳ Représentations : library(gplots)

• ballonplot(as.table(table))

• table.cont(table, row.labels = rownames(table), col.labels = colnames(table), csize = 2)

• mosaicplot(table, shade = TRUE)

→ PROFILS LIGNES / COLONNES : Variables V₁ / V₂

• LIGNES : $f_{j|i} = P(V_2=j | V_1=i)$

$$\hookrightarrow f_{j|i} = \frac{n_{ij}}{n_i}$$

profLignes $\leftarrow \text{prop.table}(\text{table}, 1)$

• COLUMNES : $f_{i|j} = P(V_1=i | V_2=j)$

$$\hookrightarrow f_{i|j} = \frac{n_{ij}}{n_j}$$

profColonnes $\leftarrow \text{prop.table}(\text{table}, 2)$

> TEST DU χ^2

© Théo Jalabert

→ Degré de liberté associé à une table à I lignes, J colonnes

$$\hookrightarrow (I-1)(J-1)$$

→ Test du χ^2 : chisq.test()

Exemple : table1 = matrix(runif(12), 3, 4)
chisq.test(table1)

X-squared = 12.345, df = 4, p-value = 0.01234

Si si l'hypothèse n'est pas validée, on rejette l'hypothèse d'indépendance.

→ Lien entre Khi-Berno et enrhé totale

$$I_T = \frac{\chi^2}{n} \Rightarrow \chi^2 = I_T \cdot n$$

khitest ← chisq.test(table1)

I_T → khitest\$statistic / sum(table1)

pour vérifier : sum(afc\$eig)