

Manipuler des données volumineuses : les génomés bactériens

J.R. Lobry & D. Chessel

Pour tester l'aptitude de R à manipuler de grands ensembles de données, la fiche propose d'implanter un tableau séquences - codons de taille respectable.

Table des matières

1	Introduction	2
2	Les données	3
3	Importer un grand tableau	4
4	Sauver et restaurer un grand tableau	5
5	Taille mémoire	6
6	Tableaux dérivés	6
6.1	Créer le tableau espèces-codons	7
6.2	Créer le tableau séquences-acides aminés	7
6.3	Créer le tableau espèces-acides aminés	8
6.4	Créer le tableau séquences-bases	8
6.5	Créer le tableau espèces-bases	9
7	Exercices	9
	Références	10


```

21 .....W.....M.....N...SS.....
9 .....W.....N...SS.....
14 .....Y...W.....N...SS.....
13 .....W.....M.....GG.....
16 .....L.....
22 .....?...L.....
23 .....?.....
15 .....Q.....
6 .....QQ.....
12 .....S.....





```

1. The Standard Code. 4. The Mold, Protozoan, and Coelenterate Mitochondrial Code and the Mycoplasma/Spiroplasma Code.
10. The Euplotid Nuclear Code. 2. The Vertebrate Mitochondrial Code. 3. The Yeast Mitochondrial Code.
5. The Invertebrate Mitochondrial Code. 21. Trematode Mitochondrial Code. 9. The Echinoderm Mitochondrial Code.
14. The Flatworm Mitochondrial Code. 13. The Ascidian Mitochondrial Code. 16. Chlorophycean Mitochondrial Code.
22. Scenedesmus obliquus mitochondrial Code. 23. Thraustochytrium Mitochondrial Code. 15. Blepharisma Nuclear Code.
6. The Ciliate, Dasycladacean and Hexamita Nuclear Code. 12. The Alternative Yeast Nuclear Code.
(source : <http://www3.ncbi.nlm.nih.gov/htbin-post/Taxonomy/wprintgc?mode=c>).

La plupart des codes variants sont trouvés dans les génomes mitochondriaux. Chez les bactéries, seul le code standard est utilisé à l'exception des Mycoplasmes et Spiroplasmes qui utilisent le plus courant des codes variants (numéro 4) où le codon stop standard TGA code pour le tryptophane (W). Nous ferons l'impasse sur cette particularité.

2 Les données

Elles sont dans quatre fichiers à récupérer dans <http://pbil.univ-lyon1.fr/R/donnees>.

	fracod.txt	10-Nov-01 16:00	8.9M
	fracod aa.txt	12-Nov-01 10:57	1k
	fracod cod.txt	12-Nov-01 11:14	1k
	fracod esp.txt	12-Nov-01 10:58	530k

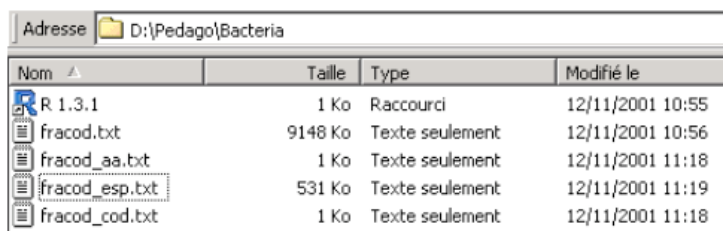
Récupérer ces quatre fichiers dans un nouveau dossier de travail :

```

path <- "http://pbil.univ-lyon1.fr/R/donnees"
telecharger <- function(quoi) {
  download.file(url = paste(path, quoi, sep = "/"), destfile = paste(getwd(),
    quoi, sep = .Platform$file.sep))
}
telecharger("fracod.txt")
telecharger("fracod_aa.txt")
telecharger("fracod_cod.txt")
telecharger("fracod_esp.txt")

```

Il se peut que ce code ne marche pas si vous passez par un proxy pour votre connexion internet. Vous pouvez essayer de neutraliser le proxy avec `Sys.putenv(no_proxy = "")`, ou télécharger les fichiers à la main dans votre espace de travail. Vous devez avoir les quatre fichiers dans votre espace de travail (donné par `getwd()`) pour pouvoir continuer cette fiche de TD :



Nom	Taille	Type	Modifié le
R 1.3.1	1 Ko	Raccourci	12/11/2001 10:55
fracod.txt	9148 Ko	Texte seulement	12/11/2001 10:56
fracod_aa.txt	1 Ko	Texte seulement	12/11/2001 11:18
fracod_esp.txt	531 Ko	Texte seulement	12/11/2001 11:19
fracod_cod.txt	1 Ko	Texte seulement	12/11/2001 11:18

Lire directement `fracod_cod.txt` comme vecteur de chaînes de caractères :

```
codcol <- readLines("fracod_cod.txt")
codcol
[1] "TTT" "TTC" "TTA" "TTG" "TCT" "TCC" "TCA" "TCG" "TAT" "TAC" "TAA" "TAG" "TGT"
[14] "TGC" "TGA" "TGG" "CTT" "CTC" "CTA" "CTG" "CCT" "CCC" "CCA" "CCG" "CAT" "CAC"
[27] "CAA" "CAG" "CGT" "CGC" "CGA" "CGG" "ATT" "ATC" "ATA" "ATG" "ACT" "ACC" "ACA"
[40] "ACG" "AAT" "AAC" "AAA" "AAG" "AGT" "AGC" "AGA" "AGG" "GTT" "GTC" "GTA" "GTG"
[53] "GCT" "GCC" "GCA" "GCG" "GAT" "GAC" "GAA" "GAG" "GGT" "GGC" "GGA" "GGG"
```

On a les 64 codons. Faire avec `fracod_esp.txt` un facteur appelé `codlig` :

```
codlig <- as.factor(readLines("fracod_esp.txt"))
summary(codlig)
AERPECG AQUAECG ARCFUCG BACHDCG BACSUCG BORBUGG BUCAICG CAMJECG CHLMUCG
2619 1489 2088 3558 3627 772 523 1497 743
CHLPNACG CHLPNCCG CHLPNJCG CHLTRCG DEIRAC1 DEIRAC2 DEIRACP1 DEIRAMP1 ECOLICG
853 968 982 832 2423 346 35 123 3914
HAEINCG HALSPCG HALSPP1 HALSPP2 HELPJCG HELPYCG METJACG METHCG MYCGECG
1505 1761 163 291 1372 1392 1516 1646 451
MYCPNCG MYCTUCG NEIMACG NMENCG PSEAECG PYRABCG PYRHOCG RICPRCG SYNY3CG
657 3679 1753 1706 5255 1692 1973 773 2908
THEACCG THEMACG TREPACG UREURCG VIBCHC1 VIBCHC2 XYLFCAG
1388 1686 917 560 2365 870 2041
```

Les codes des espèces sont les suivants :

```
AERPECG Aeropyrum pernix K1 complete genome.
AQUAECG Aquifex aeolicus complete genome.
ARCFUCG Archaeoglobus fulgidus complete genome.
BACHDCG Bacillus halodurans C-125, complete genome.
BACSUCG Bacillus subtilis complete genome.
BORBUGG Borrelia burgdorferi complete genome.
BUCAICG Buchnera sp. APS complete genome.
CAMJECG Campylobacter jejuni complete genome.
CHLMUCG Chlamydia muridarum complete genome.
CHLPNACG Chlamydia pneumoniae AR39 complete genome.
CHLPNCCG Chlamydia pneumoniae CWL029 complete genome.
CHLPNJCG Chlamydia pneumoniae J138 complete genome.
CHLTRCG Chlamydia trachomatis complete genome.
DEIRAC1 Deinococcus radiodurans R1 complete chromosome 1.
DEIRAC2 Deinococcus radiodurans R1 complete chromosome 2.
DEIRACP1 Deinococcus radiodurans Plasmid CP1.
DEIRAMP1 Deinococcus radiodurans Plasmid MP1.
ECOLICG Escherichia coli K-12 MG1655.
HAEINCG Haemophilus influenzae Rd complete genome.
HALSPCG Halobacterium sp. NRC-1 complete genome.
HALSPP1 Halobacterium sp. NRC-1 plasmid pNRC100, complete sequence.
HALSPP2 Halobacterium sp. NRC-1 plasmid pNRC200 complete genome.
HELPCG Helicobacter pylori, strain J99 complete genome.
HELPCG Helicobacter pylori 26695.
METJACG Methanococcus jannaschii complete genome.
METHCG Genome of Methanobacterium thermoautotrophicum delta H.
MYCGECG Mycoplasma genitalium complete genome.
MYCPNCG Mycoplasma pneumoniae complete genome.
MYCTUCG Mycobacterium tuberculosis H37Rv complete genome.
NEIMACG Neisseria meningitidis serogroup A strain Z2491 complete genome.
NMENCG Neisseria meningitidis serogroup B strain MC58 complete genome.
PSEAECG Pseudomonas aeruginosa PA01, complete genome.
PYRABCG Pyrococcus abyssii complete genome.
PYRHOCG Pyrococcus horikoshii OT3 complete genome.
RICPRCG Rickettsia prowazekii strain Madrid E, complete genome.
SYNY3CG Synechocystis PC6803 complete genome.
THEACCG Thermoplasma acidophilum, complete genome.
THEMACG Thermotoga maritima complete genome.
TREPACG Treponema pallidum complete genome.
UREURCG Ureaplasma urealyticum complete genome.
VIBCHC1 Vibrio cholerae chromosome I, complete chromosome.
VIBCHC2 Vibrio cholerae chromosome II, complete chromosome.
XYLFCAG Xylella fastidiosa, complete genome.
```

On pourra consulter le lien utile pour en savoir plus sur ces génomes :

<http://www.tigr.org/tdb/mdb/mdbcomplete.html>

3 Importer un grand tableau

Il faut savoir que la fonction `read.table()`, qui est très pratique pour les petits jeux de données, n'est pas adaptée aux jeux de données de taille plus conséquente. En effet, `read.table()` fait appel à la fonction `scan()` pour lire les données puis fait une analyse de ce qui a été lu pour essayer de déterminer

au mieux les types des colonnes du `data.frame` qui va être construit. Cette opération d'analyse demande beaucoup de temps calcul. Quand on sait ce que l'on veut importer, il est préférable d'utiliser directement la fonction `scan()`. Cette dernière renvoie un vecteur que l'on transforme en matrice puis en `data.frame`.

```
tempsimport <- system.time(fracod <- as.data.frame(matrix(data = scan(file = "fracod.txt",
  what = integer(0)), nrow = 67712, ncol = 64, byrow = TRUE)))
tempsimport
  user system elapsed
1.659  0.268  1.962
```

Ainsi, pour importer un tableau de 67,712 lignes et 64 colonnes, soit 4,333,568 éléments en tout, nous avons consommé ici 1.66 secondes de CPU utilisateur (ce qui correspond dans la vie réelle à une attente de 1.96 secondes lors de la dernière compilation de ce document). A titre indicatif, une importation du même tableau dans les mêmes conditions avec `fracod <- read.table("fracod.txt")` demande environ 92.33 secondes de CPU utilisateur et 237.97 secondes d'attente réelle, soit approximativement 4 minutes. Il est donc clair que `scan()` est beaucoup plus rapide que `read.table()`, mais on peut encore faire mieux.

4 Sauver et restaurer un grand tableau

L'idée est qu'après avoir importé des données sous R, on aime bien en général les "polir". Après cette phase de "polissage" il suffit de les sauvegarder comme des objets R (en binaire compatible multi-ordinateur) pour pouvoir ultérieurement les recharger très rapidement. Par exemple, dans notre cas l'objet `fracod` est un `data.frame` dont le nom des colonnes n'est pas très explicite :

```
names(fracod)
 [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9" "V10" "V11" "V12" "V13"
 [14] "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24" "V25" "V26"
 [27] "V27" "V28" "V29" "V30" "V31" "V32" "V33" "V34" "V35" "V36" "V37" "V38" "V39"
 [40] "V40" "V41" "V42" "V43" "V44" "V45" "V46" "V47" "V48" "V49" "V50" "V51" "V52"
 [53] "V53" "V54" "V55" "V56" "V57" "V58" "V59" "V60" "V61" "V62" "V63" "V64"
```

On décide donc de mieux documenter l'objet :

```
names(fracod) <- readLines("fracod_cod.txt")
names(fracod)
 [1] "TTT" "TTC" "TTA" "TTG" "TCT" "TCC" "TCA" "TCG" "TAT" "TAC" "TAA" "TAG" "TGT"
 [14] "TGC" "TGA" "TGG" "CTT" "CTC" "CTA" "CTG" "CCT" "CCC" "CCA" "CCG" "CAT" "CAC"
 [27] "CAA" "CAG" "CGT" "CGC" "CGA" "CGG" "ATT" "ATC" "ATA" "ATG" "ACT" "ACC" "ACA"
 [40] "ACG" "AAT" "AAC" "AAA" "AAG" "AGT" "AGC" "AGA" "AGG" "GTT" "GTC" "GTA" "GTG"
 [53] "GCT" "GCC" "GCA" "GCG" "GAT" "GAC" "GAA" "GAG" "GGT" "GGC" "GGA" "GGG"
```

Maintenant que nous sommes satisfaits, nous sauvegardons l'objet `fracod` en binaire au format XDR (c'est un format utilisé par toutes les variétés de R, on peut sauvegarder sous Linux et relire sous Mac ou PC, c'est rapide et portable) :

```
save(fracod, file = "fracod.RData")
```

Vous pouvez maintenant quitter votre session R, puis relancer R pour recharger le jeu de données (n'oubliez pas de sélectionner le bon répertoire de travail avant d'exécuter le code suivant, sinon le fichier ne sera pas trouvé) :

```
tempsXRD <- system.time(load("fracod.Rdata"))
tempsXRD
```

```
user system elapsed
0.626 0.085 0.713
```

Ainsi, pour restaurer un tableau de 67,712 lignes et 64 colonnes, soit 4,333,568 éléments en tout, correspondant à 22,619,749 observations, nous avons consommé ici 0.63 secondes de CPU utilisateur (ce qui correspond dans la vie réelle à une attente de 0.71 secondes lors de la dernière compilation de ce document). R est donc bien capable de manipuler de grands ensembles de données, cependant un couplage avec un SGBD sera préférable pour extraire les informations pertinentes à la volée pour des applications plus conséquentes (il y a des packages R pour cela).

5 Taille mémoire

Tous les objets R sont placés dans la mémoire vive de l'ordinateur. Une estimation de la place occupée par un objet est donnée par la fonction `object.size()` :

```
taillefracod <- object.size(fracod)
taillefracod
34945216 bytes
```

Ainsi, notre objet `fracod` occupe environ 34,945,216 octets en mémoire (soit 33.33 Mio). La taille mémoire maximum disponible dépend de votre machine et de son système d'exploitation. Si vous dépassez cette capacité vous aurez un message du type :

```
Error: cannot allocate vector of size 33856 Kb
In addition: Warning message:
Reached total allocation of 476Mb: see help(memory.size)
```

6 Tableaux dérivés

Faire avec `fracod_aa.txt` un facteur appelé `codaa` :

```
codaa <- factor(readLines("fracod_aa.txt"))
codaa
[1] Phe Phe Leu Leu Ser Ser Ser Ser Tyr Tyr Stp Stp Cys Cys Stp Trp Leu Leu Leu
[21] Pro Pro Pro Pro His His Gln Gln Arg Arg Arg Arg Ile Ile Ile Met Thr Thr Thr
[41] Asn Asn Lys Lys Ser Ser Arg Arg Val Val Val Val Ala Ala Ala Ala Asp Asp Glu Glu
[61] Gly Gly Gly Gly
21 Levels: Ala Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser ... Val
```

Réorganiser le tableau pour que les codons apparaissent par blocs d'acides aminés et vérifier.

```
frasort <- fracod[, order(codaa)]
codsort <- codaa[order(codaa)]
names(frasort)
[1] "GCT" "GCC" "GCA" "GCG" "CGT" "CGC" "CGA" "CGG" "AGA" "AGG" "AAT" "AAC" "GAT" "GAC"
[15] "TGT" "TGC" "CAA" "CAG" "GAA" "GAG" "GGT" "GGC" "GGA" "GGG" "CAT" "CAC" "ATT" "ATC"
[29] "ATA" "TTA" "TTG" "CTT" "CTC" "CTA" "CTG" "AAA" "AAG" "ATG" "TTT" "TTC" "CCT" "CCC"
[43] "CCA" "CCG" "TCT" "TCC" "TCA" "TCG" "AGT" "AGC" "TAA" "TAG" "TGA" "ACT" "ACC" "ACA"
[57] "ACG" "TGG" "TAT" "TAC" "GTT" "GTC" "GTA" "GTG"
dim(frasort)
[1] 67712 64
sum(frasort)
```

[1] 22619749

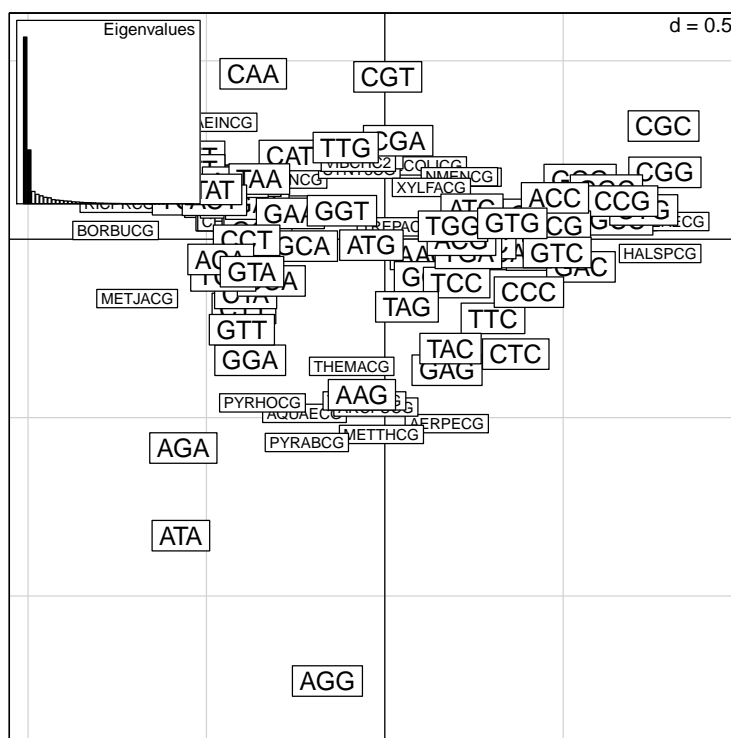
`codsor`

```
[1] Ala Ala Ala Ala Arg Arg Arg Arg Arg Arg Asn Asn Asp Asp Cys Cys Gln Gln Glu Glu Gly
[22] Gly Gly Gly His His Ile Ile Ile Leu Leu Leu Leu Leu Lys Lys Met Phe Phe Pro Pro
[43] Pro Pro Ser Ser Ser Ser Ser Ser Stp Stp Stp Thr Thr Thr Thr Trp Tyr Tyr Val Val Val
[64] Val
```

21 Levels: Ala Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser Stp ... Val

6.1 Créer le tableau espèces-codons

```
espcod <- aggregate(x = frasort, by = list(espece = codlig), FUN = sum)
rownames(espcod) <- espcod$espece
espcod <- espcod[, -1]
library(ade4)
scatter(dudi.coa(espcod, scann = FALSE, nf = 2))
```



6.2 Créer le tableau séquences-acides aminés

```
fbact1 <- function() {
  w <- data.frame(matrix(0, nrow(frasort), length(levels(codsor))))
  names(w) <- levels(codsor)
  row.names(w) <- row.names(frasort)
  for (i in 1:21) {
    a0 <- levels(codsor)[i]
    c0 <- which(codsor == a0)
    w[, i] <- apply(data.frame(frasort[, c0]), 1, sum)
  }
  return(w)
}
fraa <- fbact1()
frasort[1, ]
```



```

    } return(w)
  }
  w <- t(apply(frasort, 1, loc1))
  return(w)
}
frabase <- fbact2()
frabase[1, ]
  A C G T
198 124 150 254
sum(frabase)/3
[1] 22619749

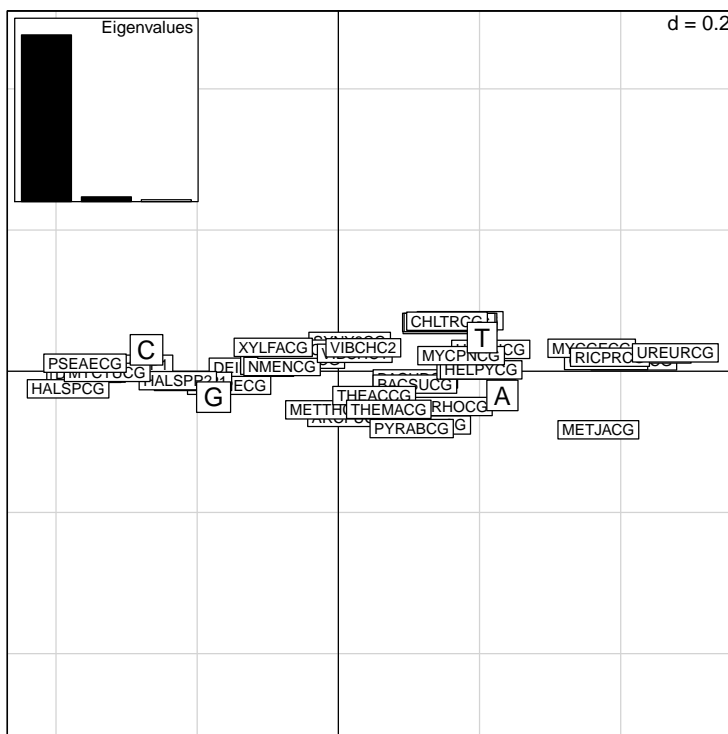
```

6.5 Créer le tableau espèces-bases

```

espbase <- aggregate(x = frabase, by = list(espece = codlig), FUN = sum)
rownames(espbase) <- espbase$espece
espbase <- espbase[, -1]
scatter(dudi.coa(espbase, scann = FALSE, nf = 2))

```



7 Exercices

Vérifiez sur ces données le résultat classique [1] selon lequel les organismes qui ont un fort taux de G+C utilisent préférentiellement des acides aminés codés par des codons riches en G+C. Pour cela, vous agrérez les données par espèces avant de représenter l'évolution des fréquences des acides-aminés en fonction du taux de G+C. Question subsidiaire : peut-on dire que le taux de G+C est le facteur majeur de variabilité inter-spécifique au niveau des codons, au niveau des acides-aminés ?

Références

- [1] N. Sueoka. Correlation between base composition of deoxyribonucleic acid and amino acid composition of protein. *Proc. Natl. Acad. Sci. USA*, 47 :1141–1149, 1961.