Quelques commandes du logiciel R ; Version 17 novembre 2019

Une fonction est définie par

```
nom=function(){}
```

"nom" est le nom de la fonction, sous lequel celle-ci est enregistrée. Entre parentèses on met les variables pour les arguments de la fonction. Entre accolades on défini la fonction. Le nom peut être choisi librement, mais ne devrait pas être le nom d'une commande de R.

Exemple 1 `f=function(x) 5*x^3-7*x+2`

"f" est le nom de la fonction. "x" est l'argument. Il n'est pas nécessaire de mettre les accolades, si la définition de la fonction se limite à une ligne. On peut calculer la valeur de fonction de 3 après l'exécution de la ligne prècèdente par

```
f(3)

## [1] 116
```

Après "##" sont indiqués les résultats d'une commande.

Exemple 2 `Cap=function(Co,p,t) Co*(1+p)^t`

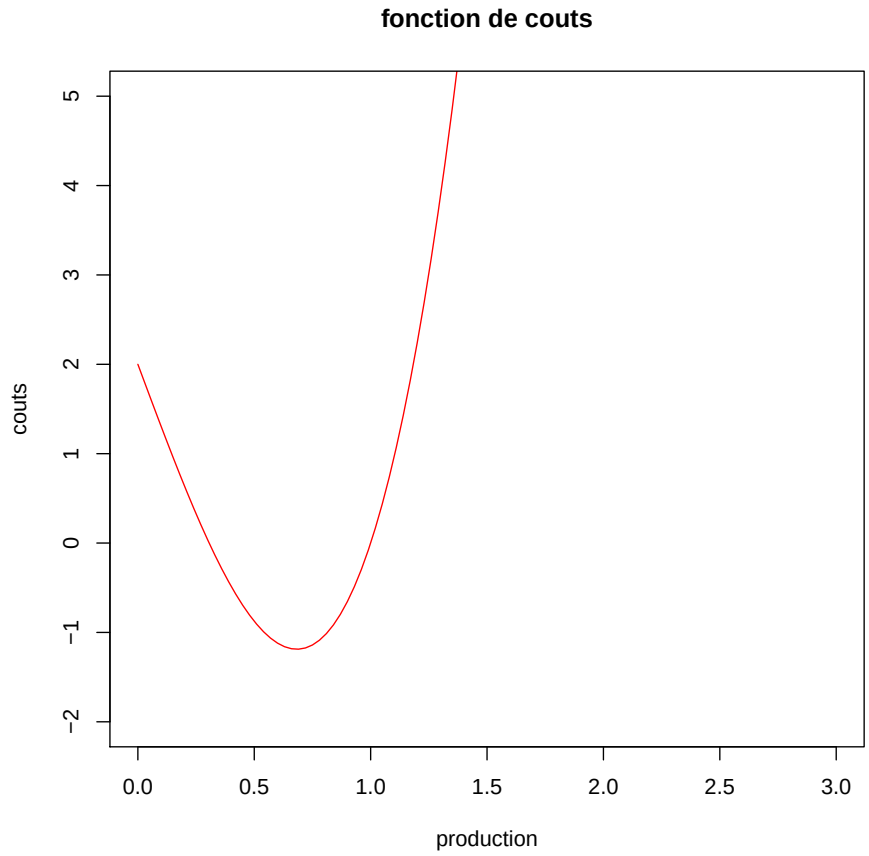
La fonction d'intérêt composé : attribue au capital initial, au taux d'intérêt et au temps t le capital final. Utilisation (au capital initial de 500, au taux d'intérêt 3% et la durée du placement 5.5 est attribué 588.2673) :

```
Cap(500,0.03,5.5)

## [1] 588.2673
```

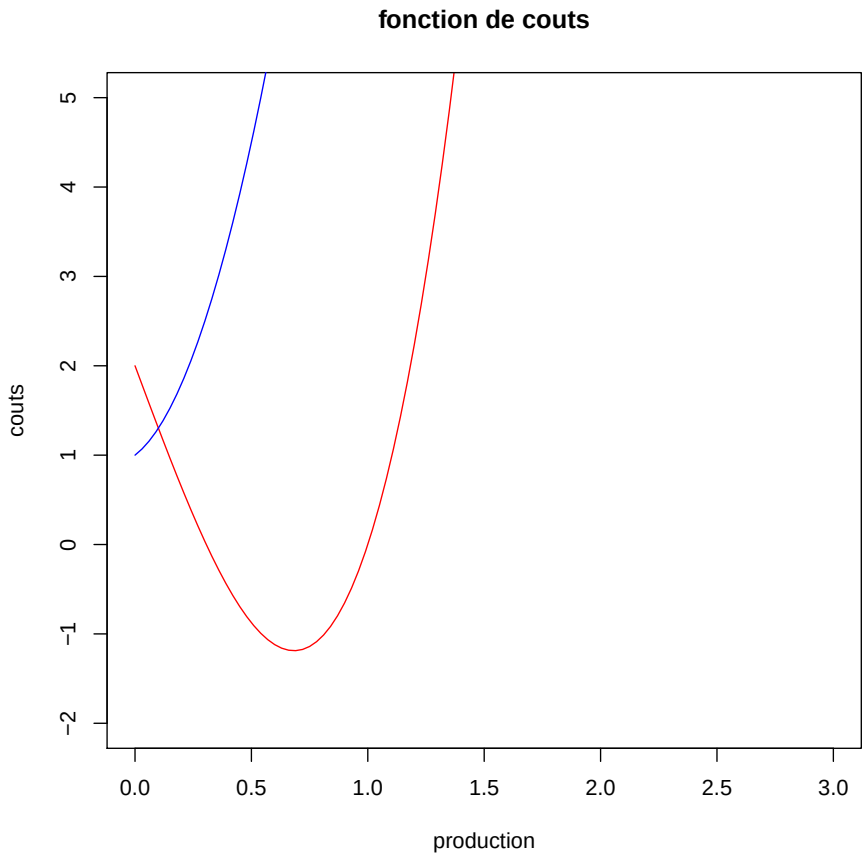
1.1 représentations graphiques

```
plot(f, xlim=c(0,3), ylim=c(-2,5), col = "red",
     xlab="production",ylab= "couts",
     main="fonction de couts")
```



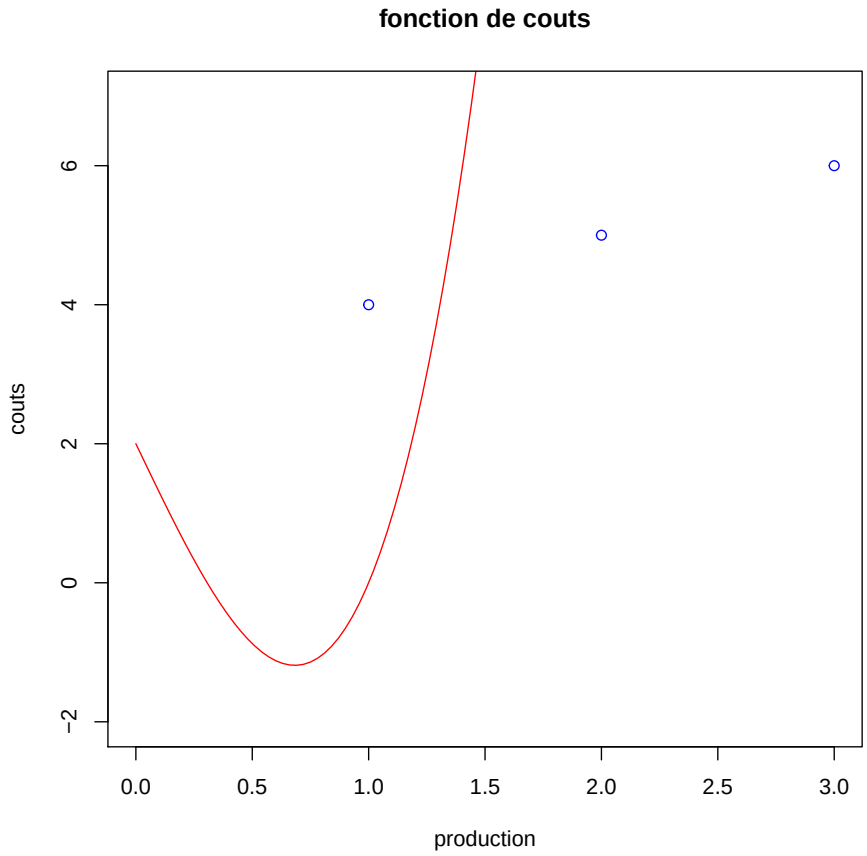
— "plot" ouvre une surface graphique est dessine la représentation graphique de la fonction f dedans, si f est une fonction réelle. En l'occurrence, f est la fonction f définie ci-dessus.
— "xlim" et "ylim" indiquent les dimensions du plot. Si on ne définit pas "xlim" et "ylim" c. à d. ces sous-commandes ne surviennent pas dans la commande "plot, "plot" desssine dans le carré [0,1]x[0,1]. En général on ne définit que "xlim".
— Si on ne définit pas "col" (pour couleur), R dessine en noir.
— Avec "xlab" et "ylab" on définit les étiquettes des axes ("lab" pour "label").
— "main" contient le titre du graphique.

```
plot(f, xlim=c(0,3), ylim=c(-2,5), col = "red",
     xlab="production",ylab= "couts",
     main="fonction de couts")
g=function(x) 10*x^2+2*x+1
curve(g,add=T,col="blue")
```



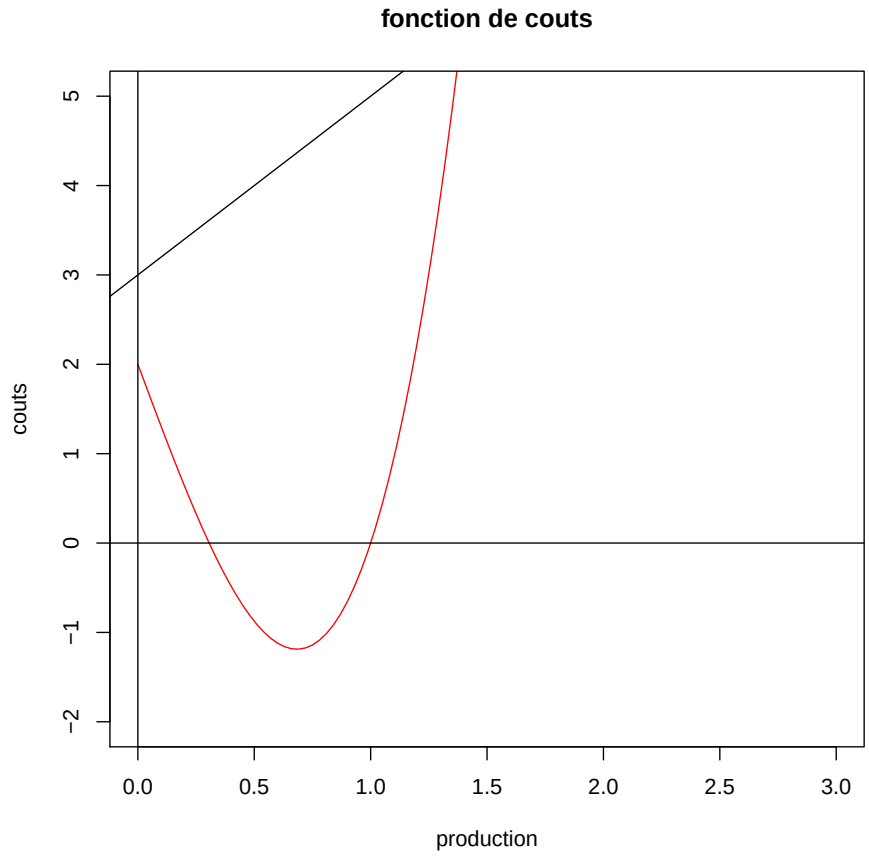
”curve” ajoute la représentation graphique de la fonction g à une surface ouverte par la commande ”plot”.

```
plot(f, xlim=c(0,3), ylim=c(-2,7), col = "red",
     xlab="production",ylab= "couts",
     main="fonction de couts")
points(x=c(1,2,3),y=c(4,5,6),col="blue")
```



"points" ajoute les points (1,4), (2,5), et (3,6) à une surface ouverte par "plot" (le nombre de composantes de x et de y doit être identique).

```
plot(f, xlim=c(0,3), ylim=c(-2,5), col = "red",
     xlab="production",ylab= "couts",
     main="fonction de couts")
abline(h=0)
abline(v=0)
abline(3,2)
```



"abline(h=0)" ajoute une droite horizontale qui passe par le point (0,0). "abline (v=0)" ajoute une droite verticale qui passe par le point (0,0). "abline(3,2)" ajoute une droite de pente 2. L'ordonnée à l'origine est 3.

1.2 n-uplets

"c()" s'utilise pour construire des n-uplets (= vecteurs), p.ex.

```
c(5,6,3,8)
```

```
## [1] 5 6 3 8
```

représente le 4-uplet (5,6,3,8) Nous pouvons enregistrer un n-uplet sous un mot ou une lettre quelconque (sans espace vide), par exemple sous "a"

```
a=c(2,1,8.2,39,8)
```

Après l'exécution de "a=c(2,1,8.2,39,8)", apparamment rien ne se passe. En effet, le logiciel enregistre le 5-uplet c(2,1,8.2,39,8) sous "a". En exécutant "a" on obtient :

```
a

## [1]  2.0  1.0  8.2 39.0  8.0
```

Il n'est pas nécessaire de réécrire "a". On peut sélectionner "a" dans la définition de "a" et l'éxecuter. Pour choisir dans le 5-uplet "a" le nombre à la troisième position de ce n-uplet, on écrit et on exécute

```
a[3]

## [1] 8.2
```

Si on veut choisir le nombre à la deuxième et à la quatrième position de "a" on écrit

```
a[c(2,4)]

## [1]  1 39
```

Si on veut choisir ces nombres dans l'ordre inverse on écrit

```
a[c(4,2)]

## [1] 39  1
```

"c(sd,sg[1])" forme à partir du m-uplet sd et de la première composante du k-uplet sg un m+1-uplet dans l'ordre (sd, sg[1]). Si on veut faire créer (4,1,5,8,9.9) à partir de a=c(4,1,5,5,6.5,6.7) et de b=c(8,7,9.9,7.5) on écrit

```
a=c(4,1,5,5,6.5,6.7)
b=c(8,7,9.9,7.5)
c(a[c(1,2,3)],b[c(1,3)])

## [1] 4.0 1.0 5.0 8.0 9.9
```

"order()" attribue à un n-uplet "a" un $n-$uplet de nombres naturels tel que ce n-uplet nous indique la position des nombres du vecteur "a" selon l'ordre croissant des composantes de "a".

```
a=c(5.2,2.1,8.9,3.3)
order(a)

## [1] 2 4 1 3
```

Le nombre le plus petit 2.1 se trouve à la deuxième position de "a", le nombre suivant 3.3 se trouve à la quatrième position de "a", le nombre 5.2 à la première et le nombre 8.9 à la troisième de "a". Par conséquent les commandes suivantes produisent un vecteur qui contient les composantes de "a" en ordre croissant

```
a=c(5.2,2.1,8.9,3.3)
a[order(a)]

## [1] 2.1 3.3 5.2 8.9
```

1.3 zéros de fonctions

Pour des polynômes de forme standardisée croissante $a_0+a_1x+a_2x^2+,...+,a_nx^n$ on entre "polyroot()". Il faut indiquer tous les a_i entre a_0 et a_n (n est le degré du polynôme), même si $a_i = 0$ ou $a_i = 1$. Pour les zéros de $f(x) = 5x^5-x^3+4x^2+x$ il faut entrer et exécuter :

```
polyroot(c(0,1,4,-1,0,5))

## [1]  0.0000000+0.0000000i -0.2397594+0.0000000i -0.9142078-0.0000000i
## [4]  0.5769836-0.7612755i  0.5769836+0.7612755i
```

Il y a dans l'exemple trois zéros réels et deux zéros complexes. Dans "x+yi" on appelle "x" la partie réelle et "yi" la partie imaginaire. Le nombre est réel, si la partie imaginaire est 0 (= 0.00000i). Le nombre est complexe, si la partie imaginaire est différente de 0 (p.ex. 0.2345i). On ne tient compte que des nombres réels. Pour les extraire du résultat et pour les ordonner en ordre croissant on peut procéder de la manière suivante :

```
sol=polyroot(c(0,1,4,-1,0,5))
solr = Re(sol)[abs(Im(sol)) < 1e-6]
solr[order(solr)]

## [1] -0.9142078 -0.2397594  0.0000000
```

"Re" considère la partie réelle. "abs" calcule la valeur absolue. "Im" considère la partie imaginaire. La commande retient du vecteur "sol" la partie réelle des nombres, aux endroits où la valeur absolue de la partie imaginaire est inférieure à 0.000006.

1.4 matrices

La matrice

$$\begin{array}{cccccc} 4 & 6 & 8 & 3 & 7 & 2 \\ 5 & 11 & \sqrt{2} & \log_2 3 & 5 & 6 \\ 7 & 1 & 1 & 0 & 2 & 2 \end{array}$$

est entrée par

```
matrix(c(4,6,8,3,7,2,5,11,2^(1/2),log(3,2),5,6,7,1,1,0,2,2),
       ncol=6,byrow=T)
```

```
##      [,1] [,2]     [,3]     [,4] [,5] [,6]
## [1,]    4    6 8.000000 3.000000    7    2
## [2,]    5   11 1.414214 1.584963    5    6
## [3,]    7    1 1.000000 0.000000    2    2
```

On peut l'enregistrer sous un nom, par ex. m1 = matrix(....). On peut aussi indiquer le nombre de lignes, pour l'exemple "nrow=3"). Si on entre les données dans c(...) par colonnes, il ne faut pas indiquer byrow=T. On accède la cellule d'une matrice enregistrée sous "m1" par m1[3,4] (pour la cellule au croisement de la troisième ligne et de la quatrième colonne)
La matrice

$$\begin{matrix} 1 & 5 & 5^2 & 5^3 \\ 1 & 4 & 4^2 & 4^3 \\ 1 & 3 & 3^2 & 3^3 \\ 1 & 6 & 6^2 & 6^3 \end{matrix}$$

peut être entrée par

```
x=c(5,4,3,6)
mat1=cbind(x^0,x,x^2,x^3)
mat1
```

```
##      x
## [1,] 1 5 25 125
## [2,] 1 4 16  64
## [3,] 1 3  9  27
## [4,] 1 6 36 216
```

Multiplication d'une matrix **A** avec un n-uplet **y** :

```
y=c(5,4,3,6)
A=cbind(x^0,x,x^2,x^3)
A%*%y
```

```
##      [,1]
## [1,]  850
## [2,]  453
## [3,]  206
## [4,] 1433
```

Solution du système d'équation Ax=y

```
x=c(4,5,6,7)
y=c(5,4,3,6)
A=cbind(x^0,x,x^2,x^3)
solve(A,y)

##                           x
## -71.0000000  48.3333333 -10.0000000   0.6666667
```

ou par

```
solve(A)%*%y

##           [,1]
##   -71.0000000
## x  48.3333333
##   -10.0000000
##     0.6666667
```

"solve(A)" calcule la matrice inverse de A.
La commande ("ls" pour "list")

```
ls()

##  [1] "a"    "A"    "b"    "Cap"  "f"    "g"    "mat1" "nom"  "sol"  "solr"
## [11] "x"    "y"
```

indique les objets enregistrés de la console. La commande ("rm" pour "remove")

```
rm(a)
```

enlève l'objet "a" de la console.