

Table d'annotations génomique

Probabilités et statistique pour la biologie (STAT1)

Jacques van Helden

2017-09-21

Contents

But de ce TP	1
Rendu	1
Attendus pour le code	2
Attendus pour le rapport d'interprétation	2
Exemple historique: génome de la levure	2
Analyse de la longueur des gènes de la levure du boulanger	3
Tutoriel	3
Chargement d'un tableau de données	3
Exploration du contenu d'un tableau de données	4
Sélection de sous-ensembles d'un tableau	6
Sélection d'un sous-ensemble de lignes sur base du contenu d'une colonne	7
Décompte par valeur	8
Exercices	9
1. Spécifications du format GTF	9
2. Création d'un dossier local pour le TP	9
3. Localisation du fichier d'annotations	9
4. Téléchargement d'un fichier à partir d'un site ftp	9
5. Chargement d'une table de données en R	9
5. Calcul de la longueur des gènes	10
6. Histogramme de la longueur des gènes	10
7. Paramètres descriptifs	10
8. Intervalle de confiance	10
9. Distribution de la longueur des gènes	11
10. Distribution attendue au hasard pour la longueur des gènes	11
11. Avant de terminer : conservez la trace de votre session	11
Rendu	12

But de ce TP

Durant ce TP, vous serez amenés à effectuer les tâches suivantes:

1. Manipuler une table de données génomique (les annotations du génome de la levure).
2. Sélectionner un sous-ensemble des données en filtrant les lignes sur base d'un critère déterminé (type d'annotation, chromosome).
3. Générer des graphiques pour représenter différents aspects liés à ces données.
4. Calculer les estimateurs de tendance centrale et dispersion.
5. Calculer un intervalle de confiance autour de la moyenne.

Rendu

A la fin du TP, vous déposerez deux fichiers sur Ametice.

1. Votre **code R**.
2. Un **rapport d' Synthétique** qui inclura une présentation des principaux résultats (figures, statistiques descriptives) et votre interprétation.

Attendus pour le code

1. Le code doit être **lisible et compréhensible**: donnez à vos variables des noms indiquant explicitement ce qu'elles contiennent.
2. Le code devra être **correctement documenté** (le symbole **#** en début ou en milieu de ligne indique que le reste de cette ligne est un commentaire).
 - avant chaque bloc de code, expliquer ce que vous comptez faire, à quoi sert ce bloc de code;
 - si c'est utile, ajoutez quelques mots de commentaires pour justifier l'approche choisie;
 - chaque fois que vous définissez une variable, ajoutez sur la même ligne un commentaire indiquant ce que cette variable représente.
3. Le code doit être **transportable**: après l'avoir téléchargé, on doit pouvoir l'exécuter sur une autre machine. Je testerai systématiquement si les fichiers de code peuvent être exécutés sur ma machine. Evitez donc tout recours à des chemins absolus (nous indiquons ci-dessous comment définir des chemins relatifs par rapport à la racine de votre compte).

Attendus pour le rapport d'interprétation

Le rapport doit être synthétique (1 page de texte maximum + autant de figures et tables que vous le désirez).

Chaque question doit être exprimée explicitement avant de présenter les résultats qui y répondent et de fournir l'interprétation de ces résultats.

Chaque figure ou table doit être documentée par une légende permettant à un lecteur naïf de comprendre ce qu'elle représente. L'interprétation des résultats affichés sur une figure ou table se trouvera dans le texte principal (avec une référence au numéro de figure ou table).

Exemple historique: génome de la levure

- 1992: publication du premier chromosome eucaryote complet, le 3ème chromosome de la levure.
- 1996: publication du génome complet.

Sur base des gènes du 3ème chromosome (échantillon) on peut estimer la taille moyenne d'un gène de levure.

Questions:

- (a) La moyenne d'échantillon (chromosome III) permettait-elle de prédire la moyenne de la population (génom complet) ?

Pour répondre à cette question, nous imaginerons que nous sommes revenus en 1992, et utiliserons l'ensemble des gènes du chromosome III (considérés ici comme un échantillon du génome) pour estimer la taille moyenne des gènes pour l'ensemble du génome (la "population" de gènes).

- (b) Cet échantillon peut-il être qualifié de "simple et indépendant" ?

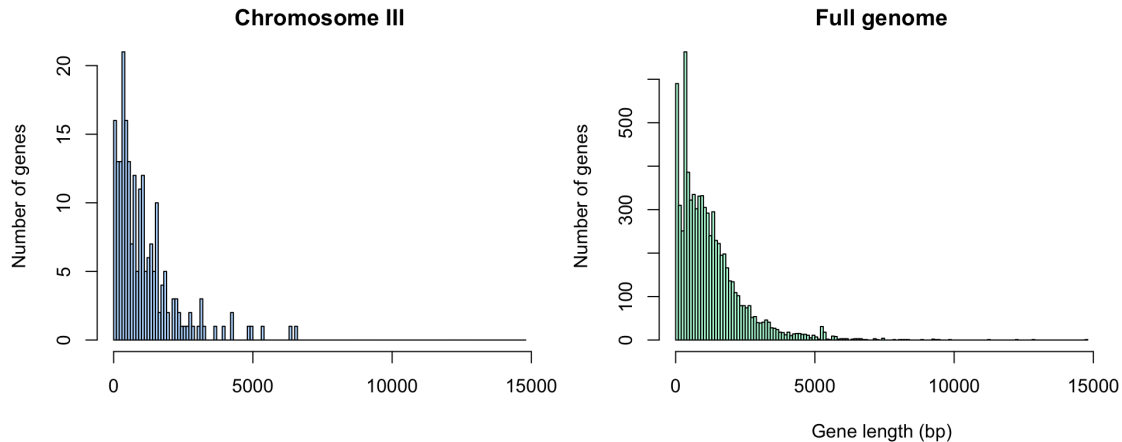


Figure 1: Distribution of gene lengths for *Saccharomyces cerevisiae*.

Analyse de la longueur des gènes de la levure du boulanger

Tutoriel

Avant de passer aux exercices, nous vous montrons ici quelques éléments de base concernant la lecture, la manipulation et l'écriture des tableaux de données avec R.

Chargement d'un tableau de données

R comporte plusieurs types de structures tabulaires (matrix, data.frame, table).

La structure la plus couramment utilisée est le `data.frame`, qui consiste en un tableau de valeurs (numériques ou chaînes de caractères) dont les lignes et les colonnes sont associées à des noms.

La fonction `read.table()` permet de lire un fichier texte contenant un tableau de données, et de stocker le contenu dans une variable.

Plusieurs fonctions dérivées de `read.table()` facilitent la lecture de différents types de formats:

- `read.delim()` pour les fichiers dont les colonnes sont délimitées par un caractère particulier (généralement la tabulation, représentée par “`\t`”).
- `read.csv()` pour les fichiers “comma-searated values”.

1. Téléchargez le fichier suivant sur votre ordinateur:

- `Saccharomyces_cerevisiae.R64-1-1.37.gtf`

2. Chargez-le au moyen de la fonction `read.table` (pour cela vous devez remplacer le chemin ci-dessous par celui de votre ordinateur).

```
## Read a GTF file with yeast genome annotations
##

## Build the full path to the local file (TO BE ADAPTED FOR YOUR COMPUTER)
gtf.file <- file.path(
  Sys.getenv("HOME"),
  "stat1", "data", "Saccharomyces_cerevisiae",
  "Saccharomyces_cerevisiae.R64-1-1.37.gtf")
```

```
## Load the feature table
feature.table <- read.delim(gtf.file, comment.char = "#", sep="\t", header=FALSE, row.names=NULL)

## The bed format does not contain any column header,
## so we set it manually based on the description of the format,
## found here:
## http://www.ensembl.org/info/website/upload/gff.html
names(feature.table) <- c("seqname", "source", "feature", "start", "end", "score", "strand", "frame", "score2")
```

Exploration du contenu d'un tableau de données

La première chose à faire après avoir chargé un tableau de données est de vérifier ses dimensions

```
dim(feature.table) ## Dimensions of the table
```

```
[1] 43028      9
```

```
nrow(feature.table) ## Number of rows
```

```
[1] 43028
```

```
ncol(feature.table) ## Number of columns
```

```
[1] 9
```

L'affichage du tableau d'annotations complet ne serait pas très lisible, puisqu'il comporte des dizaines de milliers de lignes.

Nous pouvons afficher les premières lignes avec la fonction `head()`.

```
## Display the 20 first rows of the feature table
head(feature.table, n = 20)
```

	seqname	source	feature	start	end	score	strand	frame
1	IV	SGD	gene	1802	2953	.	+	.
2	IV	SGD	transcript	1802	2953	.	+	.
3	IV	SGD	exon	1802	2953	.	+	.
4	IV	SGD	CDS	1802	2950	.	+	0
5	IV	SGD	start_codon	1802	1804	.	+	0
6	IV	SGD	stop_codon	2951	2953	.	+	0
7	IV	SGD	gene	3762	3836	.	+	.
8	IV	SGD	transcript	3762	3836	.	+	.
9	IV	SGD	exon	3762	3836	.	+	.
10	IV	SGD	CDS	3762	3833	.	+	0
11	IV	SGD	start_codon	3762	3764	.	+	0
12	IV	SGD	stop_codon	3834	3836	.	+	0
13	IV	SGD	gene	5985	7814	.	+	.
14	IV	SGD	transcript	5985	7814	.	+	.
15	IV	SGD	exon	5985	7814	.	+	.
16	IV	SGD	CDS	5985	7811	.	+	0
17	IV	SGD	start_codon	5985	5987	.	+	0
18	IV	SGD	stop_codon	7812	7814	.	+	0
19	IV	SGD	gene	8683	9756	.	-	.
20	IV	SGD	transcript	8683	9756	.	-	.

```
1
```

```
2 gene_id YDL248W; transcript_id YDL248W; gene_name YDL248W
```

```

3         gene_id YDL248W; transcript_id YDL248W; exon_number 1; gene_name COS7; gene_sour
4 gene_id YDL248W; transcript_id YDL248W; exon_number 1; gene_name COS7; gene_source SGD; gene_biotype
5         gene_id YDL248W; transcript_id YDL248W; exon_number 1; gene_na
6         gene_id YDL248W; transcript_id YDL248W; exon_number 1; gene_na
7
8
9         gene_id YDL247W-A; transcript_id YDL247W-A; exon
10        gene_id YDL247W-A; transcript_id YDL247W-A; exon_number 1; gene_sour
11        gene_id YDL247W-A; transcript
12        gene_id YDL247W-A; transcript
13
14        gene_id YDL247W; transcript_id YDL247W; gene_na
15        gene_id YDL247W; transcript_id YDL247W; exon_number 1; gene_name MPH2; gene_sour
16 gene_id YDL247W; transcript_id YDL247W; exon_number 1; gene_name MPH2; gene_source SGD; gene_biotype
17        gene_id YDL247W; transcript_id YDL247W; exon_number 1; gene_na
18        gene_id YDL247W; transcript_id YDL247W; exon_number 1; gene_na
19
20        gene_id YDL246C; transcript_id YDL246C; gene_na

```

La fonction `tail()` affiche les dernières lignes:

```
## Display the 20 first rows of the feature table
tail(feature.table, n = 20)
```

	seqname	source	feature	start	end	score	strand	frame
43009	Mito	Ensembl_Fungi	transcript	78533	78605	.	+	.
43010	Mito	Ensembl_Fungi	exon	78533	78605	.	+	.
43011	Mito	SGD	gene	79213	80022	.	+	.
43012	Mito	SGD	transcript	79213	80022	.	+	.
43013	Mito	SGD	exon	79213	80022	.	+	.
43014	Mito	SGD	CDS	79213	80019	.	+	0
43015	Mito	SGD	start_codon	79213	79215	.	+	0
43016	Mito	SGD	stop_codon	80020	80022	.	+	0
43017	Mito	SGD	gene	85035	85112	.	+	.
43018	Mito	SGD	transcript	85035	85112	.	+	.
43019	Mito	SGD	exon	85035	85112	.	+	.
43020	Mito	SGD	gene	85295	85777	.	+	.
43021	Mito	SGD	transcript	85295	85777	.	+	.
43022	Mito	SGD	exon	85295	85777	.	+	.
43023	Mito	SGD	gene	85554	85709	.	+	.
43024	Mito	SGD	transcript	85554	85709	.	+	.
43025	Mito	SGD	exon	85554	85709	.	+	.
43026	Mito	SGD	CDS	85554	85706	.	+	0
43027	Mito	SGD	start_codon	85554	85556	.	+	0
43028	Mito	SGD	stop_codon	85707	85709	.	+	0

```

43009        gene_id ENSRNA049602365; transcript_id ENSRNA04960
43010        gene_id ENSRNA049602365; transcript_id ENSRNA049602365-T1; exon_number 1; gene_name tRNA-Val;
43011
43012        gene_id Q0275; transcript_id Q0275; gene_name
43013        gene_id Q0275; transcript_id Q0275; exon_number 1; gene_name COX3; gene_sourc
43014 gene_id Q0275; transcript_id Q0275; exon_number 1; gene_name COX3; gene_source SGD; gene_biotype p
43015        gene_id Q0275; transcript_id Q0275; exon_number 1; gene_name
43016        gene_id Q0275; transcript_id Q0275; exon_number 1; gene_name
43017

```

```

43018
43019 gene_id tM(CAU)Q2; transcript_id
43020
43021
43022 gene_id RPM1; tran
43023
43024 gene_id
43025 gene_id Q0297; transcript_id Q0297; exon
43026 gene_id Q0297; transcript_id Q0297; exon_number 1; gene_sou
43027 gene_id Q0297; transcr
43028 gene_id Q0297; transcr

```

If you are using the **RStudio** environment, you can display the table in a dynamic viewer pane with the function `View()`.

```
## In RStudio, display the table in a separate tab
View(feature.table)
```

Sélection de sous-ensembles d'un tableau

Sélection d'une ligne par son indice.

```
feature.table[12,]
```

```

      seqname source    feature start  end score strand frame
12      IV    SGD stop_codon 3834 3836      .      +      0

```

```
12 gene_id YDL247W-A; transcript_id YDL247W-A; exon_number 1; gene_source SGD; gene_biotype protein_cod
```

Sélection d'une colonne par son indice (affichage des premières valeurs seulement).

```
head(feature.table[,3])
```

```

[1] gene      transcript exon      CDS      start_codon stop_codon
Levels: CDS exon gene start_codon stop_codon transcript

```

Sélection d'une cellule par indices de ligne et colonne.

```
feature.table[12, 3]
```

```

[1] stop_codon
Levels: CDS exon gene start_codon stop_codon transcript

```

Sélection d'un bloc de colonnes et/ou de lignes.

```
feature.table[100:105, 1:6]
```

```

      seqname source    feature start  end score
100      IV    SGD      CDS 34240 36477      .
101      IV    SGD start_codon 36475 36477      .
102      IV    SGD stop_codon 34237 34239      .
103      IV    SGD      gene 36797 38173      .
104      IV    SGD transcript 36797 38173      .
105      IV    SGD      exon 36797 38173      .

```

Sélection de colonnes "à la carte" (ici, les coordonnées génomiques de chaque "feature"): chromosome, début, fin, brin.

```
feature.table[100:105, c(1,4,5,7)]
```

	seqname	start	end	strand
100	IV	34240	36477	-
101	IV	36475	36477	-
102	IV	34237	34239	-
103	IV	36797	38173	+
104	IV	36797	38173	+
105	IV	36797	38173	+

Sélectionner une colonne sur base de son nom.

```
## Select the "start" column and print the 100 first results
head(feature.table$start, n=100)
```

```
[1] 1802 1802 1802 1802 1802 2951 3762 3762 3762 3762 3762
[12] 3834 5985 5985 5985 5985 5985 7812 8683 8683 8683 8686
[23] 9754 8683 11657 11657 11657 11660 13358 11657 16204 16204 16204
[34] 16204 16204 17224 17577 17577 17577 17580 18564 17577 18959 18959
[45] 18959 18959 18959 19310 20635 20635 20635 20635 20635 21004 22471
[56] 22471 22471 22474 22606 22471 22823 22823 22823 22823 22823 25874
[67] 26403 26403 26403 26406 28773 26403 28985 28985 28985 28988 30452
[78] 28985 30657 30657 30657 30657 30657 31827 32296 32296 32296 32296
[89] 32296 33232 33415 33415 33415 33418 33916 33415 34237 34237 34237
[100] 34240
```

```
## Print the 20 first values of the "feature" field, which indicates the feature type
head(feature.table$feature, n=20)
```

```
[1] gene      transcript exon      CDS      start_codon
[6] stop_codon gene      transcript exon      CDS
[11] start_codon stop_codon gene      transcript exon
[16] CDS      start_codon stop_codon gene      transcript
Levels: CDS exon gene start_codon stop_codon transcript
```

Sélection de plusieurs colonnes sur base de leurs noms.

```
## Select the "start" column and print the 100 first results
feature.table[100:106, c("seqname", "start", "end", "strand")]
```

	seqname	start	end	strand
100	IV	34240	36477	-
101	IV	36475	36477	-
102	IV	34237	34239	-
103	IV	36797	38173	+
104	IV	36797	38173	+
105	IV	36797	38173	+
106	IV	36797	38170	+

Note: il est également possible de nommer les lignes d'un data.frame mais le tableau GTF ne se prête pas à cela. Nous verrons d'autres exemples ultérieurement.

Sélection d'un sous-ensemble de lignes sur base du contenu d'une colonne

```
## Select subset of features having "gene" as "feature" attribute
genes <- subset(feature.table, feature=="gene")

nrow(feature.table) ## Count the number of features
```

```
[1] 43028
```

```
nrow(genes) ## Count the number of genes
```

```
[1] 7445
```

Décompte par valeur

La fonction `table()` permet de compter le nombre d'occurrences de chaque valeur dans un vecteur ou un tableau. Quelques exemples d'utilisation ci-dessous.

```
## Count the number of features per chromosome
table(feature.table$seqname)
```

```
  I   II  III   IV   IX Mito   V   VI  VII VIII   X   XI  XII XIII XIV
759 2912 1210 5374 1567   327 2159  946 3856 2054 2617 2231 3789 3311 2774
 XV  XVI
3846 3296
```

```
## Count the number of features per type
table(feature.table$feature)
```

```
      CDS      exon      gene start_codon stop_codon transcript
7050      7872      7445         6700         6516         7445
```

On peut calculer des tables de contingence en comptant le nombre de combinaisons entre 2 vecteurs (ou 2 colonnes d'un tableau).

```
## Table with two vectors
table(feature.table$feature, feature.table$seqname)
```

```
      I  II III  IV  IX Mito   V  VI VII VIII   X  XI XII XIII
CDS    122 492 194 895 255   59 345 151 619  346 422 361 615  544
exon   137 525 224 961 288   94 400 180 710  373 480 404 698  610
gene   132 494 213 914 274   62 383 167 676  349 458 388 658  573
start_codon 119 464 185 853 243  28 328 143 593  325 406 348 586  514
stop_codon  117 443 181 837 233  22 320 138 582  312 393 342 574  497
transcript 132 494 213 914 274   62 383 167 676  349 458 388 658  573
```

```
      XIV XV XVI
CDS    458 623 549
exon   500 689 599
gene   475 665 564
start_codon 438 607 520
stop_codon  428 597 500
transcript 475 665 564
```

```
## Same result with a 2-column data frame
table(feature.table[, c("feature", "seqname")])
```

```
      feature      seqname
      I  II III  IV  IX Mito   V  VI VII VIII   X  XI XII XIII
CDS    122 492 194 895 255   59 345 151 619  346 422 361 615  544
exon   137 525 224 961 288   94 400 180 710  373 480 404 698  610
gene   132 494 213 914 274   62 383 167 676  349 458 388 658  573
```


start_codon	119	464	185	853	243	28	328	143	593	325	406	348	586	514
stop_codon	117	443	181	837	233	22	320	138	582	312	393	342	574	497
transcript	132	494	213	914	274	62	383	167	676	349	458	388	658	573
	seqname													
feature	XIV	XV	XVI											
CDS	458	623	549											
exon	500	689	599											
gene	475	665	564											
start_codon	438	607	520											
stop_codon	428	597	500											
transcript	475	665	564											

Exercices

1. Spécifications du format GTF

Lisez les spécifications du format GTF.

- Ensembl (<http://www.ensembl.org/info/website/upload/gff.html>)
- UCSC (<https://genome.ucsc.edu/FAQ/FAQformat.html#format4>)

2. Création d'un dossier local pour le TP

Créez un dossier local (par exemple: `stat1/TP_levure` à partir de la racine de votre compte). Nous vous suggérons d'utiliser les fonctions suivantes:

- `Sys.getenv("HOME")` (Linux et Mac OS X), pour obtenir la racine de votre compte utilisateur;
- `file.path()` pour construire un chemin;
- `dir.create()` pour créer le dossier de ce TP. Lisez attentivement les options de cette fonction avec `help(dir.create)`

3. Localisation du fichier d'annotations

Localisez le fichier d'annotations du génome de la levure en format GTF dans ce dossier local.

- Site Ensembl Fungi: <http://fungi.ensembl.org/>
- Cliquez "Downloads" pour accéder au site ftp
- Dans la boîte de recherche, tapez "*saccharomyces cerevisiae*" et suivez le lien "GTF"
- Copiez l'adresse (URL) du fichier *Saccharomyces_cerevisiae.R64-1-1.37.gtf.gz*

4. Téléchargement d'un fichier à partir d'un site ftp

Fonctions suggérées:

- `download.file()` (lisez l'aide pour connaître les arguments)

5. Chargement d'une table de données en R

Ecrivez un script qui charge la table de données dans une variable nommée `feature.table`, en utilisant la fonction R `read.delim()`.

Veillez à ignorer les lignes de commentaires (qui commencent par un caractère #).

5. Calcul de la longueur des gènes

- Ajoutez à la table une colonne intitulée “length” qui indique la longueur de chaque élément génomique annoté.
 - fonction `table()`
- Comptez le nombre de lignes de la table correspondant à chaque type d’annotation (3ème colonne du GTF, “feature”).
 - fonction `subset()`
- Comptez le nombre de gènes par chromosome.
 - fonction `table()`
- Chargez la table des tailles de chromosomes `chrom_sizes.tsv`, et calculez la densité de gènes pour chaque chromosome (nombre de gènes par Mb).

6. Histogramme de la longueur des gènes

Au moyen de la fonction `hist()`, dessinez un histogramme représentant la distribution de longueur des gènes. Choisissez les intervalles de classe de façon à ce que l’histogramme soit informatif (ni trop ni trop peu de classes).

Récupérez le résultat de `hist()` dans une variable nommée `gene.length.hist`.

Imprimez le résultat à l’écran (`print()`) et analysez la structure de la variable `gene.length.hist` (il s’agit d’une variable de type liste).

Fonctions utiles:

- `class(gene.length.hist)`
- `attributes(gene.length.hist)`

7. Paramètres descriptifs

Calculez les paramètres de tendance centrale (moyenne, médiane, mode) et de dispersion (variance, écart-type, écart inter-quartile)

- pour les gènes du chromosome III;
- pour l’ensemble des gènes de la levure.

Affichez ces paramètres sur l’histogramme de la longueur des gènes, en utilisant la fonction `arrows()`

8. Intervalle de confiance

A partir des gènes du chromosome III (considérés comme l’échantillon disponible en 1992), calculez un intervalle de confiance autour de la moyenne, et formulez l’interprétation de cet intervalle de confiance. Évaluez ensuite si cet intervalle de confiance recouvrait ou non la moyenne de la population (tous les gènes du génome de la levure, qui devint disponible 4 ans après le chromosome III).

9. Distribution de la longueur des gènes

- A partir du résultat de `hist()`, récupérez un tableau (dans une variable de type `data.frame`) indiquant les fréquences absolues (`count`) en fonction de la taille médiane des classes (`mids`),
- Ajoutez à ce tableau une colonne indiquant la fréquence relative de chaque classe de longueurs de gènes.
- Ajoutez à ce tableau des colonnes indiquant la **fonction de répartition empirique** des longueurs de gènes (nombre de gènes d'une taille inférieure ou égale à chaque valeur x observée, et fréquence relative de ce nombre).
 - fonction de base: `cumsum()`
 - fonction avancée: `ecdf()`
- Au moyen des fonctions `plot()` et `lines()`, dessinez un graphe représentant la fréquence absolue par classe (médianes de classes en X , comptages en Y), et la fonction de répartition empirique.
 - suggestion: superposez les utilisez le type de lignes “h” pour les fréquences de classe, et “l” ou “s” pour la fonction de répartition.

10. Distribution attendue au hasard pour la longueur des gènes

Sur base de la taille du génome (12.156.679 bp) et des fréquences génomiques de codons définies ci-dessous, calculez la distribution de longueurs de gènes attendue au hasard, et ajoutez-là au graphique.

Vous pouvez télécharger les fréquences génomiques de tous les trinuécléotides ici: `3nt_genomic_Saccharomyces_cerevisiae-ovlp-1str.tab`

Alternative: créez une variable `freq.3nt` et assignez-y manuellement les valeurs pour les 4 nucléotides nécessaires, à partir de la table ci-dessous.

sequence	frequency	occurrences
AAA	0.0394	478708
ATG	0.0183	221902
TAA	0.0224	272041
TAG	0.0129	156668
TGA	0.0201	244627

11. Avant de terminer : conservez la trace de votre session

La traçabilité constitue un enjeu essentiel en sciences. La fonction `R sessionInfo()` fournit un résumé des conditions d'une session de travail: version de R, système opérateur, bibliothèques de fonctions utilisées.

```
sessionInfo()
```

```
R version 3.3.2 (2016-10-31)
```

```
Platform: x86_64-apple-darwin13.4.0 (64-bit)
```

```
Running under: macOS Sierra 10.12.6
```

```
locale:
```

```
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods   base
```

```
other attached packages:
```

```
[1] knitr_1.17
```

```
loaded via a namespace (and not attached):
```

```
[1] backports_1.1.0 magrittr_1.5      rprojroot_1.2  tools_3.3.2  
[5] htmltools_0.3.6 yaml_2.1.14    Rcpp_0.12.12   stringi_1.1.5  
[9] rmarkdown_1.6   highr_0.6      stringr_1.2.0  digest_0.6.12  
[13] evaluate_0.10.1
```

Rendu