Fiche TD avec le logiciel R : tdr74

Analyse multivariée de séquences génomiques

D. Charif & Pr Jean R. Lobry

Le paquet `seqinr` pour le logiciel R est une bibliothèque de fonctions dédiée à l'importation et l'analyse des séquences génomiques. Il permet d'interfacer le logiciel R avec les banques de séquences structurées sous ACNUC [8]. ACNUC est un système de stockage des banques de séquences (EMBL, GenBank, SwissProt, etc.) efficace pour extraire des sous-séquences d'intérêt biologique (des séquences codantes, des ARNt, etc). Grâce à un langage de requête assez simple, il est alors facile depuis R, de sélectionner des séquences puis d'utiliser toute la puissance et les fonctionnalités du langage R pour les analyser.

Table des matières

1 Introduction

Au cours de ce TP, nous allons utiliser les paquets `seqinr` [3] et `ade4` [5] de R [21]. Il faut qu'ils aient été installés au préalable pour que vous n'ayez plus qu'à les invoquer avec :

```
library(seqinr)
library(ade4)
```

L'exemple utilisé est le même que celui analysé dans [14] et présente l'avantage d'avoir trois facteurs interprétables du point de vue biologique. Il permet d'illustrer des effets de type pression de mutation et de type pression de sélection, et ce aussi bien au niveau nucléique que des acides-aminés. Il est d'une taille raisonnable : 850 gènes $\times$ 64 codons. Si le temps vient à manquer en TP, les principaux résultats intermédiaires ont été sauvegardés et peuvent être restaurés pour gagner du temps. Par exemple, pour charger la table donnant l'usage du code sans avoir à la calculer :

```
load(url("http://pbil.univ-lyon1.fr/R/donnees/tdr74/tuco.rda"))
dim(tuco)
```
```
[1] 850  64
```

D'un point de vue méthodologique la structure particulière de ce jeu de données permet d'illustrer l'intérêt des méthodes de couplage pour aider à l'interprétation, mais cela sort du cadre de ce TP de niveau M1. Il y a néanmoins une section « pour aller plus loin » page 25 où elles sont abordées pour les curieux, mais elle peut être oubliée sans dommage en première lecture.

2 Effectuer des requêtes dans les banques

Sélectionner la banque à partir de laquelle vous souhaitez récupérer les séquences avec la fonction `choosebank()`. Cette fonction initialise l'accès à distance vers les banques ACNUC situées sur le serveur du PRABI (un acronyme pour « Pôle Rhône-Alpes de Bioinformatique »). Pour avoir la liste des banques disponibles, appeler la fonction `choosebank()` sans argument.

```
choosebank()
```

Par exemple, pour travailler avec genbank, appeler `choosebank()` avec l'argument `"genbankseqinr"` :

```
choosebank("genbankseqinr")
```

Puis, il va s'agir de composer une requête afin de sélectionner les séquences d'intérêt. La documentation en ligne de `query` explique en détail la façon de faire. Dans la requête qui suit, il s'agit de sélectionner toutes les séquences codantes (`t=cds`) du génome du chat (`sp=felis catus`) qui ne sont pas (`et no`) partielles (`k=partial`). Le résultat est stocké dans un objet appelé `list1`.

```
list1 <- query("list1", "sp=felis catus et t=cds et no k=partial")
```

Désormais, il y a un objet appelé `list1` dans l'environnement de travail qui ne contient pas les séquences elles-mêmes mais *les noms des séquences* qui répondent à la requête. Ils sont stockés dans la liste `req` de l'objet `list1`. Les trois premiers sont :

```
getName(list1)[1:3]
```
```
[1] "AB000483.PE1" "AB000484.PE1" "AB000485.PE1"
```

En pratique, les requêtes peuvent être effectuées en plusieurs pas. Il est possible de raffiner une requête en utilisant le résultat d'une précédente requête. Par exemple, à partir de `list1`, on va pouvoir récupérer uniquement les séquences qui ont été publiées en 2004 (`y=2004`).

```
list2 <- query("list2", "list1 ET y=2004")
list2$nelem
```
```
[1] 60
```

Il y a 60 séquences complètes qui on été publiées en 2004. Les séquences seront rapatriées grâce à la fonction `getSequence()`. Par exemple les 50 premiers nucléotides de la séquence 1 sont :

```
myseq <- getSequence(list1$req[[1]])
myseq[1:50]
```
```
 [1] "a" "t" "g" "a" "a" "t" "c" "a" "a" "g" "g" "a" "g" "c" "c" "g" "t" "t" "t" "t"
[21] "t" "a" "g" "g" "c" "a" "c" "c" "t" "g" "c" "t" "c" "c" "t" "g" "g" "t" "g" "c"
[41] "t" "g" "c" "a" "g" "c" "t" "g" "g" "t"
```

La couverture du manuel de `seqinr`

Nous ne détaillerons pas ici plus avant les possibilités du langage de requête. Pour les curieux se reporter au chapitre « the query language » du manuel de `seqinr` [1].

3 Manipulations de base des séquences génomiques

On travaillera ici avec la séquence codante du gène *malM* de la bactérie *Escherichia coli*. Cette séquence est distribuée en standard avec le paquet `seqinr` sous la forme classique d'un fichier au format fasta, vous pouvez donc importer les données dans R avec la fonction `read.fasta()` même si vous êtes hors ligne :

```
malM <- read.fasta(file = system.file("sequences/malM.fasta", package = "seqinr"))
myseq <- getSequence(malM)[[1]]
myseq[1:50]
```
```
 [1] "a" "t" "g" "a" "a" "a" "a" "t" "g" "a" "a" "t" "a" "a" "a" "a" "g" "t" "c" "t"
[21] "c" "a" "t" "c" "g" "t" "c" "c" "t" "c" "t" "g" "t" "t" "t" "a" "t" "c" "a" "g"
[41] "c" "a" "g" "g" "g" "t" "t" "a" "c" "t"
```

Donner le nombre de nucléotides du gène *malM* :

```
length(myseq)
```
```
[1] 921
```

Le gène *malM* code pour une protéine, on s'attend donc à ce que sa longueur soit un multiple de trois. Vérifier que c'est bien le cas avec l'opérateur modulo `%%` de R :

```
length(myseq) %% 3
```
```
[1] 0
```

1. `http://seqinr.r-forge.r-project.org/`

Donner la composition en bases du gène *malM* :

```
table(myseq)
```

```
myseq
  a   c   g   t
238 265 228 190
```

Calculer le taux de G+C du gène *malM* :

```
GC(myseq)
```

```
[1] 0.5352877
```

Calculer le taux de G+C en position 3 des codons du gène *malM* :

```
GC3(myseq)
```

```
[1] 0.5667752
```

Découper la séquence en codons :

```
splitseq(myseq)[1:30]
```

```
 [1] "atg" "aaa" "atg" "aat" "aaa" "agt" "ctc" "atc" "gtc" "ctc" "tgt" "tta" "tca"
[14] "gca" "ggg" "tta" "ctg" "gca" "agc" "gcg" "cct" "gga" "att" "agc" "ctt" "gcc"
[27] "gat" "gtt" "aac" "tac"
```

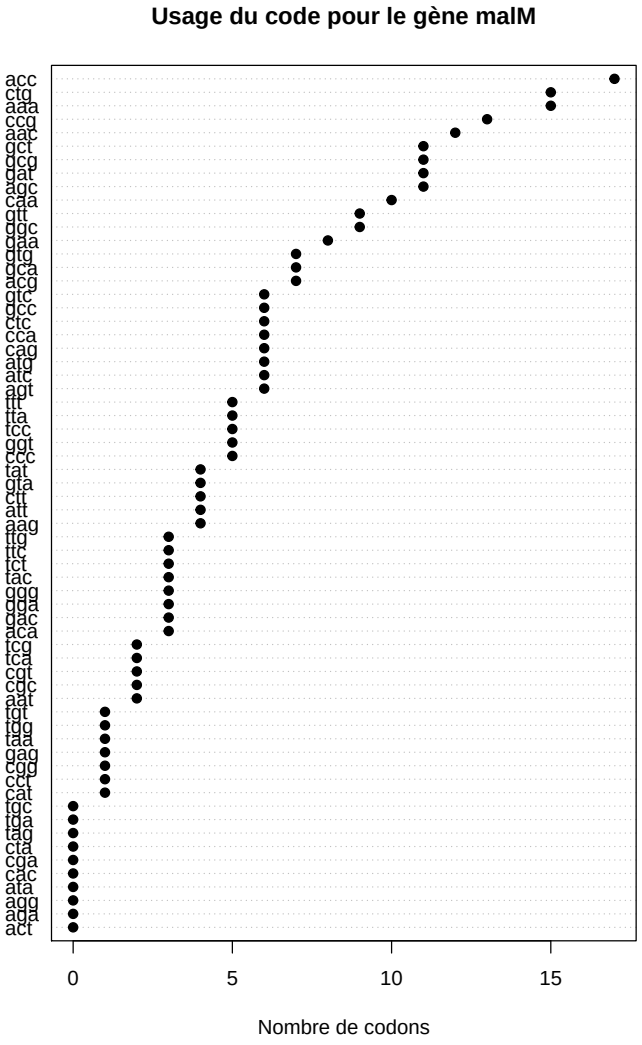
Calculer l'usage du code du gène *malM* :

```
uco(myseq)
```

```
aaa aac aag aat aca acc acg act aga agc agg agt ata atc atg att caa cac cag cat cca
 15  12   4   2   3  17   7   0   0  11   0   6   0   6   6   4  10   0   6   1   6
ccc ccg cct cga cgc cgg cgt cta ctc ctg ctt gaa gac gag gat gca gcc gcg gct gga ggc
  5  13   1   0   2   1   2   0   6  15   4   8   3   1  11   7   6  11  11   3   9
ggg ggt gta gtc gtg gtt taa tac tag tat tca tcc tcg tct tga tgc tgg tgt tta ttc ttg
  3   5   4   6   7   9   1   3   0   4   2   5   2   3   0   0   1   1   5   3   3
ttt
  5
```

Représenter graphiquement l'usage du code du gène *malM* :

```
dotchart(sort(uco(myseq)), main = "Usage du code pour le gène malM",
  xlab = "Nombre de codons", pch = 19)
```



On constate ici à l'évidence que l'utilisation des codons est loin d'être uniforme : il y a des codons très fréquents comme le codon `aac` et des codons très rares, comme le codon `act`. C'est la nature des données que nous allons analyser ci-après.

Traduire le gène *malM* pour obtenir la protéine MalM :

```
translate(myseq)[1:50]
```

```
 [1] "M" "K" "M" "N" "K" "S" "L" "I" "V" "L" "C" "L" "S" "A" "G" "L" "L" "A" "S" "A"
[21] "P" "G" "I" "S" "L" "A" "D" "V" "N" "Y" "V" "P" "Q" "N" "T" "S" "D" "A" "P" "A"
[41] "I" "P" "S" "A" "A" "L" "Q" "Q" "L" "T"
```

Utiliser à la place le code 3 lettres pour les acides-aminés :

```
aaa(translate(myseq))[1:30]
```

```
 [1] "Met" "Lys" "Met" "Asn" "Lys" "Ser" "Leu" "Ile" "Val" "Leu" "Cys" "Leu" "Ser"
[14] "Ala" "Gly" "Leu" "Leu" "Ala" "Ser" "Ala" "Pro" "Gly" "Ile" "Ser" "Leu" "Ala"
[27] "Asp" "Val" "Asn" "Tyr"
```

Visualiser le code génétique standard pour vérifier qu'il n'y a pas eu de problème lors de la traduction :

```
tablecode()
```

Genetic code 1 : standard

TTT	Phe	TCT	Ser	TAT	Tyr	TGT	Cys
TTC	Phe	TCC	Ser	TAC	Tyr	TGC	Cys
TTA	Leu	TCA	Ser	TAA	Stp	TGA	Stp
TTG	Leu	TCG	Ser	TAG	Stp	TGG	Trp
CTT	Leu	CCT	Pro	CAT	His	CGT	Arg
CTC	Leu	CCC	Pro	CAC	His	CGC	Arg
CTA	Leu	CCA	Pro	CAA	Gln	CGA	Arg
CTG	Leu	CCG	Pro	CAG	Gln	CGG	Arg
ATT	Ile	ACT	Thr	AAT	Asn	AGT	Ser
ATC	Ile	ACC	Thr	AAC	Asn	AGC	Ser
ATA	Ile	ACA	Thr	AAA	Lys	AGA	Arg
ATG	Met	ACG	Thr	AAG	Lys	AGG	Arg
GTT	Val	GCT	Ala	GAT	Asp	GGT	Gly
GTC	Val	GCC	Ala	GAC	Asp	GGC	Gly
GTA	Val	GCA	Ala	GAA	Glu	GGA	Gly
GTG	Val	GCG	Ala	GAG	Glu	GGG	Gly

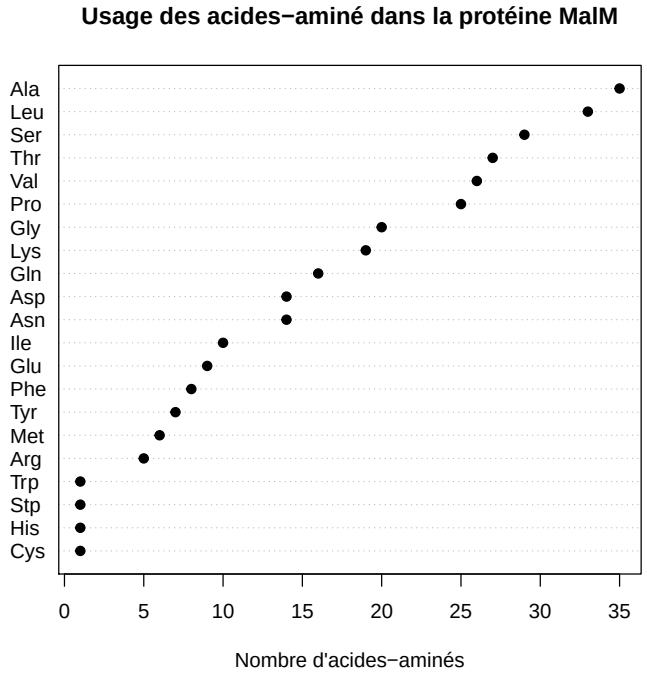
Calculer l'usage des acides-aminés dans la protéine malM :

```
table(aaa(translate(myseq)))
```

```
Ala Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser Stp Thr Trp Tyr Val
 35   5  14  14   1  16   9  20   1  10  33  19   6   8  25  29   1  27   1   7  26
```

Représenter graphiquement les résultats :

```
dotchart(sort(table(aaa(translate(myseq)))), xlab = "Nombre d'acides-aminés",
  main = "Usage des acides-aminé dans la protéine MalM", pch = 19)
```



On constate ici à l'évidence que l'utilisation des acides-aminés est loin d'être uniforme : il y a des acides-aminés très fréquents comme `Ala` et des acides-aminés très rares, comme `Cys`. C'est un aspect de la nature des données qui va nous intéresser ci-après.

4 Analyse multivariée de séquences génomiques

4.1 Introduction

Les analyses multivariées figurent parmi les outils de choix utilisés pour l'analyse des séquences génomiques. L'analyse factorielle des correspondances [9] a ainsi été largement employée dans la détection des biais d'usage des codons, dans la prédiction de la localisation des séquences codantes ou encore dans la détection d'erreurs de séquençage.

Nous allons voir un exemple portant sur l'usage du code chez la bactérie *Borrelia burgdorferi*. Le genre *Borrelia*, classe des spirochètes, comporte une vingtaine d'espèces. *Borrelia burgdorferi* est l'espèce responsable de la maladie de Lyme chez l'homme. Le génome de cette espèce est linéaire et long de 1,5 méga-bases. Il a été séquencé en 1997 [6]. Son origine de réplication, au centre du chromosome, a été expérimentalement localisée [20] ce qui permet d'orienter les gènes par rapport au brin précoce (leading) ou au brin tardif (lagging) de la réplication (*cf.* figure 1 page 8).

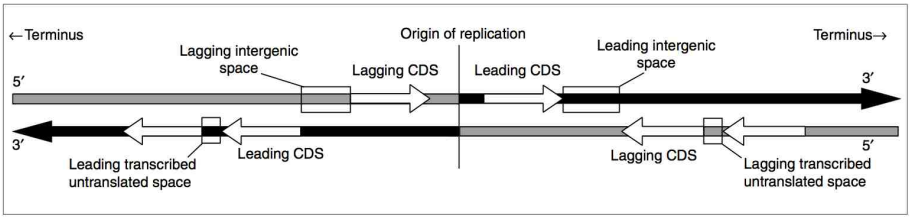


FIGURE 1 – Copie d'écran de la figure 1 dans Lobry & Sueoka (2002) [17]. Inutile d'entrer dans les détails de la biologie moléculaire pour ce TP : il suffit de savoir que la machinerie réplication de l'ADN définit deux groupes de gènes : « précoce » (*leading* en anglais) ou bien « tardif » (*lagging* en anglais) en fonction de leur orientation par rapport au sens de progression de la fourche de réplication. C'est comme dans la fiche `tdr601` d'initiation à l'ACP où nous avions des individus soit ♂ soit ♀, ici nous avons des gènes soit « précoces » soit « tardifs ». Notez que cela n'a rien à voir avec le positionnement du pic d'expression d'un gène dans le cylce de reproduction d'une cellule eucaryote.

4.2 Constitution du jeu de données

OUVRIR la banque de séquences `emglib` [18]. Il s'agit d'une banque de génomes complets d'organismes procaryotes. Récupérer toutes les séquences codantes et non partielles du genome de *Borrelia burgdorferi* (`n=BORBUCG`). Vous stockerez le résultat dans l'objet `borre`.

```
choosebank("emglib")
borre <- query("borre","n=BORBUCG et t=cds et no k=partial")
```

CONSTRUIRE un tableau (que l'on nommera `tuco`) qui donnera pour chaque gène, l'effectif des codons. Les noms de gènes seront en ligne et les codons en colonnes.

```
tuco <- matrix(NA, nrow = borre$nelem, ncol = 64)
rownames(tuco) <- getName(borre)
colnames(tuco) <- words()
for(i in seq(1:borre$nelem)){
        print(paste("Calcul pour la séquence", rownames(tuco)[i]))
        tuco[i, ] <- uco(getSequence(borre$req[[i]]))
}
```

Sauvegarder l'objet `tuco` dans un fichier :

```
save(tuco, file = "tuco.rda")
```

On pourra ainsi recharger l'objet `tuco` avec la commande :

```
load(url("http://pbil.univ-lyon1.fr/R/donnees/tdr74/tuco.rda"))
dim(tuco)
```

```
[1] 850  64
```

```
sum(tuco)
```

```
[1] 284122
```

NOUS avons donc construit une table de contingence dans laquelle nous avons ventilés 284122 codons en croisant les 64 modalités possibles pour la nature du codon et les 850 modalités possibles pour la nature du gène.

FIGURE 2 – Copie d'écran du début de l'entrée pour la séquence du génome complet de *Borrelia burgdorferi* dans la banque `emglib` [18]. Par rapport aux banques généralistes on trouve ici des annotations supplémentaires que nous allons exploiter dans ce TP. La localisation des gènes sur le brin précoce (`leading`) ou tardif (`lagging`) pour la réplication est documenté sous `/strand=`, ainsi que la valeur du « Codon Adaptation Index » [22] sous `/CAI=`.

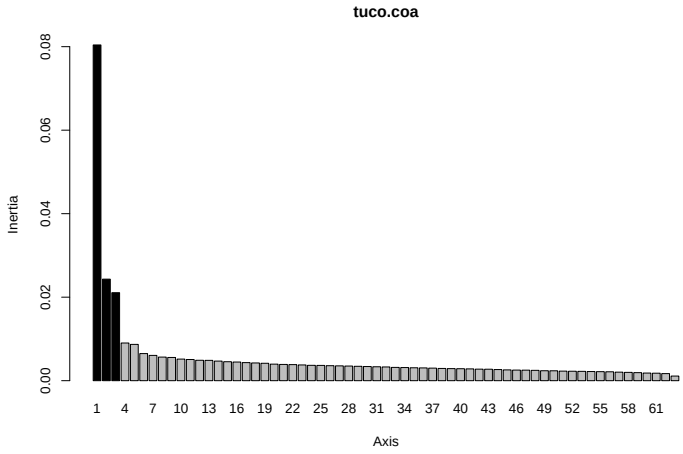
4.3 Réalisation de l'AFC

RÉALISER l'analyse factorielle des correspondances sur ce tableau en utilisant la fonction `dudi.coa()` du paquet `ade4` en conservant trois facteurs :

```
tuco.coa <- dudi.coa(tuco, scan = FALSE, nf = 3)
```

Représenter le graphe des valeurs propres pour justifier le fait que nous allons nous intéresser aux trois premiers facteurs :

```
screeplot(tuco.coa)
```



Calculer le pourcentage de variance pris en compte par les trois premiers facteurs :

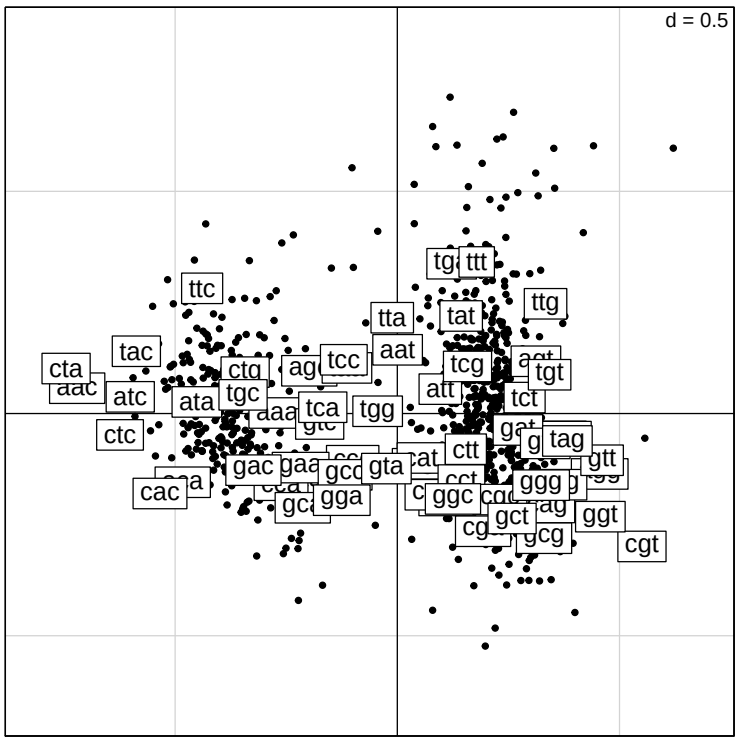
```
 100*tuco.coa$eig[1:3]/sum(tuco.coa$eig)
```

```
[1] 23.909870  7.231221  6.265265
```

Représentez le premier plan factoriel sans mettre d'étiquette pour les lignes ni de représentation du graphe des valeurs propres :

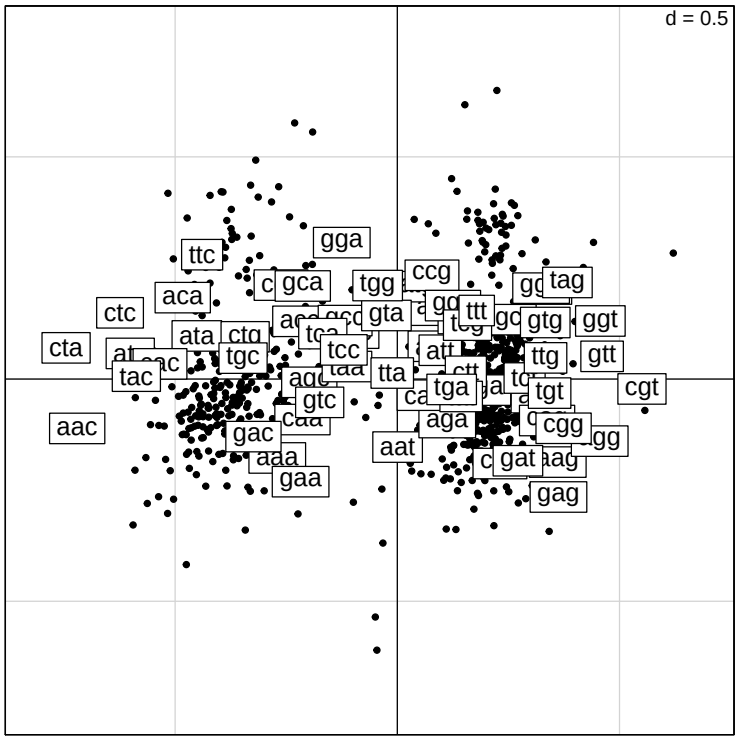
```
 scatter(tuco.coa, xax = 1, yax = 2, clab.row = 0, posieig = "none")
```

```
NULL
```



Représentez de même le plan (1, 3) :

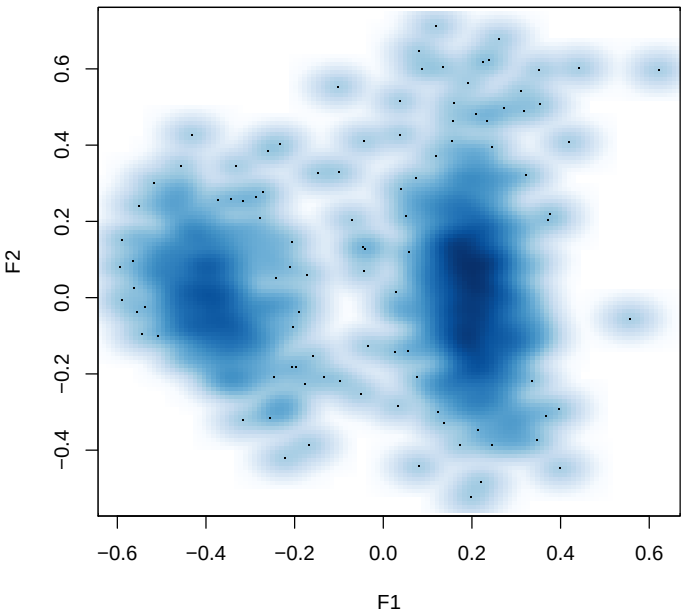
```
scatter(tuco.coa, xax = 1, yax = 3, clab.row = 0, posieig = "none")
```
```
NULL
```



QUAND on approche le millier d'invidus il devient intéressant d'utiliser au lieu des nuages de points classiques des représentations faisant appel à une estimation de la densité locale des points :

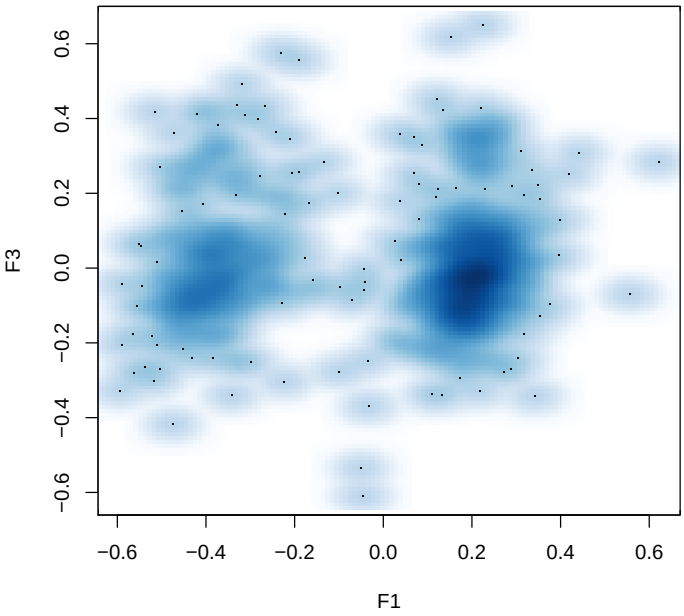
```
x <- tuco.coa$li[, 1] ; y <- tuco.coa$li[, 2]
smoothScatter(x, y, xlab = "F1", ylab = "F2", main = "Le premier plan factoriel")
```

Le premier plan factoriel




```
x <- tuco.coa$li[, 1] ; y <- tuco.coa$li[, 3]
smoothScatter(x, y, xlab = "F1", ylab = "F3", main = "Le deuxième plan factoriel")
```

Le deuxième plan factoriel

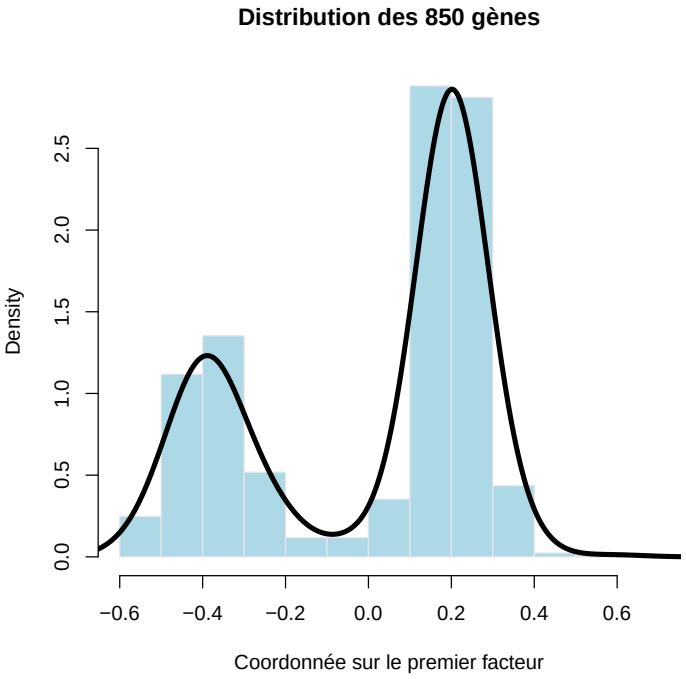


On voit que l'on a une distribution bimodale sur le premier et le troisième facteur. Reste à interpréter ces facteurs pour leur donner un sens biologique.

4.4 Interprétation du premier facteur

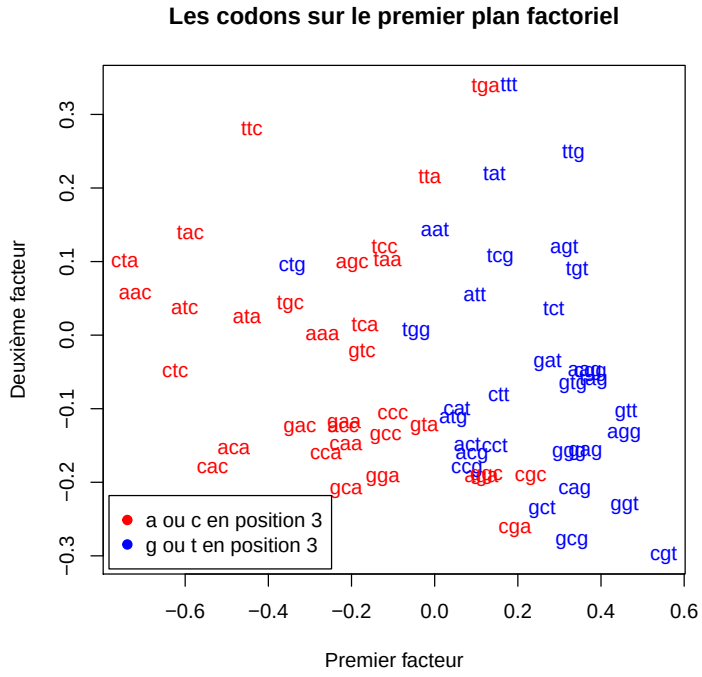
Représentez la distribution des gènes sur le premier facteur :

```
x <- tuco.coa$li[, 1]
hist(x, col = "lightblue", proba = TRUE, border = grey(0.9),
xlab = "Coordonnée sur le premier facteur", main = "Distribution des 850 gènes")
lines(density(x), lwd = 4)
```



Ce que l'AFC nous apprend ici c'est que le plus gros facteur de variabilité de l'usage du code chez *Borrelia burgdorferi* est la conséquence de l'existence de deux sous-populations de gènes utilisant leur propre dialecte. Pour aider à l'interprétation on représente les codons avec une variable illustrative :

```
x <- tuco.coa$co[, 1]
y <- tuco.coa$co[, 2]
coul <- ifelse(substr(colnames(tuco), 3, 3) %in% c("a", "c"), "red", "blue")
plot(x, y, pch = "", xlab = "Premier facteur", ylab = "Deuxième facteur",
  main = "Les codons sur le premier plan factoriel")
text(x, y, colnames(tuco), col = coul)
legend("bottomleft", inset = 0.01, col = c("red", "blue"), pch = 19,
  legend = c("a ou c en position 3", "g ou t en position 3"))
```

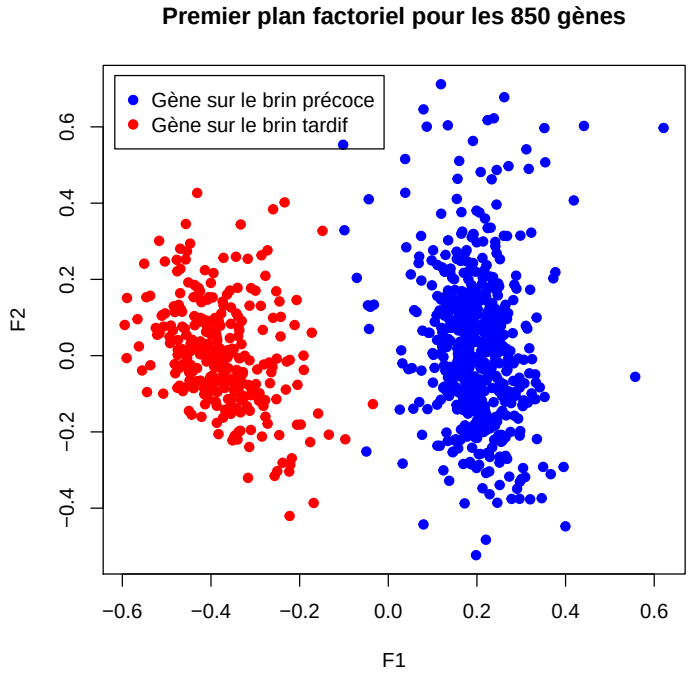


Pour confirmer cette interprétation nous allons extraire des annotations des gènes leur localisation sur le brin précoce ou tardif de la réplication :

```
leading <- logical(borre$nelem)
for(i in seq(1:borre$nelem)){
  print(paste("Traitement de la séquence numéro", i))
  annot <- getAnnot(borre$req[[i]], nbl = 8)
  if(length(grep("leading", annot)) == 1) leading[i] <- TRUE
}
save(leading, file = "leading.rda")
```

On repart de la sauvegarde et on représente les gènes sur le premier plan factorielle avec comme variable illustrative la localisation sur le brin précoce ou tardif de la réplication :

```
load(url("http://pbil.univ-lyon1.fr/R/donnees/tdr74/leading.rda"))
x <- tuco.coa$li[, 1] ; y <- tuco.coa$li[, 2]
plot(x, y, pch = 19, col = ifelse(leading, "blue", "red"), xlab = "F1",
  ylab = "F2", main = "Premier plan factoriel pour les 850 gènes")
legend("topleft", inset = 0.02, legend = c("Gène sur le brin précoce",
  "Gène sur le brin tardif"), pch = 19, col = c("blue", "red"))
```



Nous pouvons donc interpréter le premier facteur comme étant l'opposition entre les gènes situés sur le brin précoce qui utilisent préférentiellement des codons terminant par G ou T de ceux situés sur le brin tardif qui utilisent préférentiellement des codons terminant par A ou C. C'est l'exemple typique du résultat d'une pression de mutation asymétrique (*cf.* figure 3 page 16). Notez que les effets sélectifs ne sont pas complètement absents puisqu'il y a plus de gènes sur le brin précoce que sur le brin tardif :

```
barplot(table(leading), las = 1, ylab = "Nombre de gènes",
  main = "L'excès de gènes sur le brin précoce",
  xlab = "Le gène est sur le brin précoce", col = "lightblue")
```

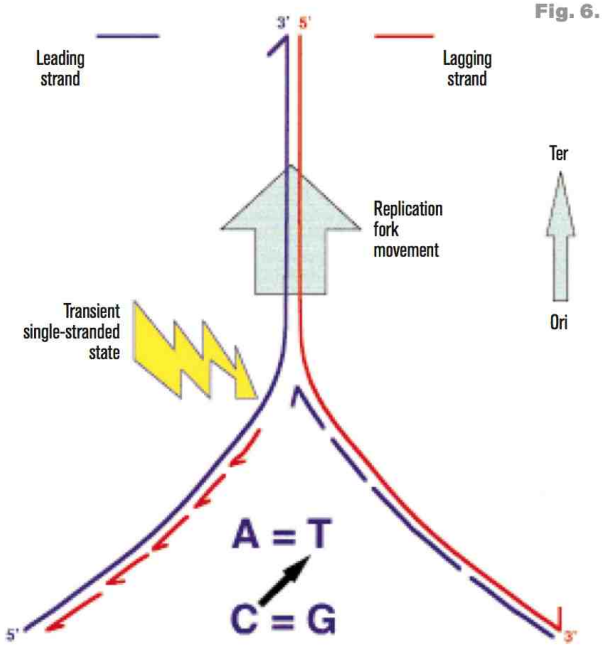
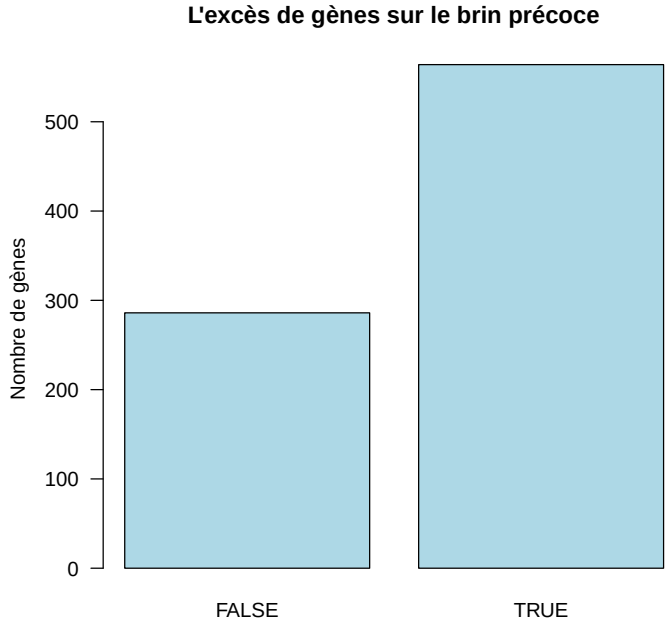


FIGURE 3 – Copie d'écran de la figure 6 dans [13]. Le mécanisme de réplication de l'ADN est fondamentalement asymétrique parce que les polymérases ne travaillent que dans le sens 5'→3'. Durant la réplication le brin précoce (*leading*) reste plus longtemps exposé au solvant ce qui augmente d'un facteur 100 les mutations C→T [1]. Le brin précoce se trouve alors enrichi en bases G et T. C'est un exemple de pression de mutation asymétrique [12].



On peut interpréter ceci comme étant l'effet d'une pression sélective pour éviter les collisions frontales entre le système assurant la transcription des gènes et le sens de progression de la fourche de réplication [11]. Si tel est bien le cas, on devrait s'attendre à ce que cet effet soit exacerbé pour les gènes codant pour des protéines fortement exprimées. Ce qui nous fait une transition toute trouvée pour l'interprétation du deuxième facteur.

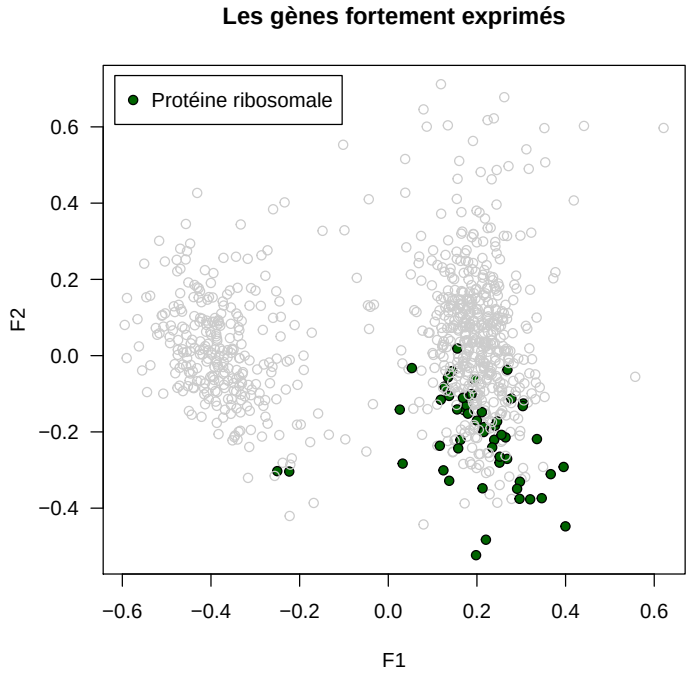
4.5 Interprétation du deuxième facteur

Le cas de *Borrelia burgdorferi* est un peu particulier. Chez les bactéries le premier facteur de la variabilité de l'usage du code est en général lié au niveau d'expression des gènes : les gènes fortement exprimés ont tendance à utiliser des codons pris en charge par des tRNA à forte concentration intracellulaire pour maximiser l'efficacité de la traduction [7]. Il est donc naturel de chercher si le deuxième facteur ne serait pas lié à cet effet. Mais comment faire ? Une première approche consiste à repérer les gènes dont on sait qu'il sont fortement exprimés : ceux qui codent pour des protéines ribosomales. En effet, en phase de croissance exponentielle, les protéines ribosomales représentent à elles seules de l'ordre de 50 % de la biomasse des bactéries. Pour les récuperer, il faut composer la requête suivante :

```
rib <- query("rib", "borre et k=@ribosomal@protein@")
ribnames <- getName(rib)
save(ribnames, file = "ribnames.rda")
```

Le critère `k=@ribosomal@protein@` permet de ne récupérer dans la liste complète des CDS que ceux qui sont associés aux mots-clés contenant la chaîne de caractères `@ribosomal@protein@`. Le caractère `@` est un *joker*, c'est-à-dire qu'il peut remplacer n'importe quelle chaîne. Attention, dans la vraie vie, lorsque l'on veut récupérer des gènes appartenant à une famille bien précise, il faut toujours - après une requête basée sur l'emploi de mots-clés - veiller à éliminer les entrées pour lesquelles les annotations montrent que l'on est pas en présence d'un représentant de la dite famille. Dans le cadre de ce TP, vous pouvez directement exploiter l'information, mais ce n'est pas le cas en général. Représenter les gènes en portant cette information illustrative :

```
load(url("http://pbil.univ-lyon1.fr/R/donnees/tdr74/ribnames.rda"))
mescouleurs <- ifelse(rownames(tuco) %in% ribnames, "darkgreen", "transparent")
mespourtours <- ifelse(rownames(tuco) %in% ribnames, "black", grey(0.8))
x <- tuco.coa$li[, 1] ; y <- tuco.coa$li[, 2]
plot(x, y, bg = mescouleurs, pch = 21, xlab = "F1", ylab = "F2", las = 1,
  col = mespourtours,
  main = "Les gènes fortement exprimés")
legend("topleft", inset = 0.02, legend = "Protéine ribosomale", pch = 21, pt.bg = "darkgreen")
```



Ce que l'AFC nous apprend ici c'est qu'il y a un facteur opposant les protéines fortement exprimées aux autres. Nous n'avons pas comme pour le premier facteur deux sous-populations, il s'agit plus d'un gradient. Au vu de cette représentation graphique il semble évident que la pression de sélection pour mettre les gènes sur le brin précoce semble exacerbé pour ceux qui sont fortement exprimés. Cependant, les effectifs ne sont pas très élevés, on décide donc de faire un test statistique. On pose H_0 : « il y a indépendance entre la localisation des gènes et leur niveau d'expression » contre H_1 son alternative. On décide classiquement de prendre un risque de première espèce $\alpha = 0.05$. On calcule la table de contingence qui ventile les gènes entre les modalités croisées de ces deux variable qualitatives :

```
tdc <- matrix(c(850 - sum(leading) - 2, sum(leading) - 51,
                2, 51), nrow = 2, byrow = TRUE)
colnames(tdc) <- c("Tardif", "Précoce")
rownames(tdc) <- c("Faiblement","Fortement")
tdc
```

```
           Tardif Précoce
Faiblement    284     513
Fortement       2      51
```

Le résultat du test du χ^2 d'indépendance est le suivant :

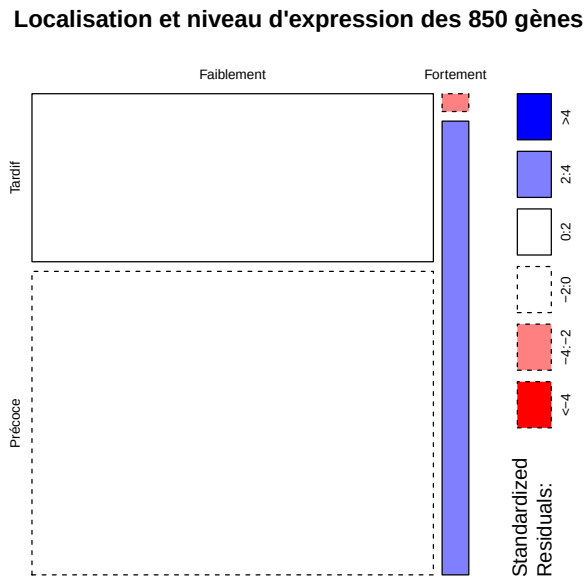
```
chisq.test(tdc)
```

```
        Pearson's Chi-squared test with Yates' continuity correction
data:  tdc
X-squared = 21.19, df = 1, p-value = 4.16e-06
```

On est bien en dessous des 5 %, donc avec un risque de première espèce $\alpha = 0.05$ les données expérimentales sont en contradiction avec l'hypothèse d'indépendance indépendance entre la localisation des gènes et leur niveau

d'expression. Mais avant de conclure il faut vérifier que la dépendance mise en évidence va bien dans le sens attendu :

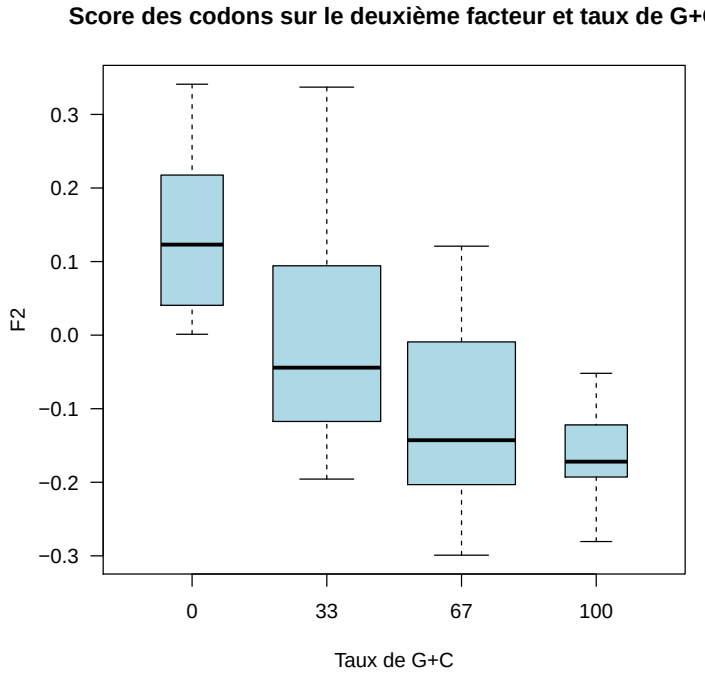
```
mosaicplot(tdc, shade=T,
           main = "Localisation et niveau d'expression des 850 gènes")
```



C'est bien ça : il y a (en bleu) un excès significatif des gènes fortement exprimés dans le brin précoce et *a contrario* un déficit (en rouge) des gènes fortement exprimés dans le brin tardif. L'analyse du second facteur nous permet ici de conforter l'interprétation que nous avons fait du premier facteur.

Il nous faut préciser maintenant en quoi les gènes fortement exprimés diffèrent des autres. Pour comprendre quels sont leurs codons préférés, on fait la représentation suivante :

```
y <- tuco.coa$co[, 2]
gccodon <- sapply(colnames(tuco), function(x) round(100*GC(s2c(x))))
boxplot(y~gccodon, col = "lightblue", varwidth = TRUE,
  xlab = "Taux de G+C", las = 1, ylab = "F2",
  main = "Score des codons sur le deuxième facteur et taux de G+C")
```

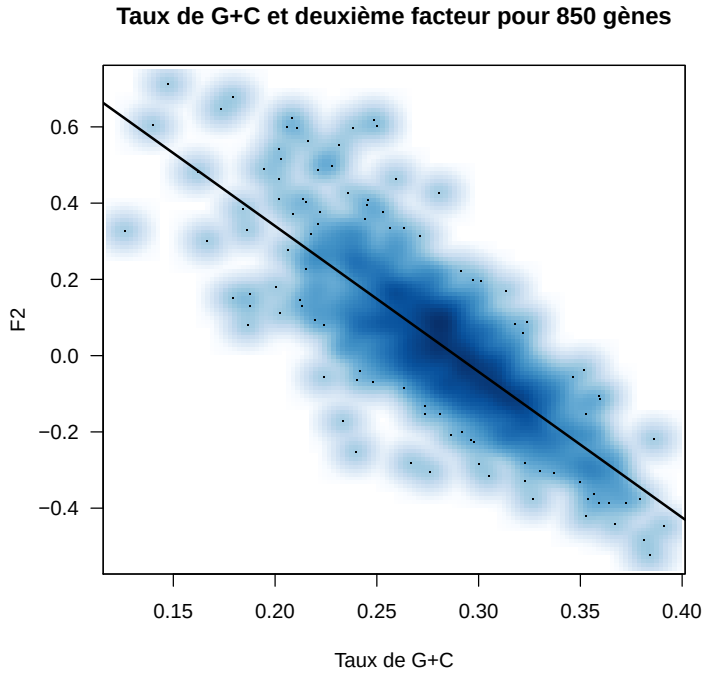


On constate que les codons avec un score négatif sur F2 ont tendance à avoir un taux de G+C élevé. Essayons de confirmer ceci en calculant le taux de G+C de tous les gènes :

```
tgc <- numeric(borre$nelem)
for(i in seq(1:borre$nelem)){
  print(paste("Calcul du taux de G+C de la séquence numéro", i))
  tgc[i] <- GC(getSequence(borre$req[[i]]))
}
save(tgc, file = "tgc.rda")
```

On peut maintenant confronter le taux de G+C des gènes avec leurs coordonnées sur le deuxième facteur :

```
load(url("http://pbil.univ-lyon1.fr/R/donnees/tdr74/tgc.rda"))
y <- tuco.coa$li[, 2]
smoothScatter(tgc, y, xlab = "Taux de G+C", ylab = "F2", las = 1,
  main = "Taux de G+C et deuxième facteur pour 850 gènes")
abline(lm(y~tgc), lwd = 2)
```

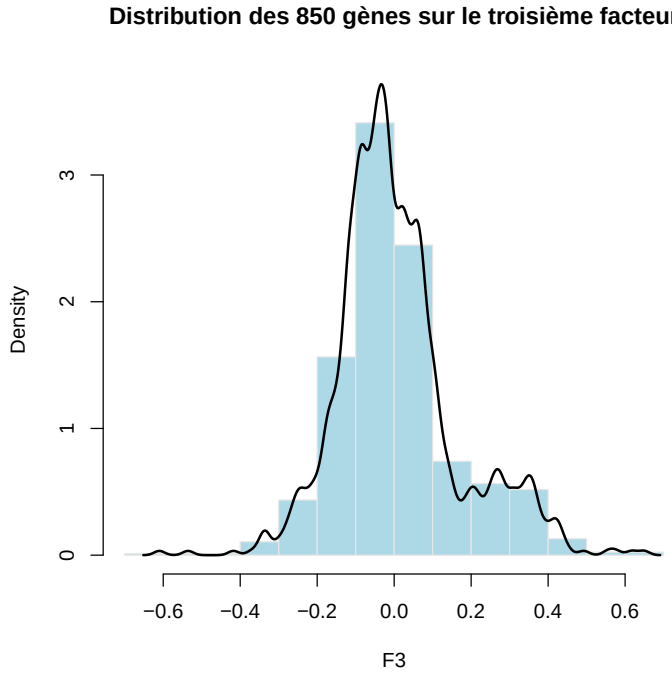


On constate qu'il y a une forte corrélation : les gènes fortement exprimés, avec des coordonnées négatives sur F2 on tendance à avoir un taux de G+C élevé. Notez que ce taux de G+C élevé est tout relatif : le taux de G+C moyen des gènes de *Borrelia burgdorferi* est de 28.4 %, ce qui est extrêmement faible, on dit que c'est une bactérie AT riche. Les gènes fortement exprimés montent ici à 40 % de taux de G+C maximum, ce qui est beaucoup plus que les autres gènes, mais reste néamoins faible. Ce que nous avons ici c'est un génome qui est soumis à une pression de mutation directionelle [23] intense vers les bas G+C. On rencontre souvent ce phénomène chez les bactéries endosymbiontiques qui avec des effectifs de population efficaces faibles et leur génome réduit ne sont pas dans les conditions pour contrer ce biais mutationnel. L'AFC ne nous permet pas de détecter directement ce phénomène parce que c'est l'ensemble des gènes qui y sont soumis, ce n'est pas un facteur structurant de la variabilité entre les gènes. Il faudrait analyser simultanément plusieurs génomes ayant des taux de G+C différents pour le voir apparaître comme un facteur majeur de la variabilité de l'usage du code [15] chez les bactéries.

4.6 Interprétation du troisième facteur

Nous avons déjà vu en examinant le deuxième plan factoriel que la distribution des gènes était bimodale sur le troisième facteur. Précisons les choses :

```
x <- tuco.coa$li[, 3]
dstx <- density(x, adjust = 0.5)
hist(x, col = "lightblue", proba = TRUE, ylim = c(0, max(dstx$y)), border = grey(0.9),
  main = "Distribution des 850 gènes sur le troisième facteur", xlab = "F3")
lines(dstx, lwd = 2)
```



On voit qu'il y a un groupe qui se singularise avec des valeurs sur F3 supérieures à 0.2. Ce groupe représente de l'ordre de 10 % des gènes :

```
100*sum(x>0.2)/length(x)
```

```
[1] 12.58824
```

Comment interpréter le troisième facteur ? Dans ce cas de figure, lorsque l'on a un groupe de faible effectif qui se singularise, une approche possible est de retourner aux données de départ pour essayer de comprendre ce que ces individus ont de particulier en commun. On commence par récupérer la liste des noms des séquences qui nous intéressent :

```
imp <- rownames(tuco)[tuco.coa$li[, 3] > 0.2]
length(imp)
```

```
[1] 107
```

```
imp[1:10]
```

```
 [1] "BORBUCG.BB0006" "BORBUCG.BB0017" "BORBUCG.BB0027" "BORBUCG.BB0034"
 [5] "BORBUCG.BB0037" "BORBUCG.BB0050" "BORBUCG.BB0051" "BORBUCG.BB0070"
 [9] "BORBUCG.BB0073" "BORBUCG.BB0090"
```

Le moins que l'on puisse dire c'est que le nom des gènes de ce génome n'est pas très informatif. Il va nous falloir inspecter les annotations des gènes. On commence par sauvegarder les noms des séquences dans un fichier texte :

```
writeLines(imp, "imp.txt")
```

On utilise ensuite ce fichier pour construire une requête et récupérer les annotations. Cette requête étant longue vous pouvez la sauter pour partir directement du résultat sauvegardé.

```
listIMP <- crelistfromclientdata("listIMP", file = "imp.txt", type = "SQ")
impann <- getAnnot(listIMP$req)
save(impann, file = "imp.rda")
```

L'examen des annotations pour le premier gène peut déjà nous mettre la puce à l'oreille :

```
load(url("http://pbil.univ-lyon1.fr/R/donnees/tdr74/imp.rda"))
impann[[1]][1:10]
```

```
 [1] "     CDS             complement(6309..7433)"
 [2] "                     /strand=\"leading\""
 [3] "                     /CAI=\"0.679117\""
 [4] "                     /gene=\"BB0006\""
 [5] "                     /note=\"similar to GB:L42023 SP:P44646 PID:1003567"
 [6] "                     PID:1222263 PID:1204590 percent identity: 28.88;"
 [7] "                     identified by sequence similarity; putative\""
 [8] "                     /product=\"conserved hypothetical integral membrane"
 [9] "                     protein\""
[10] "                     /db_xref=\"PID:g2687886\""
```

On constate qu'il s'agit d'une protéine membranaire intégrale. Comptons le nombre de fois où le mot « membrane » apparaît dans les annotations :

```
sum(unlist(sapply(impann, function(x) grep("membrane", x))))
```

```
[1] 120
```

Nous avons donc 120 fois le mot « membrane », ce qui fait beaucoup pour 107 gènes. On a ici un phénomène bien connu : les protéines membranaires intégrales sont enrichies en acides-aminés hydrophobes pour assurer leur localisation au sein des phospholipides constitutifs des membranes cellulaires. Il s'agit donc d'une pression de sélection sur la composition globale en acides-aminés des protéines. Pour essayer de confirmer cette interprétation, nous allons calculer l'indice d'hydrophobicité de Kyte et Doolittle [10], aussi connu sous le nom de « GRAVY score ». C'est une forme linéaire sur les fréquences des codons :

$$s = \sum_{i=1}^{64} \alpha_i f_i$$

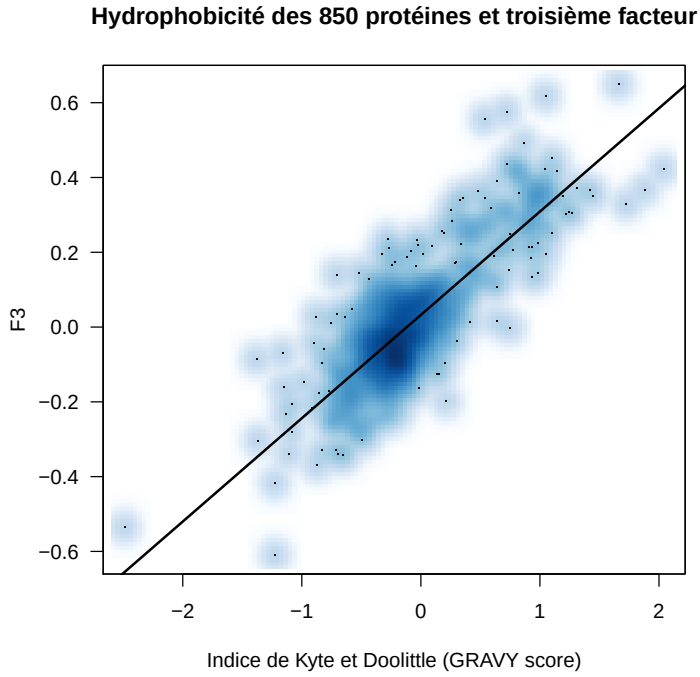
où α_i est le coefficient pour l'acide-aminé codé par le codon numéro i et f_i la fréquence relative du codon numéro i. Les coefficients α_i sont donnés dans l'élément `KD` du jeu de données `EXP` :

```
data(EXP)
EXP$KD
```

```
 [1] -3.9 -3.5 -3.9 -3.5 -0.7 -0.7 -0.7 -0.7 -4.5 -0.8 -4.5 -0.8  4.5  4.5  1.9  4.5
[17] -3.5 -3.2 -3.5 -3.2 -1.6 -1.6 -1.6 -1.6 -4.5 -4.5 -4.5 -4.5  3.8  3.8  3.8  3.8
[33] -3.5 -3.5 -3.5 -3.5  1.8  1.8  1.8  1.8 -0.4 -0.4 -0.4 -0.4  4.2  4.2  4.2  4.2
[49]  0.0 -1.3  0.0 -1.3 -0.8 -0.8 -0.8 -0.8  0.0  2.5 -0.9  2.5  3.8  2.8  3.8  2.8
```

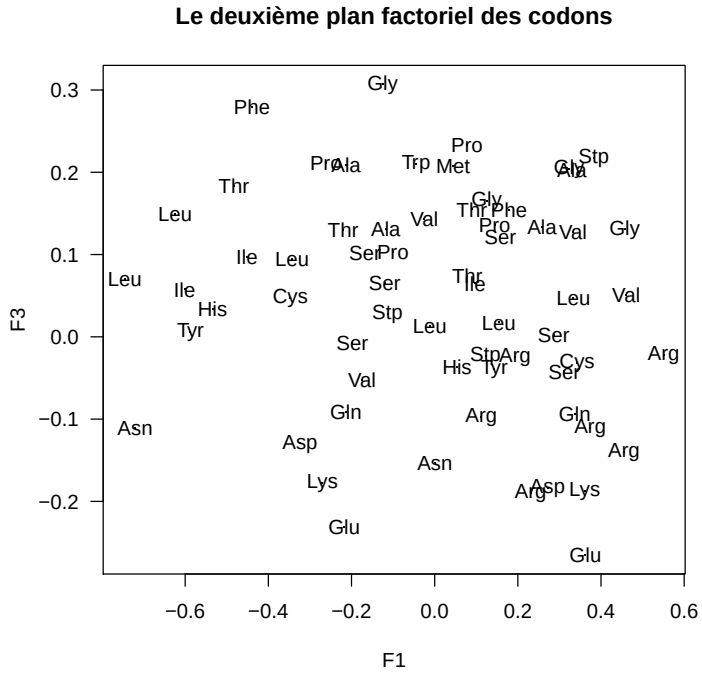
Grace aux notations matricielles de R le calcul d'une forme linéaire s'écrit de façon très compacte :

```
rtuco <- tuco/rowSums(tuco)
kd <- rtuco %*% EXP$KD
y <- tuco.coa$li[, 3]
smoothScatter(kd, y,
  main = "Hydrophobicité des 850 protéines et troisième facteur",
  xlab = "Indice de Kyte et Doolittle (GRAVY score)", ylab = "F3", las = 1)
abline(lm(y~kd), lwd = 2)
```



On constate donc que le troisième facteur correspond à un gradient hydrophile-hydrophobe qui singularise les protéines membranaires intégrales. On revient au point de vue des codons mais en les traduisants pour faciliter la lecture :

```
aaname <- sapply(colnames(tuco), function(x) aaa(translate(s2c(x))))
x <- tuco.coa$co[, 1] ; y <- tuco.coa$co[, 3]
plot(x, y, pch = ".", main = "Le deuxième plan factoriel des codons",
  las = 1, xlab = "F1", ylab = "F3")
text(x, y, aaname)
```



Ceci confirme notre interprétation puisque les protéines hydrophobes ont tendance à éviter, en bas, les acides-aminés chargés tels que l'aspartate, le glutamate, la lysine ou l'arginine. Elles préfèrent, en haut, les acides-aminés hydrophobes tels que la phénylalanine ou le tryptophane.

5 Pour aller plus loin

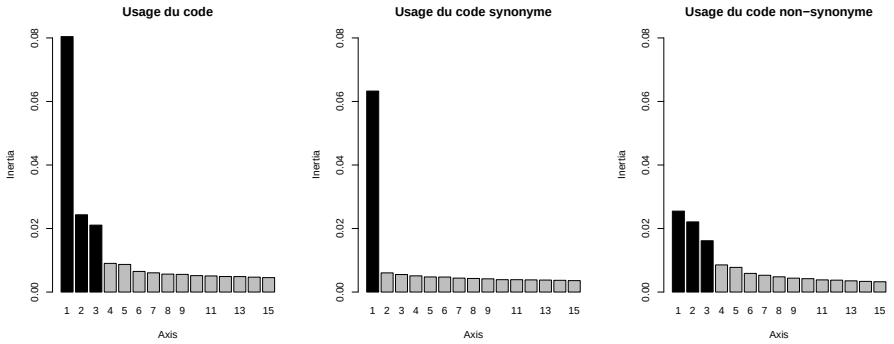
5.1 Analyse factorielle des correspondances inter-intra

Nous avons vu dans la section précédente toute la puisance de l'AFC pour mettre en évidence des structures dans notre table d'usage du code. L'AFC fait parti des méthodes d'analyse multivariées dites mono-tableau : on n'analyse qu'une seule table à la fois. Mais les méthodes d'analyse multivariées ne s'arrêtent pas là, il existe en effet des méthodes de couplage qui permettent d'analyse plusieurs tables à la fois. Nous allons déborder un petit peu du cadre de ce TP en utilisant une méthode multi-tableau particulière : l'analyse inter-intra. Pour étudier la variabilité synonyme de l'usage du code on voit parfois dans la littérature l'utilisation de l'AFC sur des tables de fréquences relatives des codons par acide-aminé, mais c'est une très mauvaise idée [19], c'est bien l'AFC intra qu'il faut utiliser [4], comme le démontre ses performances par rapport aux autres méthodes [24].

Quand on analyse une table d'usage du code on aimerait bien distinguer dans la variabilité celle qui est synonyme, dans le sens où elle n'a pas d'impact au niveau des acides-aminés, de celle qui est non-synonyme et porte donc sur l'analyse de l'usage des acides-aminés. Nous allons donc coupler notre table d'usage du code avec un facteur donnant pour chaque codon l'acide-aminé

correspondant. Dans `ade4` le facteur doit correspondre aux lignes du tableau, on transpose donc la table avant de lancer les analyses.

```
ttuco <- t(tuco)
ttuco.coa <- dudi.coa(ttuco, scan = FALSE, nf = 3)
facaa <- as.factor(aaname)
ttuco.wca <- wca(ttuco.coa, facaa, scan = FALSE, nf = 1)
ttuco.bca <- bca(ttuco.coa, facaa, scan = FALSE, nf = 3)
par(mfrow = c(1,3))
maxi <- ttuco.coa$eig[1]
screeplot(ttuco.coa, npcs = 15, ylim = c(0, maxi), main = "Usage du code")
screeplot(ttuco.wca, npcs = 15, ylim = c(0, maxi), main = "Usage du code synonyme")
screeplot(ttuco.bca, npcs = 15, ylim = c(0, maxi), main = "Usage du code non-synonyme")
```



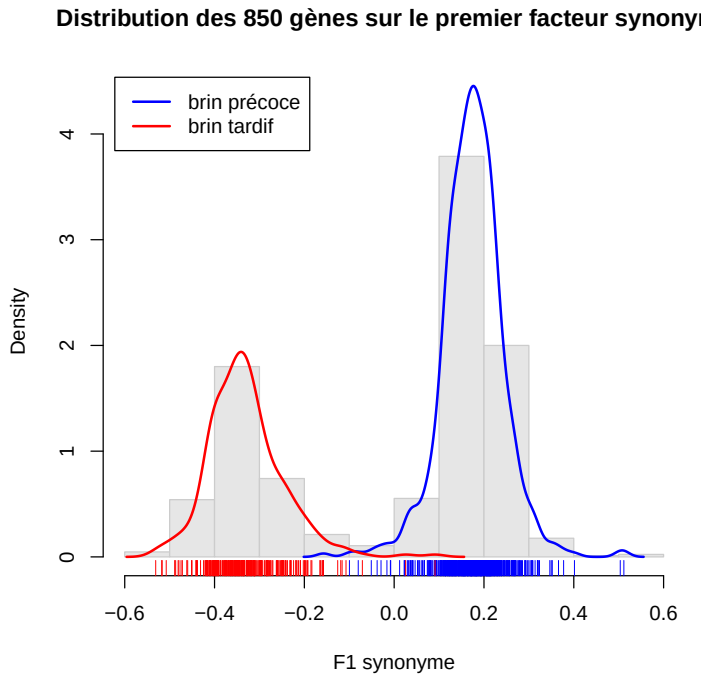
On retrouve sur la figure de gauche les résultats de l'AFC effectuée précédemment. Le fait d'avoir transposé la table n'a strictement aucun effet parce que l'AFC traite de façon parfaitement symétrique les lignes et les colonnes du tableau analysé. Nous avons donc pour rappel trois facteurs dominant la variabilité de l'usage du code chez *Borrelia burgdorferi* que nous avons interprétés pour rappel comme étant, dans l'ordre d'importance décroissante, comme le résultat : i) d'une pression de mutation asymétrique ii) d'une pression de sélection pour l'optimisation de la traduction ii) d'une pression de sélection sur l'hydrophobicité des protéines.

Cette variabilité globale de l'usage peut se décomposer grâce à l'analyse inter-intra comme la somme de la variabilité synonyme (graphe du milieu) et de la variabilité non-synonyme (graphe de droite). L'analyse de la variabilité non-synonyme revient exactement à faire l'AFC de la table d'usage des acides-aminés. On remarque tout de suite des choses intéressantes. Premièrement, il y a plus de variabilité au niveau synonyme que non-synonyme, c'est un phénomène bien connu, les substitutions synonymes ne changeant pas par définition l'acide-aminé codé sont moins accessibles aux pressions de sélection. Deuxièmement, il n'y a plus qu'un seul facteur dominant au niveau synonyme alors que trois restent au niveau non-synonyme.

On regarde la distribution des 850 gènes sur le premier facteur en fonction de leur localisation par rapport au brin précoce et tardif de la réplication :

```
x <- ttuco.wca$co[, 1]
lea <- x[leading] ; dlea <- density(lea) ; ylea <- dlea$y*length(lea)/length(x)
lag <- x[!leading] ; dlag <- density(lag) ; ylag <- dlag$y*length(lag)/length(x)
hist(x, border = grey(0.8), proba = TRUE, ylim = c(0, max(ylea)),
  xlab = "F1 synonyme", col = grey(0.9),
  main = "Distribution des 850 gènes sur le premier facteur synonyme")
lines(dlea$x, ylea, col = "blue", lwd = 2)
lines(dlag$x, ylag, col = "red", lwd = 2)
rug(lea, col = "blue") ; rug(lag, col = "red")
```

```
legend("topleft", inset = 0.02, legend = c("brin précoce", "brin tardif"),
  col = c("blue", "red"), lwd = 2)
```

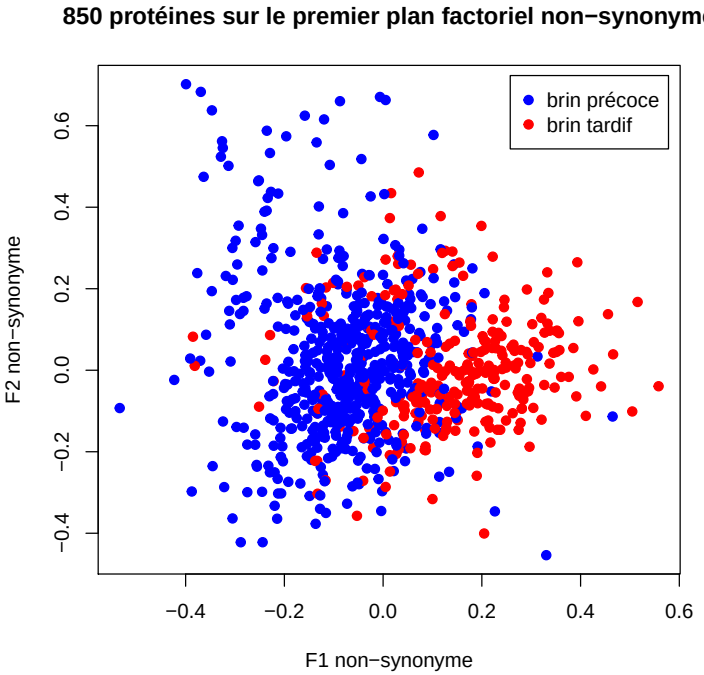


On retrouve donc ici l'effet de la pression de mutation asymétrique. On peut prédire sans grande chance se tromper l'orientation des gènes par rapport à la réplication à partir de leur coordonnées sur ce premier facteur [2]. On retrouve la pression de sélection en faveur la co-orientation des gènes avec le sens de progression de la fourche de réplication puisqu'il y a plus de gènes sur le brin précoce. Il n'y avait qu'un facteur à interpréter, nous en avons fini pour l'analyse de la variabilité synonyme.

On représente le premier plan factoriel des individus, ici des protéines puisque nous travaillons en fait avec les fréquences des acides-aminés des gènes les codants, en portant comme information supplémentaire l'orientation par rapport à la réplication :

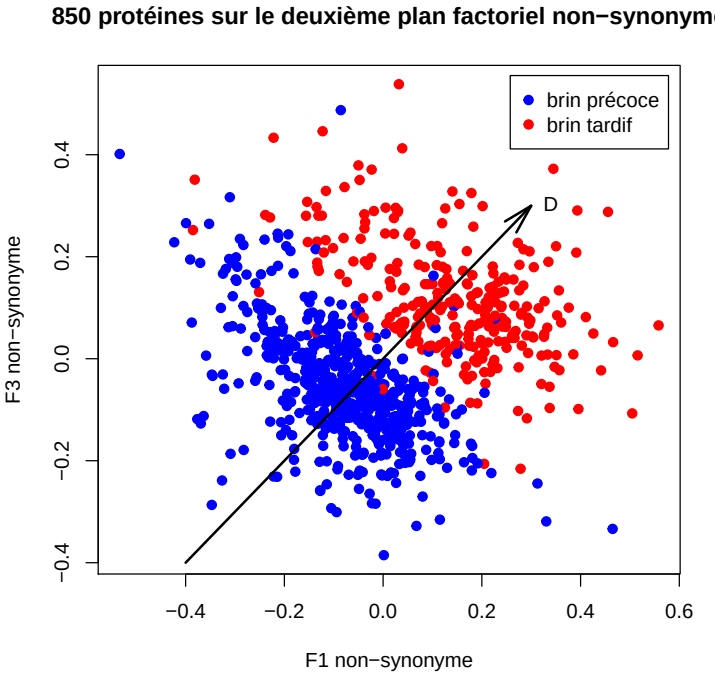
```
x <- ttuco.bca$co[, 1] ; y <- ttuco.bca$co[, 2]
plot(x, y, pch = 19, col = ifelse(leading, "blue", "red"),
  xlab = "F1 non-synonyme", ylab = "F2 non-synonyme",
  main = "850 protéines sur le premier plan factoriel non-synonyme")
legend("topright", inset = 0.02, legend = c("brin précoce", "brin tardif"),
  col = c("blue", "red"), pch = 19)
```

2. Attention : si votre objectif est de prédire la localisation des gènes à partir de leur composition en codons il vaut mieux utiliser une méthode d'analyse discriminante utilisant la distance de Mahalanobis pour tenir compte de l'hétéroscédasticité.



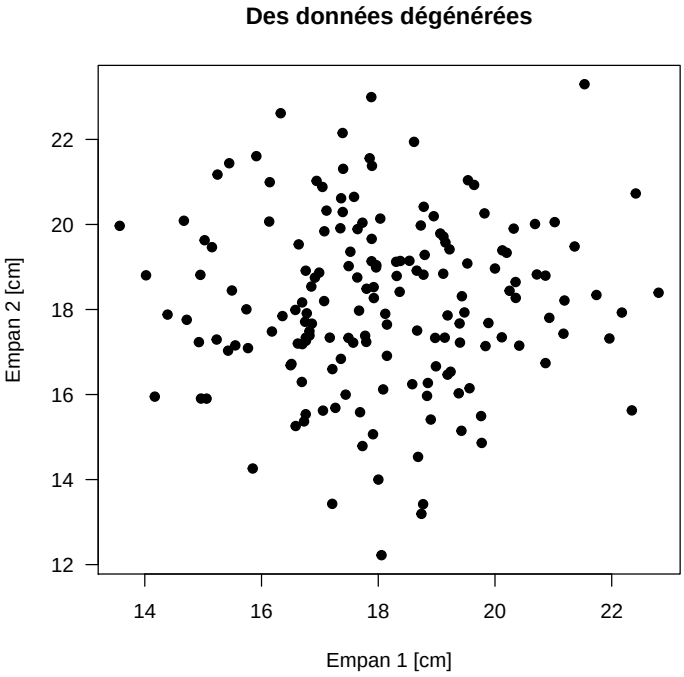
On retrouve donc ici au niveau des acides-aminés l'effet de la pression de mutation asymétrique. Ce qui est étonnant c'est que ce soit le premier facteur de la variabilité de la composition des protéines de *Borrelia burgdorferi*. Pour des bactéries ayant des génomes moins extrêmes le premier facteur sortant au niveau des acides-aminés est plutôt lié à des phénomènes sélectifs comme l'opposition entre les protéines cytoplasmique hydrophiles et les protéines membranaires intégrales hydrophobes chez *Escherichia coli* [16, 15]. Mais la situation est un petit plus compliquée, on va s'en appercevoir via l'examen du deuxième plan factoriel :

```
x <- ttuco.bca$co[, 1] ; y <- ttuco.bca$co[, 3]
plot(x, y, pch = 19, col = ifelse(leading, "blue", "red"),
  xlab = "F1 non-synonyme", ylab = "F3 non-synonyme",
  main = "850 protéines sur le deuxième plan factoriel non-synonyme")
legend("topright", inset = 0.02, legend = c("brin précoce", "brin tardif"),
  col = c("blue", "red"), pch = 19)
arrows(x0 = -0.4, y0 = -0.4, x1 = 0.3, y1 = 0.3, lwd = 2, angle = 15)
text(0.3, 0.3, "D", pos = 4)
```



On voit ici que les groupes de gènes « précoces » et « tardifs » sont bien séparés sur le plan (F1, F3). Si on imagine de définir un facteur D égal à la somme de F1 et F3 (*cf.* flêche) on voit que la projection des individus sur cet axe donnerait deux groupes bien distinct. Mais alors pourquoi est-ce que l'AFC ne m'a pas donné directement D comme premier facteur de l'analyse ? C'est une question intéressante car on va toucher du doigt une limite intrinsèque des méthodes multivariées. Construisons un petit jeu de données dégénéré pour avoir une intuition du phénomène. On imagine que l'on a mesuré la taille (en cm) de l'empan de la main dominante de 168 individus, mais dans univers parallèle un peu bizarre où il n'y à plus de symétrie : la taille de la main droite est complètement indépendante de celle de la main gauche ! On tire les valeurs au hasard dans une même loi normale pour que l'inertie totale de chaque variable soit la même :

```
set.seed(1)
empan1 <- rnorm(n = 168, mean = 18, sd = 2)
empan2 <- rnorm(n = 168, mean = 18, sd = 2)
plot(empan1, empan2, pch = 19, las = 1, main = "Des données dégénérées",
  xlab = "Empan 1 [cm]", ylab = "Empan 2 [cm]")
```

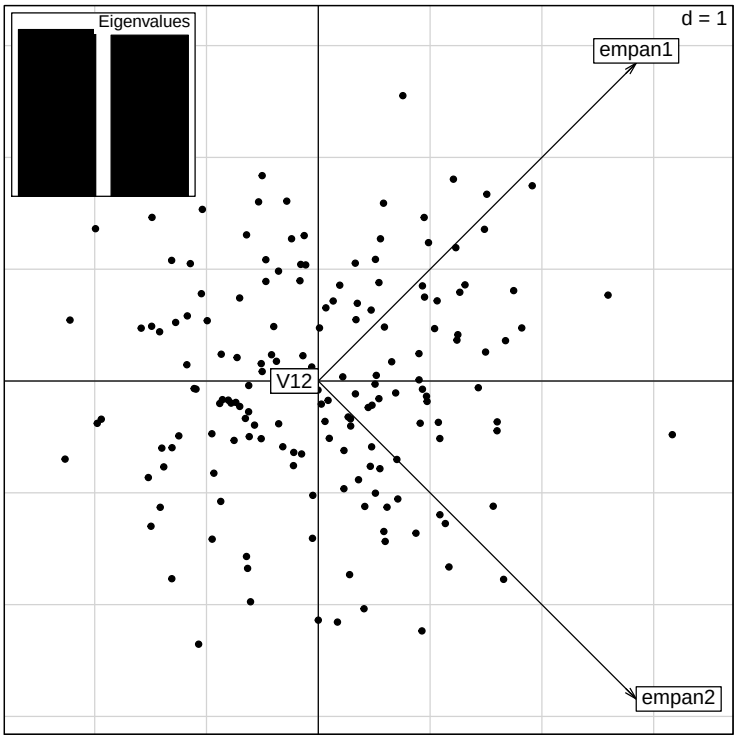


Pour simuler une analyse multivariée on ajoute 10 variables strictement non informatives puisqu'elles ont la même valeur pour tous les individus :

```
deg <- cbind(empan1, empan2, matrix(0, nrow = 168, ncol = 10))
dim(deg)
```

```
[1] 168  12
```

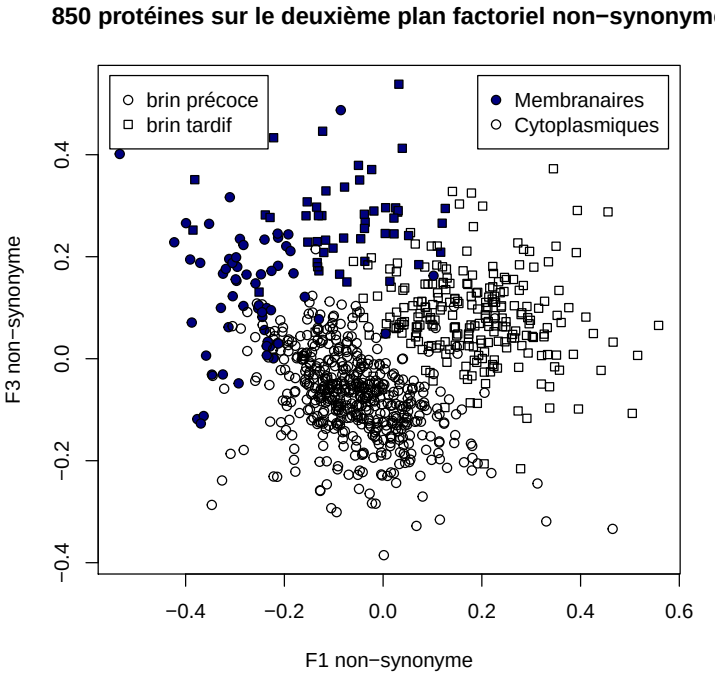
```
deg.pca <- dudi.pca(deg, scan = FALSE, nf = 2)
scatter(deg.pca, clab.row = 0)
```



L'analyse détecte bien qu'il n'y a que deux facteurs seulement puisqu'elle ne donne que deux valeurs propres. Mais ces deux valeurs propres ont des valeurs très voisines puisque que nos deux facteurs biologiques expliquent la même fraction de la variabilité totale. Il n'y a aucun moyen d'ordonner l'un par rapport à l'autre. L'analyse donne arbitrairement une solution possible où F1 est la somme des empans et F2 la différence des empans, mais ce n'est qu'une solution possible : on peut faire une rotation des vecteurs initiaux autour de l'origine et avoir des solutions tout autant satisfaisantes.

Nous sommes proches d'un cas dégénéré de ce type pour l'analyse de l'usage des acides-aminés. On peut déjà le suspecter parce que le graphe des valeurs propres est assez plat pour les trois premiers facteurs. Si on reprend le deuxième plan factoriel en ajoutant comme information supplémentaire les protéines intégralement membranaires et la localisation des gènes par rapport à la réplication, on a la représentation suivante :

```
x <- ttuco.bca$co[, 1] ; y <- ttuco.bca$co[, 3]
mybg <- ifelse(rownames(tuco) %in% imp, "navy", "transparent")
mypch <- ifelse(leading, 21, 22)
plot(x, y, pch = mypch, bg = mybg,
  xlab = "F1 non-synonyme", ylab = "F3 non-synonyme",
  main = "850 protéines sur le deuxième plan factoriel non-synonyme")
legend("topright", inset = 0.02, legend = c("Membranaires", "Cytoplasmiques"),
  pt.bg = c("navy", "transparent"), pch = 21)
legend("topleft", inset = 0.02, legend = c("brin précoce", "brin tardif"),
   pch = c(21, 22))
```

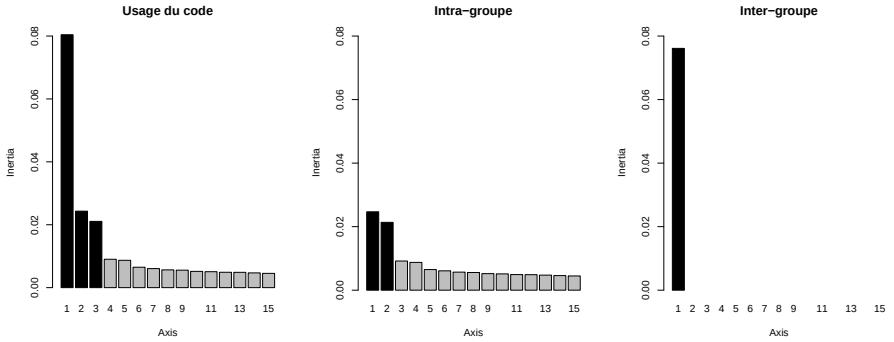


On voit que les facteurs F1 et F3 sont des facteurs composites qui tous deux prennent en compte une part de la variabilité induite par les deux phénomènes biologiques sous-jacents.

5.2 Analyse des correspondances internes

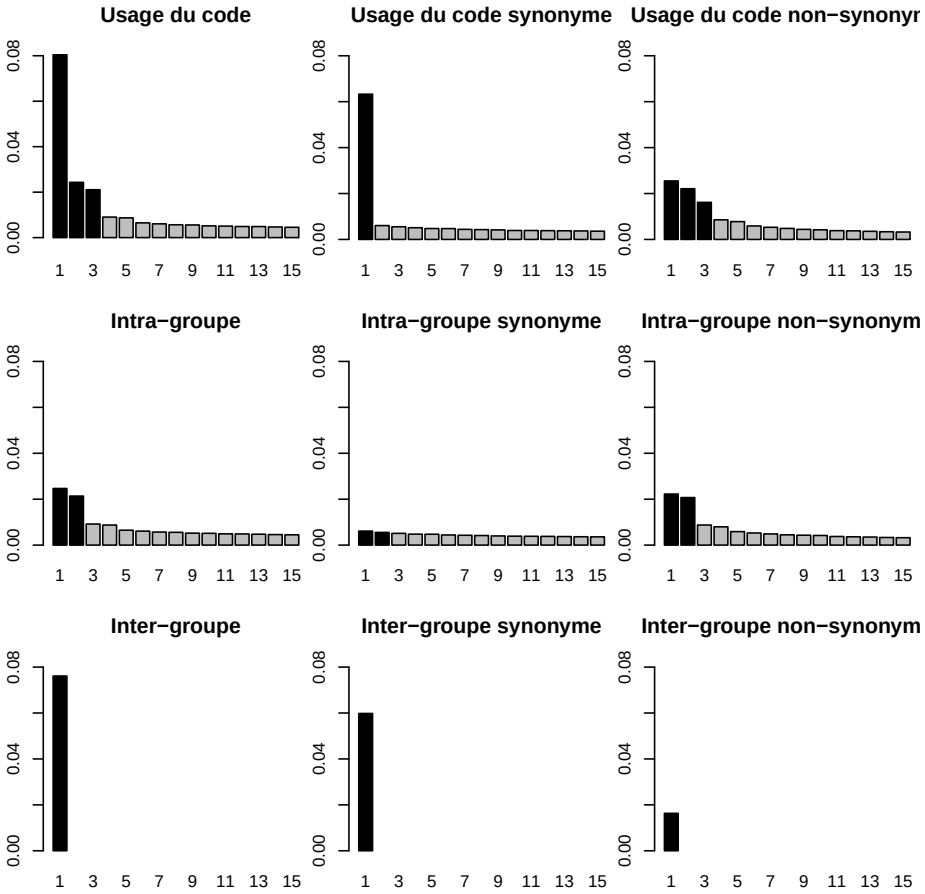
Nous avons vu dans la partie précédente que l'on pouvait décomposer la variabilité en utilisant la structuration en bloc-colonnes définie par le code génétique. Mais aimerait pouvoir le faire également en décomposant la variabilité en utilisant la structuration en bloc-lignes définie par l'appartenance des gènes au groupe « précoce » ou « tardif ». Et on aimerait bien aussi faire les deux simultanément, c'est le but de l'analyse des correspondances internes [2]. On commence par décomposer la variabilité entres les brin précoces et tardifs :

```
tuco.wca <- wca(tuco.coa, as.factor(leading), scannf = FALSE, nf = 2)
tuco.bca <- bca(tuco.coa, as.factor(leading), scannf = FALSE, nf = 1)
par(mfrow = c(1, 3))
screeplot(tuco.coa, npcs = 15, main = "Usage du code") ; ymax <- tuco.coa$eig[1]
screeplot(tuco.wca, npcs = 15, main = "Intra-groupe", ylim = c(0, ymax))
screeplot(tuco.bca, npcs = 15, main = "Inter-groupe", ylim = c(0, ymax))
```



On voit donc ici que le gros de la variabilité s'explique par la différence d'usage du code entre les gènes « précoces » et « tardifs » et qu'il reste deux facteurs interprétables en variabilité intra-groupe. Dans chaque sous-analyse on décompose maintenant la variabilité en variabilité synonyme et non-synonyme :

```
ttuco.wca.wca <- wca(t(tuco.wca), facaa, scan = F, nf = 0)
ttuco.wca.bca <- bca(t(tuco.wca), facaa, scan = F, nf = 2)
ttuco.bca.wca <- wca(t(tuco.bca), facaa, scan = F, nf = 1)
ttuco.bca.bca <- bca(t(tuco.bca), facaa, scan = F, nf = 1)
par(mfrow = c(3, 3), mar = c(2, 2, 4, 0))
screeplot(tuco.coa, npcs = 15, main = "Usage du code") ; ymax <- tuco.coa$eig[1]
screeplot(ttuco.wca, npcs = 15, ylim = c(0, maxi), main = "Usage du code synonyme")
screeplot(ttuco.bca, npcs = 15, ylim = c(0, maxi), main = "Usage du code non-synonyme")
screeplot(tuco.wca, npcs = 15, main = "Intra-groupe", ylim = c(0, ymax))
screeplot(ttuco.wca.wca, npcs = 15, main = "Intra-groupe synonyme", ylim = c(0, ymax))
screeplot(ttuco.wca.bca, npcs = 15, main = "Intra-groupe non-synonyme", ylim = c(0, ymax))
screeplot(tuco.bca, npcs = 15, main = "Inter-groupe", ylim = c(0, ymax))
screeplot(ttuco.bca.wca, npcs = 15, main = "Inter-groupe synonyme", ylim = c(0, ymax))
screeplot(ttuco.bca.bca, npcs = 15, main = "Inter-groupe non-synonyme", ylim = c(0, ymax))
```



Nous pouvons faire la lecture suivante de cette représentation graphique synthétique :

— On part de l'analyse globale en haut à gauche. Il y a trois facteurs dominants qui représentent 12.6 % de la variabilité totale. Pour faciliter les comparaisons on ramène tout à cette varibilité structurée. Les trois premiers facteurs expliquent donc 100 % de cette variabilité structurée avec le premier facteur qui domine largement avec 63.9 % contre seulement 19.3 % et 16.7% pour suivants.

— On cherche ensuite le graphique où il y a le plus d'inertie prise en compte. C'est le graphe en bas à gauche pour la variabilité inter-groupes qui explique 60.5 % de la variabilité structurée. Donc le facteur majeur de la variabilité de l'usage du code chez *Borrelia burgdorferi* est la différence entre les gènes « précoces » et *tardifs*.

— Les deux graphes juste à droite de ce dernier nous montrent que ce facteur à 60.5 % se décompose essentiellement en variabilité synonyme : 47.5 % pour la variabilité synonyme contre 13.0 % pour la variabilité non-synonyme, c'est à dire au niveau des acides-aminés.

— On cherche le graphique suivant où il y a le plus d'inertie prise en compte. C'est celui du milieu en haut qui nous dit donc que c'est au niveau synonyme qu'il y a facteur dominant pour l'usage du code. Plus précisément nous avons 50 % pour le seul facteur de la variabilité synonyme et 50 %

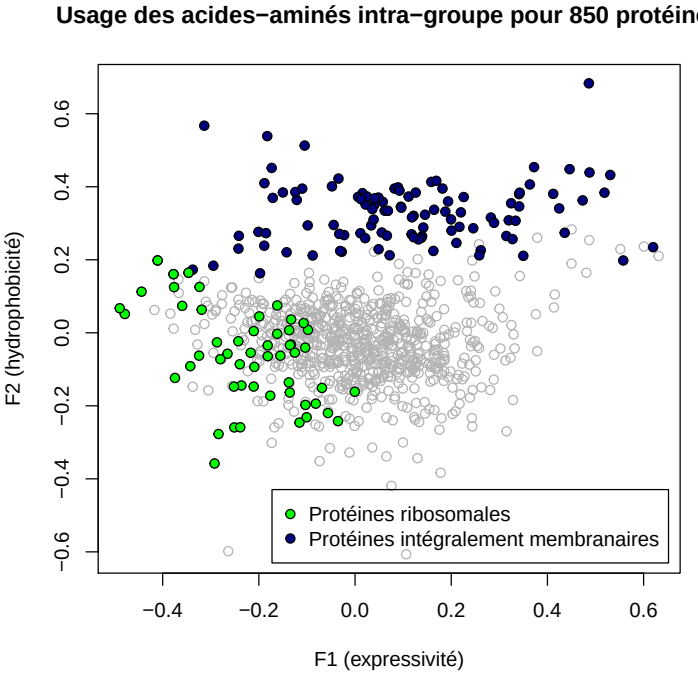
pour les trois facteurs de la variabilité non-synonyme.

— En regardant les deux graphiques juste en bas on voit que cette variabilité synonyme s'explique quasiment exclusiment par l'effet inter-groupe. On a 47.5 % pour l'effet inter-groupe et un résiduel 4.8 % pour le premier facteur de l'effet intra-groupe mais qui ne semble pas être dominant. Autrement dit, la quasi-totalité de la variabilité synonyme de l'usage du code s'explique par l'opposition entre les gènes « précoces » et « tardifs ».

— Il nous reste à analyser l'usage des acides-aminés. On a déjà vu qu'il y avait un effet inter-groupe pour 13.0 %, reste à anlyser les deux facteurs restants quand cet effet est neutralisé (graphe de droite au milieu). Les deux facteurs dominants représentent 17.7 % et 16.5 % de la variabilité initiale structurée.

On représente le premier plan factoriel de l'usage intra-groupe des acides-aminés en portant comme information supplémentaire les protéines membranaires intégrales et les protéines ribosomales comme exemple de protéines fortement exprimées :

```
x <- ttuco.wca.bca$co[, 1] ; y <- ttuco.wca.bca$co[, 2]
plot(x, y, main = "Usage des acides-aminés intra-groupe pour 850 protéines",
  xlab = "F1 (expressivité)", ylab = "F2 (hydrophobicité)", pch = 21, col = grey(0.7))
ind <- rownames(tuco) %in% imp
points(x[ind], y[ind], bg = "navy", pch = 21)
ind <- rownames(tuco) %in% ribnames
points(x[ind], y[ind], bg = "green", pch = 21)
legend("bottomright", inset = 0.02, c("Protéines ribosomales",
  "Protéines intégralement membranaires"), pch = 21, pt.bg = c("green", "navy"))
```



Il n'y a maintenant plus aucune ambiguité pour l'interprétation de ces deux facteurs : toutes les protéines membranaires sont en haut, toutes les protéines ribosomales sont en bas. Nous pouvons interpréter le premier facteur comme un

axe d'expressivité opposant les protéines faiblement exprimées à celles qui le sont fortement. Nous pouvons interpréter le deuxième facteur comme un axe d'hydrophobicité séparant les protéines cytoplasmiques de celles qui sont membranaires. Notez que les protéines ribosomales, cytoplasmiques, sont correctement placées sur le F2. Par rapport à l'analyse globale de l'usage des acides-aminés nous avons une situation beaucoup plus confortable pour interpréter les facteurs grâce à la neutralisation de l'effet inter-groupes.

En résumé, pour ce qui est de l'usage des acides-aminés chez *Borrelia burgdorferi* nous avons trois facteurs structurant la variabilité qui par ordre d'importance décroissante sont :

1. Un gradient d'expressivité qui oppose les protéines fortement exprimées au protéines faiblement exprimées.
2. Un gradient d'hydrophobicité qui oppose les protéines membranaires intégrales riches en acides-aminés hydrophobes aux protéines cytoplasmiques riches en acides-aminés hydrophiles.
3. Une opposition entre les protéines codées par des gènes situés sur le brin précoce pour la réplication et celles codées par des gènes siués sur le brin tardif.

L'interprétation, et l'appréciation de l'importance relative des ces facteurs, est beaucoup plus délicate quand on fait une analyse mono-tableau du type AFC de l'usage des acides-aminés parce que la structure particulière des données fait que l'analyse renvoie des facteurs composites. C'est tout l'intérêt des méthodes de couplage que d'aider à l'interprétation et à la quantification de l'importance des variables latentes.

5.3 De l'échec du CAI

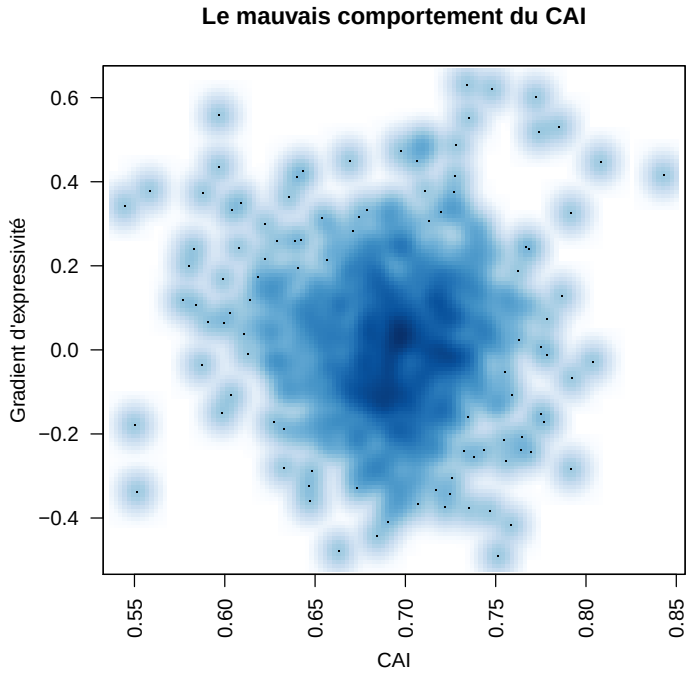
Une statistique très populaire dans les analyses de l'usage du code est le CAI [22] qui est un acronyme pour l'anglais « Codon Adaptation Index » soit indice d'adaptation des codons. Un exemple complet en français de la façon dont il est calculé est donné dans la fiche « Analyse de l'usage du code des génomes bactériens » [3]. Retenons simplement que ses valeurs peuvent varier entre 0 et 1 et seront d'autant plus proches de 1 que l'usage synonyme du code ressemble à un ensemble de gène de référence. Si on utilise comme ensemble de référence des gènes fortement exprimés, comme les protéines ribosomale, le CAI devient alors un indice d'expressivité du gène. Nous n'allons pas le calculer ici puisqu'il est déjà renseigné dans `emglib`, nous nous contentons de l'extraire des annotations :

```
cai <- logical(borre$nelem)
for(i in seq(1:borre$nelem)){
  print(paste("Traitement de la séquence numéro", i))
  annot <- getAnnot(borre$req[[i]], nbl = 3)
  cai[i] <- as.numeric(unlist(strsplit(annot[3], split = '"'))[2])
}
save(cai, file = "cai.rda")
```

On compare maintenant pour tous les gènes leur position sur le gradient d'expressivité que l'analyse des correspondances internes nous a permis de mettre en évidence avec la valeur du CAI :

3. `http://pbil.univ-lyon1.fr/R/pdf/tdr77.pdf`

```
load(url("http://pbil.univ-lyon1.fr/R/donnees/tdr74/cai.rda"))
x <- cai ; y <- ttuco.wca.bca$co[ , 1]
smoothScatter(x, y, las = 2, xlab = "CAI", ylab = "Gradient d'expressivité",
  main = "Le mauvais comportement du CAI")
```



On constate qu'il n'y a aucune relation entre les deux, ce qui d'un point de vue pédagogique n'est pas une situation très confortable si on voulait utiliser le CAI pour aider à l'interprétation des facteurs. Normalement, ça marche, mais avec *Borrelia burgdorferi* on est dans un cas un peu particulier puisque le gradient d'expressivité se trouve exclusivement au niveau non-synonyme. Comme le CAI se base uniquement sur l'usage synonyme des codons, il ne peut pas retrouver ce gradient d'expressivité. Ceci dit, même pour des génomes moins pathologiques comme celui d'*Escherichia coli*, on trouve une trace du gradient d'expressivité au niveau non-synonyme [16]. Si votre objectif est de prédire le niveau d'expression des gènes à partir de l'usage du code il est dommage de perdre une partie du signal en se restreignant d'entrée de jeu au niveau synonyme avec le CAI.

Références

[1] A.S. Bhagwat, W. Weilong Hao, J.P. Townes, H. Lee, H. Tang, and P.L. Foster. Strand-biased cytosine deamination at the replication fork causes cytosine to thymine mutations in *Escherichia coli*. *Proceedings of the National Academy of Sciences of the United States of America*, 113 :2176–2181, 2016.

[2] P. Cazes, D. Chessel, and S. Dolédec. L'analyse des correspondances internes d'un tableau partitionné : son usage en hydrobiologie. *Revue de Statistique Appliquée*, 36 :39–54, 1988.

[3] D. Charif and J.R. Lobry. SeqinR 1.0-2 : a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis. In H.E. Roman U. Bastolla, M. Porto and M. Vendruscolo, editors, *Structural approaches to sequence evolution : Molecules, networks, populations*, Biological and Medical Physics, Biomedical Engineering, pages 207–232. Springer Verlag, New York, USA, 2007. ISBN 978-3-540-35305-8.

[4] D. Charif, J. Thioulouse, J.R. Lobry, and G. Perrière. Online synonymous codon usage analyses with the ade4 and seqinR packages. *Bioinformatics*, 21(4) :545–7, 2005.

[5] D. Chessel, A.-.B. Dufour, and J. Thioulouse. The ade4 package-I- One-table methods. *R News*, 4 :5–10, 2004.

[6] C.M. Fraser, S. Casjens, W.M. Huang, G.G. Sutton, R. Clayton, R. Lathigra, O. White, K.A. Ketchum, R. Dodson, E.K. Hickey, M. Gwinn, B. Dougherty, J.F. Tomb, R.D. Fleischmann, D. Richardson, J. Peterson, A.R. Kerlavage, J. Quackenbush, S. Salzberg, M. Hanson, R. van Vugt, N. Palmer, M.D. Adams, J. Gocayne, J. Weidman, T. Utterback, L. Watthey, L. McDonald, P. Artiach, C. Bowman, S. Garland, C. Fuji, M.D. Cotton, K. Horst, K. Roberts, B. Hatch, H.O. Smith, and J.C. Venter. Genomic sequence of a Lyme disease spirochaete, *Borrelia burgdorferi*. *Nature*, 390 :580–586, 1997.

[7] M. Gouy and C. Gautier. Codon usage in bacteria : correlation with gene expressivity. *Nucleic Acids Research*, 10 :7055–7073, 1982.

[8] M. Gouy, F. Milleret, C. Mugnier, M. Jacobzone, and C. Gautier. ACNUC : a nucleic acid sequence data base and analysis system. *Nucleic Acids Res.*, 12 :121–127, 1984.

[9] M.O. Hill. Correspondence analysis : A neglected multivariate method. *Applied Statistics*, 23 :340–354, 1974.

[10] J. Kyte and R.F. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157 :105–132, 1982.

[11] B. Liu and M. Alberts. Head-on collision between a DNA replication apparatus and RNA polymerase transcription complex. *Science*, 267 :1131–1137, 1995.

[12] J.R. Lobry. Asymmetric substitution patterns in the two DNA strands of bacteria. *Molecular Biology and Evolution*, 13 :660–665, 1996.

[13] J.R. Lobry. Genomic landscapes. *Microbiology today*, 26 :164–165, 1999.

[14] J.R. Lobry. *The black hole of symmetric molecular evolution. Habilitation thesis*. PhD thesis, Université Claude Bernard - Lyon 1, 2000.

[15] J.R. Lobry and D. Chessel. Internal correspondence analysis of codon and amino-acid usage in thermophilic bacteria. *Journal of Applied Genetics*, 44 :235–261, 2003.

[16] J.R. Lobry and C. Gautier. Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 *Escherichia coli* chromosome-encoded genes. *Nucleic Acids Research*, 22 :3174–3180, 1994.

[17] J.R. Lobry and N. Sueoka. Asymmetric directional mutation pressures in bacteria. *Genome Biology*, 3(10) :research0058.1–research0058.14, 2002.

[18] G. Perrière, P. Bessières, and B. Labedan. EMGLib : the enhanced microbial genomes library (update 2000). *Nucleic Acids Res.*, 28 :68–71, 2000.

[19] G. Perrière and J. Thioulouse. Use and misuse of correspondence analysis in codon usage studies. *Nucleic Acids Res.*, 30 :4548–4555, 2002.

[20] M. Picardeau, J.R. Lobry, and B.J. Hinnebusch. Physical mapping of an origin of bidirectional replication at the centre of the *Borrelia burgdorferi* linear chromosome. *Molecular Microbiology*, 32 :437–445, 1999.

[21] R Core Team. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[22] P.M. Sharp and W.-H. Li. The codon adaptation index - a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Research*, 15 :1281–1295, 1987.

[23] N. Sueoka. On the genetic basis of variation and heterogeneity of DNA base composition. *Proceedings of the National Academy of Sciences of the United States of America*, 48 :582–592, 1962.

[24] H. Suzuki, C.J. Brown, L.J. Forney, and E. Top. Comparison of correspondence analysis methods for synonymous codon usage in bacteria. *DNA Research*, 15 :357–365, 2008.