Fiche TD avec le logiciel  : tdr313

Simulation avec un peu de programmation : la plus longue suite

C. Gautier & J.R. Lobry

Examen Magister (2002) statistiques non-paramétriques (avec solutions)

Table des matières

1 Introduction

L'objectif est de construire par simulation un test pour décider si la plus longue suite homogène (soit entièrement de purines soit entièrement de pyrimidine) dans une séquence est significative ou pas, compte tenu bien entendu des fréquences respectives des purines et pyrimidines.

Remarque : il existe une solution mathématique exacte au problème qui dépasse le cadre de cet examen.

On se place dans une région du génome dans laquelle on suppose que la fréquence des purines est de 0.45 (et donc celle des pyrimidines est de 0.55 !). En notant 1 les purines et 2 les pyrimidines la séquence observée est :

```
12111222212211112121112122222222222211212221111121211111
```

2 Tester si la fréquence des bases est compatible avec les fréquences dans la région

Vous regarderez la documentation de la fonction R `chisq.test` et l'utiliserez pour tester si on peut accepter que les fréquences dans la séquence sont 0.45 et 0.55.

On commence par entrer les données sous la forme d'un vecteur d'entiers :

```
data <- strsplit("12111222212211112121112122222222222211212221111112121111",
    split = "")
data <- as.integer(unlist(data))
n <- length(data)
data
```

```
 [1] 1 2 1 1 1 2 2 2 2 1 2 2 1 1 1 1 2 1 2 1 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1
[41] 2 2 2 1 1 1 1 1 2 1 2 1 1 1 1
```

Nous avons donc $n = 55$ nucléotides dans notre séquence. La documentation de la fonction `chisq.test` nous dit :

```
Details:

     If 'x' is a matrix with one row or column, or if 'x' is a vector
     and 'y' is not given, 'x' is treated as a one-dimensional
     contingency table.  In this case, the hypothesis tested is whether
     the population probabilities equal those in 'p', or are all equal
     if 'p' is not given.
```

```
table(data)
```

```
data
 1  2
27 28
```

```
chisq.test(table(data), p = c(0.45, 0.55))
```

```
        Chi-squared test for given probabilities

data:  table(data)
X-squared = 0.3719, df = 1, p-value = 0.542
```

Avec un risque de première espèce de 5% les fréquences observées dans notre séquence ne nous permettent pas de rejeter l'hypothèse que celles ci soient égales à 0.45 et 0.55. On accepte donc, faute de mieux, que les fréquences dans la séquence soient compatibles avec les fréquences dans la région.

3 Écrire une fonction qui simule une séquence de fréqunce de pyrimidine donnée

Vous utiliserez la fonction de génération au hasard du résultat d'une loi binomiale.

On peut le faire directement avec la fonction `sample` :

```
sim <- function(p = 0.45) {
    sample(x = c(1, 2), size = n, replace = TRUE, prob = c(p, 1 -
        p))
}
sim()
```

```
 [1] 1 2 1 1 2 2 1 2 2 2 2 2 1 2 2 1 2 2 1 2 2 1 1 2 1 1 1 2 2 1 2 1 2 1 2 2 1 2 2 2
[41] 2 2 2 2 2 1 2 2 2 2 1 1 2 1 1
```

Mais on demandait d'utiliser la fonction `rbinom`, donc on l'utilise pour construire l'urne dans laquelle on pioche sans remise :

```
sim2 <- function(p = 0.45) {
    npyr <- rbinom(n = 1, size = n, prob = p)
    urne <- rep(c(1, 2), c(npyr, n - npyr))
    sample(x = urne, size = n, replace = FALSE)
}
sim2()
```

```
 [1] 2 2 1 2 1 1 2 1 2 2 1 1 1 2 1 2 2 1 2 2 2 1 2 1 1 1 2 2 1 2 1 1 1 2 2 2 2 1 1 1
[41] 1 1 2 2 2 1 1 2 1 2 1 2 1 1 2
```

4 Écrire une fonction qui calcule la plus longue suite d'une séquence

Vous appellerez `pls` cette fonction et en recopierez le listing sur la copie.

Si on se souvient que suite se dit *run* en anglais, un appel à `help.search("run")` nous donne une solution immédiate :

```
pls <- function(x = data) {
    max(rle(x)$lengths)
}
pls()
```

```
[1] 12
```

On peut aussi implémenter l'algorithme à la main :

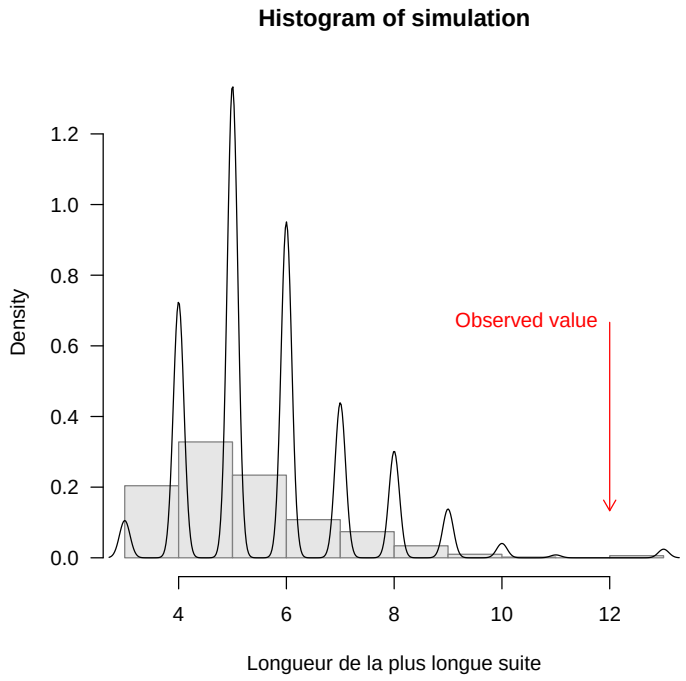
```
pls2 <- function(x = data) {
    n <- length(x)
    global.max <- 1
    local.max <- 1
    for (i in 1:(n - 1)) {
        if (x[i] == x[i + 1]) {
            local.max <- local.max + 1
        }
        else {
            if (local.max > global.max) {
                global.max <- local.max
            }
            local.max <- 1
        }
    }
    max(local.max, global.max)
}
pls2()
```

```
[1] 12
```

5 Simulation

Vous simulerez 500 séquences de même longueur que la séquence observée avec comme fréquence 0.45 et 0.55, pour chacune d'elle vous calculerez la longueur de la plus longue suite. Tracer l'histogramme de ces longueurs.

```
simulation <- sapply(1:500, function(x) {
    pls(sim())
})
dst <- density(simulation, adjust = 0.5)
hist(simulation, proba = TRUE, col = grey(0.9), border = grey(0.5),
    las = 1, ylim = c(0, max(dst$y)), xlim = range(c(simulation,
        pls(data))), xlab = "Longueur de la plus longue suite")
lines(dst$x, dst$y)
arrows(x0 = pls(data), x1 = pls(data), y0 = max(dst$y/2), y1 = max(dst$y/10),
    col = "red", length = 0.1)
text(pls(data), max(dst$y/2), "Observed value", col = "red", pos = 2)
```

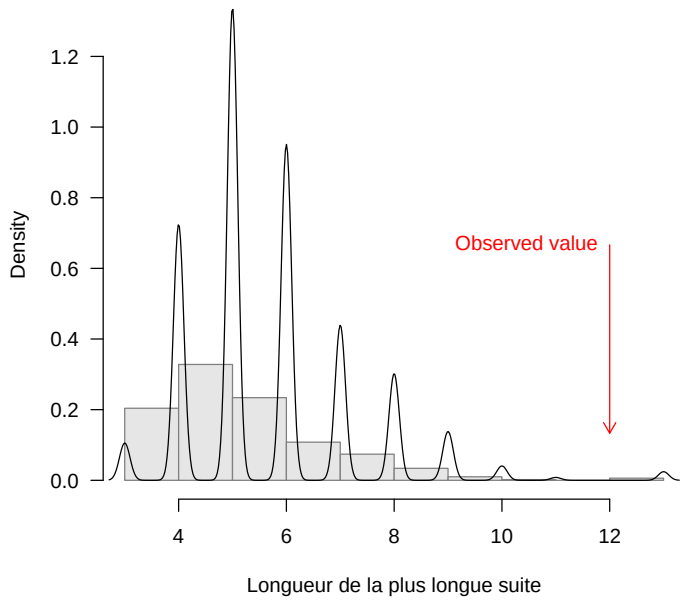


La valeur observée peut s'obtenir par hasard, mais ça n'arrive pas très souvent. Plus précisément, dans notre simulation nous avons $100\frac{7}{500} \approx 1.4$ % de cas où la plus longue suite est supérieure ou égale à la valeur observée. Dans ces conditions, avec un risque de première espèce de 5 % on rejette l'hypothèse nulle : la plus longue suite observée est anormalement longue.

Pour construire ce test nous avons fait l'hypothèse que les séquences sont construites par tirage avec remise dans une urne de composition donnée *a priori*. Le résultat du test de la question 1 est rassurant mais pas probant : on peut s'interroger sur la puissance de ce premier test pour une séquence aussi courte. Affranchissons nous de cette hypothèse en travaillant avec des séquences simulées par simple permutation, ainsi toutes les séquences auront exactement la même composition globale que la séquence observée :

```
simulation <- sapply(1:500, function(x) {
    pls(sample(data))
})
dst <- density(simulation, adjust = 0.5)
hist(simulation, proba = TRUE, col = grey(0.9), border = grey(0.5),
    las = 1, ylim = c(0, max(dst$y)), xlim = range(c(simulation,
        pls(data))), xlab = "Longueur de la plus longue suite")
lines(dst$x, dst$y)
```

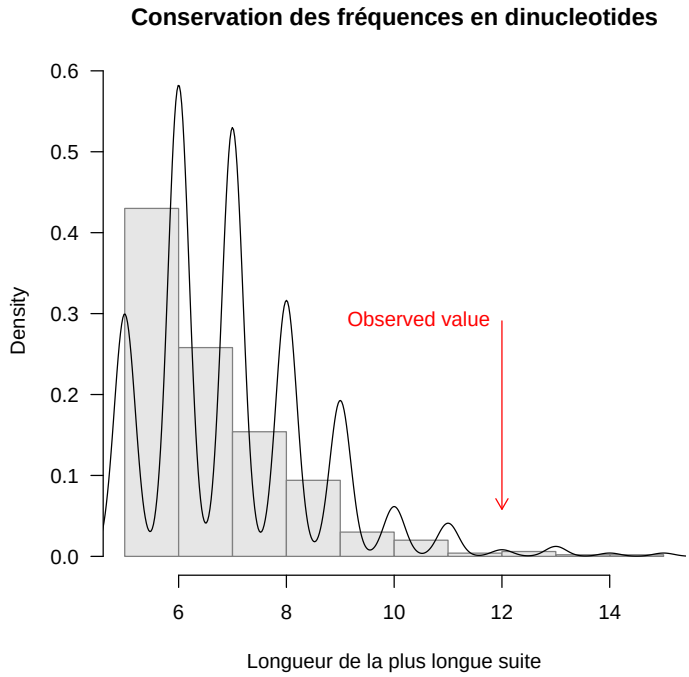
```
arrows(x0 = pls(data), x1 = pls(data), y0 = max(dst$y/2), y1 = max(dst$y/10),
    col = "red", length = 0.1)
text(pls(data), max(dst$y/2), "Observed value", col = "red", pos = 2)
```



C'est rassurant, quand on détruit l'ordre de la séquence observée, la longueur de la plus longue suite chute de façon significative.

Le modèle utilisé pour la simulation est assez simpliste puisque la seule contrainte est la conservation de la fréquence des nucléotides. On pourrait imaginer qu'il y a des contraintes locales, par exemple sur la fréquence en dinucléotides. Que se passe-t-il si on permute la séquence sous la contrainte de la conservation exacte des fréquences de dinucléotides ?

```
tabledn <- function(x = data) {
    tmp <- paste(x[-n], x[-1], sep = "")
    table(factor(tmp, levels = c(11, 12, 21, 22)))
}
simdn <- function(x = data) {
    refdn <- tabledn(x)
    result <- sample(x)
    while (!identical(refdn, tabledn(result))) {
        result <- sample(x)
    }
    result
}
simulation <- sapply(1:500, function(x) {
    pls(simdn(data))
})
dst <- density(simulation, adjust = 0.5)
hist(simulation, proba = TRUE, col = grey(0.9), border = grey(0.5),
    las = 1, ylim = c(0, max(dst$y)), xlim = range(c(simulation,
        pls(data))), xlab = "Longueur de la plus longue suite",
    main = "Conservation des fréquences en dinucleotides")
lines(dst$x, dst$y)
arrows(x0 = pls(data), x1 = pls(data), y0 = max(dst$y/2), y1 = max(dst$y/10),
    col = "red", length = 0.1)
text(pls(data), max(dst$y/2), "Observed value", col = "red", pos = 2)
```



C'est toujours assez bon, même si l'on conserve les fréquences en dinucléotides, la plus longue suite observée est anormalement grande.