Mini tutoriel R pour l'analyse de données

Jean-Marc Lasgouttes — Inria Paris
Jean-Marc.Lasgouttes@insa-rouen.fr

http://ana-donnees.lasgouttes.net/

Partie I. R vite fait

Qu'est-ce que R ?

Une copie de S S est un logiciel de statistiques développé aux Bell Labs ; R est une implémentation libre distribuée sous la licence GNU GPL.

Un logiciel populaire beaucoup de fonctionnalités ont été développées par la communauté R ; c'est sa grande force.

Un logiciel portable R existe sous Windows, macOS et Linux.

Un logiciel libre Le code source de R et des binaires sont disponibles sur https://cran.org/

CRAN ? C'est le site qui regroupe tous les *packages* R développés par la communauté

Installer R

Les binaires sont là : https://pbil.univ-lyon1.fr/CRAN/

Windows choisir de télécharger « base »

macOS Deux versions, la première pour les ordinateurs avec puce M1 ou M2, la seconde pour les processeurs Intel.

Ubuntu on installe le package `r-base`

```
sudo apt install r-base
```

IDE RStudio Environnement de travail
- console R pour taper les commandes
- panneaux pour visualiser les variables, les tables, les graphiques
- possibilité de faire des *notebooks* en langage markdown combinées à du code R.

Lire des données

Commande de base

```
read.table(file, header = FALSE, sep = "",
    quote = "\"'", dec = ".", numerals = c(...),
    row.names, col.names,
    as.is = !stringsAsFactors, na.strings = "NA",
    colClasses = NA, nrows = -1, skip = 0,
    check.names = TRUE, fill = !blank.lines.skip,
    strip.white = FALSE, blank.lines.skip = TRUE,
    comment.char = "#",
    allowEscapes = FALSE, flush = FALSE,
    stringsAsFactors = FALSE,
    fileEncoding = "", encoding = "unknown", text,
    skipNul = FALSE)
```

Comment lire ça ?

- les paramètres peuvent avoir une valeur par défaut (`FALSE` pour `header`)
  Seules les quelques premières sont obligatoires
- si on ne donne pas le nom, ce sera le premier, le second...
- on peut abréger le nom si ce n'est pas ambigu (`head` pour `header`, par ex.)
- en général, on ne donne que quelques paramètres

Lire des données (suite)

Commande de base version courte

```
read.table(file, header = FALSE, sep = "",
           quote = "\"'", ...)
```

Explications afficher l'aide avec la commande `?read.table` ou `help(read.table)`

Et le résultat ? uniquement retour de fonction (souvent un objet complexe)

On peut afficher l'objet avec la commande `print(obj)` (ou juste `obj`) ou alors `summary(obj)`

Exemple parfois aussi simple que

```
sympa <- read.table("sympa.txt")
```

« <- » ? ? C'est l'opérateur d'affectation (on peut aussi utiliser `=`)

Autre type de fichiers

- comma separated variables (csv) : `read.csv()`, `read.csv2()`
- Libre Office Calc avec `read_ods()` du package `readODS`

Manipuler une table

Comme un tableau

- `sympa[3,2]` donne la case de troisième ligne et seconde colonne
- `sympa["PAYS",]` donne la ligne correspondant à `PAYS`
- `sympa[1,]`, pareil (`PAYS` est la première ligne)
- `sympa[1:3,2:4]` donne le bloc composé des lignes 1 à 3 et des colonnes 2 à 4

Comme un enregistrement en utilisant `$` pour accéder à un champ

— `sympa$GENE` donne la colonne correspondant à la colonne `GENE` (sans les noms de ligne/colonnes)

Type de données connues de R

— réels, entiers, booléens (`TRUE` ou `FALSE`), caractères (chaîne) ou facteurs (valeurs qualitatives)
— vecteur de ces variables : `c(3,2, 1.5, FALSE)`
— matrice à deux dimensions

```
matrix(c(1,2,3, 11,12,13),
       nrow=2, ncol=3, byrow=TRUE)
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 11 & 12 & 13 \end{pmatrix}$$

Les classes en R

Principe L'objet `sympa` est de classe `data.frame`. Pour toute fonction dont le premier argument est `sympa`, R execute si possible *`fonction`*`.data.frame`.

Example deux fonctions `print`

La fonction spécialisée

```
> print(sympa)
      SERI GENE GAI HONN INTL SERV COUR COMP DISC TOTAL
PAYS    20    9   9   27   10   16   20    4    8   123
OUVR    42   10  22   51   18   28   38   12   22   243
VEND    11    2   5   14    8    7    5    8    6    66
COMM     8    9  12   23   14   16   14   12   12   120
EMPL    19   10  16   52   32   25   22   25   30   231
TECH    10    5  12   23   20   13   11   13   10   117
UNIV     2    8   7    6   15    6    6    9    4    63
LIBE     8   42  23   24   46   22   22   34   16   237
TOTAL  120   95 106  220  163  133  138  117  108  1200
```

La fonction utilisée est `print.data.frame`

La fonction par défaut

```
> print.default(sympa)
$SERI
[1]  20  42  11   8  19  10   2   8 120
$GENE
[1]  9 10  2  9 10  5  8 42 95
$GAI
[1]   9  22   5  12  16  12   7  23 106
$HONN
[1]  27  51  14  23  52  23   6  24 220
$INTL
[1]  10  18   8  14  32  20  15  46 163
$SERV
[1]  16  28   7  16  25  13   6  22 133
$COUR
[1]  20  38   5  14  22  11   6  22 138
$COMP
[1]   4  12   8  12  25  13   9  34 117
$DISC
[1]   8  22   6  12  30  10   4  16 108
$TOTAL
[1]  123  243   66  120  231  117   63  237 1200
attr(,"class")
[1] "data.frame"
```

Les noms sont obtenus avec

```
> names(sympa)
 [1] "SERI"  "GENE"  "GAI"   "HONN"  "INTL"  "SERV"  "COUR"
 [8] "COMP"  "DISC"  "TOTAL"
> rownames(sympa)
[1] "PAYS"  "OUVR"  "VEND"  "COMM"  "EMPL"  "TECH"  "UNIV"
[8] "LIBE"  "TOTAL"
```

Partie II. Analyse de données en R

Les outils disponibles

Dans ce cours : `ade4`

— disponible sur CRAN
— développé à l'université Lyon I

Autres possibilités

— livré avec R : fonction `prcomp` pour l'ACP ou `corresp` (du package MASS) pour l'AFC
  Assez basique
— package `FactoMineR` : peut-être plus moderne

Utiliser ade4

Installer le package à faire une seule fois

```
install.packages("ade4")
```

ou alors dans RStudio le menu « Tools ▷ Install Package ».

Charger le package là, plus besoin de guillemets

```
require(ade4)
```

Concept central de ade4 Duality diagram (DUDI)

Faire une AFC

AFC COrrespondence Analysis

```
dudi.coa(df, scannf = TRUE, nf = 2)
```

Paramètres

— `df` : les données comme retournées par `read.table` (data frame)
— `scannf` : si vrai (`TRUE` ou `T`), affiche un graphique des valeurs propres et demande le nombre d'axes à conserver
— `nf` : quand `scannf=F`, contient le nombre d'axes à conserver

Exemple typique

```
coa1 <- dudi.coa(mesdonnees, scannf = F, nf = 4)
```

L'objet `coa1` a, entre autres champs

— `coa1$eig` : valeurs propres
— `coa1$co`, `coa1$li` : coordonnées des colonnes et des lignes
— `coa1$cw`, `coa1$lw` : poids des colonnes et des lignes

Faire une ACP

ACP Principal Components Analysis

```
dudi.pca(df, row.w = rep(1, nrow(df))/nrow(df),
    col.w = rep(1, ncol(df)),
    center = TRUE, scale = TRUE,
    scannf = TRUE, nf = 2)
```

Paramètres comme l'AFC, avec en plus
- `row.w`, `col.w` : les poids des individus (mais aussi des variables !)
- `center`, `scale` : permet de faire une ACP non centrée et/ou non réduite

Exemple typique

```
pca1 <- dudi.pca(mesdonnees, scannf = F, nf = 4)
```

L'objet `pca1` a les mêmes champs que pour l'AFC, plus, mais `co` représente les corrélations des variables aux axes.

Faire une ACM

ACM c'est comme l'AFC, mais on peut donner des poids aux individus

```
dudi.acm(df, row.w = rep(1, nrow(df))/nrow(df),
    scannf = TRUE, nf = 2)
```

Le tableau de Burt On l'a directement avec

```
acm.burt (df1, df2, counts = rep(1, nrow(df1)))
```

Les variables supplémentaires qualitatives On peut utiliser la petite fonction `acm.suppl` qui est dans dans `fonctions.R`

```
acm.suppl(acm, suppl)
```

où `acm` est un objet retourné par `dudi.acm` et `suppl` est le tableau de variables supplémentaires.

L'objet retourné contient les champs
- `li` : coordonnées des variables supplémentaires
- `eff` : effectifs des catégories supplémentaires
- `eff.tot` : effectif total
- `test` : valeurs test des catégories supplémentaires

Des graphiques !

Les valeurs propres

```
barplot(coa1$eig)
```

La projection des individus on affiche les modalités de colonne du premier plan avec

```
s.label(coa1$co)
```

et on ajoute les modalités de ligne avec

```
s.label(coa1$li, add.plot=T)
```

Paramètres `s.label(x, xax=1, yax=2, add.plot=F, ...)`
- `x` : la table qu'on veut représenter (`$li` ou `$co`)
- `xax`, `yax` : indice de l'axe principal en abscisse, ordonnée
- `add.plot=T` indique que les nouveaux points seront sur le graphe préexistant

Le cercle des corrélations pour l'ACP seulement, avec `s.corcircle(pca1$co, xax=1, yax=2, ...)`

Données liées à l'inertie

Commande de base calcul de l'inertie totale et autres

```
inert1 <- inertia(x, row.inertia = FALSE,
                  col.inertia = FALSE)
```

- `x` : objet retourné par `dudi.coa`
- `row.inertia` : calculs détaillés par ligne si `TRUE`
- `col.inertia` : calculs détaillés par colonne si `FALSE`

Valeur retournée objet avec les champs
- `tot.inertia` : répartition de l'inertie entre les axes
- `row.contrib` : contribution des lignes aux axes
- `row.rel` : qualité de représentation des lignes par les axes
- `row.cum` : qualité de représentation cumulée des lignes par les sous-espaces
- et les mêmes avec `col.xxx` pour les colonnes