Quelques manipulations avec R et utilisation de la « librairie » IPErad (version avril 2010)

Ce document constitue une brève introduction à l'utilisation de **R**. Elle rassemble des éléments épars présentés dans le reste du cours et détaille les outils disponibles dans la librairie IPErad. Pour les manipulations de base, on pourra aussi se référer à des exemples liés à des travaux pratiques en psychologie proposés par Pallier & Lalanne (2005).

Installer R

Téléchargez et installez **R** à partir de l'adresse `http://www.r-project.org/` (choisissez sous CRAN mirror un site de téléchargement proche de chez vous, l'Université de Berne ou l'ETHZ pour la Suisse, le CICT à Toulouse ou le Département de biométrie et de biologie évolutionniste de l'université de Lyon pour la France, etc.).

Pour les manipulations, utilisez le jeu de données et la bibliothèque dont les dernières versions peuvent être téléchargées sur le site de l'Institut de psychologie et éducation de l'université de Neuchâtel[1].

Choisissez un répertoire de travail (appelé aussi répertoire courant) (par exemple `\Mes documents\R` sous windows ou `/Documents/R` sous Mac OS) dans lequel les fichiers de données (`exGraph.dat` ; `auteurDoc.txt` ; `dataQuest.txt` ; `dataRT.txt` ; `dataRT3.txt`)[2] et de fonctions supplémentaires (IPErad ou IPErad.r) seront copiés. Une fois que **R** a été lancé, sélectionnez le répertoire de travail.

Manipulations de base

Ces manipulations permettent d'effectuer des traitements simples de données issues d'un questionnaire.

Charger les données et rendre les variables directement accessibles par leur nom

```
> quest <- read.table("dataQuest.txt")
> attach(quest)
```

Visualiser toutes les données ou en partie

```
> quest
> quest[1:10,]
```

Créer et visualiser une table

```
> table(age,op1)
> matable <- table(sexe,op1)
> matable
> pretty.table(matable)
```

Appliquez un test

```
> chisq.test(matable)
```

Utiliser la « bibliothèque » IPErad

Copier dans le répertoire de travail IPErad.r (IPErad). Une fois que **R** a été lancé, sélectionnez le répertoire de travail et exécutez la requête :

```
> source("IPErad")
```

Cette bibliothèque permet d'effectuer les tests présentés dans cet ouvrage. Par exemple pour effectuer le test de Jonkheere (S)

```
> S.test(matable)
```

Les fonctions principales seront présentées systématiquement.

[1] http://www2.unine.ch/ipe/page-8565.html (consulté : février 2011).

[2] Les fichiers avec l'extension « .dat » contiennent plusieurs tables en format **R** (chargement à l'aide de la fonction load). Les fichiers avec l'extension « .txt » contiennent une table en format texte (chargement à l'aide de la fonction `read.table`).

Diagrammes divers

Représentation triangulaire

* `init.RT()` : prépare une zone graphique (anciennement `init.graph()`)

* `affiche.RT(df,ordre=1,pct=T)` : effectue l'affichage des points à partir d'un « dataframe » construit selon la structure :

- Entête : `"var1" "var2" "var3"`

- Chaque ligne : `"id" v1 v2 v3`

- `pct=T` (vrai par défaut) indique que les données sont déjà des pourcentages (compris entre 0 et 1). Dans ce cas `v3 = 1-(v1+v2)` (cette variable peut être omise dans les données).

Cette fonction utilise : `pos.x(ksi,eta)` ; `marque(ksi,eta,c="x",col="red")`

Exemple

```
> donnees <- read.table("dataRT.txt")
> init.graph()
> affiche.RT(donnees)
```

* `marque(ksi,eta,c="x",col="red")` : permet l'affichage d'un seul point donné par deux de ces coordonnées « triangulaires », son nom et sa couleur.

Représentation tétraédrique

Nécessite la librairie `rgl` (installation : `install.packages("rgl")` puis `library(rgl)`)

* `init.RT3d()` : prépare une zone graphique

* `affiche.RT3d(df,ordre=1,pct=T)` : effectue l'affichage des points à partir d'un « dataframe » construit selon la structure :

- Entête : `"var1" "var2" "var3" "var4"`

- Chaque ligne : `"id" v1 v2 v3 v4`

- `pct=T` (défaut) indique que les données sont déjà des pourcentages (compris entre 0 et 1). Dans ce cas `v4` = 1-(`v1+v2+v3`) (il peut être omis) .

Cette fonction utilise : `pos.3d(a,b,c,d)` ; `marque.3d(a,b,c,d,lab="x",col="red")`

Exemple

```
> donnees <- read.table("dataRT3d.txt")
> init.RT3d()
> affiche.RT3d(donnees)
```

Diagramme de profils

* `radar(tab, labels= dimnames(tab)[2], glabels= dimnames(tab)[1], main=NULL,couleur=c("red","blue","green","violet","orange", "maroon"),relative=FALSE,rayon=TRUE)`

* `profil(tab, labels= dimnames(tab)[2], glabels= dimnames(tab)[1], main=NULL,couleur=c("red","blue","green","violet","orange", "maroon"),relative=FALSE,rayon=TRUE, radar=TRUE)` : affiche un diagramme de profil, chaque ligne de la matrice ou du dataframe tab correspond à un profil.

- `labels` : les noms des variables

- `glabels` : les noms des observations (les groupes à comparer)

- `main` : titre principal

- `couleur` : les couleurs des lignes de profil

- `relative` : FALSE le cercle extérieur représente 100% ; TRUE le cercle extérieur est adapté au plus grand pourcentage observé

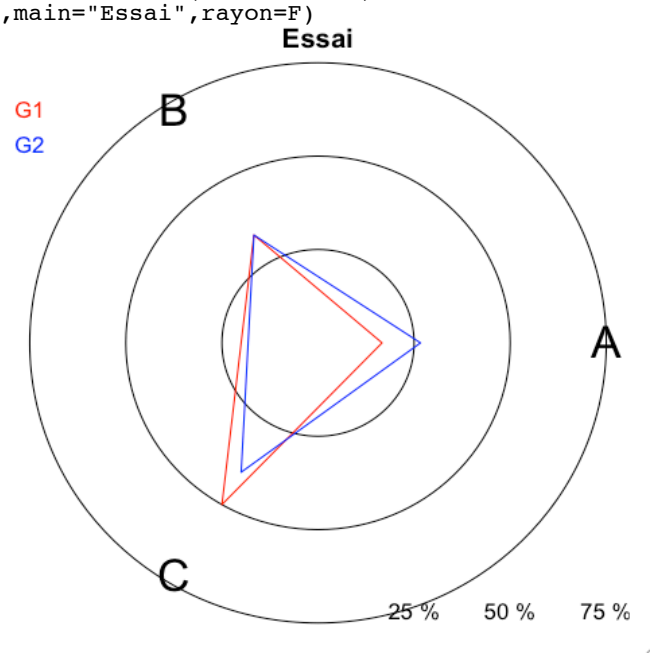
- `radar` : si TRUE en format « radar » ou « toile d'araignée »

- `rayon` : si TRUE, dessine les rayon de la toile

Exemple

```
> dfmm
  X1 X2 X3
1  1  2  3
2  4  5  6

> profil(dfmm,relative=T,main="Essai")
> profil(dfmm,relative=T,labels=c("A","B","C"),
glabels=c("G1","G2"),main="Essai",rayon=F)
```

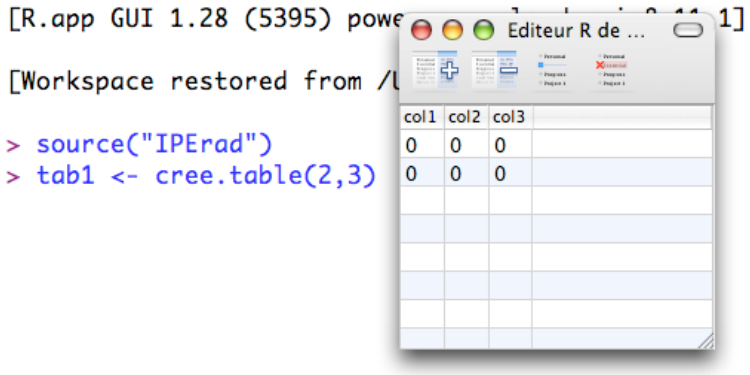


Utilitaires

* `cree.table(ligne,colonne)` permet de créer une table

Exemple

```
> tab <- cree.table(2,3)
> tab
```



Pour créer une table avec des noms de lignes et/ou colonnes, on introduit les noms à la place des valeurs, par exemple : `cree.table(c("G1","G2"), c("A","B","C"))` ou de façon mixe : `cree.table(c("G1","G2"), 3)`.

Pour éditer un tableau déjà créé, on utilisera `edit(tab)`.

* `pretty.table(tab,pct="row")` affiche une table avec les pourcentages par lignes (défaut), par colonnes (`pct="col"`) ou par rapport au total (`pct="total"`).

* `multinomial(n,decomp)` avec `decomp = c(n1,n2,…nk)` de somme égale à n (nk peut être omis). Cette fonction retourne la valeur : $n! / \prod n_i!$

* `as.boite(tab,plabel, rlabel, clabel)` fabrique une « boîte » à partir de la matrice `tab`. Cette structure est utilisée dans les plans factoriels (à faire depuis une table **R** de dimension 3).

`plabel`: étiquette des « plaques » ; `rlabel` : étiquettes des lignes ; `clabel` : étiquettes des colonnes. Attention, les labels ne doivent pas être des nombres.

Exemple

```
> tab26.K <- as.boite(tab26.K,c("S","M","I"),c("F","D"),c("NC","I","C"))
```

De façon interne, une boîte est un objet qui sépare les informations de structures (nombre d'éléments, étiquettes) des données. Une boîte correspond un tableau de dimension 3.

* `aff.boite(boite)` renvoie les données de la « boîte »

Statistique non-paramétrique

Test basé sur le S de Kendall

* S.`kendall(tab)` donne la valeur du S liée à la table 'tab'.

* `S.test(tab)` effectue un test basé sur le S. Elle utilise : `S.kendall(tab)` ; `prod.S(v,w)` ; `prod.S1(v,w)` ; `var.S(tab)` ; `correc.S(tab)`

Exemple

```
> tab
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    3    2    0
[3,]    4    0    1

> S.test(tab)
S = -39, corr = 1, Sc = -38
variance = 363.71, ecart-type = 19.071
z = -1.99, p-value = 0.023

> S.test(table(age,op2))
S = -110, corr = 1, Sc = -109
variance = 15014.59, ecart-type = 122.5
z = -0.89, p-value = 0.19
```

* `distrib.exacte.S(t,u)` donne la distribution exacte de S avec 't' somme de colonnes sous la forme c(t1, …,tn) et 'u' somme des lignes. Cette fonction utilise : `distrib0.S(ltab)` ; `multiplicite(tab)` ; `multinomial(n,decomp)` ; `cree.pattern(t,u)` ; `regroupe.tab(tab).`

* `distrib.exacte.S(tab)` donne la distribution exacte de S pour les tableaux qui ont mêmes totaux marginaux que `tab`.

Exemple

```
> distrib.exacte.S(c(3,3,2),c(3,2,3))

     [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7] [,8]
[1,] -19   -14   -11   -10    -9    -8    -7   -6
[2,]   3    24     9     18    10    18    36   18

     [,9]  [,10] [,11] [,12] [,13] [,14] [,15]
[1,]  -4    -3    -1     0     1     3     4
[2,]  42    45     3   108     3    45    42

     [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23]
[1,]   6     7     8     9    10    11    14    19
[2,]  18    36    18    10    18     9    24     3
```

* `S.ftest(boite,plaque2=null)` effectue un test pour une plan factoriel (2 x n). Avec le paramètre 'boite' (liste de deux matrices) ou les deux matrices.

Exemple avec les données de la table II.24 (du chapitre II.6)

```
> S.ftest(tab24.fi,tab24.ga)

Analyse de l'interaction avec delta
***********************************
D1 =  -0.292  D2 =  0.00833
delta =  0.301  v =  0.0822
z =  1.05 p =  0.147

> S.ftest(tab24.D,tab24.F)

Analyse de l'interaction avec delta
***********************************
D1 =  -0.385  D2 =  -0.1
delta =  0.285  v =  0.0829
z =  0.989 p =  0.161
```

Test basé sur le K de Kruskal

* `K.kruskal(tab)` retourne le vecteur contenant nombre de lignes, nombre de colonnes, valeur de K et facteur correctif associés à '`tab`'.

* `K.test(tab)` effectue le test de Kruskal-Wallis. Elle utilise : `K.kruskal(tab)` qui retourne `c(nc,nl,K,Tc)`

* `K.ftest(boite,np=NULL,nl=NULL,nc=ncol(boite))` effectue un test pour une plan factoriel (analyse de variance non paramétrique). Avec le paramètre 'boite ', une boîte ou une matrice. Dans ce dernier cas, il faut suppléer les paramètres np, nl, nc (respectivement nombre de plaques, de lignes et de colonnes ou listes d'étiquettes).

Exemple

Les données sont reprises du chapitre II.8.

```
> K.ftest(tab26.K)
Analyse de variance non parametrique
************************************
T = 0.88  (facteur correctif)
Effet total
***********
Ktotal = 25.13  ( 22.00 ), df = 5, p-value = 0.00013
Effet  S M I
*************
Kp = 18.94  ( 16.58 ), df = 2, p-value = 7.7e-05
Effet  F D
************
Kr = 1.447  ( 1.267 ), df = 1, p-value = 0.23
Effet d'interaction
*******************
Kint = 4.739 , df = 2, p-value = 0.094
```

Statistiques basées sur le L de Meddis

* `L.meddis(tab,lambdas)` retourne le vecteur contenant L, moyenne, variance et facteur correctif associés à 'tab'.

Exemple

```
> L.meddis(tab21.L,c(1,2,3))
[1] 138 132  88   1
```

* `L.test(tab,lambdas)` effectue un test basé sur le coefficient L.

Exemple

```
> L.test(tab21.L,c(1,2,3))
T =  1  L =  138 , mL =  132 ,vL =  88
sc =  9.3808 , z =  0.64 , p-value =  0.26
```

La même fonction permet de traiter les plans factoriels en jouant sur les coefficients lambda.

Exemple

```
> L.test(tab25.L,c(1,-1,1,-1))
L =  -103 , mL =  -183 , vL =  18117 ,
sc  =  125.7137 , z =  0.6363668 , p-value =  0.26

> L.test(tab25.L,c(1,1,-1,-1))
L =  331 , mL =  122 , vL =  18218.67
sc  =  126.0659 , z =  1.657863 , p-value =  0.049

> L.test(tab25.L,c(1,-1,-1,1) )
L =  -446 , mL =  -305 , vL =  17791.67
sc  =  124.5798 , z =  1.131805 , p-value =  0.13
```

Statistiques basées sur le chi2

* `Chi2.test(tab)` effectue le test du chi2. Elle utilise `Chi2(tab)` qui retourne `c(nl, nc, chi2)`. La fonction R de base, chisq.test, est plus complète.

Statistiques pour échantillons appareillés

* `z.test(tab,av,ap,a,pct=TRUE)` effectue un test sur une progression de 'av' à 'ap' avec les probabilités de progressions données par 'tab' en pourcent ou non. Elle passe par une approximation normale et utilise `distrib.Z(tab,av,a,pct)`

Exemple

Les données sont reprises du chapitre II.4.

La probabilité d'évolution « naturelle » est donnée dans deux cas :

```
> prog1
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    0
> prog2
     [,1]  [,2]  [,3]
[1,]  0.5 0.333 0.1667
[2,]  0.5 0.500 0.0000
```

`prog1` indique que les progressions de 0, 1 ou 2 pour un individu de niveau 1 ont le même poids (1e ligne). Il en va de même pour un individu de niveau 2 (mais il ne peut progresser de 2 niveaux).

`prog2` indique les probabilités de progressions.

Probabilité de l'évaluation constatée :

```
> Z.test(prog1,c(7,11,4),c(1,7,14),2,pct=F)
moy:  12.5 ; ecart-type:  2.723356
diff:  16 ; z:  1.285179 ; p-value:  0.09936485

> Z.test(prog2,c(7,11,4),c(1,7,14),2)
moy:  10.16667 ; ecart-type:  2.576604
diff:  16 ; z:  2.263962 ; p-value:  0.01178823
```

* `distrib.exacte.Z(v,a)` donne la distribution exacte des progressions possibles d'ampant 'a' à partir de la répartition 'v'. Elle utilise : `distrib0.Z(tab,a)` (la ligne i de 'tab' contient la place du sujet i) ; `distrib1.Z(tab,a)` ; `enumLM(v)` (passage d'une énumération à une forme matricielle) ; `somme.rang(tab)`.

Calcul des effets

* `effet(tab,coef,dim=1)` calcul de l'effet. Les coefficients (coef) peuvent être "gamma", "delta" (ligne, colonne ou symétrique pour 'dim' valant respectivement 1, 2 ou 3), "Phi", "C", "tau.b", "tau.c".

Pour chaque coefficient 'coef', la fonction utilise la fonction IPE.<coef> .

Fonctions statistiques diverses

* `cree.pattern(t,u)` : crée tous les tableaux possibles avec 'u' étant somme des colonnes et 't' la somme des lignes.

* `cree.pattern2(t,m)` : crée tous les tableaux possibles de m colonnes avec 't' valant la somme des lignes.

* `moy.tab(tab)` : crée le tableau avec les fréquences proportionnelles aux totaux lignes et colonnes de 'tab' (tableau de contingence).

* `valeurs.chi()` : utiliser pour des tests.

* `cronbach(tab)` : calcul du coefficient de Cronbach

* `pct.table(tab,pct="row")` : calcul des pourcentages de `tab` en ligne, colonne ou selon total

Calcul de l'entropie

`H(v,relative=F)` : entropie de Shannon, avec `v = c(`n_1`, `n_2`, ..., `n_K`)`. La valeur relative est obtenue en faisant le rapport à la valeur maximale.

`HP(v,relative=F)` : entropie selon la formule de Shannon avec utilisation du logarithme naturelle (valeur de l'entropie de Shannon divisée par ln 2). La valeur relative est obtenue en faisant le rapport à la valeur maximale.

`SB(v)` : entropie calculée selon la formule de Boltzmann (notée H_P dans le texte.

`HH(v,exact=T)` : entropie selon la qualité calculée de manière exacte ou approchée.

`HV(x,v)` : entropie sur l'utilité.

Représentation de graphes

Nécessite la librairie `igraph` et accessoirement la librairie `tcltk` .

* `core(adj,arr=0)` : donne la cardinalité du « core » lié à chaque sommets du graphe donné par la matrice d'adjacence adj . (utilise mcore).

* `fermeture(adj,thres=0,arr=0)` : fermeture transitive du graphe selon la technique proposée par Pochon & Favre (2009).

* `centralite(adj,thres=0,ferm=F,arr=0)` : donne les coefficients de centralité entrante et sortante pour chaque sommet du graphe de matrice adjacente `adj`. `ferm` indique si la fermeture a déjà été effectuée (par défaut une fermeture transitive est effectuée). Les autres paramètres sont destinés à la fermeture.

* `gr <- graph.adjacency(adj,…)` : crée un graphe à partir d'une matrice d'adjacence (`igraph`) que l'on peut afficher avec plot.

* `tklplot(gr)` : affiche un graphe dans un format qu'il est permis de modifier (librairie `tcltk` après avoir précisé le *device* `X11()`)

Exemple

```
> load("exGraph.dat")
> library(igraph)
> gr20 <- graph.adjacency(mat20,mode="dir")
> plot(gr20)

> centralite(mat20,ferm=F)
> centralite(mat20)
> core(mat20)
> library(tcltk)
> X11()
> tkplot(gr20)
```