Programmer avec R

Lise Vaudor

2016-09-12

Contents

Contexte de l'exercice

Dans ce tutoriel nous allons construire un programme par étapes successives. Chaque exercice correspond à une de ces étapes, il sont donc destinés à être résolus dans l'ordre.

Vous tiendrez tout au long de ce TD le rôle d'un gérant de zoo, qui s'interroge sur la comptabilité de son établissement... Vous serez ainsi amenés à manipuler différents tableaux de données, de faire des petits calculs, quelques graphiques de synthèse, etc.

En ce sens, il est conçu pour être le plus proche possible de ce que vous êtes amenés à faire assez classiquement dans vos recherches (tout en traitant d'objets un peu plus glamours au yeux de l'individu lambda si je puis me permettre-).

Votre **script sera le support de votre travail** durant toute la formation, aussi n'oubliez pas de l'enregistrer, de le commenter, de le bichonner, en un mot, de lui accorder toute votre attention!

Objectifs

- Etre en mesure d'automatiser des traitements répétitifs (boucles "for" et "while")
- Pouvoir réaliser des traitements spécifiques en fonction de la réalisation de certaines conditions (structures de contrôle "if")
- Savoir créer des fonctions
- Savoir manipuler/découper/recoller des chaînes de caractère
- Savoir gérer les environnements, savoir assigner de manière automatique des noms à des objets.
- Savoir manipuler (lister/copier/supprimer, etc.) des fichiers et dossiers
- Savoir lire/écrire des fichiers de formats particuliers (par exemple du texte brut, des images, etc.)

1 Mettre en forme les données de consommation alimentaire des différentes espèces

Le dossier fourni contient un sous-dossier "fiches_especes". Ouvrez quelques unes de ces fiches. Elles sont toutes organisées selon un modèle commun. Nous souhaiterions mettre toutes les informations qu'elles contiennent sous forme d'un seul et même tableau, qui ressemblerait à ceci:

```
  categorie   espece effectifs pourcentage poids prix poids_total
1  fourrage     lion         6           0     0 0.14           0
2  fourrage    hyene         5           0     0 0.14           0
3  fourrage   iguane         5           0     0 0.14           0
```

```
4  fourrage     lynx         4           0     0 0.14           0
5  fourrage  chameau         6           1    16 0.14          96
6  fourrage antilope         7           1     3 0.14          21
  prix_total
1       0.00
2       0.00
3       0.00
4       0.00
5      13.44
6       2.94
```

(Ici je n'ai affiché que les premières lignes du tableau en question...)

Nous allons pour cela procéder en plusieurs étapes...

1.1 liste des fichiers

Créer un vecteur correspondant aux noms des fichiers contenus dans le dossier "data/fiche_especes". Donner un nom à ce vecteur (par exemple, "fiches")

Intérêt:

- chemins (relatif ou absolu) vers un dossier
- exploration de dossiers (fonctions getwd, setwd, dir)

1.2 liste des espèces

Créer un vecteur correspondant aux noms de tous les animaux faisant l'objet d'une fiche. Donner un nom à ce vecteur (par exemple, "especes").

Intérêt:

- manipulation de chaînes de caractère (gsub)

1.3 lecture d'une fiche

Lire et afficher le contenu du fichier "fiche_antilope.csv" dans la console

Intérêt:

- lecture de fichiers de données

1.4 fonction pour lecture d'une fiche

Créer une fonction "data_espece" ayant pour argument le nom d'un animal et renvoyant en sortie la table de données associée à cet animal

Intérêt:

- écriture d'une fonction
- paste

1.5 utilisation de la fonction pour lecture d'une fiche

Lire et afficher les données relatives à chaque animal. On se servira du vecteur "especes", et de la fonction "data_espece"

Intérêt:

- boucle for

1.6 info sur la quantité de nourriture pour une espèce

Afficher la ligne donnant la quantité de nourriture quotidienne pour chaque espèce.

Intérêt:

- classes des objets
- système d'indexation

1.7 quantité de nourriture pour une espèce

Afficher la quantité de nourriture quotidienne (par exemple, "300g" et non comme auparavant "quantite: 300g") pour chaque espèce.

Intérêt:

- manipulation de chaînes de caractère (fonction strsplit)

1.8 quantité de nourriture pour une espèce (en kg)

Répéter l'exercice précédent, mais faire en sorte que toutes les quantités soient exprimées en kg.

Intérêt:

- manipulation de chaînes de caractère (fonction grep, gsub)
- opérateurs arithmétiques

1.9 création d'un tableau (espèce,quantité)

Répéter l'exercice précédent, mais stocker le résultat dans un tableau ayant deux colonnes:

- une première colonne correspondant aux noms des espèces
- une deuxième colonne correspondant aux quantités de nourriture quotidienne (en kg)

```
     especes quantites
1   antilope     3.000
2   autruche    11.000
3    chameau    16.000
4 crocodiles     0.400
5   elephant   200.000
6     fennec     0.055
```

1.10 écriture de la fonction traite_quantite

Créer une fonction “traite_quantite” qui opère la transformation suivante:

```
traite_quantite("quantite: 320g")
```

```
[1] 0.32
```

```
traite_quantite("quantite: 3kg")
```

```
[1] 3
```

```
traite_quantite("quantite: 800g")
```

```
[1] 0.8
```

Intérêt:

- écriture d’une fonction

1.11 utilisation de la fonction traite_quantite

Simplifier le résultat de l’exercice n-2 en utilisant la fonction traite_quantite.

Intérêt:

- simplification d’un code à l’aide d’une fonction

1.12 écriture de la fonction traite_chaine

Créer une fonction “traite_chaine” qui opère la transformation suivante:

```
traite_chaine("viande: 20%")
```

```
$aliment
[1] "viande"

$proportion
[1] 0.2
```

```
traite_chaine("poisson: 10%")
```

```
$aliment
[1] "poisson"

$proportion
[1] 0.1
```

Intérêt:

- listes et système d’indexation
- écriture d’une fonction

1.13 info quant aux catégories d'aliment pour une espèce

Pour une espèce donnée (par exemple, l'autruche), créer la table suivante, qui donne le pourcentage et le poids de chaque catégorie d'aliment pour un individu de l'espèce considérée.

```
    espece categorie pourcentage poids
1 autruche    viande        0.05  0.55
2 autruche   poisson        0.00  0.00
3 autruche    fruits        0.20  2.20
4 autruche   legumes        0.75  8.25
5 autruche  fourrage        0.00  0.00
```

1.14 création de la table alim (toutes espèces)

Créer la table "alim" qui donne le pourcentage de la ration et le poids de chaque catégorie d'aliment pour un individu de chaque espèce considérée.

```
     espece categorie pourcentage poids
1  antilope    viande        0.00  0.00
2  antilope   poisson        0.00  0.00
3  antilope    fruits        0.00  0.00
4  antilope   legumes        0.00  0.00
5  antilope  fourrage        1.00  3.00
6  autruche    viande        0.05  0.55
7  autruche   poisson        0.00  0.00
8  autruche    fruits        0.20  2.20
9  autruche   legumes        0.75  8.25
10 autruche  fourrage        0.00  0.00
11  chameau    viande        0.00  0.00
12  chameau   poisson        0.00  0.00
13  chameau    fruits        0.00  0.00
14  chameau   legumes        0.00  0.00
15  chameau  fourrage        1.00 16.00
```

(Ici je n'ai affiché que les 15 premières lignes de cette table)

Intérêt:

- écrire des boucles imbriquées
- "augmenter" un jeu de données (fonction "rbind")

1.15 écriture de la table alim dans un fichier

Ecrire la table alim dans un fichier (format .csv).

Intérêt:

- écriture de fichiers

2 Analyser le budget alimentaire du zoo

2.1 faire une jointure de deux tables

Charger la table “effectifs” qui indique le nombre d’individus total de chaque espèce dans le zoo. Fusionner cette table avec la table “alim” pour créer la table “alim_totale”.

Intérêt:

- jointure de deux tables (fonction “merge”)

2.2 consommation totale (par enclos)

Calculer pour chaque type d’aliment, la consommation totale de chaque enclos -c’est-à-dire pour l’ensemble des individus de cette espèce- (ajouter une nouvelle variable “poids_total” dans “alim_totale”)

- ajout de variables à une table (fonction `data.frame`)

2.3 prix approximatifs

Charger la table “prix_approx” qui indique le prix approximatif au kg de chaque catégorie d’aliment. Calculer, pour chaque type d’aliment, le coût en aliments de chaque enclos.

Intérêt:

- retrouver la solution la plus simple parmi les techniques déjà utilisées...

2.4 coût total des aliments (toutes catégories d’aliments confondus, par espèce)

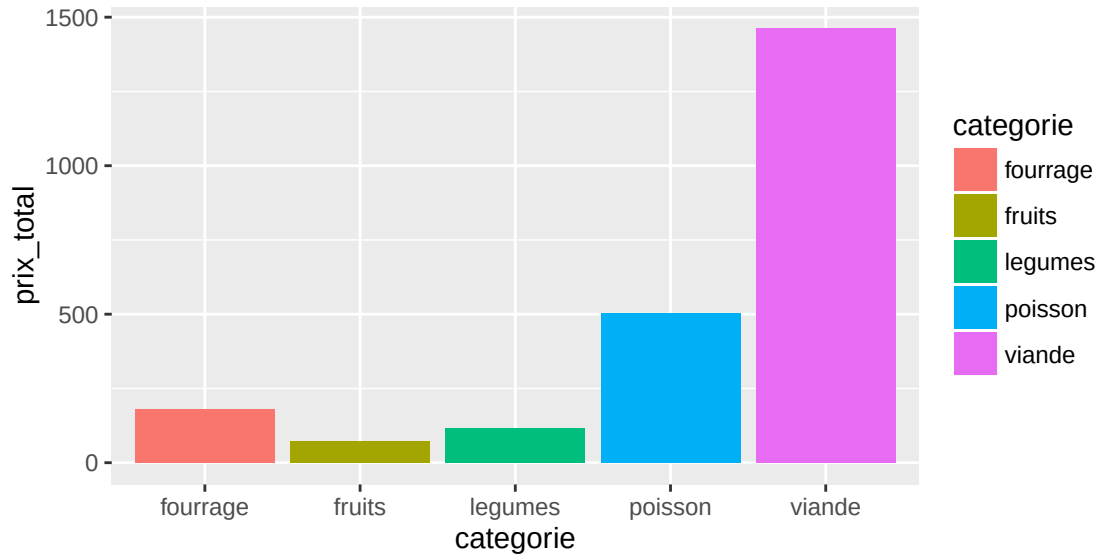
Calculer le coût total des aliments pour chacune des espèces. Quelle est l’espèce dont l’alimentation coûte le plus cher?

Intérêt:

- faire une boucle ou une “boucle cachée” via la fonction `tapply`

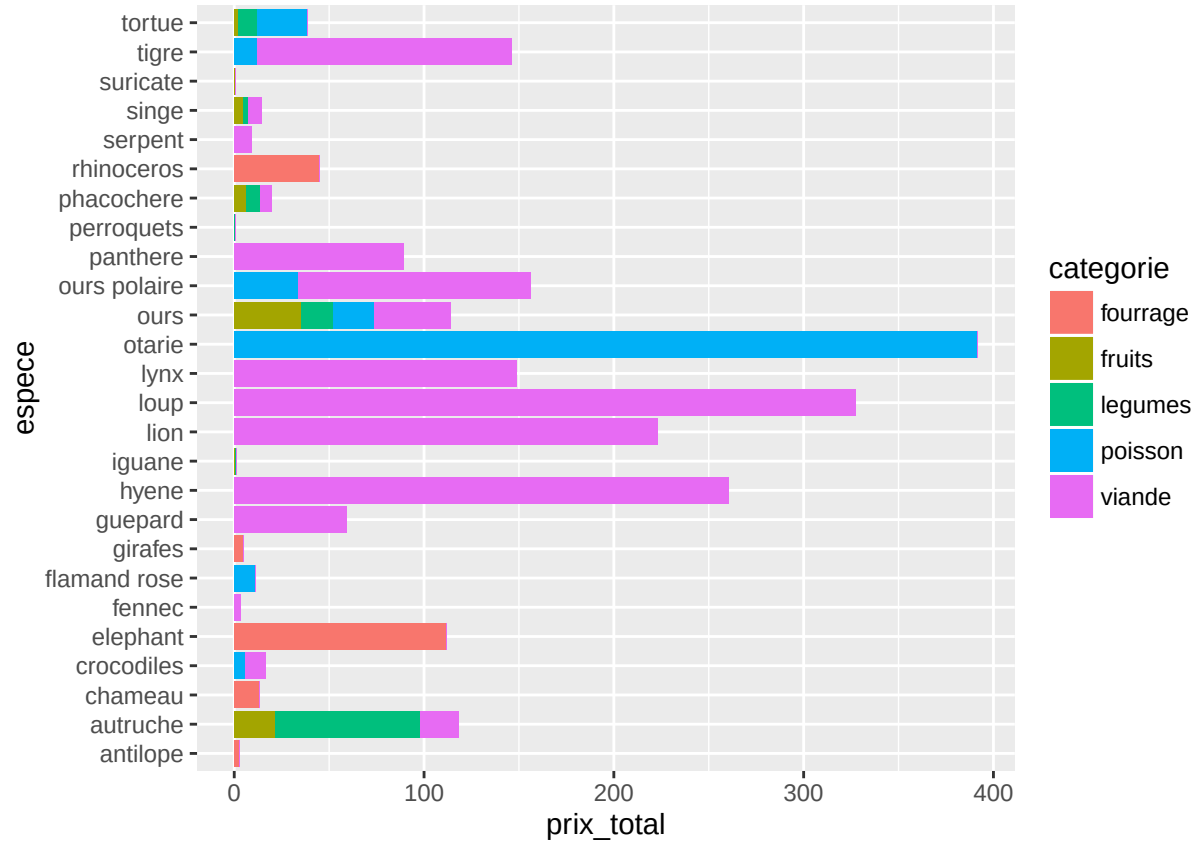
2.5 graphique : coût total (toutes espèces confondues) par type d’aliment

Reproduisez le graphique suivant:



2.6 graphique: coût total (par espèce) par type d'aliment

Reproduisez le graphique suivant (indice: on utilisera la fonction `coord_flip`):



3 Affiner le budget avec les prix de gros, détaillés par produit

On va maintenant affiner cette analyse budgétaire en travaillant sur le prix des aliments (exemple saumon, sardine, etc.) plutôt que sur un prix moyen par catégorie d'aliment (exemple, poisson).

La table prixdetailles fournit le prix au kilo des différents aliments, et fait correspondre à chaque aliment sa catégorie (viande, poisson, fruits, légumes ou fourrages).

En outre, cette table montre les réductions obtenues en fonction du volume des achats. Par exemple, le prix du premier kilo de viande de mouton acheté est 7,46. Le deuxième kilo voit son prix réduit de 1.1%, son prix est donc de $(100 - 1.1)/100 * 7.46 = 7.38$ euros, le troisième kilo voit son prix réduit de 1.1% à nouveau, son prix est donc de $(100 - 1.1)/100 * 7.37 = 7.30$ euros, etc. Le prix du kilo ne peut cependant pas descendre en dessous du prix indiqué dans la colonne prixmin.

3.1 lecture de prixdetailles

Charger la table prixdetailles.

3.2 écriture de la fonction prixaukilo

Ecrire une fonction prixaukilo qui a pour entrée

- le nom d'un aliment
- un nombre n

et qui a en sortie le prix du n-ième kilo de cet aliment.

```
prixaukilo("mouton",3)
```

```
[1] 7.296783
```

3.3 écriture de la fonction prixpourpoids

Ecrire une fonction prixpourpoids qui a en entrée

- le nom d'un aliment
- un poids

et en sortie le prix correspondant.

(Cette fonction doit vous permettre de répondre à la question -par exemple- "Combien me coûteraient 195 kilos de saumon?")

```
prixpourpoids("saumon",195)
```

```
[1] 1017.434
```

3.4 modification des fonctions prixaukilo et prixpourpoids

Modifier les fonctions prixaukilo et prixpourpoids de sorte qu'elles puissent avoir un vecteur comme argument (pour le n-ième kilo -fonction prixaukilo- ou pour le poids -fonction prixpourpoids-):

```r
prixaukilo("luzerne",c(10,100,1000))
```

```
[1] 0.09135172 0.08000000 0.08000000
```

```r
prixpourpoids("luzerne",0:5)
```

```
[1] 0.0000000 0.1000000 0.1990000 0.2970100 0.3940399 0.4900995
```

3.5 graphique: prix en fonction du poids

Représenter graphiquement le prix payé pour un aliment (par exemple, le saumon) en fonction du poids acheté (le prix total et non le prix au kilo).

3.6 création d'un tableau pour utilisation par ggplot2

Préparer un tableau de données donnant le prix de 1 à 100 kilos de poisson pour chaque type de poisson possible (tilapia, morue, sardine au saumon).

Attention à la manière dont vous mettrez ces données en forme! Jetez un coup d'oeil à l'exercice suivant, qui utilisera ce jeu de données...

3.7 graphique: prix en fonction du poids pour plusieurs aliments

Réalisez un graphique semblable à celui réalisé dans l'exercice n-2, mais renseignant le prix des quatre types de poisson (tilapia, morue, sardine au saumon) en fonction du poids acheté.

3.8 poisson le moins cher pour les besoins du zoo

Pour notre zoo, quel type de poisson est-il préférable d'acheter (d'un point de vue strictement économique, et considérant que les achats se font tous les jours pour une consommation le jour-même)?

3.9 exercice facultatif: fruit, légume et viandes les moins chers pour les besoins du zoo

Quel type de fourrage, de fruit, de légume et de viande est-il préférable d'acheter pour notre zoo?

4 Aide-mémoire

4.1 Exemples de structure conditionnelle (if):

```r
x=33
if(x<0){print("x est negatif")}
if(x>0){print("x est positif")}
```

```
[1] "x est positif"
```

```
smiley=function(humeur){
  if(humeur=="content"){print(":-)")}
  if(humeur=="triste"){print(":-(")}
  if(humeur!="content" & humeur!="triste"){print(":-?")}
}
smiley("content")
```

```
[1] ":-)"
```

```
smiley("triste")
```

```
[1] ":-("
```

```
smiley("mi-figue mi-raisin")
```

```
[1] ":-?"
```

4.2 Exemple de jointure de tables (merge):

```
print(tablenoms)
```

```
    espece        nom
1     lion     Simone
2     lion   Bertrand
3     ours  Papa Ours
4     ours Maman Ours
5     ours  Bébé Ours
6     ours     Winnie
7     ours      Baloo
8   canard       Gwek
9   canard     Bwagak
10 serpent  Saucisson
11 serpent    Sensass
12 serpent    Sucette
13 serpent    Salsifi
14    loup      Ghost
15    loup    Nymeria
16    loup  Grey Wind
```

```
print(tablecris)
```

```
   espece           cri
1    lion        groar!
2    ours   grougrou...
3  canard     coincoin!
4 serpent   ssssssss...
5    loup wouoouuuou!!!
```

```
merge(tablenoms,tablecris, by="espece")
```

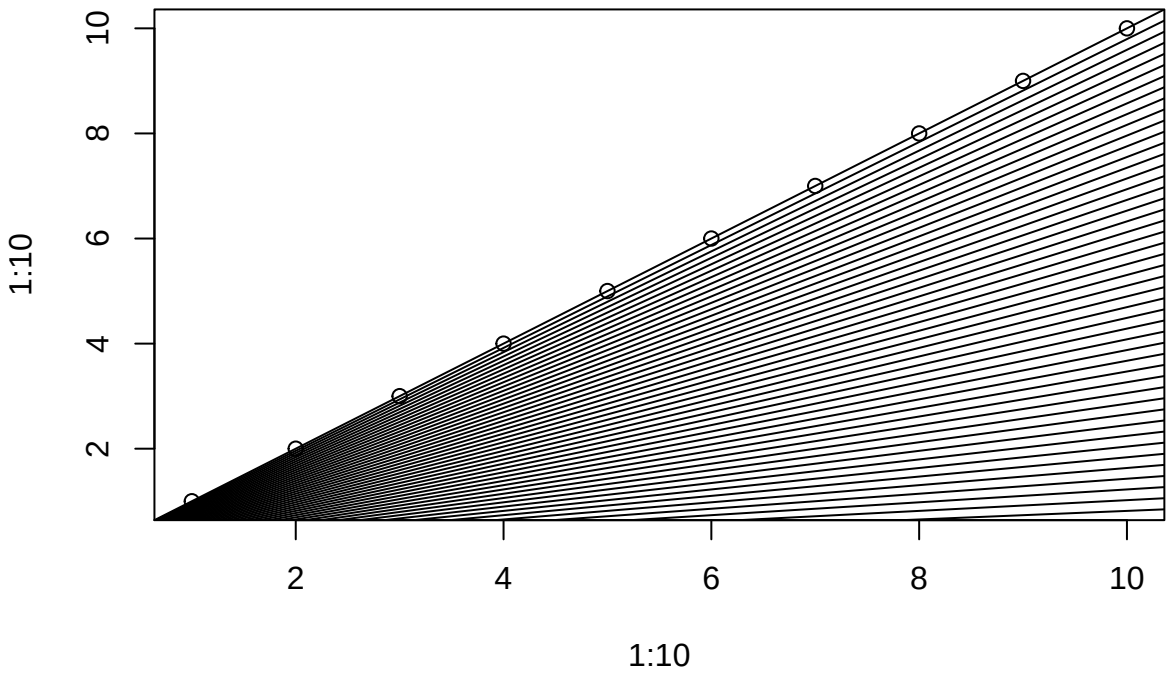
```
    espece        nom           cri
1   canard       Gwek     coincoin!
2   canard     Bwagak     coincoin!
3     lion     Simone        groar!
4     lion   Bertrand        groar!
5     loup      Ghost wouoouuuou!!!
6     loup    Nymeria wouoouuuou!!!
7     loup  Grey Wind wouoouuuou!!!
8     ours  Papa Ours   grougrou...
9     ours Maman Ours   grougrou...
10    ours  Bébé Ours   grougrou...
11    ours     Winnie   grougrou...
12    ours      Baloo   grougrou...
13 serpent  Saucisson   ssssssss...
14 serpent    Sensass   ssssssss...
15 serpent    Sucette   ssssssss...
16 serpent    Salsifi   ssssssss...
```

4.3 Exemples de boucle for

```
mots=c("SHEBAM!","POW!","BLOP!","WIZZ!")
for (i in 1:length(mots)){
  print(mots[i])
}
```

```
[1] "SHEBAM!"
[1] "POW!"
[1] "BLOP!"
[1] "WIZZ!"
```

```
plot(1:10,1:10)
coeffs=seq(0,1,length.out=50)
for(i in 1:50){
 abline(a=0,b=coeffs[i])
}
```



4.4 Exemples de manipulation de chaînes de caractère (grep, gsub, paste, strsplit)

`grep` trouve les emplacements dans un vecteur où se trouve un motif :

```
mots=c("Les",
       "chaussettes",
       "de",
       "l'archiduchesse",
       "sont",
       "-",
       "elles",
       "sèches",
       "?")

grep("ss",mots)
```

```
[1] 2 4
```

Ici on a deux s (“ss”) dans le deuxième et quatrième mot.

`gsub` trouve les emplacements dans un vecteur où se trouve un motif et le remplace par un autre motif:

```
gsub("s",mots,replacement="*")
```

```
[1] "Le*"              "chau**ette*"      "de"               "l'archiduche**e"
[5] "*ont"             "-"                "elle*"            "*èche*"
[9] "?"
```

`paste` concatène plusieurs chaînes de caractères en une seule:

```
paste("pouet","pouet",sep="-")
```

```
[1] "pouet-pouet"
```

```
#séparateur vide:
paste0("gouzi","gouzi")
```

```
[1] "gouzigouzi"
```

```
#concatène tous les éléments d'un vecteur
paste(mots,collapse=" ")
```

```
[1] "Les chaussettes de l'archiduchesse sont - elles sèches ?"
```

`strsplit` découpe une chaîne de caractères en plusieurs:

```
decoupe= strsplit(c("hip-hip-hip-Hourrah",
                    "pouet-pouet",
                    "youp-la-boum"),
                  split="-")
print(decoupe)
```

```
[[1]]
[1] "hip"     "hip"     "hip"     "Hourrah"

[[2]]
[1] "pouet" "pouet"

[[3]]
[1] "youp" "la"   "boum"
```

```
print(decoupe[[1]])
```

```
[1] "hip"     "hip"     "hip"     "Hourrah"
```

4.5 Exemple d'assignation de nom (`assign`) et de récupération d'objet par nom (`get`)

La fonction `assign` permet d'associer une chaîne de caractère à un objet. Elle est utile notamment dans une boucle lorsque l'on crée des objets à la chaîne!

```
formulemagique= "Per Horus et per Rha et per solem invictus duceres"
mots_formule=strsplit(formulemagique," ")[[1]]
for (i in 1:10){
  mot=mots_formule[i]
  assign(paste0("mot",i),mot)
}
print(mot5)
```

```
[1] "Rha"
```

Réciproquement, la fonction `get` permet de récupérer l'objet associé à une chaîne de caractère (son nom):

```
for (i in 1:10) {
  mot=get(paste0("mot",i))
  print(mot)
}
```

```
[1] "Per"
[1] "Horus"
[1] "et"
[1] "per"
[1] "Rha"
[1] "et"
[1] "per"
[1] "solem"
[1] "invictus"
[1] "duceres"
```