


TD6 MODÈLE LOGISTIQUE DISCRET

Le TD est en lien avec le **problème 5** de la matière « Introduction à la modélisation en SVT ». Les questions avec  sont notées uniquement pour les étudiants de la licence de mathématiques et d'informatique.

Exercice 6.1 Évolution temporelle

Soit un modèle logistique

$$p_{n+1} = (1 + b - m - \alpha \times p_n) \times p_n \quad (9)$$

décrivant l'évolution journalière de bactéries sur un biofilm. Les coefficients sont p_n le nombre de bactéries à un jour donné et p_{n+1} le lendemain, b le taux de division (équivalent à un taux de reproduction), m le taux de mortalité et α le coefficient lié à la compétition entre les bactéries.

6.1.1 Affecter les valeurs aux variables suivantes :

- $m = 0.2$, le taux de mortalité ;
- $\alpha = 0.004$, compétition entraînant une mortalité additionnelle.
- $p_0 = 10$, le nombre de bactéries initiales

6.1.2 Construire une fonction `evolution` qui a pour :

- ➡ **argument** une valeur numérique nommée `actuel` : la population à un instant t
- ➡ **argument** une valeur numérique nommée `b` : le paramètre b
- retour** ➡ une valeur numérique entière qui est l'évolution à $t + 1$ selon (9)

Utiliser la fonction `floor` pour transformer un nombre décimal en un nombre entier.

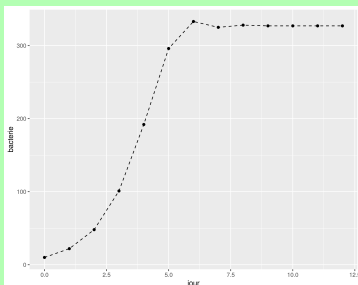
6.1.3 Construire un tibble `donnee` qui contient les données de l'évolution des bactéries en prenant en compte la valeur de b . La construction peut se faire avec l'algorithme A1 ou la fonction `accumulate` mais doit utiliser la fonction `evolution`. Ce tibble possède trois *variables* :

- `jour` de 0 à 12
- `b` qui vaut 1.5
- `bacterie` qui contient le nombre de bactéries

6.1.4 Tracer l'évolution du nombre de bactéries en fonction des jours avec des lignes en pointillé et des points.

Correction 6.1

```
1 setwd(dirname(rstudioapi::getActiveDocumentContext())$path))
2 library(tidyverse)
3
4 # Q.6.1.1
5 m = 0.2
6 alpha = 0.004
7 p_0 = 10
8
9 # Q 6.1.2
10 evolution <- function(actuel, b){
11   return(floor((1 + b - m - alpha * actuel) * actuel ))
12 }
13
14 # Q 6.1.3
15 # Avec une boucle pour
16 bacterie <- numeric(12)
17 bacterie[1] <- p_0
18 for (jour in 1:12){
19   bacterie[jour+1] = evolution(bacterie[jour], 1.51)
20 }
21 # Puis la création du tibble
22 donnee <- tibble(jour = 0:12,
23                  b = rep(1.51, 13),
24                  bacterie)
25
26 # En programmation fonctionnelle
27 donnee <- tibble(jour = 0:12) %>%
28   mutate(b = 1.51) %>%
29   mutate(bacterie = accumulate(.x = b[-1],
30                                .f = evolution,
31                                .init = p_0))
32
33 # Q 6.1.4
34 ggplot(data = donnee) +
35   aes(x = jour,
36        y = bacterie) +
37   geom_point() +
38   geom_line(linetype = "dashed")
```



Exercice 6.2 Évolution temporelle

L'objectif est de faire une représentation graphique de l'évolution temporelle des données en faisant varier le paramètre b parmi les valeurs suivantes : $b \in \{1.51, 2.5, 3\}$.

6.2.1 En reprenant l'algorithme A2 du cours, utiliser une boucle *pour* / boucle **for** pour :

- construire trois tibbles avec les paramètres différents jusqu'au 21^{ème} jour
- combiner ces tibbles pour créer un unique tibble `donnee_plot`

Au final, il faut avoir un tibble ayant 66 observations des 3 *variables* suivantes :

- `jour`
- `b`
- `bacterie`

A Attention à bien différencier les variables quantitatives des variables qualitatives.

6.2.2 Tracer l'évolution des bactéries en fonction du temps avec des lignes et des points.

6.2.3 Tracer l'évolution des bactéries en fonction du temps avec des lignes et des points en précisant que l'allure des lignes est différente pour chaque paramètre.

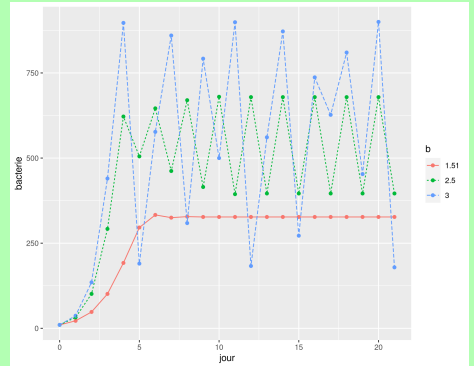
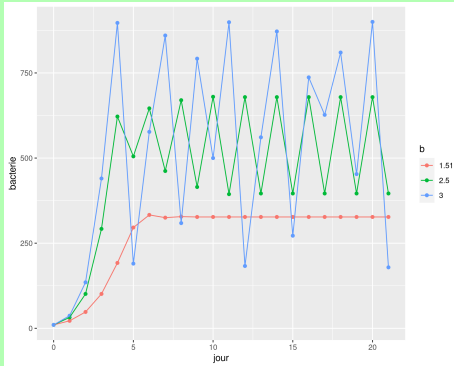
Correction 6.2

```
1 setwd(dirname(rstudioapi::getActiveDocumentContext())$path))
2 library(tidyverse)
3
4 m = 0.2
5 alpha = 0.004
6 p_0 = 10
7
8 evolution <- function(actuel, b){
9   return(floor((1 + b - m - alpha * actuel) * actuel))
10 }
11
12 # Avec une boucle pour
13 donnee_plot <- tibble()
14 for (b in c(1.51, 2.5, 3)){
15   bacterie <- numeric(20)
16   bacterie[1] <- p_0
17   for (jour in 1:20){
18     bacterie[jour+1] = evolution(bacterie[jour], b)
19   }
20   donnee <- tibble(jour = 0:20,
21                   bacterie) %>%
22     mutate(b = b)
23   donnee_plot <- bind_rows(donnee_plot, donnee)
24 }
25 donnee_plot <- mutate(donnee_plot, b = factor(b))
26
27 # En programmation fonctionnelle
28 donnee_plot <- tibble(jour = 0:21) %>%
```

```

29 expand(jour, b = c(1.51, 2.5, 3.0)) %>%
30 group_by(b) %>%
31 mutate(bacterie = accumulate(.x = b[-1],
32                               .f = evolution,
33                               .init = p_0)) %>%
34 mutate(b = factor(b))
35
36 ggplot(data = donnee_plot) +
37   aes(x = jour,
38       y = bacterie,
39       color = b) +
40   geom_point() +
41   geom_line()
42 ggsave("factor_1.png")
43
44 ggplot(data = donnee_plot) +
45   aes(x = jour,
46       y = bacterie,
47       linetype = b,
48       color = b) +
49   geom_point() +
50   geom_line()
51 ggsave("factor_2.png")

```

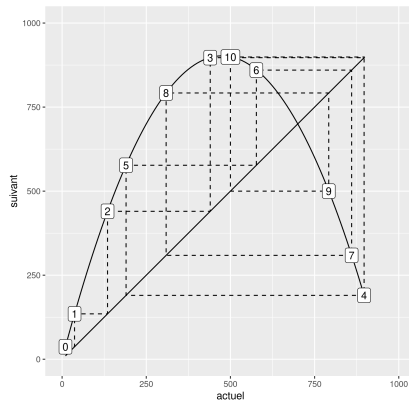


Exercice 6.3 🧩 Évolution temporelle

L'objectif est de reproduire le graphique où l'on reporte les points successifs de l'évolution temporelle sur l'application logistique. Cette courbe donne le nombre de bactéries suivant en fonction du nombre actuel de bactéries :

$$f(x) = (1 + b - m - \alpha \times x) \times x \quad (10)$$

Les paramètres sont ceux de l'exercice 6.1 et $b = 3$. Voici les opérations à effectuer pour obtenir la visualisation suivante



6.3.1 Comme dans les exercices précédents, créer des données pour 11 jours et stocker ces valeurs dans un tibble `donnee` : il a 12 *observations* de 3 *variables* (`jour`, `b`, `actuel`).

6.3.2 En utilisant la fonction `lead` et le tibble `donnee` créer un tibble `donnee_decale` où les valeurs sont dans deux variables `actuel` et `suivant`. Voici un aperçu des premières valeur :

```
1 > head(donnee_decale)
2 # A tibble: 6 × 4
3   jour     b actuel suivant
4   <int> <dbl> <dbl>   <dbl>
5 1     0     3    10     37
6 2     1     3    37    135
7 3     2     3   135   440
8 4     3     3  440   897
9 5     4     3  897   190
10 6     5     3  190   577
```

6.3.3 Créer un tibble `donnee_bissectrice` à partir de `donnee_decale` ayant les valeurs de `suivant` dans `actuel` pour avoir les points sur la bissectrice.

```
1 > head(donnee_bissectrice)
2 # A tibble: 6 × 4
3   jour     b actuel suivant
4   <int> <dbl> <dbl>   <dbl>
5 1     0     3    37    37
6 2     1     3   135   135
7 3     2     3  440   440
8 4     3     3  897   897
9 5     4     3  190   190
10 6     5     3  577   577
```

6.3.4 Créer un tibble `donnee_application` qui en réalisant séquentiellement les opérations suivantes

1. Combine le tibble `donnee_decale` avec ensuite le tibble `donnee_bissectrice`

2. Trie l'ensemble selon la *variable jour*
3. Supprime tous les `NA` du tibble
4. Ajoute une nouvelle variable `etape` en utilisant `case_when` : si `actuel` est différent de `suivant` alors `etape` vaut `jour`, sinon, `etape` vaut `NA`

6.3.5 Reproduire la figure présentée au début de l'exercice en utilisant :

- `geom_path` pour relier les points dans l'ordre d'apparition dans le tibble
- `geom_function` pour tracer la bissectrice et la fonction (10)
- `geom_label` pour afficher l'ordre d'apparition des points.

Correction 6.3

```
1 setwd(dirname(rstudioapi::getActiveDocumentContext())$path))
2 library(tidyverse)
3
4 # Q 6.3.1
5 b <- 3
6 m <- 0.2
7 alpha <- 0.004
8 p_0 <- 10
9
10 evolution <- function(actuel, b){
11   return(floor((1 + b - m - alpha * actuel) * actuel))
12 }
13
14 donnee <- tibble(jour = 0:11) %>%
15   mutate(b = 3) %>%
16   mutate(bacterie = accumulate(.x = b[-1],
17                               .f = evolution,
18                               .init = p_0))
19
20 # Q 6.3.2
21 donnee_decale <- donnee %>%
22   mutate(suivant = lead(bacterie)) %>%
23   rename(actuel = bacterie)
24
25 # Q 6.3.3
26 donnee_bissectrice <- donnee_decale %>%
27   mutate(actuel = suivant)
28
29 # Q 6.3.4
30 donnee_application <- bind_rows(donnee_decale, donnee_bissectrice) %>%
31   arrange(jour) %>%
32   drop_na() %>%
33   mutate(etape = case_when(actuel != suivant ~ jour))
34
35 # Q 6.1.3
36 ggplot(data = donnee_application) +
37   aes(x = actuel,
38       y = suivant,
```

```

39   label = etape) +
40   geom_path(size = 0.5, linetype = "dashed") +
41   geom_function(fun = ~ .x) +
42   geom_function(fun = ~ (1 + b - m - alpha *.x)*.x) +
43   geom_label() +
44   coord_fixed(xlim = c(0, 1000),
45               ylim = c(0, 1000))
46   ggsave("application_1.png")
47
48   # Alternative colorée
49   ggplot(data = donnee_application) +
50     aes(x = actuel,
51         y = suivant,
52         color = etape) +
53     geom_path(size = 0.5, linetype = "dashed", color = "black") +
54     geom_function(fun = ~ .x) +
55     geom_function(fun = ~ (1 + b - m - alpha *.x)*.x) +
56     geom_point(size = 4) +
57     coord_fixed(xlim = c(0, 1000),
58                 ylim = c(0, 1000))
59     ggsave("application_2.png")

```

