Les vecteurs

Création d'un vecteur

Pour créer un vecteur, la façon la plus simple est utiliser la fonction `c()`.

```
# Exemple x=[1 4 5]
x=c(1,4,5)
x
```

```
[1] 1 4 5
```

Pour créer une série régulière, `R` propose les fonctions `:`, `seq()` et la fonction `rep()`.

```
# exemple série de 1 à 10
1:10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
# série de 20 à 2
20:2
```

```
 [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2
```

```
1.2:5.7
```

```
[1] 1.2 2.2 3.2 4.2 5.2
```

```
# générer une séquence à l'aide de la fonction `seq()`
# taper `?seq` pour consulter l'aide de la fonction
seq(from=1,to=10)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
# introduisons les arguments `by' et `len`
seq(1,3,by=0.5)  # création d'une séquence de 1 à 3 avec un pas de 0.5
```

```
[1] 1.0 1.5 2.0 2.5 3.0
```

```
seq(1,3, len=5) # création d'une séquence de 1 à 2 de longueur 10
```

```
[1] 1.0 1.5 2.0 2.5 3.0
```

```
# vecteur contenant des répétitions
rep(5,times=5)
```

```
[1] 5 5 5 5 5
```

```
rep(c(1,2),6) # répéter (1,2) 6 fois.
```

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

```
rep(c(1,2),each=3) # répéter chaque éléments du vecteur (1,2) 3 fois
```

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

```
rep(c(1,2),each=3,len=10)
```

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

Indexation et accés

```
# indexation
x=seq(1,5,len=15)
x
```

```
 [1] 1.000000 1.285714 1.571429 1.857143 2.142857 2.428571 2.714286
 [8] 3.000000 3.285714 3.571429 3.857143 4.142857 4.428571 4.714286
[15] 5.000000
```

```
x[5] # extraction de la 5ième valeur
```

```
[1] 2.142857
```

```
x[-4] # éliminer la valeur du 4ième index
```

```
 [1] 1.000000 1.285714 1.571429 2.142857 2.428571 2.714286 3.000000
 [8] 3.285714 3.571429 3.857143 4.142857 4.428571 4.714286 5.000000
```

```
x[x>=1.2] # extraire les valeurs >= à 1.2
```

```
 [1] 1.285714 1.571429 1.857143 2.142857 2.428571 2.714286 3.000000
 [8] 3.285714 3.571429 3.857143 4.142857 4.428571 4.714286 5.000000
```

```
x[1:4]      # extraire les éléments de x de la première position à la 4ième.
```

```
[1] 1.000000 1.285714 1.571429 1.857143
```

```
x[-c(1,5)]  # afficher toutes les valeurs de x suaf la 1ière et la 5ième valeur.
```

```
 [1] 1.285714 1.571429 1.857143 2.428571 2.714286 3.000000 3.285714
 [8] 3.571429 3.857143 4.142857 4.428571 4.714286 5.000000
```

Statistiques élémentaires

```
# la longueur d'un vecteur est obtenu à l'aide de la fonction length
length(x)
```

```
[1] 15
```

```
# maximum
max(x)
```

```
[1] 5
```

```
min(x)
```

```
[1] 1
```

```
sum(x) # la somme des val de x
```

```
[1] 45
```

```
cumsum(x) # la somme cumulée
```

```
 [1]  1.000000  2.285714  3.857143  5.714286  7.857143 10.285714 13.000000
 [8] 16.000000 19.285714 22.857143 26.714286 30.857143 35.285714 40.000000
[15] 45.000000
```

```r
prod(x) # produit des val de x
```

```
[1] 3112378
```

```r
cumprod(x) # produit cumulé
```

```
 [1] 1.000000e+00 1.285714e+00 2.020408e+00 3.752187e+00 8.040400e+00
 [6] 1.952669e+01 5.300100e+01 1.590030e+02 5.224385e+02 1.865852e+03
[11] 7.196856e+03 2.981555e+04 1.320403e+05 6.224756e+05 3.112378e+06
```

```r
which.min(x)     # donne la position pour laquelle x est minimale
```

```
[1] 1
```

```r
which(x>3.3)  # donne les positions pour lesquelles les valeurs de x sont supérieur à 3.3
```

```
[1] 10 11 12 13 14 15
```

```r
# la moyenne empirique
mean(x)
```

```
[1] 3
```

```r
# multiplication d'un vecteur par un scalaire
3*x
```

```
 [1]  3.000000  3.857143  4.714286  5.571429  6.428571  7.285714  8.142857
 [8]  9.000000  9.857143 10.714286 11.571429 12.428571 13.285714 14.142857
[15] 15.000000
```

```r
# x'x produit scalaire
crossprod(x)
```

```
         [,1]
[1,] 157.8571
```

```r
# ou encore
t(x)%*%x
```

```
         [,1]
[1,] 157.8571
```