Projet Semestriel : Mathématiques pour le Big Data

Nom	Prénom
CAI	Eddy
MBAE	Hakim
SHEIKH	Rakib
TARANTO	Tom

Exercice 1 : Cas d'une matrice non diagonalisable Soit la matrice

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 2 \\ -1 & 2 & 2 \\ -1 & 1 & 3 \end{bmatrix}$$

1. Ecrire une fonction qui calcule les valeurs propres ainsi que leur multiplicité d'une matrice carrée quelconque. Appliquer sur A

```
library(limSolve)
library(expm)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'expm'
```

```
## The following object is masked from 'package:Matrix':
##
##     expm
```

```
#Q1
A= matrix(c(0,2,2,-1,2,2,-1,1,3), nrow = 3, byrow = T)
q1<-function(A){
  valeurs<-Re(eigen(A)$values)
  print(valeurs)
  res<-table(valeurs)
  print(res)
  return(res)
}

A
```

```
##      [,1] [,2] [,3]
## [1,]    0    2    2
## [2,]   -1    2    2
## [3,]   -1    1    3
```

```
res<-as.matrix(q1(A))
```

```
## [1] 2 2 1
## valeurs
```

```
## 1 2
## 1 2
```

```
res
```

```
##   [,1]
## 1    1
## 2    2
```

```
dim(res)
```

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

```
eigen(A)$values
```

```
## [1] 2+0i 2-0i 1+0i
```

2. Donner les vecteurs propres de A pour chaque valeur propre. Pourquoi A n'est pas diagonalisable ? Justifier.

```
vec_propre<-Re(eigen(A)$vectors)
vec_propre
```

```
##           [,1]      [,2]          [,3]
## [1,] 0.8164966 0.8164966  8.944272e-01
## [2,] 0.4082483 0.4082483 -4.213000e-16
## [3,] 0.4082483 0.4082483  4.472136e-01
```

La multiplicité de la valeur propre 2 est 2, elle n'est donc pas inversible. On retrouve une valeur propre double donc non diagonalisable.

3. Donner la matrice de Jordan J de A ainsi qu'une matrice de passage P permettant d'effectuer le changement de base entre A et J . Calculer $P-1$ et vérifier le changement de base

```
#Q3
#On réarrange les vecteurs propres pour les mettre dans le bon ordre
vec_propre<-vec_propre[,c(3,1,2)]

I3<-diag(1,nrow = 3)


#Pour trouver une base du sous-espace propre associé à lambda = 2,
#On cherche un antécédent du vecteur propre associé à lambda=2 par A-2*I3
new_vect<-Solve(A-2*I3, vec_propre[,2])

#On remplace ce nouveau vecteur dans la matrice de passage
vec_propre[,3]<-new_vect

P<-vec_propre
#La matrice de Jordan est composée des blocs de Jordan

#retourne le bloc de Jordan correspondant à vp l de multiplicité m
jordan_block<-function(l,m){
  if( m == 1) return(l)
  else{
    res<-matrix(rep(0,m*m), nrow = m)
    for (i in c(1:m)) {
      res[i,i]<-l
    }
```

```
    for (i in c(1:(m-1))) {
      res[i,i+1]<-1
    }
    return(res)
    }

}

#Retourne la matrice de Jordan associée aux valeurs propres
jordan_matrix<-function(val, multi){
  n<-sum(multi)
  res<-matrix(rep(0,n*n), nrow = n)
  idx<-1
  for (i in c(1:length(multi))) {
    jb<-jordan_block(val[i], multi[i])
    res[idx:(idx+multi[i]-1),idx:(idx+multi[i]-1)]<-jb
    idx<-idx + multi[i]
  }
  return(res)
}

#Valeurs propres : 1, 2. Multiplicité de 1:1 , multiplicité de 2 : 2
J<-jordan_matrix(c(1,2),c(1,2))
J
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    2    1
## [3,]    0    0    2
```

```
P
```

```
##               [,1]      [,2]       [,3]
## [1,]  8.944272e-01 0.8164966 -0.1360828
## [2,] -4.213000e-16 0.4082483  0.1360828
## [3,]  4.472136e-01 0.4082483  0.1360828
```

```
solve(P, tol = NULL)
```

```
##               [,1]          [,2]      [,3]
## [1,] -1.861901e-15 -2.236068e+00  2.236068
## [2,]  8.164966e-01  2.449490e+00 -1.632993
## [3,] -2.449490e+00  5.329071e-15  4.898979
```

```
#On vérifie que l'on tombe bien sur A

q <- P%*%J%*%solve(P, tol = NULL)
print("Verifions que PJP-1 = A")
```

```
## [1] "Verifions que PJP-1 = A"
```

```
q
```

```
##               [,1] [,2] [,3]
## [1,] -1.332268e-15    2    2
## [2,] -1.000000e+00    2    2
## [3,] -1.000000e+00    1    3
```

```
A
```

```
##      [,1] [,2] [,3]
## [1,]    0    2    2
## [2,]   -1    2    2
## [3,]   -1    1    3
```

4. On pose

$$\begin{cases} a_{n+1} = 2b_n + 2c_n \\ b_{n+1} = -a_n + 2b_n + 2c_n \quad \text{avec } a_0 = 1, b_0 = 2, c_0 = -3 \\ c_{n+1} = -a_n + b_n + 3c_n \end{cases}$$

En utilisant J^{25}, P et P^{-1}, écrire une fonction qui retourne a_25, b_25 , c_25

```
#Q4

library(expm)
# Fonction calculant les n-ieme termes de la suite a_n
abc<-function(J,P,N){
  X0<-c(1,2,-3)
  JN<-J%^%N
  res<-P %*% JN %*% solve(P) %*% X0
  return(res)

}
X0<-c(1,2,-3)
abc(J,P,25)
```

```
##             [,1]
## [1,] -5502926858
## [2,] -2868903936
## [3,] -2868903941
```

Exercice 2 : Interpolation de données Soit un nuage de points 2D : $\Delta = \{(x_i, y_i), i = 0, ..., n\}$. On suppose que $x_0 < x_1 < ... < x_n$ sont des entiers relatifs triés et $y_0, y_1, ..., y_n$ des entiers relatifs quelconques

On souhaite interpoler Δ par un polynôme de degré inférieure ou égal à n.

1. Créer une fonction qui génère graphiquement Δ aléatoirement tel que l'utilisateur saisisse le nombre de points et a, b, c, d pour que $\forall i = 0, ..., n, a \leq x_i \leq b, c \leq y_i \leq d$. Proposer un exemple.

```
library(polynom)

options(digits=10)

#Question 1

#Fonction de génération de delta
generate <- function(N,a,b,c,d){
  #Wichmann-Hill generator to reduce the number of duplicates
  RNGkind("Wich")
  #Xi
  X<-sort(runif(n = N, min = a, max = b))
  #Yi
```

```
  Y<-runif(n = N, min = c, max = d)
  L<-list("X" = X, "Y" = Y)
  return(L)
}

#On génère les X entre 0 et 10 et les y entre -5 et 5 avec 10 valeurs
delta<-generate(10,0,10,-5,5)
#On affiche le résultat graphiquement
#plot(delta$X, delta$Y, xlim = c(0,10), ylim = c(-5,5))
```

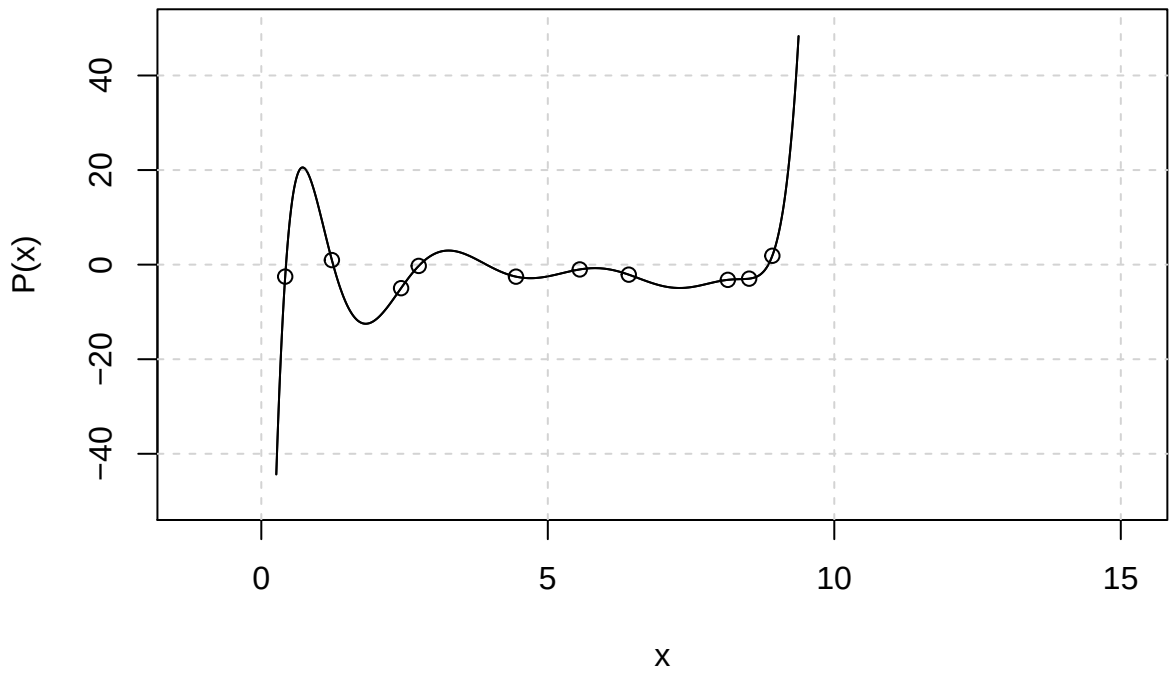
2. Donner la matrice de vandermonde permettant un première interpolation polynomiale de Δ , avec $n = 9$, puis $n = 19$ et enfin $n = 29$. Que constate-t-on ? Retourner et tracer si possible le polynôme d'interpolation.

```
#Retourne la matrice de Vandermonde pour les Xi données
vandermonde <- function(xi){
  N<-length(xi)
  #On créé la matrice contenant les xi
  res<-matrix(rep(xi,N),ncol = N)
  #On les met a la bonne puissance
  res<-res^(col(res)-1)
  return(res)
}

#Matrice de vandermonde pour nos données
vdm<-vandermonde(delta$X)
vdm
```

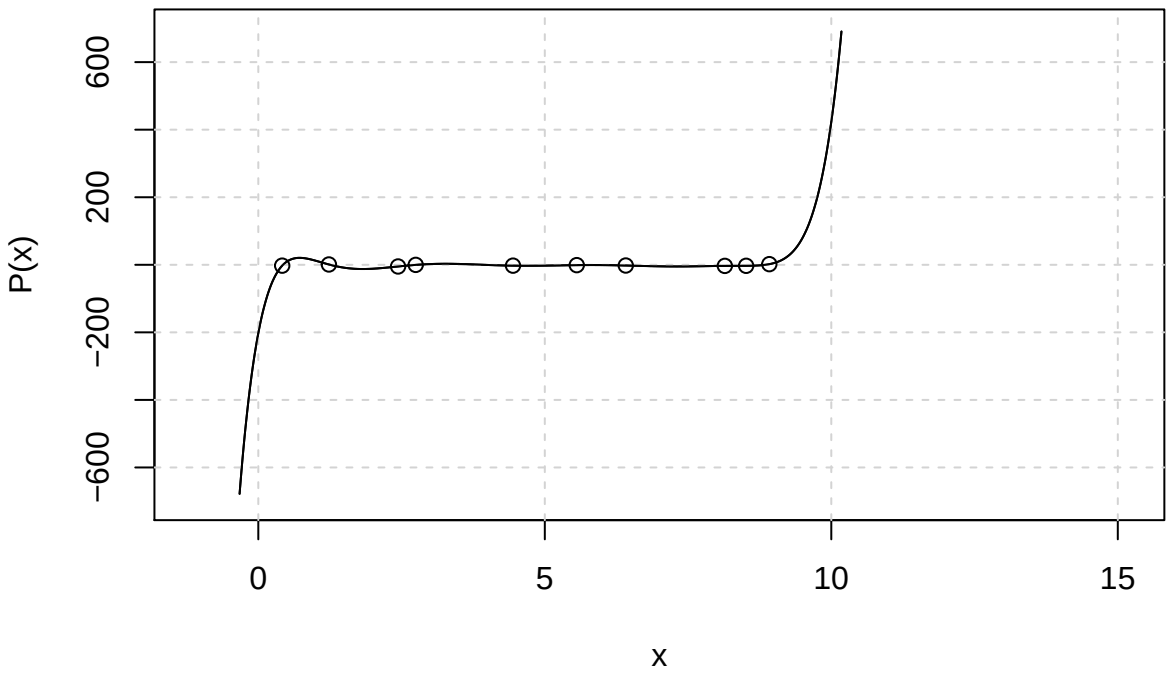
```
##       [,1]         [,2]          [,3]            [,4]            [,5]
##  [1,]    1 0.4168833565  0.1737917329   0.07245088097 3.020356644e-02
##  [2,]    1 1.2318756111  1.5175175213   1.86939282387 2.302859427e+00
##  [3,]    1 2.4393583578  5.9504691978  14.51532677059 3.540808367e+01
##  [4,]    1 2.7470166676  7.5461005720  20.72926404676 5.694363384e+01
##  [5,]    1 4.4450628277 19.7585835419  87.82814522947 3.904016236e+02
##  [6,]    1 5.5576470688 30.8874409412 171.66149560937 9.540340079e+02
##  [7,]    1 6.4132348363 41.1295810659 263.77366209568 1.691642439e+03
##  [8,]    1 8.1420101713 66.2923296296 539.75282212339 4.394672968e+03
##  [9,]    1 8.5140708351 72.4894021848 617.17990499455 5.254713429e+03
## [10,]    1 8.9186320210 79.5419971262 709.40580258512 6.326929307e+03
##                 [,6]            [,7]            [,8]            [,9]
##  [1,] 1.259136416e-02 5.249130153e-03 2.188274997e-03 9.122554257e-04
##  [2,] 2.836836364e+00 3.494629530e+00 4.304948888e+00 5.303161542e+00
##  [3,] 8.637300484e+01 2.106947113e+02 5.139599049e+02 1.253732389e+03
##  [4,] 1.564251113e+02 4.297023879e+02 1.180399622e+03 3.242577435e+03
##  [5,] 1.735359745e+03 7.713783094e+03 3.428825049e+04 1.524134277e+05
##  [6,] 5.302184308e+03 2.946766907e+04 1.637709047e+05 9.101808882e+05
##  [7,] 1.084890022e+04 6.957654482e+04 4.462107210e+05 2.861654140e+06
##  [8,] 3.578147200e+04 2.913331090e+05 2.372037137e+06 1.931315049e+07
##  [9,] 4.473900235e+04 3.809110351e+05 3.243103535e+06 2.761201322e+07
## [10,] 5.642755431e+04 5.032565927e+05 4.488360363e+06 4.003003445e+07
##                [,10]
##  [1,] 3.803041039e-04
##  [2,] 6.532835365e+00
##  [3,] 3.058302583e+03
##  [4,] 8.907414261e+03
```

```
##  [5,] 6.774872619e+05
##  [6,] 5.058464146e+06
##  [7,] 1.835246002e+07
##  [8,] 1.572478678e+08
##  [9,] 2.350906365e+08
## [10,] 3.570131471e+08
```

```r
#Fonction retournant les coefficients du polynôme d'interpolation grâce a Vandermonde
coef_vdm<-function(d){

  #mat[mat < 0.1] <- NA
  #print(vandermonde(d$X))
  coeff<- solve(round(vandermonde(d$X), digits = 8),matrix(d$Y, ncol=1), tol = NULL)
  return(coeff)
}

coef<-coef_vdm(delta)

p = as.polynomial(coef)

#Avec le détail
plot(p, ylim = c(-50,50), xlim = c(-1,15))
points(delta$X, delta$Y)
```




```r
#Avec la courbe s'affichant sur le dernier point
plot(p, ylim = c(-700,700), xlim = c(-1,15))
points(delta$X, delta$Y)
```

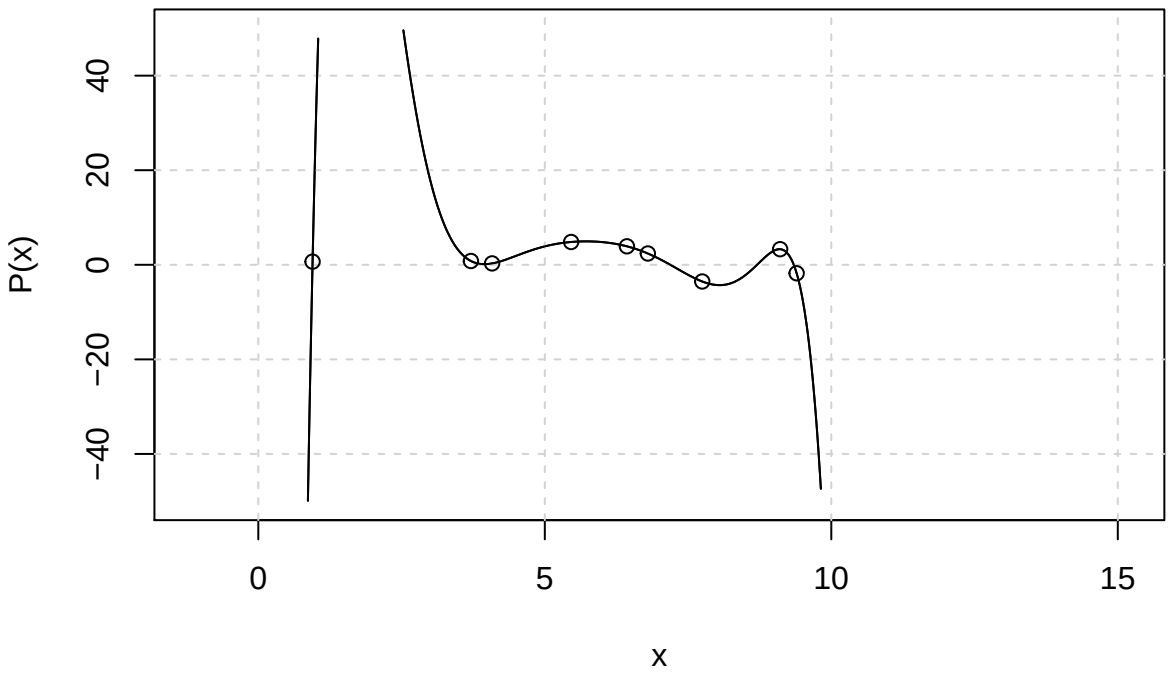



```
print(p)
```

```
## -202.3625157 + 895.9299108*x - 1344.973136*x^2 + 981.5217073*x^3 -
## 404.8418476*x^4 + 101.0782214*x^5 - 15.59804587*x^6 + 1.45532738*x^7 -
## 0.0752747703*x^8 + 0.001657487095*x^9
```

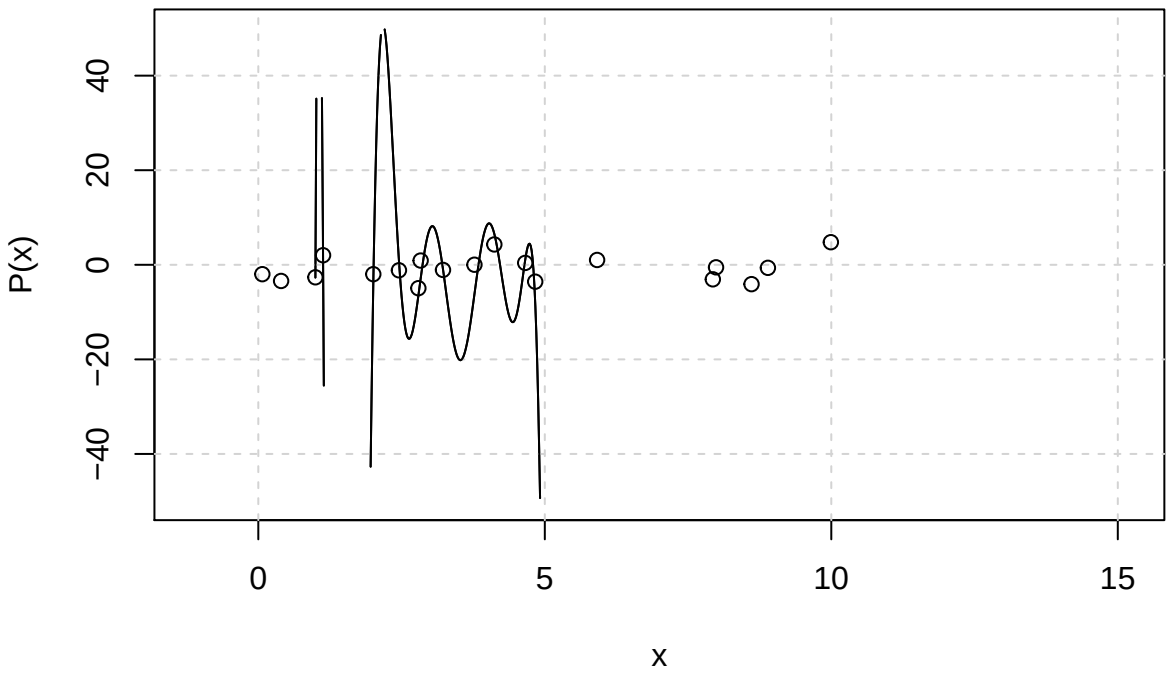
```
ex_deux_q_deux<-function(N){
  delta<-generate(N,0,10,-5,5)
  vdm<-vandermonde(delta$X)
  p<-as.polynomial(coef_vdm(delta))
  print(p)
  plot(p, ylim = c(-50,50), xlim = c(-1,15))
  points(delta$X, delta$Y)
}

ex_deux_q_deux(9)
```

```
## -1791.43773 + 4007.316152*x - 3287.700726*x^2 + 1401.615028*x^3 -
## 351.1188871*x^4 + 53.84263157*x^5 - 4.983832984*x^6 + 0.256055575*x^7 -
## 0.005610189026*x^8
```



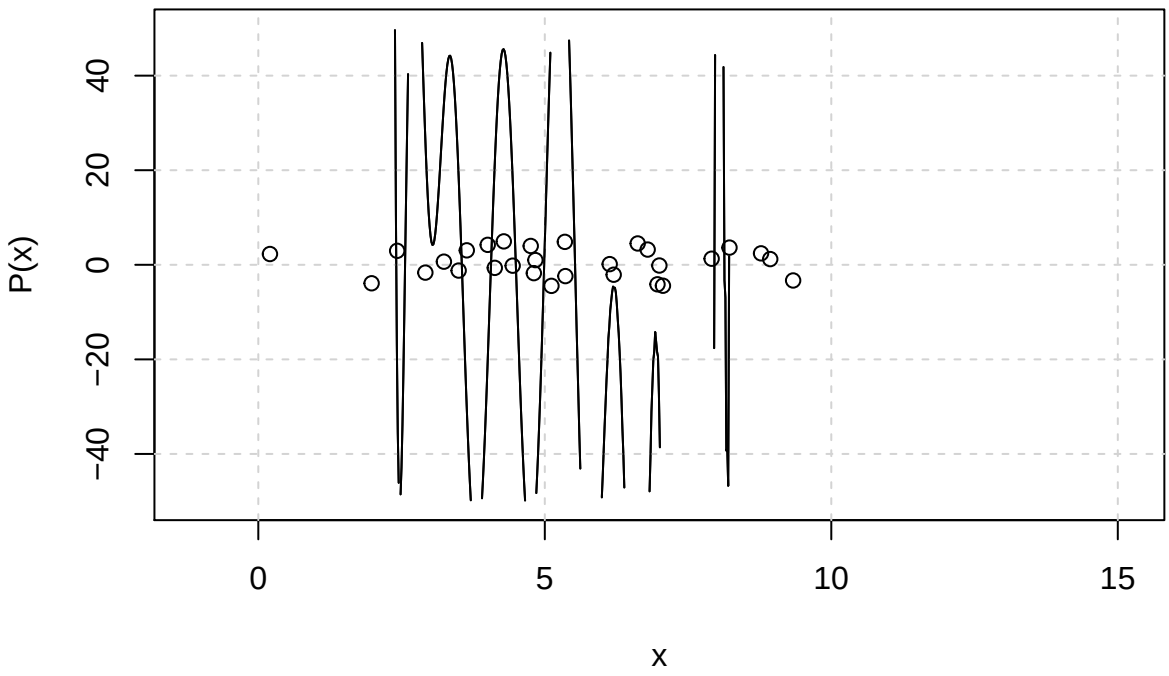

```
ex_deux_q_deux(19)
```

```
## -52949.8196 + 1160814.221*x - 7080044.38*x^2 + 21042551.33*x^3 -
## 37420183.11*x^4 + 44164844.65*x^5 - 36765056.01*x^6 + 22433085.1*x^7 -
## 10286353.5*x^8 + 3600490.119*x^9 - 970352.1574*x^10 + 201861.1257*x^11 -
## 32299.55749*x^12 + 3933.968297*x^13 - 357.6636413*x^14 + 23.47643677*x^15 -
## 1.049772778*x^16 + 0.02858922333*x^17 - 0.0003574668633*x^18
```




```
ex_deux_q_deux(29)
```

```
## 128459488.7 - 939235936.3*x + 1641772474*x^2 - 229605614.6*x^3 - 2858069215*x^4
## + 4796595642*x^5 - 4282940094*x^6 + 2529935788*x^7 - 1056828738*x^8 +
## 317841000.1*x^9 - 67463181.46*x^10 + 9199919.704*x^11 - 481072.9982*x^12 -
## 94590.8675*x^13 + 25406.61338*x^14 - 2808.913101*x^15 + 156.4455076*x^16 -
## 3.978009553*x^17 + 0.1225647925*x^18 + 0.03783201537*x^19 - 0.01342968952*x^20
## + 0.001179769347*x^21 + 4.285663189e-06*x^22 - 5.092415907e-07*x^23 -
## 1.47726562e-06*x^24 + 2.335315064e-07*x^25 - 1.621959247e-08*x^26 +
## 5.648943794e-10*x^27 - 8.100878085e-12*x^28
```



On peut remarquer que les valeurs des coefficients diminuent lorsque la puissance de x augmente.

On voit que la matrice de Vandermonde ne possède pas de creux. L'inversibilité de cette matrice est donc complexe à calculer.

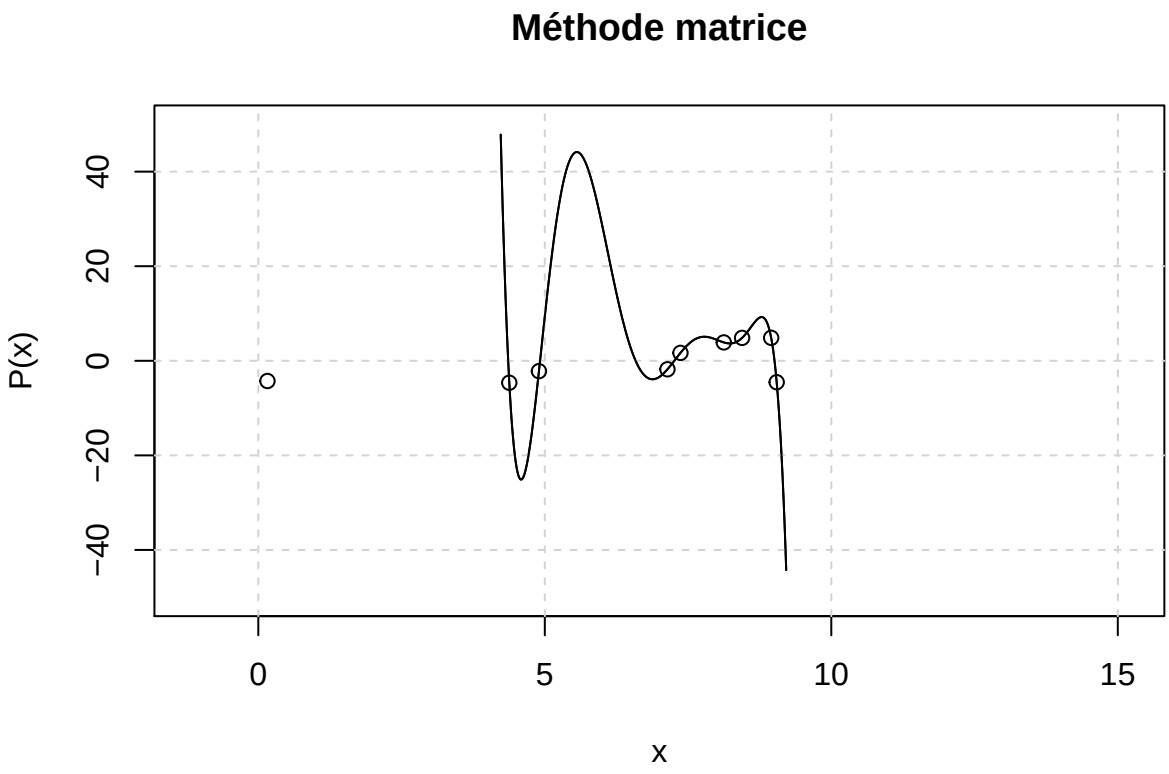
On voit aussi que l'affichage des polynômes n'est pas bien rendu. Cela est dû au moteur r, nos fonctions passent bien par les points concernés.

3. Donner la matrice de Newton permettant une seconde interpolation polynomiale de Δ , avec $n = 9$, puis $n = 19$ et enfin $n = 29$. Comparer avec la méthode précédente. Retourner et tracer si possible le polynôme d'interpolation

```r
#Fonction et création de la matrice de Newton

newton<-function(xi){
  N<-length(xi)
  res<-matrix(rep(1,N*N),ncol = N)
  #On parcourt les colonnes
  for (k in (2:N)) {
    #On parcourt les lignes
    for (j in (1:N)){
      res[j,k] <- res[j,k-1] * (xi[j] - xi[k-1])
    }
  }
  return(res)
}

#Fonction retournant les coefficients du polynôme d'interpolation grâce a la matrice de Newton
```

```r
coef_newton<-function(d){
  coeff<-solve(newton(d$X), tol = NULL)%*% d$Y
  return(coeff)
}

#Fonction retournant le polynôme d'interpolation grâce aux coefficients
polynome_newton_from_coef<-function(x,coefs){
  a<-coefs
  n<-length(x)-1
  p = a[n+1]
  for (k in c(1:n)){
    p = a[n-k+1] + (polynomial(coef = c(0,1)) - x[n-k+1]) * p
  }
  return(p)
}

#Fonction question 3 exo 2
ex_deux_q_trois<-function(N){
  delta<-generate(N,0,10,-5,5)
  #Affichage de la matrice de Newton
  print(newton(delta$X))
  #Calcul des coefs grâce a la matrice
  coefficients1<-coef_newton(delta)
  #Calcul du polynôme
  p1<-polynome_newton_from_coef(delta$X, coefs = coefficients1)
  print(p1)
  plot(p1, ylim = c(-50,50), xlim = c(-1,15), main = "Méthode matrice")
  points(delta$X, delta$Y)
}
ex_deux_q_trois(9)
```

```
##       [,1]        [,2]         [,3]         [,4]         [,5]         [,6]
##  [1,]    1 0.000000000  0.000000000   0.00000000   0.00000000   0.00000000
##  [2,]    1 4.220799930  0.000000000   0.00000000   0.00000000   0.00000000
##  [3,]    1 4.738068492  2.450853879   0.00000000   0.00000000   0.00000000
##  [4,]    1 6.980850051 19.267496033  43.21278479   0.00000000   0.00000000
##  [5,]    1 7.208086304 21.532618002  53.18595000  12.08577598   0.00000000
##  [6,]    1 7.964416399 29.815720400  96.19588710  94.61503734  71.56020017
##  [7,]    1 8.283625839 33.654929687 119.32548322 155.45435045 167.19729985
##  [8,]    1 8.791077919 40.177669912 162.84047489 294.77836559 466.63168089
##  [9,]    1 8.887300775 41.472596585 172.07943662 328.06096655 550.88472251
##               [,7]        [,8]        [,9]
##  [1,]   0.00000000   0.0000000  0.00000000
##  [2,]   0.00000000   0.0000000  0.00000000
##  [3,]   0.00000000   0.0000000  0.00000000
##  [4,]   0.00000000   0.0000000  0.00000000
##  [5,]   0.00000000   0.0000000  0.00000000
##  [6,]   0.00000000   0.0000000  0.00000000
##  [7,]  53.37095653   0.0000000  0.00000000
##  [8,] 385.74645448 195.7478404  0.00000000
##  [9,] 508.40290363 306.9100903 29.53176564
## -23616.7559 + 173233.4426*x - 170334.6355*x^2 + 75050.59919*x^3 -
## 18431.25493*x^4 + 2702.245706*x^5 - 235.956335*x^6 + 11.3544917*x^7 -
## 0.2322829409*x^8
```

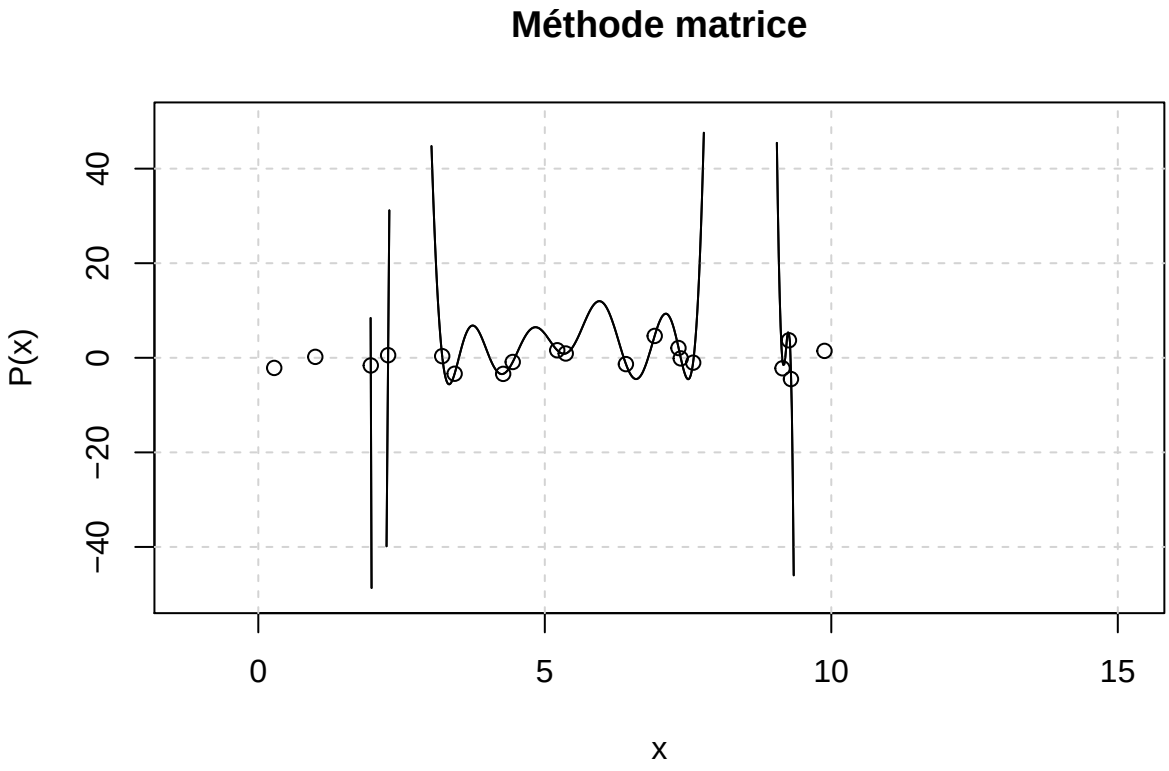



```
ex_deux_q_trois(19)
```

```
##       [,1]        [,2]         [,3]           [,4]           [,5]
##  [1,]    1 0.000000000  0.000000000   0.0000000000    0.000000000
##  [2,]    1 0.716880717  0.000000000   0.0000000000    0.000000000
##  [3,]    1 1.685184534  1.631770617   0.0000000000    0.000000000
##  [4,]    1 1.987953827  2.526834652   0.7650479403    0.000000000
##  [5,]    1 2.931481201  6.492059688   8.0910323512    7.634110511
##  [6,]    1 3.148394136  7.655362589  11.2014000461   12.998556133
##  [7,]    1 3.995675719 13.101001574  30.2697486451   60.773237017
##  [8,]    1 4.162609138 14.343220610  35.5342476351   77.274740343
##  [9,]    1 4.939483604 20.857477727  67.8764703669  200.339423467
## [10,]    1 5.087821778 22.238569123  75.6697835581  234.566336958
## [11,]    1 6.137925602 33.273970186 148.1603735315  614.861368335
## [12,]    1 6.640310637 39.333414712 194.9020299742  906.753786592
## [13,]    1 7.055243130 44.718687875 240.1419742394 1216.868857434
## [14,]    1 7.094956967 45.252176531 244.8039771418 1250.214680111
## [15,]    1 7.311987947 48.223344577 271.3432798518 1444.640880281
## [16,]    1 8.870917873 72.333793939 519.7713546177 3577.567545981
## [17,]    1 8.983119654 74.256613459 541.9199472453 3790.819896087
## [18,]    1 9.018546689 74.868962172 549.0412138022 3860.085239100
## [19,]    1 9.603386905 85.340557165 675.7438021115 5146.081703341
##                 [,6]           [,7]            [,8]           [,9]
##  [1,]     0.000000000      0.00000000 0.000000000e+00       0.0000000
##  [2,]     0.000000000      0.00000000 0.000000000e+00       0.0000000
##  [3,]     0.000000000      0.00000000 0.000000000e+00       0.0000000
```

```
##  [4,]     0.000000000      0.00000000 0.000000000e+00       0.0000000
##  [5,]     0.000000000      0.00000000 0.000000000e+00       0.0000000
##  [6,]     2.819554958      0.00000000 0.000000000e+00       0.0000000
##  [7,]    64.674545638     54.79755139 0.000000000e+00       0.0000000
##  [8,]    95.135091624     96.48743713 1.610697778e+01       0.0000000
##  [9,]   402.282043703    720.52313169 6.800354133e+02     528.3021488
## [10,]   505.804910412    980.97202493 1.071364732e+03     991.2401925
## [11,]  1971.518791610   5893.91746302 1.262624400e+04   24940.8276448
## [12,]  3362.995135203  11743.29820747 3.105673650e+04   76949.3226113
## [13,]  5018.077467128  19604.87090682 5.998242414e+04  173507.1990285
## [14,]  5205.238523151  20542.80088440 6.366791758e+04  186696.4799281
## [15,]  6328.259121525  26348.30051427 8.737919120e+04  275190.1731374
## [16,] 21248.735877425 121596.39543590 5.928118728e+05 2791141.3189903
## [17,] 22940.671450704 133852.52111520 6.675819447e+05 3218085.7847393
## [18,] 23496.591640546 137928.57741797 6.927974476e+05 3364181.1413291
## [19,] 34334.171870935 221626.83117452 1.242819260e+06 6761903.4016711
##                 [,10]           [,11]          [,12]           [,13]
##  [1,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [2,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [3,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [4,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [5,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [6,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [7,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [8,] 0.000000000e+00         0.00000         0.0000          0.0000
##  [9,] 0.000000000e+00         0.00000         0.0000          0.0000
## [10,] 1.470387605e+02         0.00000         0.0000          0.0000
## [11,] 2.989013531e+04     31387.74537         0.0000          0.0000
## [12,] 1.308774881e+05    203185.84220    102077.5266          0.0000
## [13,] 3.670995092e+05    722239.41282    662522.8733     274902.2674
## [14,] 4.024192895e+05    807709.91673    773003.7246     351443.3064
## [15,] 6.528898810e+05   1452135.58516   1704897.7109    1145141.1077
## [16,] 1.097318863e+07  41512627.05003 113453688.8754  253070619.2656
## [17,] 1.301276769e+07  50688606.34304 144218921.2831  337877389.1258
## [18,] 1.372270711e+07  53940186.67010 155381239.1955  369533264.8455
## [19,] 3.153686360e+07 142406761.48476 493505121.3079 1462293313.0388
##                 [,14]           [,15]       [,16]         [,17]         [,18]
##  [1,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [2,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [3,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [4,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [5,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [6,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [7,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [8,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
##  [9,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
## [10,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
## [11,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
## [12,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
## [13,] 0.000000000e+00 0.000000000e+00           0           0.0          0.00
## [14,] 1.395716217e+04 0.000000000e+00           0           0.0          0.00
## [15,] 2.940090438e+05 6.380907084e+04           0           0.0          0.00
## [16,] 4.594939314e+08 8.160432584e+08  1272154256           0.0          0.00
## [17,] 6.513858863e+08 1.229922525e+09  2055362529   230615336.9          0.00
```

```
## [18,] 7.255059740e+08 1.395575835e+09  2381632142   351597534.9   12456058.34
## [19,] 3.726133603e+09 9.346745083e+09 21417121946 15687378595.0 9730367203.59
##            [,19]
##  [1,]          0
##  [2,]          0
##  [3,]          0
##  [4,]          0
##  [5,]          0
##  [6,]          0
##  [7,]          0
##  [8,]          0
##  [9,]          0
## [10,]          0
## [11,]          0
## [12,]          0
## [13,]          0
## [14,]          0
## [15,]          0
## [16,]          0
## [17,]          0
## [18,]          0
## [19,] 5690710057
## 4514361.054 - 36106263.23*x + 110713157*x^2 - 187883987.1*x^3 + 205272356.9*x^4
## - 156519252.8*x^5 + 87435675.67*x^6 - 36908468.35*x^7 + 12009364.86*x^8 -
## 3048903.816*x^9 + 607594.5239*x^10 - 95098.01693*x^11 + 11631.19462*x^12 -
## 1098.796033*x^13 + 78.54813898*x^14 - 4.105966548*x^15 + 0.1479867774*x^16 -
## 0.003285506513*x^17 + 3.385040218e-05*x^18
```




```
ex_deux_q_trois(29)
```

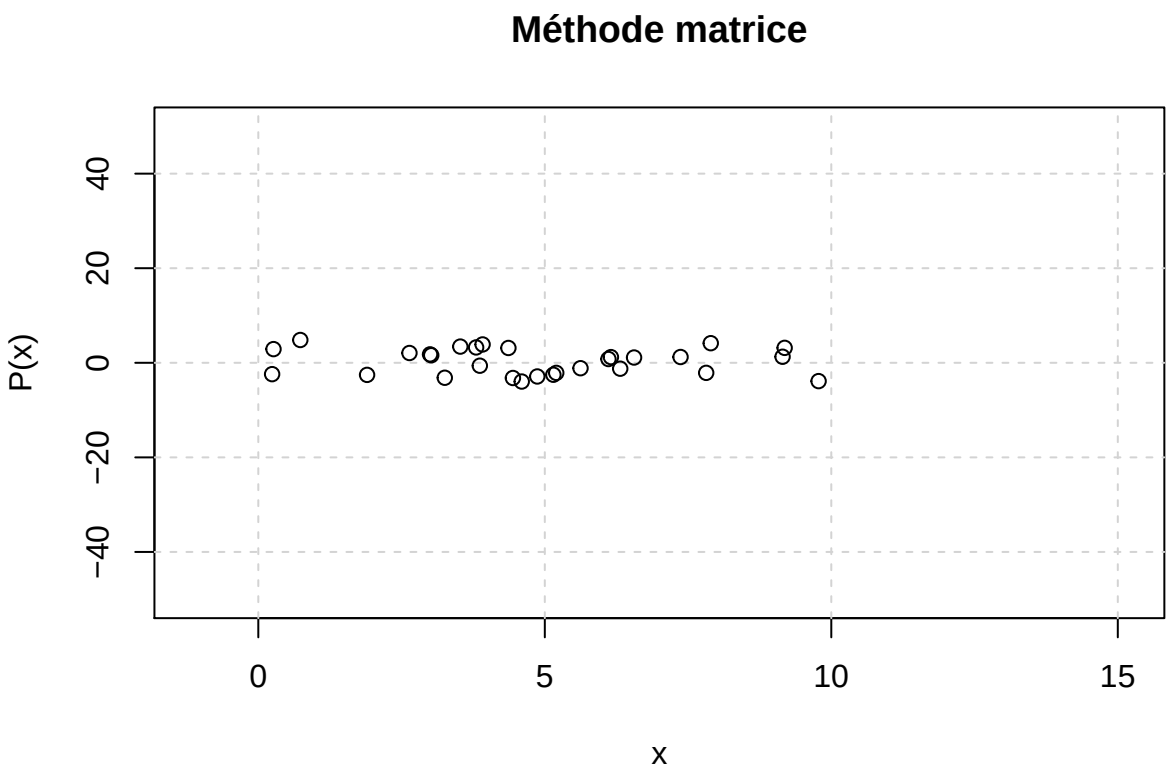
```
##       [,1]          [,2]          [,3]          [,4]          [,5]
##  [1,]    1 0.00000000000  0.0000000000   0.000000000    0.000000000
##  [2,]    1 0.02407651794  0.0000000000   0.000000000    0.000000000
##  [3,]    1 0.49189159130  0.2301143009   0.000000000    0.000000000
##  [4,]    1 1.65692941936  2.7055220098   3.152035486    0.000000000
##  [5,]    1 2.39668103774  5.6863762627  10.831349494    8.012508317
##  [6,]    1 2.75593986298  7.5288510928  17.045682304   18.733382871
##  [7,]    1 2.77696248761  7.6446610705  17.468592524   19.565401283
##  [8,]    1 3.01317794678  9.0066945061  22.708455966   30.798309964
##  [9,]    1 3.28413220235 10.7064538546  29.894995253   48.645219474
## [10,]    1 3.56031872524 12.5901493476  38.631955879   73.531651684
## [11,]    1 3.62494265952 13.0529332878  40.895506581   80.482898414
## [12,]    1 3.67306472400 13.4029698580  42.637167611   85.962298909
## [13,]    1 4.12359269904 16.9047349940  61.392944804  151.435722578
## [14,]    1 4.20235404175 17.5586014397  65.150531324  165.835766595
## [15,]    1 4.35634961584 18.8728962457  72.933515343  196.878204317
## [16,]    1 4.62772242252 21.3043953780  88.111375245  261.760657075
## [17,]    1 4.90848439061 23.9750398004 105.887988145  344.300614250
## [18,]    1 4.95871962200 24.4695115877 109.301100258  360.889301968
## [19,]    1 5.38217293630 28.8382015330 141.026918980  525.359615644
## [20,]    1 5.86963526792 34.3112977995 184.517364778  777.317381760
## [21,]    1 5.91296584003 34.8208013972 188.766149774  803.395608428
## [22,]    1 6.07529274722 36.7629100696 205.262074578  906.922422916
## [23,]    1 6.31670988250 39.7487393609 231.529184082 1078.875168631
```

```
## [24,]    1 7.12806607829 50.6377070056 336.040659310 1838.524370040
## [25,]    1 7.57548602918 57.2055972530 405.221250517 2398.324910689
## [26,]    1 7.65521746608 58.4180434724 418.467482360 2510.088497381
## [27,]    1 8.90834770669 79.1441768699 666.113491414 4830.267553076
## [28,]    1 8.94466488467 79.7916735145 674.460926916 4915.292817050
## [29,]    1 9.53716671793 90.7279272400 820.659063554 6466.988162025
##                 [,6]            [,7]            [,8]            [,9]
##  [1,]     0.000000000 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [2,]     0.000000000 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [3,]     0.000000000 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [4,]     0.000000000 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [5,]     0.000000000 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [6,]     6.730133123 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [7,]     7.440359167 1.564158779e-01 0.000000000e+00 0.000000000e+00
##  [8,]    18.987062896 4.884195677e+00 1.153722524e+00 0.000000000e+00
##  [9,]    43.170256675 2.280219886e+01 1.156458469e+01 3.133473437e+00
## [10,]    85.564201123 6.882603475e+01 5.391530363e+01 2.949926120e+01
## [11,]    98.854055330 8.590445053e+01 7.284527073e+01 4.456416612e+01
## [12,]   109.720875961 1.006277431e+02 9.017274565e+01 5.950380252e+01
## [13,]   261.516115257 3.576632567e+02 4.816401470e+02 5.348203245e+02
## [14,]   299.445166839 4.331217351e+02 6.173680631e+02 7.341593424e+02
## [15,]   385.816030710 6.174637384e+02 9.752142805e+02 1.309880193e+03
## [16,]   583.998858841 1.093118879e+03 2.023100625e+03 3.266385938e+03
## [17,]   864.815437267 1.861553737e+03 3.967942564e+03 7.520467110e+03
## [18,]   924.612316290 2.036717295e+03 4.443622490e+03 8.645252743e+03
## [19,]  1568.456876334 4.119133323e+03 1.073120917e+04 2.542218076e+04
## [20,]  2699.587689172 8.405693783e+03 2.599606036e+04 7.425663694e+04
## [21,]  2824.967768137 8.918496628e+03 2.796843532e+04 8.110253015e+04
## [22,]  3336.215444529 1.107407636e+04 3.652596115e+04 1.118466862e+05
## [23,]  4229.221780930 1.505928612e+04 5.330606882e+04 1.760983007e+05
## [24,]  8698.766701077 3.803210593e+04 1.654816327e+05 6.809384064e+05
## [25,] 12420.457018566 5.986096601e+04 2.872442546e+05 1.310496784e+06
## [26,] 13199.391801836 6.466748463e+04 3.154644788e+05 1.464398578e+06
## [27,] 31453.092227454 1.935122513e+05 1.186498158e+06 6.994608058e+06
## [28,] 32185.257968945 1.991857113e+05 1.228518189e+06 7.286939592e+06
## [29,] 46177.436364857 3.131396716e+05 2.116888132e+06 1.381055441e+07
##                [,10]           [,11]           [,12]           [,13]
##  [1,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [2,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [3,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [4,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [5,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [6,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [7,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [8,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [9,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [10,] 8.147298379e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [11,] 1.518793383e+01 9.815040376e-01 0.000000000e+00 0.000000000e+00
## [12,] 2.314296396e+01 2.609276586e+00 1.255637761e-01 0.000000000e+00
## [13,] 4.489605352e+02 2.528877848e+02 1.261025039e+02 5.681270572e+01
## [14,] 6.741211418e+02 4.328095807e+02 2.499091782e+02 1.322742584e+02
## [15,] 1.404476352e+03 1.118006562e+03 8.177177763e+02 5.587342024e+02
## [16,] 4.388684201e+03 4.684497743e+03 4.697519536e+03 4.484523189e+03
## [17,] 1.221588721e+04 1.646903970e+04 2.113869973e+04 2.611516537e+04
```

```
## [18,] 1.447723148e+04 2.024497349e+04 2.700227924e+04 3.471561257e+04
## [19,] 5.333677078e+04 9.717182045e+04 1.707532649e+05 2.918358074e+05
## [20,] 1.919907625e+05 4.433674438e+05 9.952236238e+05 2.186078897e+06
## [21,] 2.132050594e+05 5.015962677e+05 1.147663888e+06 2.570653623e+06
## [22,] 3.121820577e+05 7.851297652e+05 1.923842789e+06 4.621509060e+06
## [23,] 5.340317763e+05 1.472000466e+06 3.962282606e+06 1.047486923e+07
## [24,] 2.617482208e+06 9.338515218e+06 3.271397136e+07 1.130268153e+08
## [25,] 5.623805391e+06 2.258051953e+07 8.920532171e+07 3.481167480e+08
## [26,] 6.401011043e+06 2.621149206e+07 1.056395161e+08 4.206726887e+08
## [27,] 3.933918309e+07 2.103870913e+08 1.111560220e+09 5.819332303e+09
## [28,] 4.124795972e+07 2.220932935e+08 1.181474629e+09 6.228261846e+09
## [29,] 8.635787338e+07 5.161478822e+08 3.051581927e+09 1.789478766e+10
##                [,14]           [,15]           [,16]           [,17]
##  [1,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [2,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [3,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [4,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [5,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [6,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [7,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [8,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [9,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [10,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [11,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [12,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [13,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [14,] 1.041809820e+01 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [15,] 1.300492503e+02 2.002700895e+01 0.000000000e+00 0.000000000e+00
## [16,] 2.260781435e+03 9.616649384e+02 2.609697134e+02 0.000000000e+00
## [17,] 2.049757633e+04 1.447396072e+04 7.991577044e+03 2.243730899e+03
## [18,] 2.899194270e+04 2.192850756e+04 1.320907524e+04 4.372166911e+03
## [19,] 3.672987797e+05 4.333460402e+05 4.445364739e+05 3.353807711e+05
## [20,] 3.816986813e+06 6.363990453e+06 9.630535443e+06 1.196028567e+07
## [21,] 4.599858548e+06 7.868572303e+06 1.224834731e+07 1.574210775e+07
## [22,] 9.019799456e+06 1.689353152e+07 2.903901996e+07 4.203602356e+07
## [23,] 2.297261570e+07 4.857228418e+07 9.521917597e+07 1.608239942e+08
## [24,] 3.395860578e+08 9.935310168e+08 2.753786275e+09 6.885412043e+09
## [25,] 1.201661880e+09 4.053364127e+09 1.304833206e+10 3.846339837e+10
## [26,] 1.485658086e+09 5.129774467e+09 1.692244807e+10 5.123262765e+10
## [27,] 2.784407938e+10 1.310340612e+11 5.964667963e+11 2.553250849e+12
## [28,] 3.002689995e+10 1.423968932e+11 6.533618394e+11 2.820525468e+12
## [29,] 9.687475755e+10 5.168086846e+11 2.677491272e+12 1.314499425e+13
##                [,18]           [,19]           [,20]           [,21]
##  [1,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [2,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [3,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [4,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [5,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [6,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [7,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [8,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [9,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [10,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [11,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
```

```
## [12,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [13,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [14,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [15,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [16,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [17,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [18,] 2.196368165e+02 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [19,] 1.588660297e+05 6.727234682e+04 0.000000000e+00 0.000000000e+00
## [20,] 1.149563907e+07 1.047155749e+07 5.104489829e+06 0.000000000e+00
## [21,] 1.581265521e+07 1.508916643e+07 8.009222466e+06 3.470441916e+05
## [22,] 4.904798357e+07 5.476566030e+07 3.795916411e+07 7.806586008e+06
## [23,] 2.264764483e+08 3.075528110e+08 2.874194648e+08 1.284979465e+08
## [24,] 1.528273448e+10 3.315354589e+10 5.788254840e+10 7.284118229e+10
## [25,] 1.025819465e+11 2.684329915e+11 5.887575948e+11 1.004332591e+12
## [26,] 1.407223529e+11 3.794575212e+11 8.625238429e+11 1.540107219e+12
## [27,] 1.021265441e+13 4.033618667e+13 1.422324438e+14 4.322034961e+14
## [28,] 1.138414987e+13 4.537659826e+13 1.616537660e+14 4.970901180e+14
## [29,] 6.084400257e+13 2.785710469e+14 1.157460968e+15 4.245024501e+15
##                 [,22]           [,23]           [,24]           [,25]
##  [1,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [2,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [3,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [4,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [5,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [6,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [7,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [8,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
##  [9,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [10,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [11,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [12,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [13,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [14,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [15,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [16,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [17,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [18,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [19,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [20,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [21,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [22,] 1.267218962e+06 0.000000000e+00 0.000000000e+00 0.000000000e+00
## [23,] 5.188028035e+07 1.252478866e+07 0.000000000e+00 0.000000000e+00
## [24,] 8.850933796e+10 9.318027056e+10 7.560238984e+10 0.000000000e+00
## [25,] 1.669723210e+12 2.504907542e+12 3.153117863e+12 1.410767840e+12
## [26,] 2.683254307e+12 4.239339807e+12 5.674388481e+12 2.991261763e+12
## [27,] 1.294614515e+15 3.667714072e+15 9.505386517e+15 1.692226499e+16
## [28,] 1.507027636e+15 4.324223108e+15 1.136386375e+16 2.064358132e+16
## [29,] 1.538482152e+16 5.326031317e+16 1.715225396e+17 4.132150599e+17
##                 [,26]           [,27]           [,28]           [,29]
##  [1,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
##  [2,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
##  [3,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
##  [4,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
##  [5,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
```

```
##  [6,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
##  [7,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
##  [8,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
##  [9,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [10,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [11,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [12,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [13,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [14,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [15,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [16,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [17,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [18,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [19,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [20,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [21,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [22,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [23,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [24,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [25,] 0.000000000e+00 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [26,] 2.384975985e+11 0.000000000e+00 0.000000000e+00 0.00000000e+00
## [27,] 2.255503850e+16 2.826440082e+16 0.000000000e+00 0.00000000e+00
## [28,] 2.826475505e+16 3.644591543e+16 1.323612797e+15 0.00000000e+00
## [29,] 8.105960033e+17 1.525500542e+18 9.592637424e+17 5.68365526e+17
## 156857798300 - 2445878258000*x + 16261403090000*x^2 - 6.2083252e+13*x^3 +
## 1.5570215e+14*x^4 - 27757839420000*x^5 + 37046725390000*x^6 -
## 38381010860000*x^7 + 31689540510000*x^8 - 21264837570000*x^9 +
## 1.17702076e+14*x^10 - 54344362220000*x^11 + 21107332820000*x^12 -
## 6939001281000*x^13 + 1939041687000*x^14 - 461746348100*x^15 + 93788587660*x^16
## - 16238337190*x^17 + 2390941107*x^18 - 298136806.7*x^19 + 31283221.04*x^20 -
## 2737108.047*x^21 + 197154.9425*x^22 - 11483.46158*x^23 + 527.1290501*x^24 -
## 18.34778552*x^25 + 0.454921193*x^26 - 0.00715555021*x^27 + 5.36443235e-05*x^28
```



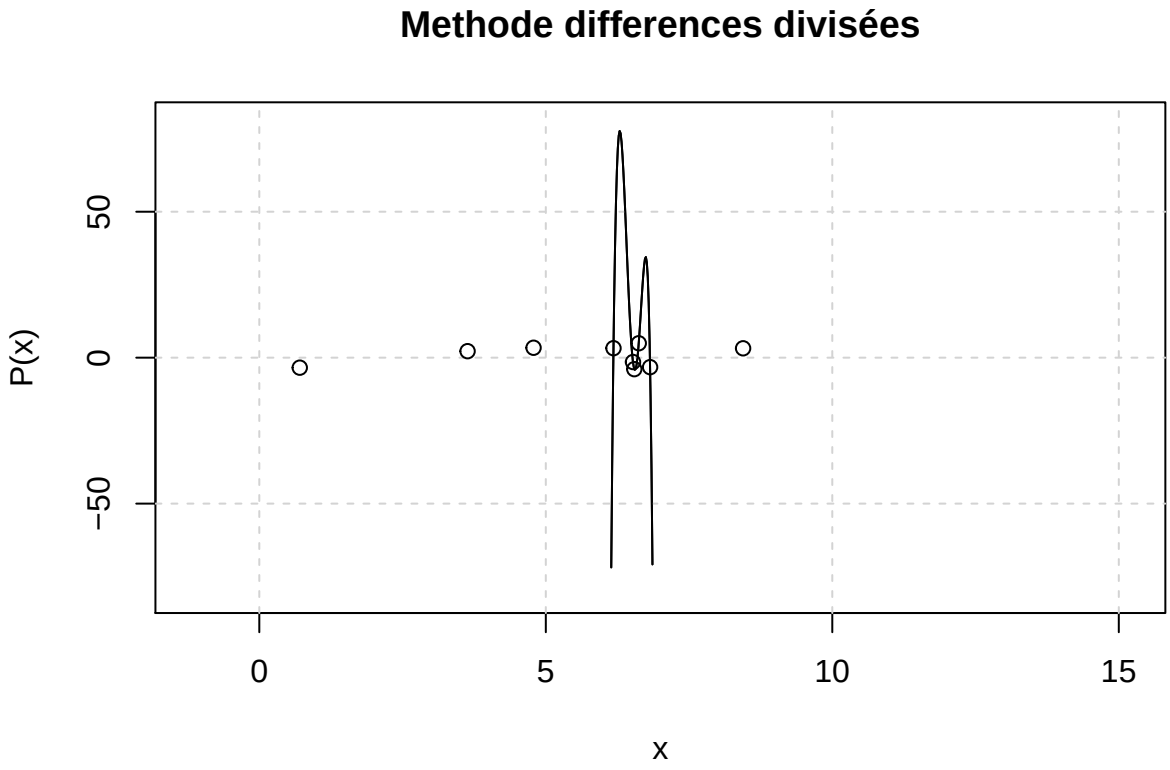
- La première calcule la matrice de Newton
- La deuxième calcule les coefficients du polynôme dans la base de Newton
- La troisième calcule les coefficients du polynôme dans la base canonique de $R[X]$
- On remarque que la matrice de Newton est triangulaire, on peut l'inverser très facilement avec le pivot de Gauss. Le calcul est donc plus rapide qu'avec la méthode de Vandermonde. En revanche, il faut projeter le polynôme obtenu dans la base canonique de $R[X]$

4. Toujours, pour l'interpolation de Newton, implémenter le tableau des différences divisées afin de calculer et tracer le polynôme d'interpolation.

```
#Fonction de calcul des différences divisées
diff_div_coef<-function(x,y){
  m <-length(x)
  res<-y
  for (k in c(2:m)) {
    res[k:m]<-(res[k:m] - res[k-1])/(x[k:m] - x[k-1])
  }
  return(res)
}

#Fonction question 4 exo 2
ex_deux_q_quatre<-function(N){
  delta<-generate(N,0,10,-5,5)

  #Cette fois-ci on calcule les coefficients grâce aux différences divisées
  coefficients2<-diff_div_coef(delta$X,delta$Y)
  #On calcule le polynô me associé
```

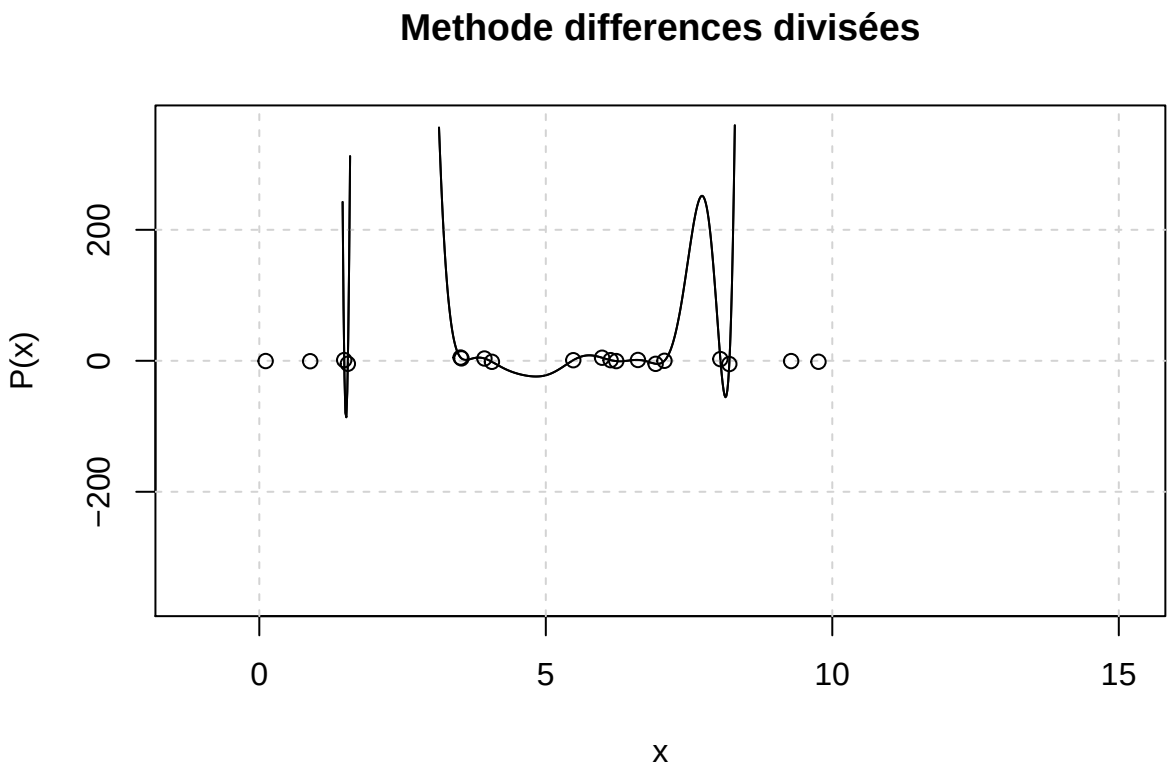
```
  p2<-polynome_newton_from_coef(delta$X, coefficients2)
  print(coefficients2)
  print(p2)
  plot(p2, ylim = c(-N*N, N*N), xlim = c(-1,15), main = "Methode differences divisées")
  points(delta$X, delta$Y)
}
ex_deux_q_quatre(9)
```

```
## [1]   -3.42799981395    1.92761193919   -0.22190533094   -0.04079041667
## [5]   -0.44019773829   -8.00854633221  179.69906654168 -746.33830779917
## [9]  400.56410889582
## 75232543.06 - 198032671.4*x + 176630325.4*x^2 - 79947047.33*x^3 +
## 21053843.54*x^4 - 3377500.508*x^5 + 326338.184*x^6 - 17496.61406*x^7 +
## 400.5641089*x^8
```



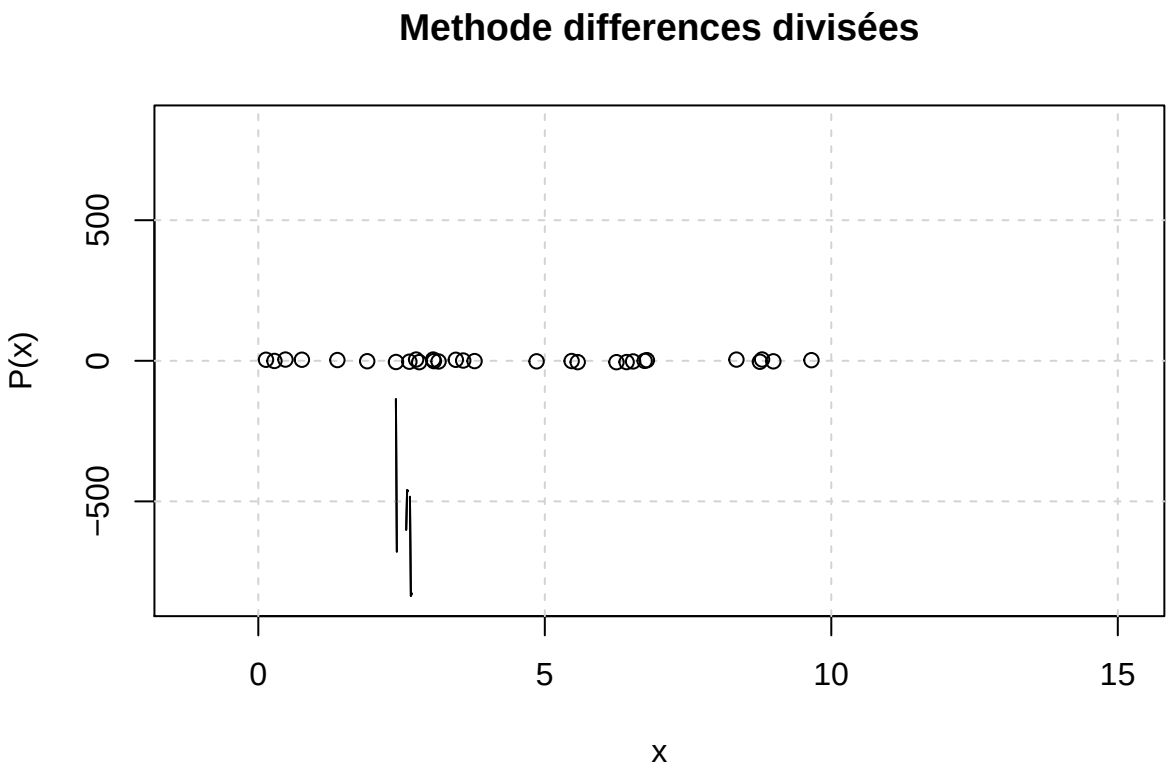

```
ex_deux_q_quatre(19)
```

```
##  [1] -3.614483008e-01 -3.920857293e-01  2.465607911e+00 -9.987510577e+01
##  [5]  5.032773605e+01 -2.760997350e+01  1.628330330e+01 -1.627752041e+01
##  [9]  7.524216561e+00 -2.985680687e+00  1.156606919e+00 -4.338514803e-01
## [13]  1.367168619e-01 -3.474168933e-02  7.088567704e-03 -1.343068602e-03
## [17]  6.835892162e-04 -3.210664986e-04  1.284914778e-04
## 1961263.592 - 27433859.8*x + 108826976.5*x^2 - 220339684.2*x^3 +
## 276564360.2*x^4 - 236851033.8*x^5 + 146356350.7*x^6 - 67597393.48*x^7 +
## 23874024.57*x^8 - 6539687.026*x^9 + 1399865.816*x^10 - 234549.4859*x^11 +
## 30631.49981*x^12 - 3083.915402*x^13 + 234.5988375*x^14 - 13.03539118*x^15 +
## 0.4989711233*x^16 - 0.01175719295*x^17 + 0.0001284914778*x^18
```




```
ex_deux_q_quatre(29)
```

```
##  [1]  3.665473036e+00 -2.848557603e+01  1.586912833e+02 -3.443722922e+02
##  [5]  3.183607360e+02 -1.998243640e+02  9.617290111e+01 -4.171130668e+01
##  [9]  2.159306215e+01 -1.132392338e+02  3.779346940e+02 -1.502276864e+03
## [13]  5.935960174e+03 -1.150723791e+04  1.865778024e+04 -2.342240690e+04
## [17]  1.251405124e+04 -5.049395439e+03  1.957712206e+03 -6.018858565e+02
## [21]  1.756576610e+02 -4.966850111e+01  1.329320616e+01 -3.523542876e+00
## [25]  6.605375916e-01 -1.150220339e-01  1.990394898e-02 -3.331571132e-03
## [29]  5.018730794e-04
## 39383137830 - 810821212600*x + 6835629640000*x^2 - 32397912230000*x^3 +
## 99316897790000*x^4 - 213681019600000*x^5 + 340444878800000*x^6 -
## 417045520500000*x^7 + 403693885400000*x^8 - 315143740300000*x^9 +
## 201487586700000*x^10 - 106746432500000*x^11 + 47277327480000*x^12 -
## 17618213620000*x^13 + 5549305803000*x^14 - 1481437785000*x^15 +
## 335574697700*x^16 - 64469102700*x^17 + 10481813150*x^18 - 1436475672*x^19 +
## 164905608.3*x^20 - 15716292.7*x^21 + 1227887.889*x^22 - 77257.48663*x^23 +
## 3815.839014*x^24 - 142.3689617*x^25 + 3.770023572*x^26 - 0.06311158397*x^27 +
## 0.0005018730794*x^28
```



5. Appliquer les fonctions des questions 2, 3 et 4 à

$$\Delta = \{(0,2),(1,6),(2,12),(3,20),(4,30),(5,42),(6,56),(7,72),(8,90),(9,110)\}$$

Comparer les résultats et donner le polyôme d'interpolation P dans la base canonique des polynômes. Prévision : calculer $P(10), P(15)$ **et** $P(20)$

```
ex_deux_q_cinq<-function(){
  delta$X<-c(0,1,2,3,4,5,6,7,8,9)
  delta$Y<-c(2,6,12,20,30,42,56,72,90,110)
  #Polynôme question 2
  vdm<-vandermonde(delta$X)
  p1<-as.polynomial(coef_vdm(delta))

  #Polynôme question 3
  coefficients1<-coef_newton(delta)
  p2<-polynome_newton_from_coef(delta$X, coefs = coefficients1)

  #Polynôme question 4
  coefficients2<-diff_div_coef(delta$X,delta$Y)
  #On calcule le polynome associé
  p3<-polynome_newton_from_coef(delta$X, coefs = coefficients2)


  print("Vandermonde : Y = ")
  print(p1)
  print("Matrice de Newton : Y = ")
```

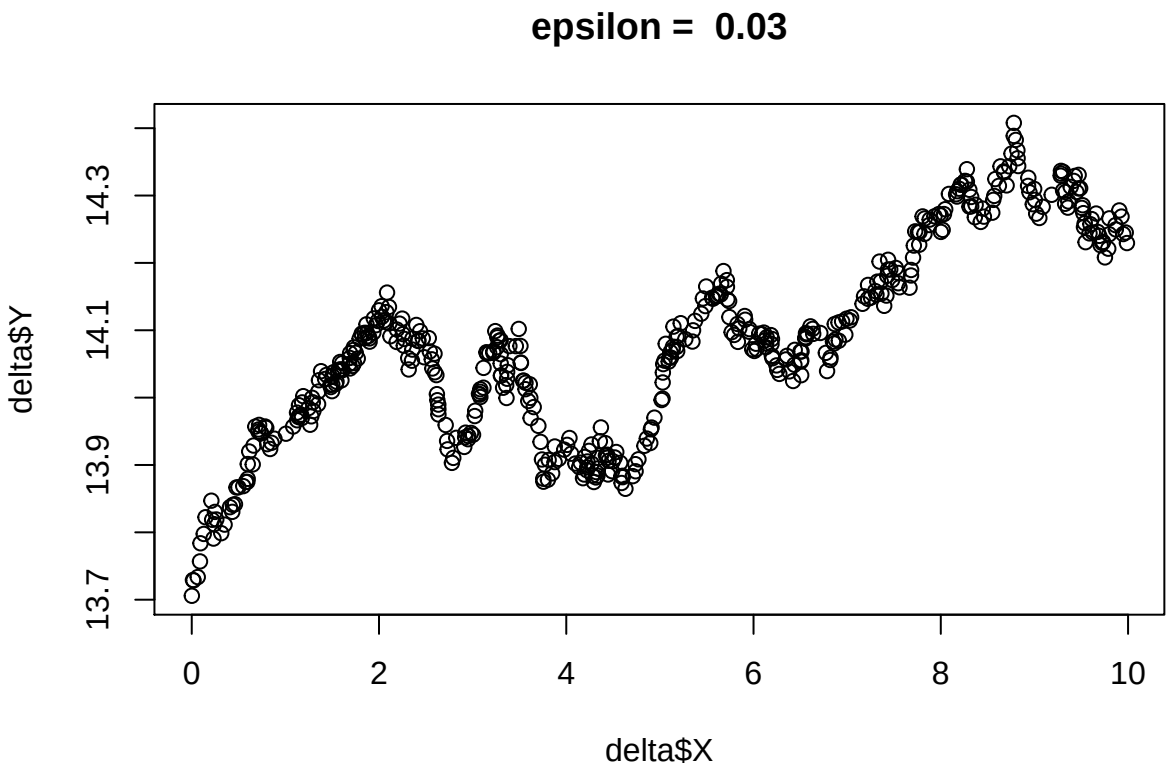
```
  print(p2)
  print("Différences divisées : Y = ")
  print(p3)

  for (new_val in c(5,10,15)) {
    cat("Vandermonde : X = ", new_val, " P(X) = ", predict(p1,new_val),"\n")
    cat("Matrice de Newton : X = ", new_val, " P(X) = ", predict(p2,new_val),"\n")
    cat("Différences divisées : X = ", new_val, " P(X) = ", predict(p3,new_val), "\n")
  }

}
ex_deux_q_cinq()
```

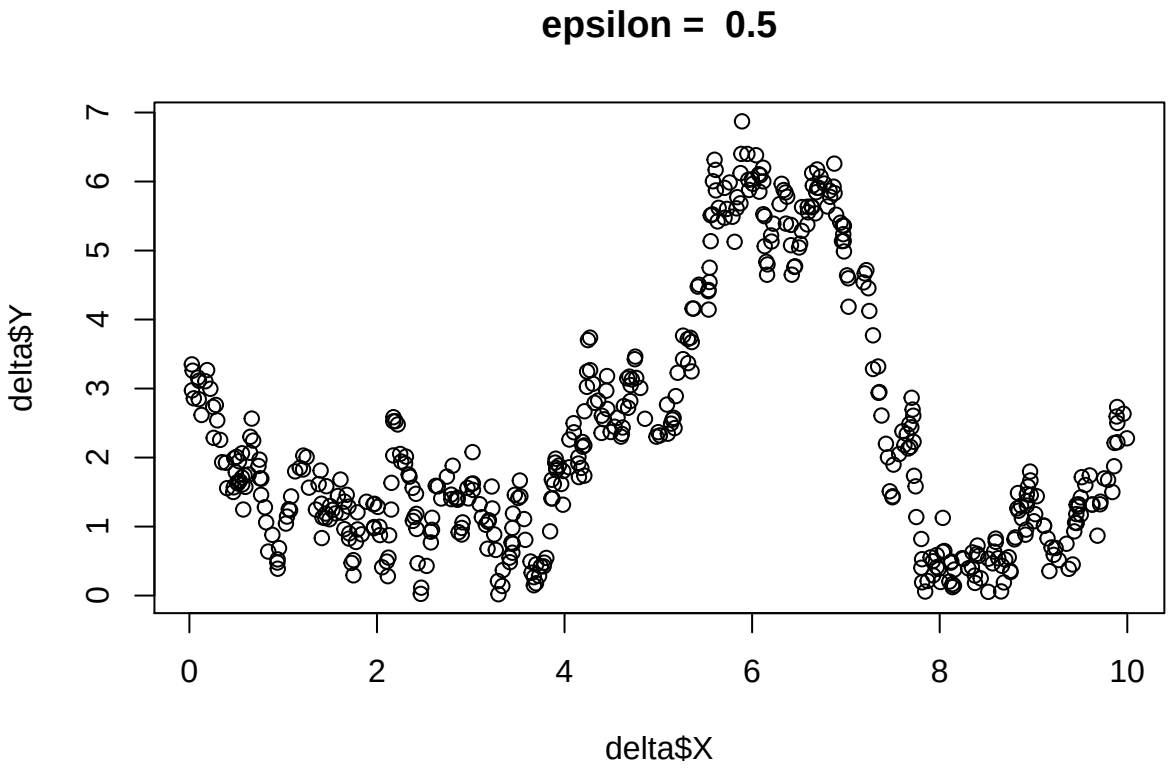
```
## [1] "Vandermonde : Y = "
## 2 + 3*x + x^2 + 6.399008307e-14*x^3 - 4.179249539e-14*x^4 + 1.517058084e-14*x^5
## - 3.212245284e-15*x^6 + 3.940410608e-16*x^7 - 2.590520391e-17*x^8 +
## 7.049035077e-19*x^9
## [1] "Matrice de Newton : Y = "
## 2 + 3*x + x^2 - 5.985026876e-13*x^3 + 3.930098193e-13*x^4 - 1.411586522e-13*x^5
## + 2.777521654e-14*x^6 - 3.044752914e-15*x^7 + 1.740264954e-16*x^8 -
## 4.011548038e-18*x^9
## [1] "Différences divisées : Y = "
## 2 + 3*x + x^2
## Vandermonde : X =  5  P(X) =  42
## Matrice de Newton : X =  5  P(X) =  42
## Différences divisées : X =  5  P(X) =  42
## Vandermonde : X =  10  P(X) =  132
## Matrice de Newton : X =  10  P(X) =  132
## Différences divisées : X =  10  P(X) =  132
## Vandermonde : X =  15  P(X) =  272
## Matrice de Newton : X =  15  P(X) =  272
## Différences divisées : X =  15  P(X) =  272
```

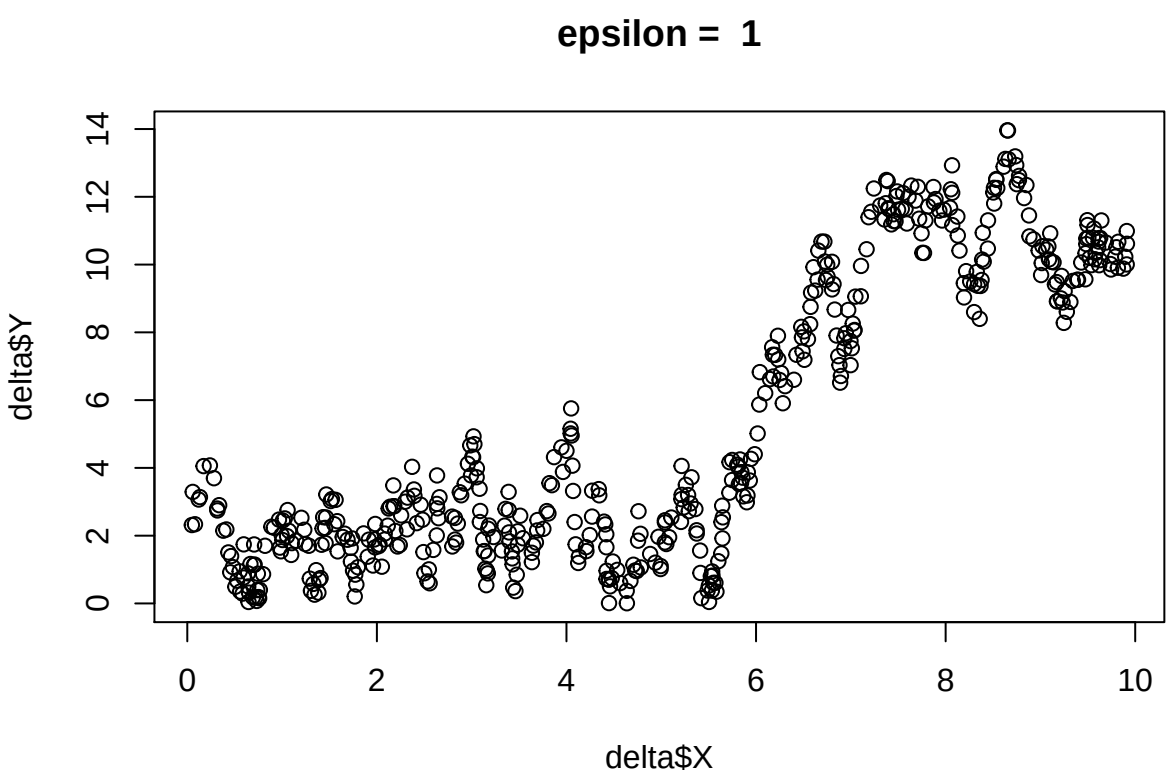
Exercice 3 : Approximation de données 1. Créer une fonction qui génère graphiquement Δ aléatoirement tel que l'utilisateur saisisse le nombre de points, l'incertitude ϵ et a, b, c, d pour que $\forall i = 0, ..., n, a \leq x_i \leq b, c \leq y_i \leq d$. Proposer plusieurs exemple avec $\epsilon = 0.03, \epsilon = 0.5, \epsilon = 1 et \epsilon = 5$

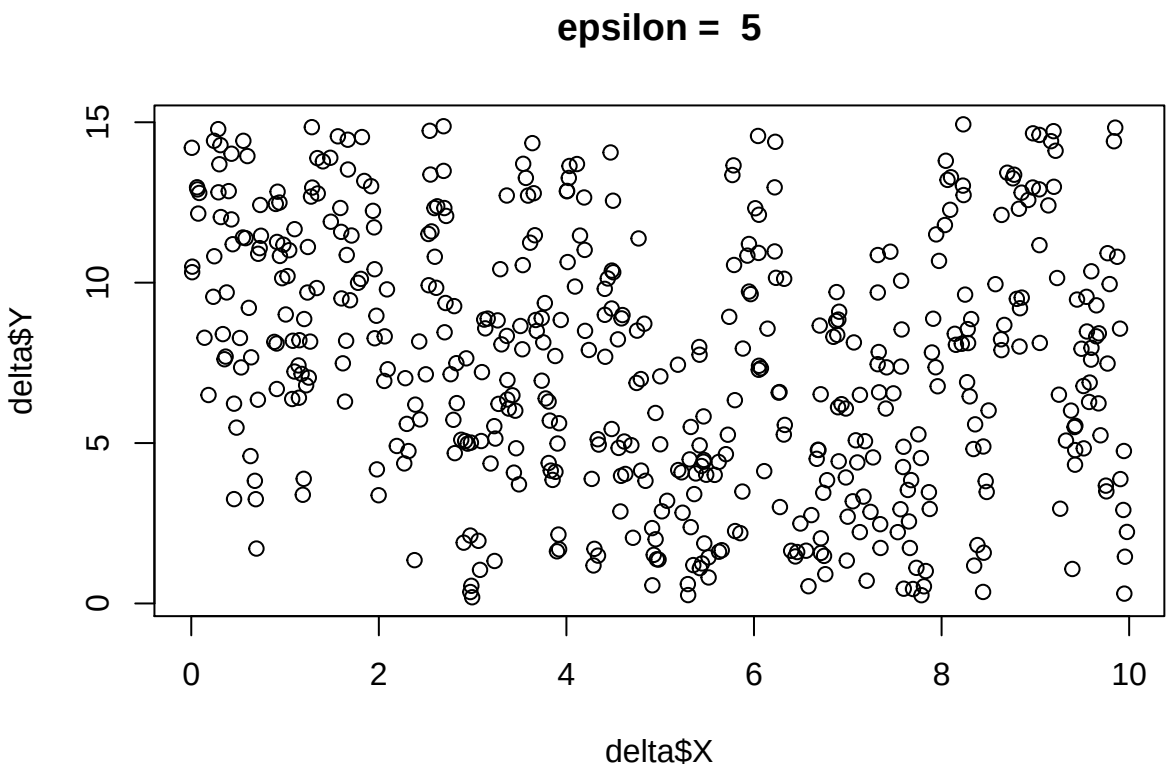
```
generate<-function(N,e,a,b,c,d){

  RNGkind("Wich")
  #Xi
  X<-sort(runif(n = N, min = a, max = b))
  #Yi
  Y<-rep(c,N)
  Y[1]<-runif(1,c,d)

  for (i in c(2:N)) {
    Y[i]<-runif(1,min = max(Y[i-1] - e,c), max = min(Y[i-1] + e, d))
  }

  L<-list("X" = X, "Y" = Y)
  return(L)
}
```

```
q1<-function(){
for (e in c(0.03,0.5,1,5)) {
  delta<-generate(500,e,0,10,0,15)
  plot(delta$X,delta$Y, main = paste("epsilon = ",e))

  }
}
q1()
```



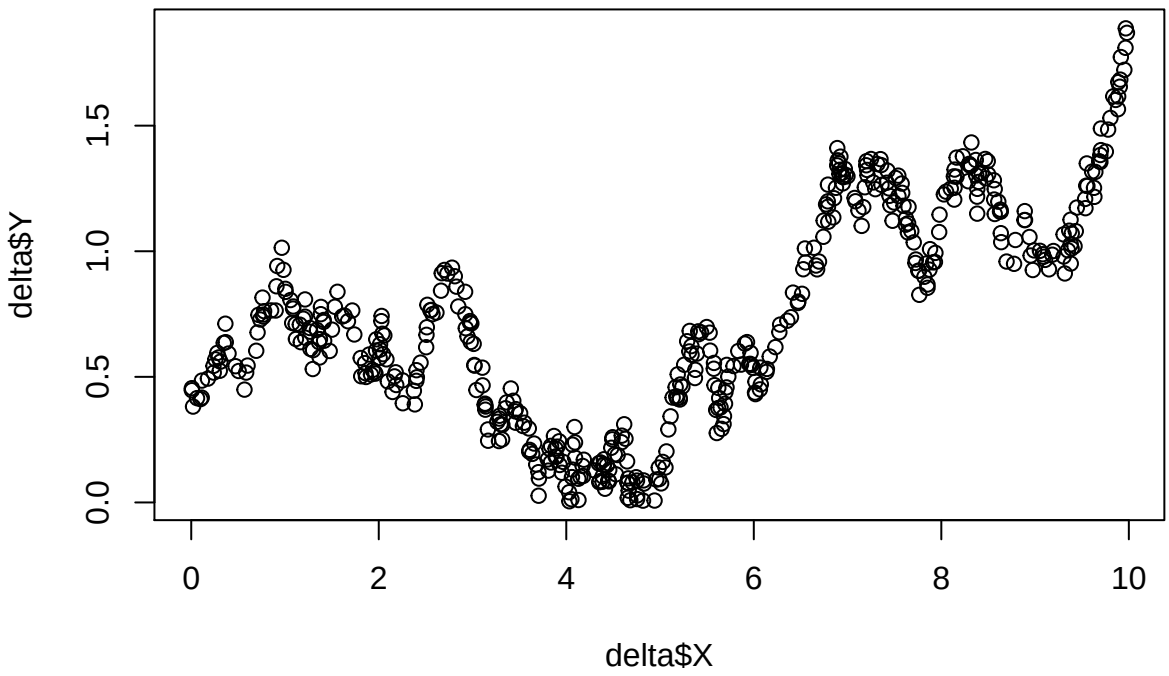

epsilon = 0.5
delta$Y
delta$X
0
1
2
3
4
5
6
7
0
2
4
6
8
10


epsilon = 1
delta$Y
0
2
4
6
8
10
12
14
delta$X
0
2
4
6
8
10



2. Comment choisir le degré du polynôme d'interpolation de Δ ? Donner un critère de choix et écrire la fonction.

```
q2<-function(){

  delta<-generate(500,0.1,0,10,0,5)
  plot(delta$X,delta$Y)

  # degre<-readline(prompt = "Combien de pics voyez vous?")

 # return((as.integer(degre)+1))


}
q2()
```

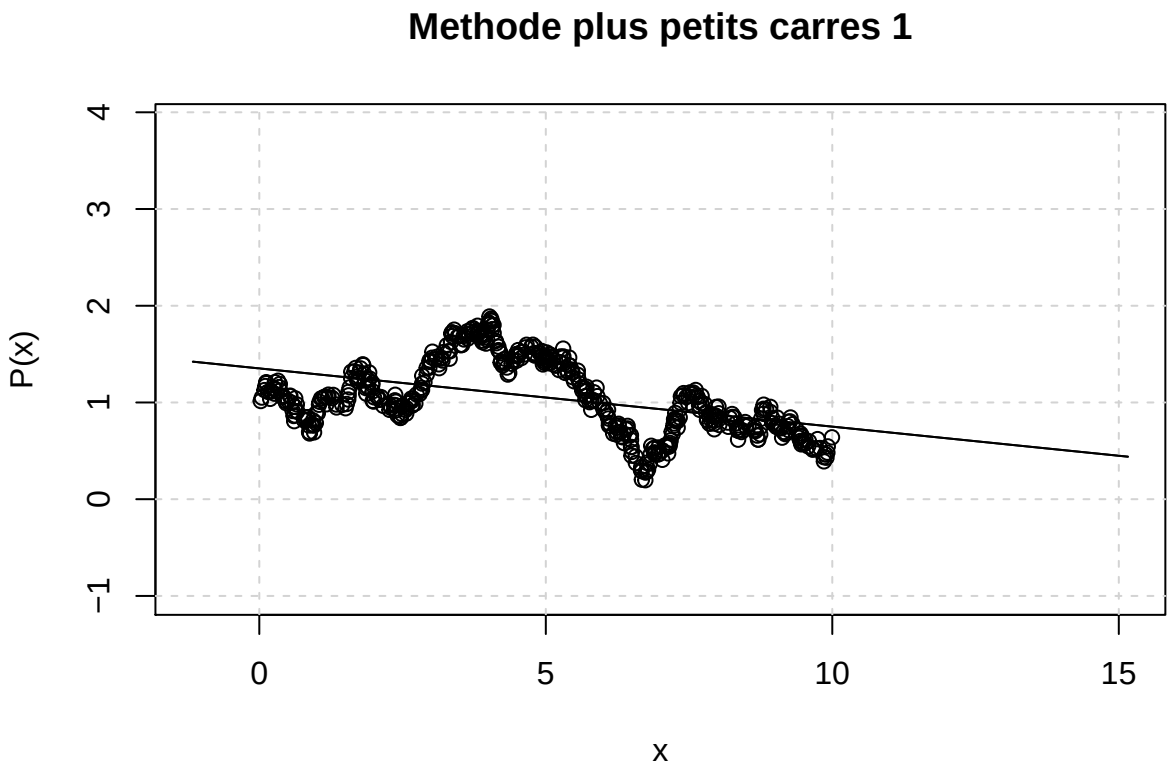


3. Déterminer le polynôme d'approximation par la méthode des moindres carrés. Effectuer le tracé.

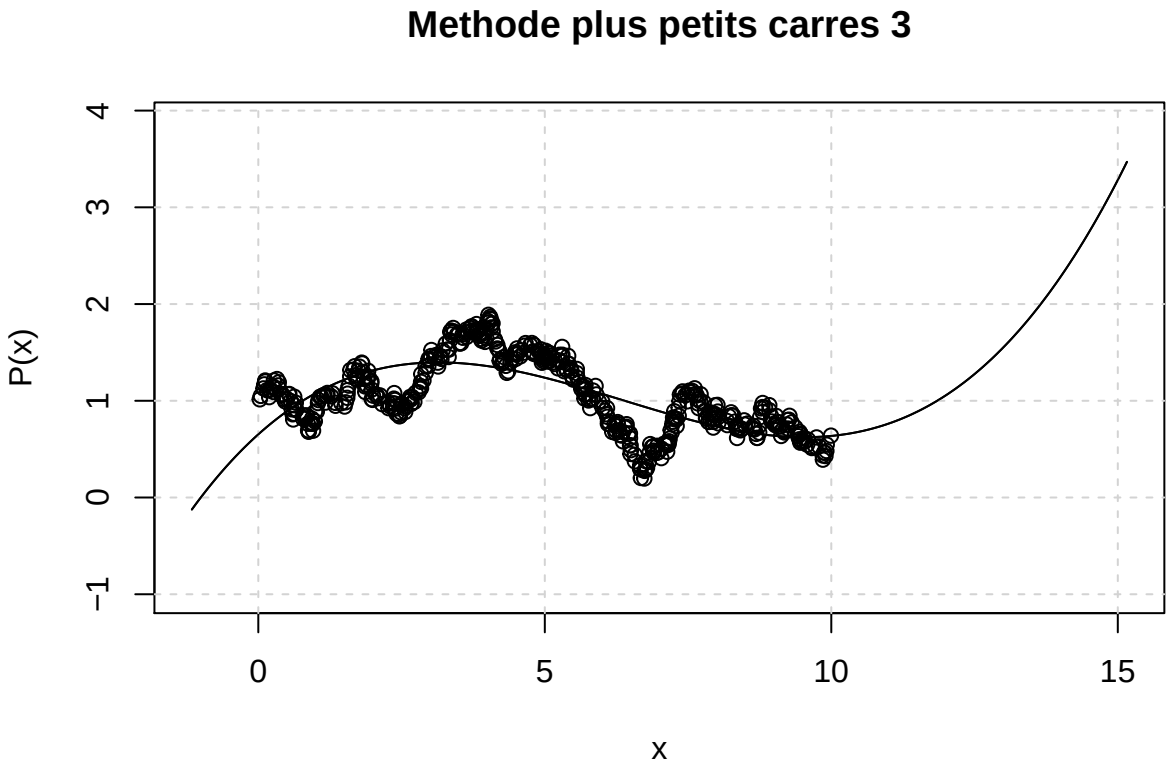
```
vandermonde <- function(N, xi) {
  N <- N+1
  #On creer la matrice contenant les xi
  res <- matrix(rep(xi, N), ncol = N)
  #On les met a la bonne puissance
  res <- res^(col(res)-1)
  return(res)
}

aux <- function(delta) {
  plot(delta$X, delta$Y)
  degre <- readline(prompt = "Combien de pics voyez vous?")
  return((as.integer(degre) + 1))
}

q3<-function(N, d) {
  m <- vandermonde(N, d$X)
  y0 <- d$Y
  m[m < 10^-6] <- 0
  res <- solve(t(m) %*% m, tol = NULL) %*% t(m) %*% y0
  return(res)
}

delta<-generate(500,0.1,0,10,0,5)
```

```r
#d<-aux(delta)
#print(d)

for (d in c(1,3,5,7,9)) {
  res <- q3(d, delta)
  print(as.polynomial(res))
  plot(as.polynomial(res), ylim = c(-1,max(delta$Y) +2), xlim = c(-1,15), main = paste("Methode plus pe
  points(delta$X, delta$Y)
}
```

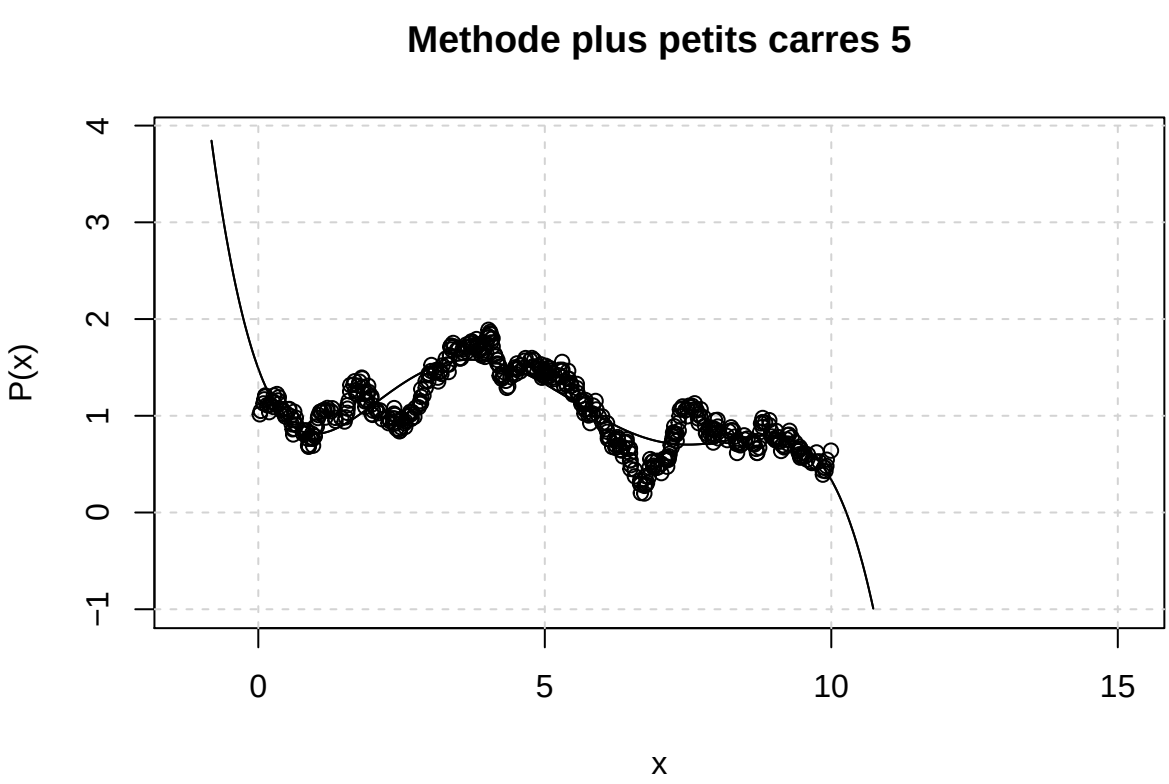
```
## 1.352120926 - 0.06019544231*x
```



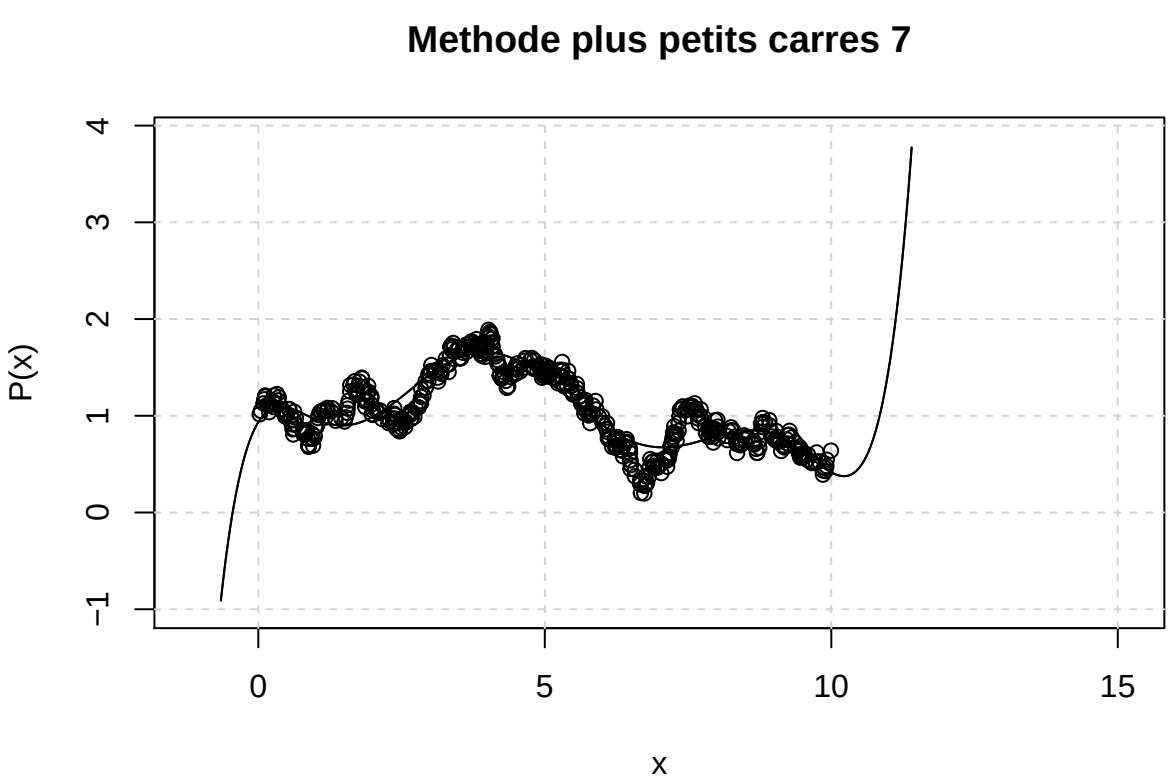

```
## 0.6529939761 + 0.5315954858*x - 0.1124610489*x^2 + 0.005913814889*x^3
```



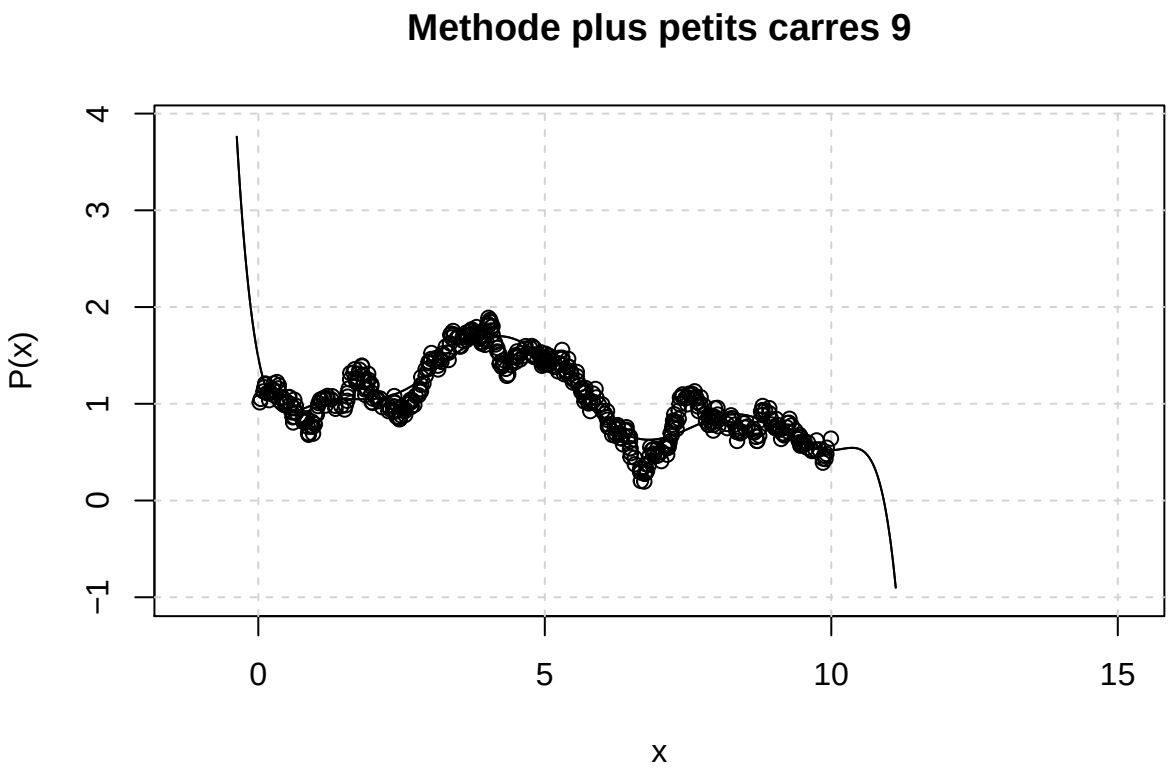

```
## 1.48830032 - 1.626988943*x + 1.245408754*x^2 - 0.3279787339*x^3 +
## 0.03522458094*x^4 - 0.001336886234*x^5
```




```
## 0.9341104215 + 0.9629428678*x - 1.862774527*x^2 + 1.244378968*x^3 -
## 0.3643566611*x^4 + 0.05239438808*x^5 - 0.003652575491*x^6 + 9.884529401e-05*x^7
```




```
## 1.47675952 - 3.030853867*x + 5.806558729*x^2 - 5.110813922*x^3 +
## 2.413607081*x^4 - 0.6501375089*x^5 + 0.1027085771*x^6 - 0.009409531293*x^7 +
## 0.0004627333242*x^8 - 9.449407603e-06*x^9
```



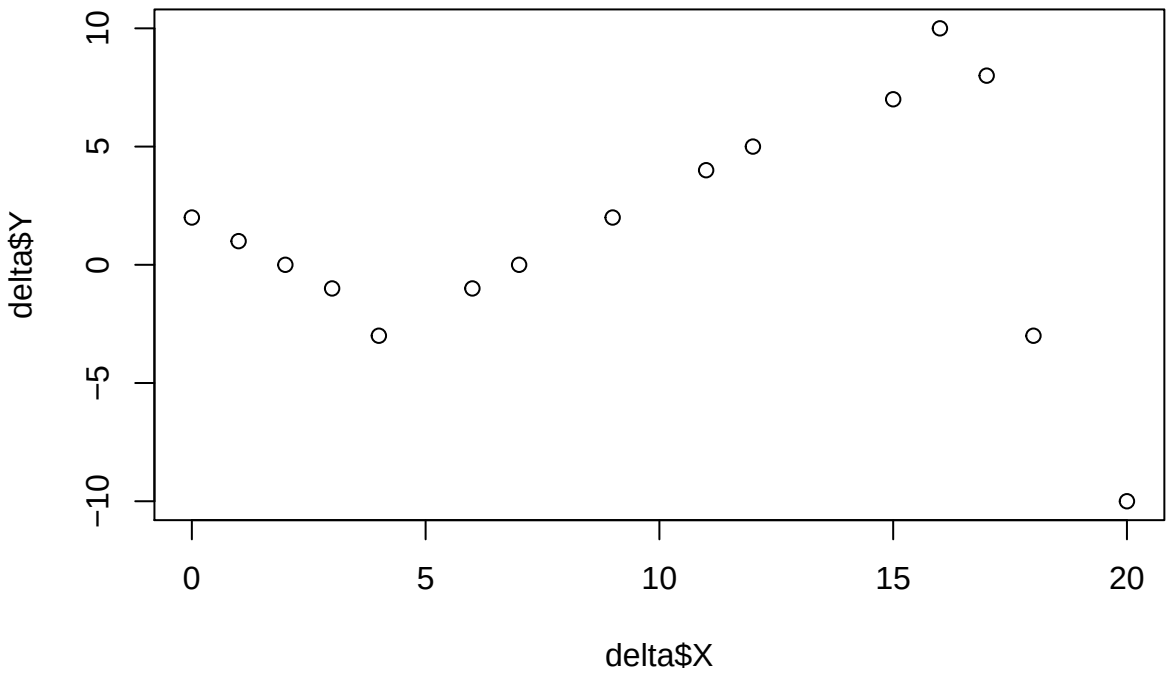
4. Soit le nuage de points suivant à traiter $\Delta = \{(0,2),(1,1),(2,0),(3,-1),(4,-3),(6,-1),(7,0),(9,2),(11,4),(12,5),(15$

Fixer ϵ **et donner un polynôme d'interpolation** P **en fonction du degré choisi. Prévision : calculer** $P(22), P(25)$ **et** $P(50)$

```
delta$X<-c(0,1,2,3,4,6,7,9,11,12,15,16,17,18,20)
delta$Y<-c(2,1,0,-1,-3,-1,0,2,4,5,7,10,8,-3,-10)


find_epsilon<-function(d){
  m<-0
  for (i in c(2:length(d$Y))){
    m<-max(abs(d$Y[i] - d$Y[i-1]), m)
  }
  return(m)

}

e<-find_epsilon(delta)
e
```

```
## [1] 11
```

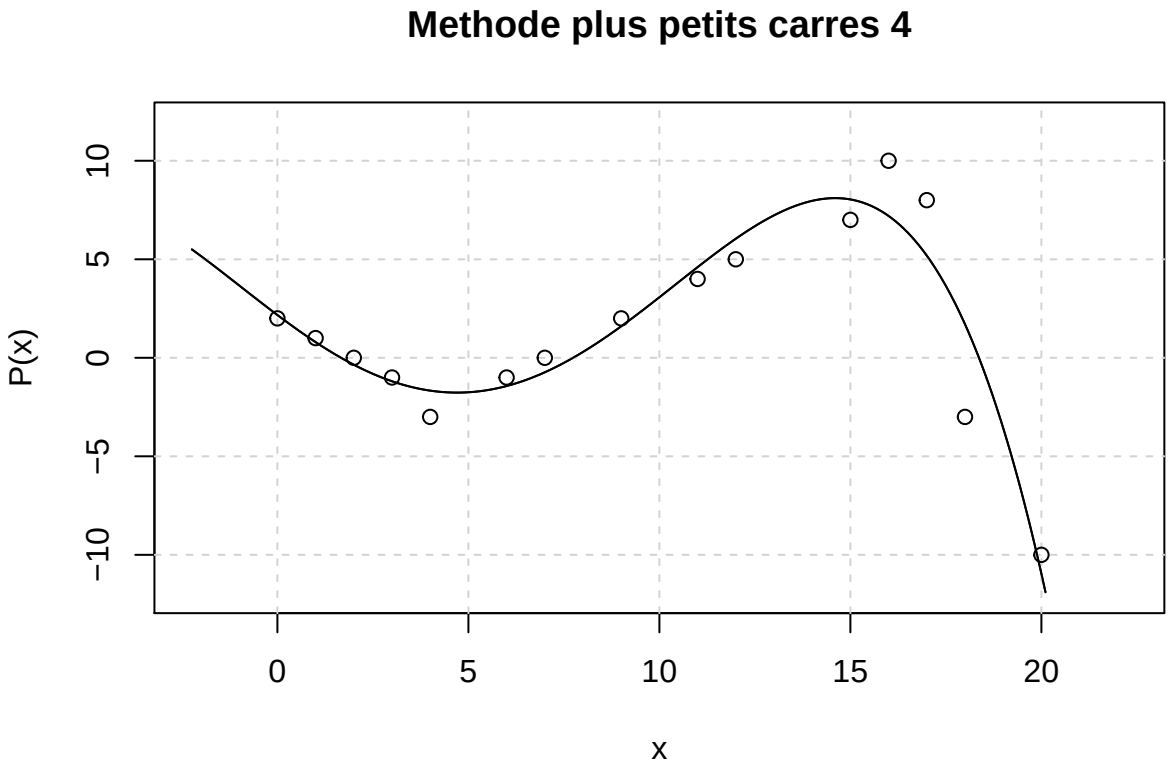
```
#On affiche le nuage de points
plot(delta$X,delta$Y)
```




```
#Un polynôme de degre 3 semble convenir
d<-4

res<-q3(d,delta)
p<-as.polynomial(res)
print(p)
```

```
## 2.171465892 - 1.4563374*x + 0.06257116153*x^2 + 0.01954282709*x^3 -
## 0.001033584341*x^4
```

```
plot(p, ylim = c(min(delta$Y) -2,max(delta$Y) +2), xlim = c(min(delta$X) -2,max(delta$X) +2), main = pas
points(delta$X, delta$Y)
```




```
for (xi in c(22,25,50)) {
  print(predict(p, xi))
}
```

```
## [1] -33.61482518
## [1] -93.51720294
## [1] -3931.266243
```

Exercice 4 : Palindromes 1. Créer une fonction qui identifie les mots ou phrases palindromiques (mots ou phrases qui se lisent dans les deux sens, par exemple RADAR).

Cette fonction prendra comme argument le mot "mot" et retournera les phrases : *"mot est un palindrome"* **si le mot = palindrome et** ***"mot n'est pas un palindrome"*****, sinon.**

```
library(stringr)

is_palindrome <- function(mot) {
  #supprime les espaces
  mot1 <- gsub(' ','',mot)
  #définition du mot inversé
  split_word <- unlist((str_split(mot1, pattern = "")))
  reverse_word <- split_word[str_length(mot1):1]
  paste_word <- paste(reverse_word, collapse = "")
  #comparaison du mot originel et de son inverse
  if (mot1 == paste_word){
    cat(mot, "\t: est un palindrome\n")
  }
```

```
  else{
    cat(mot, "\t: n'est pas un palindrome\n")
  }
}
```

2. Appliquer votre fonction sur les mots *"radar"*, *"bonne année"*, *"sept"*, *"kayak"*, *"la mariée ira mal"*, *"statistiques"*, *"engage le jeu que je le gagne"*, *"esope reste ici et se repose"*.

```
is_palindrome("radar")
```

```
## radar    : est un palindrome
```

```
is_palindrome("bonne année")
```

```
## bonne année  : n'est pas un palindrome
```

```
is_palindrome("sept")
```

```
## sept     : n'est pas un palindrome
```

```
is_palindrome("kayak")
```

```
## kayak    : est un palindrome
```

```
is_palindrome("la mariée ira mal")
```

```
## la mariée ira mal    : n'est pas un palindrome
```

```
is_palindrome("statistiques")
```

```
## statistiques     : n'est pas un palindrome
```

```
is_palindrome("engage le jeu que je le gagne")
```

```
## engage le jeu que je le gagne    : est un palindrome
```

```
is_palindrome("esope reste ici et se repose")
```

```
## esope reste ici et se repose     : est un palindrome
```

3. Créer une fonction qui retourne tous les mots palindromiques d'au plus 9 lettres dans un dictionnaire.

```
palindrome9 <- function(liste){
  for (i in liste){
    if(str_length(i) < 10){
      mot <- gsub(' ','',i)
      split_word <- unlist((str_split(mot, pattern = "")))
      reverse_word <- split_word[str_length(mot):1]
      paste_word <- paste(reverse_word, collapse = "")
      if (mot == paste_word){
        message(i)
      }
    }
  }
}

liste <- list("radar","bonne année","sept")
palindrome9(liste)
```

```
## radar
```

Exercice 5 : ACP

A. Analyse rapide 1. Récupérer les données du fichier "decathlon" et donner la matrice corrélation des variables quantitatives (ne pas prendre COMPET)

```
library(FactoMineR)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:limSolve':
##
##     resolution
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(corrplot)
```

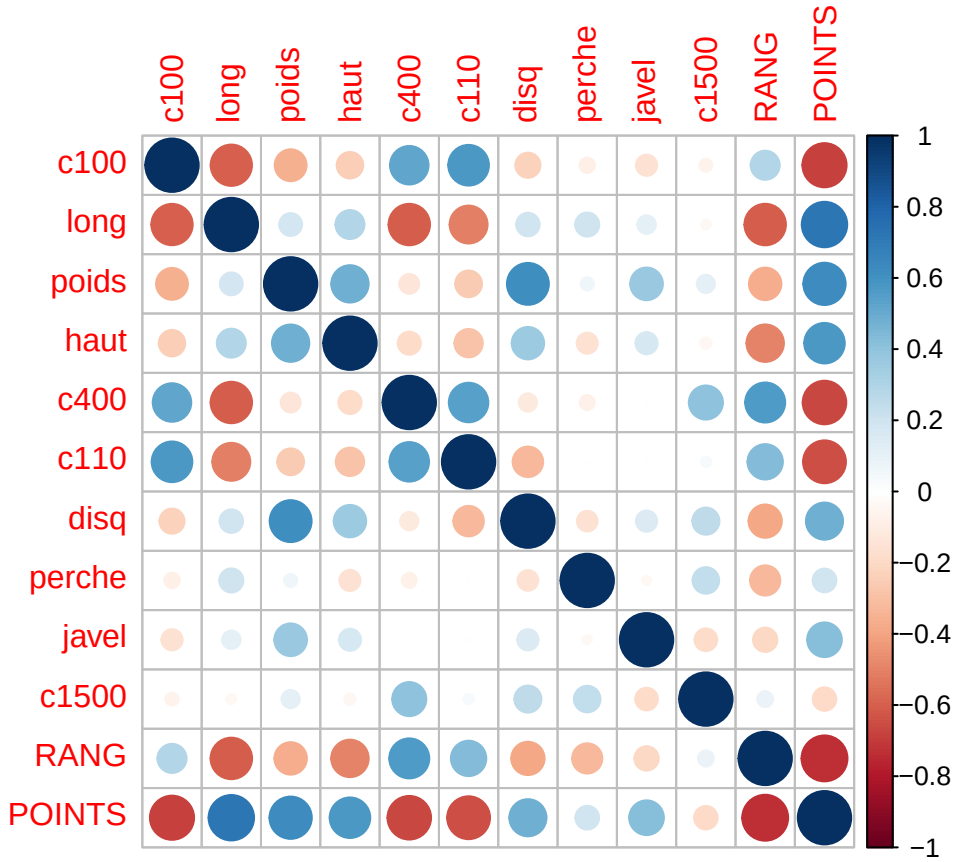
```
## corrplot 0.84 loaded
```

```
deca<-read.table("decathlon.dat")

print(deca)
```

```
##              c100 long poids haut  c400  c110  disq perche javel  c1500 RANG
## SEBRLE      11.04 7.58 14.83 2.07 49.81 14.69 43.75   5.02 63.19 291.70    1
## CLAY        10.76 7.40 14.26 1.86 49.37 14.05 50.72   4.92 60.15 301.50    2
## KARPOV      11.02 7.30 14.77 2.04 48.37 14.09 48.95   4.92 50.31 300.20    3
## BERNARD     11.02 7.23 14.25 1.92 48.93 14.99 40.87   5.32 62.77 280.10    4
## YURKOV      11.34 7.09 15.19 2.10 50.42 15.31 46.26   4.72 63.44 276.40    5
## WARNERS     11.11 7.60 14.31 1.98 48.68 14.23 41.10   4.92 51.77 278.10    6
## ZSIVOCZKY   11.13 7.30 13.48 2.01 48.62 14.17 45.67   4.42 55.37 268.00    7
## McMULLEN    10.83 7.31 13.76 2.13 49.91 14.38 44.41   4.42 56.37 285.10    8
## MARTINEAU   11.64 6.81 14.57 1.95 50.14 14.93 47.60   4.92 52.33 262.10    9
## HERNU       11.37 7.56 14.41 1.86 51.10 15.06 44.99   4.82 57.19 285.10   10
## BARRAS      11.33 6.97 14.09 1.95 49.48 14.48 42.10   4.72 55.40 282.00   11
## NOOL        11.33 7.27 12.68 1.98 49.20 15.29 37.92   4.62 57.44 266.60   12
## BOURGUIGNON 11.36 6.80 13.46 1.86 51.16 15.67 40.49   5.02 54.68 291.70   13
## Sebrle      10.85 7.84 16.36 2.12 48.36 14.05 48.72   5.00 70.52 280.01    1
## Clay        10.44 7.96 15.23 2.06 49.19 14.13 50.11   4.90 69.71 282.00    2
## Karpov      10.50 7.81 15.93 2.09 46.81 13.97 51.65   4.60 55.54 278.11    3
## Macey       10.89 7.47 15.73 2.15 48.97 14.56 48.34   4.40 58.46 265.42    4
## Warners     10.62 7.74 14.48 1.97 47.97 14.01 43.73   4.90 55.39 278.05    5
## Zsivoczky   10.91 7.14 15.31 2.12 49.40 14.95 45.62   4.70 63.45 269.54    6
## Hernu       10.97 7.19 14.65 2.03 48.73 14.25 44.72   4.80 57.76 264.35    7
## Nool        10.80 7.53 14.26 1.88 48.81 14.80 42.05   5.40 61.33 276.33    8
## Bernard     10.69 7.48 14.80 2.12 49.13 14.17 44.75   4.40 55.27 276.31    9
## Schwarzl    10.98 7.49 14.01 1.94 49.76 14.25 42.43   5.10 56.32 273.56   10
## Pogorelov   10.95 7.31 15.10 2.06 50.79 14.21 44.60   5.00 53.45 287.63   11
## Schoenbeck  10.90 7.30 14.77 1.88 50.30 14.34 44.41   5.00 60.89 278.82   12
## Barras      11.14 6.99 14.91 1.94 49.41 14.37 44.83   4.60 64.55 267.09   13
## Smith       10.85 6.81 15.24 1.91 49.27 14.01 49.02   4.20 61.52 272.74   14
## Averyanov   10.55 7.34 14.44 1.94 49.72 14.39 39.88   4.80 54.51 271.02   15
## Ojaniemi    10.68 7.50 14.97 1.94 49.12 15.01 40.35   4.60 59.26 275.71   16
```

```
## Smirnov     10.89 7.07 13.88 1.94 49.11 14.77 42.47   4.70 60.88 263.31   17
## Qi          11.06 7.34 13.55 1.97 49.65 14.78 45.13   4.50 60.79 272.63   18
## Drews       10.87 7.38 13.07 1.88 48.51 14.01 40.11   5.00 51.53 274.21   19
## Parkhomenko 11.14 6.61 15.69 2.03 51.04 14.88 41.90   4.80 65.82 277.94   20
## Terek       10.92 6.94 15.15 1.94 49.56 15.12 45.62   5.30 50.62 290.36   21
## Gomez       11.08 7.26 14.57 1.85 48.61 14.41 40.95   4.40 60.71 269.70   22
## Turi        11.08 6.91 13.62 2.03 51.67 14.26 39.83   4.80 59.34 290.01   23
## Lorenzo     11.10 7.03 13.22 1.85 49.34 15.38 40.22   4.50 58.36 263.08   24
## Karlivans   11.33 7.26 13.30 1.97 50.54 14.98 43.34   4.50 52.92 278.67   25
## Korkizoglou 10.86 7.07 14.81 1.94 51.16 14.96 46.07   4.70 53.05 317.00   26
## Uldal       11.23 6.99 13.53 1.85 50.95 15.09 43.01   4.50 60.00 281.70   27
## Casarsa     11.36 6.68 14.92 1.94 53.20 15.39 48.66   4.40 58.62 296.12   28
##             POINTS   COMPET
## SEBRLE        8217 Decastar
## CLAY          8122 Decastar
## KARPOV        8099 Decastar
## BERNARD       8067 Decastar
## YURKOV        8036 Decastar
## WARNERS       8030 Decastar
## ZSIVOCZKY     8004 Decastar
## McMULLEN      7995 Decastar
## MARTINEAU     7802 Decastar
## HERNU         7733 Decastar
## BARRAS        7708 Decastar
## NOOL          7651 Decastar
## BOURGUIGNON   7313 Decastar
## Sebrle        8893 OlympicG
## Clay          8820 OlympicG
## Karpov        8725 OlympicG
## Macey         8414 OlympicG
## Warners       8343 OlympicG
## Zsivoczky     8287 OlympicG
## Hernu         8237 OlympicG
## Nool          8235 OlympicG
## Bernard       8225 OlympicG
## Schwarzl      8102 OlympicG
## Pogorelov     8084 OlympicG
## Schoenbeck    8077 OlympicG
## Barras        8067 OlympicG
## Smith         8023 OlympicG
## Averyanov     8021 OlympicG
## Ojaniemi      8006 OlympicG
## Smirnov       7993 OlympicG
## Qi            7934 OlympicG
## Drews         7926 OlympicG
## Parkhomenko   7918 OlympicG
## Terek         7893 OlympicG
## Gomez         7865 OlympicG
## Turi          7708 OlympicG
## Lorenzo       7592 OlympicG
## Karlivans     7583 OlympicG
## Korkizoglou   7573 OlympicG
## Uldal         7495 OlympicG
## Casarsa       7404 OlympicG
```

```
corrplot(cor(deca[ , !(names(deca) %in% c("COMPET"))]))
```



2. Quelles sont les couples de variables les plus corrélées, les moins corrélées, les plus opposées ? Justifier.

Les couples de variables les plus corrélés sont :

- POINTS –> LONGEUR
- POINTS –> POIDS
- C110 –> C100
- C400 –> C100
- Disque –> Poids
- C110 –> C400
- Rang –> C400

On voit que les épreuves de lancer et les épreuves de courses sont corrélées entre elles

Les moins corrélées sont :

- C1500 –> C100
- C1500 –> Long
- Perche –> Poids
- C1500 –> Poids
- C1500 –> Haut
- Perche –> C400
- Javel –> C400
- Perche –> C110
- Javel –> C110

- C1500 –> C110
- Rang –> C1500

Les moins corrélées sont les épreuves les plus différentes.

Les plus opposées :

- C100 –> long
- C100 –> POINTS
- long –> C400
- long –> Rang
- Point –> C400
- Point –> C110
- Point –> Rang

3. Comment se groupent les variables du point de vue des signes de corrélation ? Expliquez pourquoi.

- Les épreuves concernant des sports similaires montrent une corrélation positive (course et lancer).
- Une corrélation négative apparait entre le rang et le score, cela s'explique, car il faut avoir un score élevé pour avoir un rang faible.

B. ACP : dans cette partie, vous allez procédez à une analyse en composantes principales des performances centrées-réduites, en excluant les variables RANG, POINTS et COMPET. 4. Donner les valeurs propres de la matrice de corrélation. Trier ces valeurs propres et donner le nombre de vecteurs propres qui expliquent le plus l'inertie du nuage des individus. Quelle règle peut-on utiliser ? Donner le pourcentage d'inertie totale en conservant les trois premiers vecteurs propres.

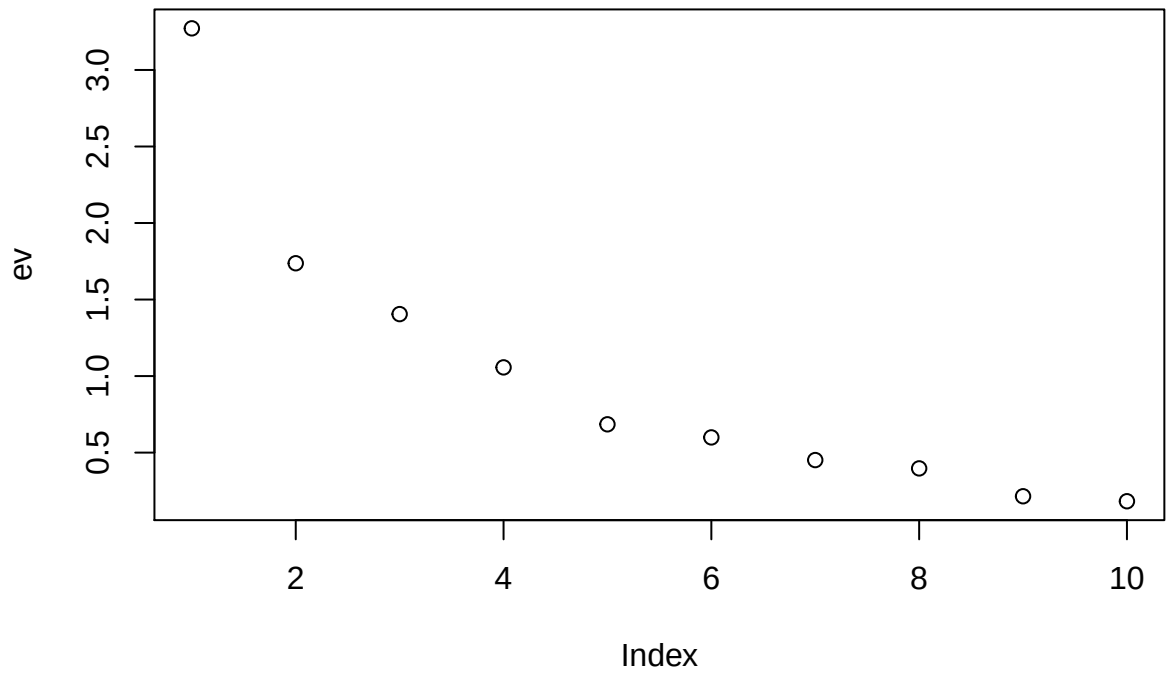
```
deca<-deca[ , !(names(deca) %in% c("COMPET","RANG","POINTS"))]
correlation<-cor(deca)
ev<-eigen(correlation)$values
ev
```

```
##  [1] 3.2719055380 1.7371310231 1.4049166821 1.0568503533 0.6847735349
##  [6] 0.5992686808 0.4512352638 0.3968765857 0.2148148532 0.1822274851
```

```
sort(ev, decreasing = T)
```

```
##  [1] 3.2719055380 1.7371310231 1.4049166821 1.0568503533 0.6847735349
##  [6] 0.5992686808 0.4512352638 0.3968765857 0.2148148532 0.1822274851
```

```
plot(ev)
```



On utilise la règle du coude, la cassure apparait pour la cinquième valeur propre.

```
sum((sort(ev,decreasing = T) / sum(ev))[1:3])
```

```
## [1] 0.6413953243
```

64% de l'inertie est expliquée par les trois premières valeurs propres.

5. Déterminer les trois composantes principales (projection des individus sur les trois vecteurs propres), que l'on note $C1$, $C2$, $C3$ dans l'ordre décroisant d'inertie.

```
res.pca <- PCA(deca, graph = F)
#Q5
#Composantes principales
ind <- get_pca_ind(res.pca)
ind$coord[,1:3]
```

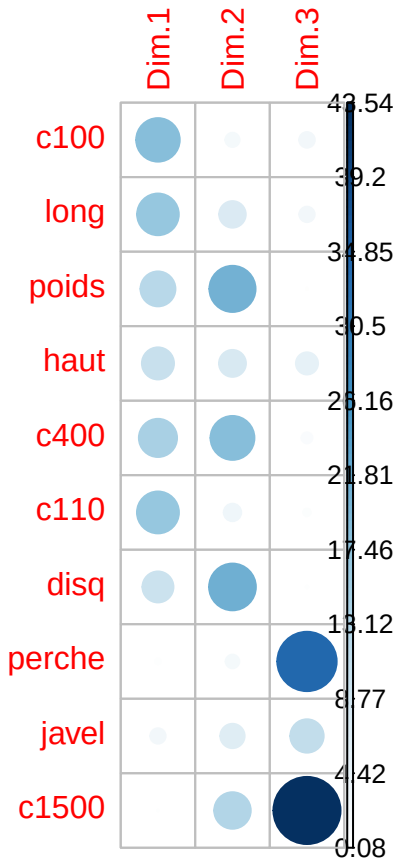
```
##                    Dim.1          Dim.2           Dim.3
## SEBRLE       0.791627716890  0.77161119552  0.8268411940268
## CLAY         1.234990562922  0.57457806534  2.1412469663713
## KARPOV       1.358214935758  0.48402090113  1.9562579868954
## BERNARD     -0.609515083106 -0.87462852884  0.8899406618588
## YURKOV      -0.585968337761  2.13095422255 -1.2251567968084
## WARNERS      0.356889530489 -1.68495666583  0.7665531449198
## ZSIVOCZKY    0.271774781024 -1.09377557750 -1.2827673831291
## McMULLEN     0.587516189056  0.23072991176 -0.4176329823038
## MARTINEAU   -1.995359298025  0.56099598158 -0.7299466010890
## HERNU       -1.546076461677  0.48838301094  0.8407858519275
## BARRAS      -1.341652726752 -0.31091157069 -0.0003683375477
```

```
## NOOL        -2.344973805586 -1.96637500055 -1.3364815492290
## BOURGUIGNON -3.979041864579  0.19986018993  1.3264851034113
## Sebrle       4.038448501441  1.36582606354 -0.2899565042806
## Clay         3.919365157228  0.83696136260  0.2311753204792
## Karpov       4.619987275045  0.03999522890 -0.0415857980014
## Macey        2.233460565598  1.04176620064 -1.8643620154049
## Warners      2.168396445406 -1.80320025033  0.8510173287098
## Zsivoczky    0.925132182894  1.16865179610 -1.4774802908286
## Hernu        0.889037851513 -0.61842521554 -0.8982953479746
## Nool         0.295305666684 -1.54561667242  1.3552601285624
## Bernard      1.906334367677 -0.08580429180 -0.7571859708851
## Schwarzl     0.081078659392 -1.35345709932  0.8224866222304
## Pogorelov    0.539677027745  0.77075098970  1.3476197769273
## Schoenbeck   0.114430984607 -0.03985060809  0.7404039810320
## Barras       0.002145202768  0.36033768481 -1.5696934887659
## Smith        0.870310569720  1.05932551998 -1.6434290616483
## Averyanov    0.349155137968 -1.55864999153  0.2825354036679
## Ojaniemi     0.380113998692 -0.77244734296 -0.3709431418934
## Smirnov     -0.484514212539 -1.06066118077 -1.2283378499303
## Qi          -0.434466690806 -0.32614689717 -1.0697978122896
## Drews       -0.248684024375 -3.08167683010  1.0548427374522
## Parkhomenko -1.069429104277  2.09318217909 -0.9999839028901
## Terek       -0.681953059148  0.53561439799  2.2091259997098
## Gomez       -0.289889207723 -1.19671610589 -1.3061025895306
## Turi        -1.541813055585  0.42716772525  0.5140859441357
## Lorenzo     -2.408509979550 -1.58292969328 -1.5023461069170
## Karlivans   -1.994368726831 -0.29418239625 -0.3427836936915
## Korkizoglou -0.957829813261  2.06638553650  2.5865525262672
## Uldal       -2.562259590728  0.24546870508 -0.4191406444668
## Casarsa     -2.857088268209  3.79784504993  0.0305611909207
```

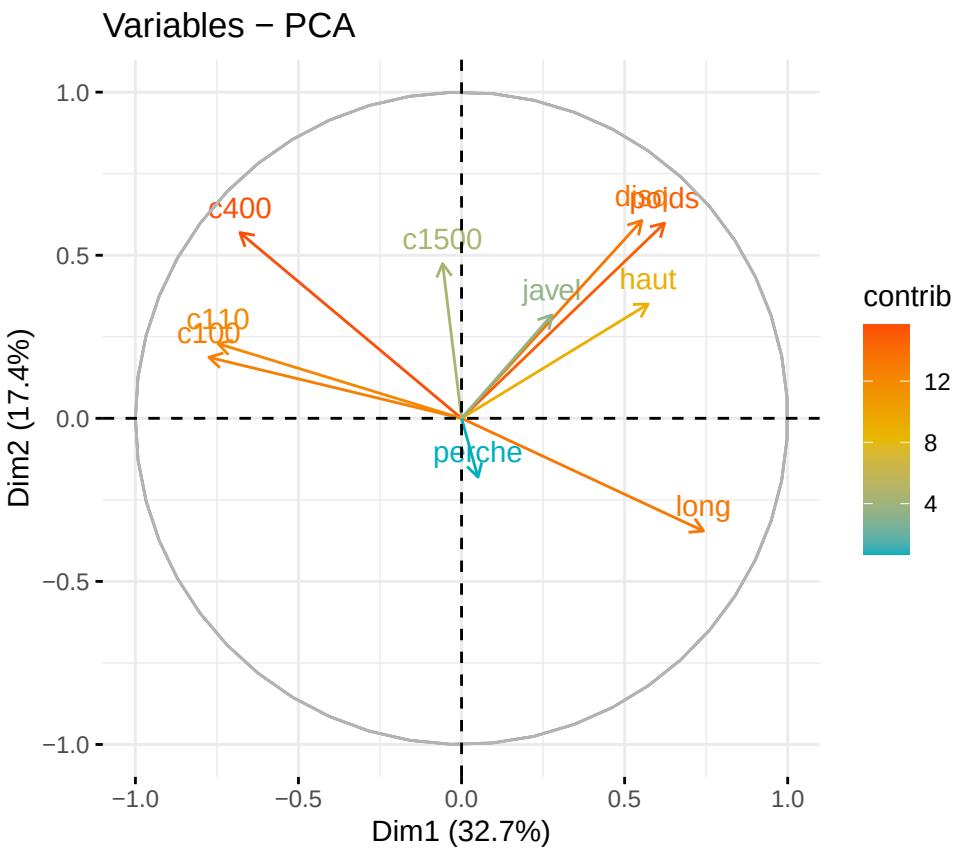
```
C1<-ind$coord[,1]
C2<-ind$coord[,2]
C3<-ind$coord[,3]
```

6. Déterminer le tableau des corrélations des variables par rapport à $C1, C2, C3$ et donner les deux cercles de corrélation des variables par rapport à $(C1, C2)$ et $(C2, C3)$

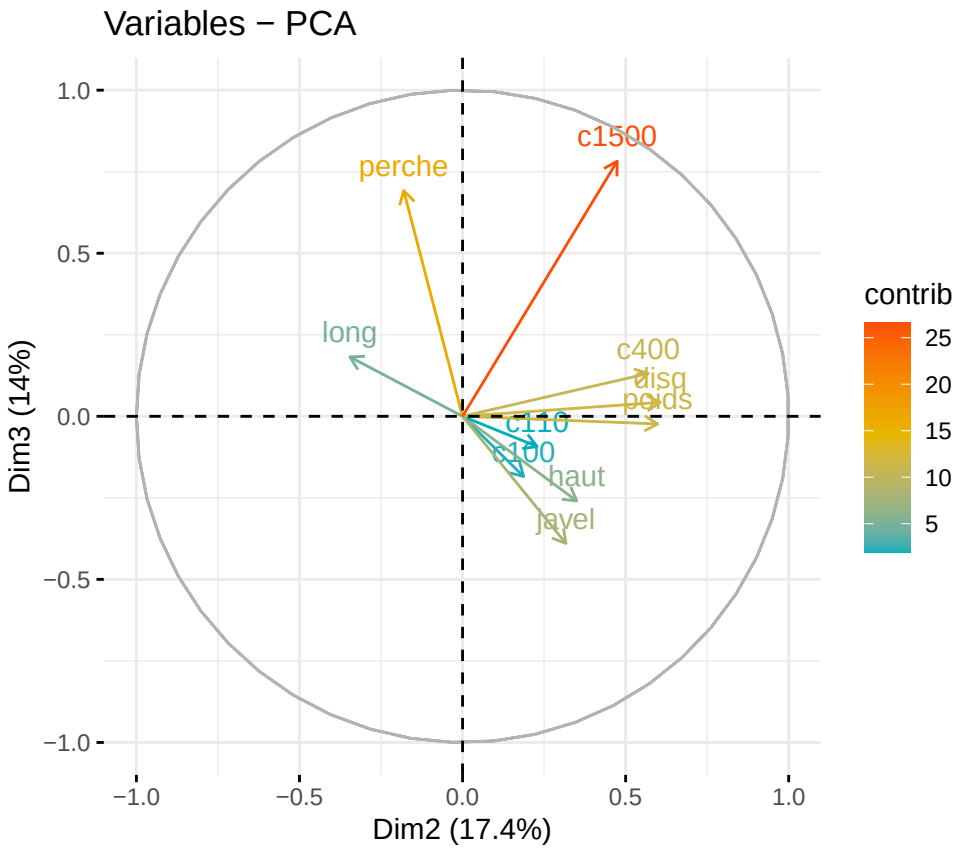
```
var <- get_pca_var(res.pca)
corrplot(var$contrib[,c(1:3)], is.corr=FALSE)
```




```
fviz_pca_var(res.pca, col.var = "contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07") , axes = c
```




```
fviz_pca_var(res.pca, col.var = "contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07") , axes = c
```



7. Quelles sont les variables qui déterminent les 3 composantes principales ? Proposez un seuil.

```
res.desc <- dimdesc(res.pca, axes = c(1,2,3), proba = 0.05)
res.desc$Dim.1$quanti
```

```
##         correlation         p.value
## long   0.7418997450 2.849885834e-08
## poids  0.6225025511 1.388320670e-05
## haut   0.5719452960 9.362284801e-05
## disq   0.5524665193 1.802219952e-04
## c400  -0.6796099427 1.028174558e-06
## c110  -0.7462453240 2.136961517e-08
## c100  -0.7747198283 2.778466580e-09
```

```
res.desc$Dim.2$quanti
```

```
##         correlation         p.value
## disq   0.6063133911 2.650744528e-05
## poids  0.5983033207 3.603567348e-05
## c400   0.5694377766 1.020941249e-04
## c1500  0.4742237687 1.734405284e-03
## haut   0.3502936078 2.475025040e-02
## javel  0.3169890605 4.344974126e-02
## long  -0.3454212938 2.696968984e-02
```

```
res.desc$Dim.3$quanti
```

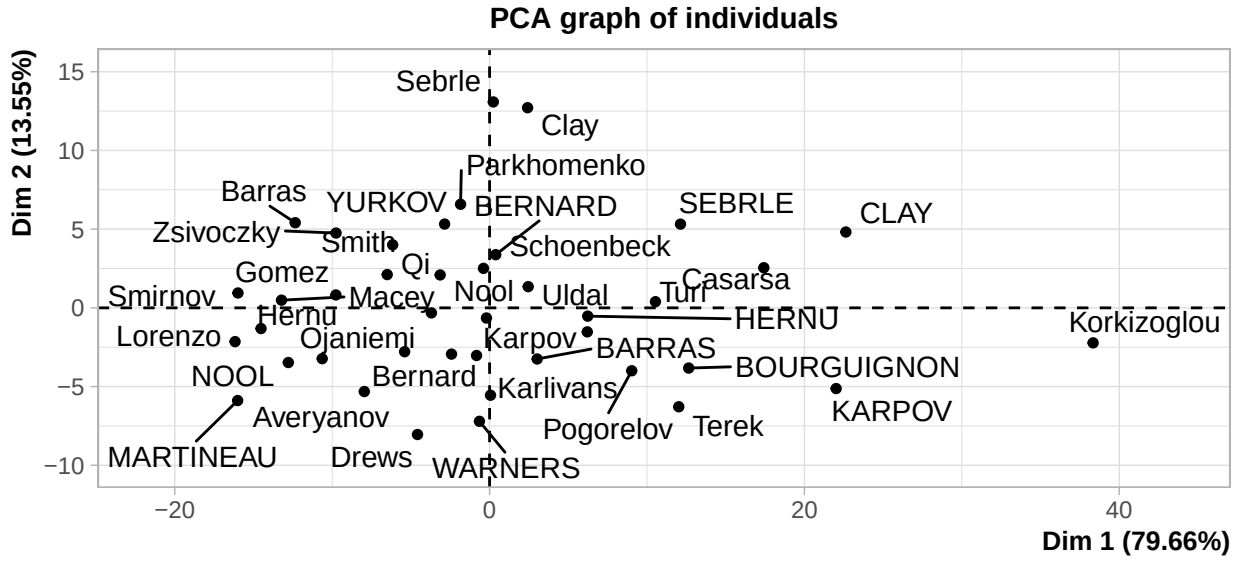
```
##          correlation         p.value
## c1500   0.7821428011 1.554449692e-09
## perche  0.6917566549 5.480171723e-07
## javel  -0.3896554074 1.179330939e-02
```

Nous proposons un seuil de 33% de la valeur absolue de la corrélation.

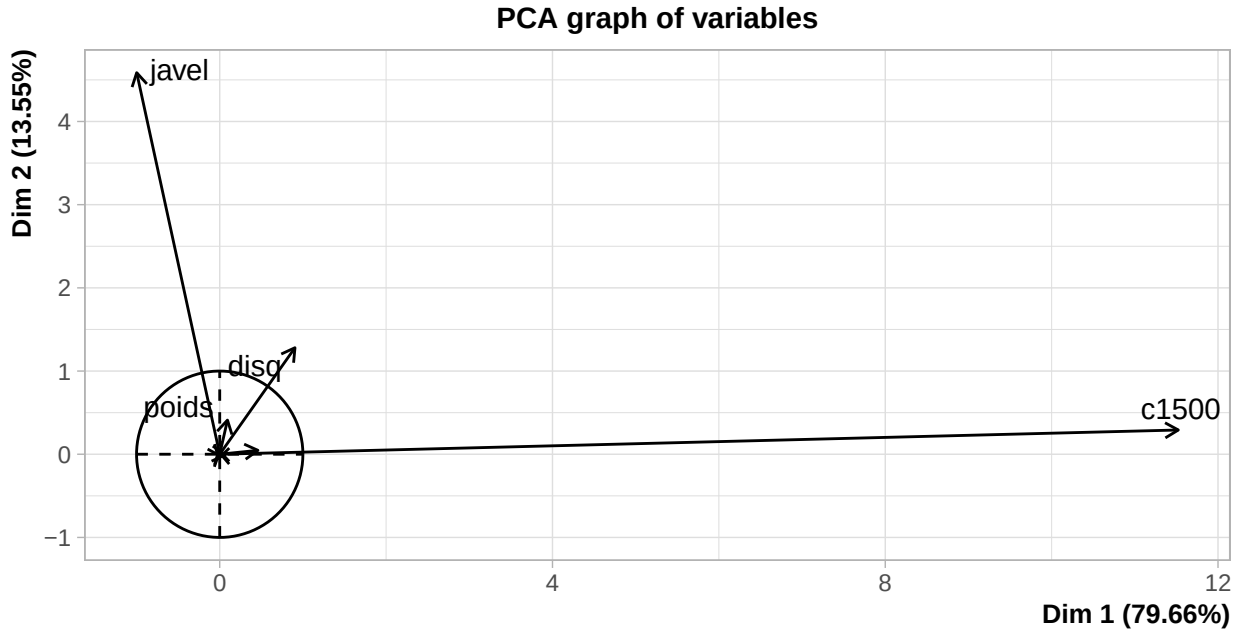
8. Expliquez comment les données peuvent être modifiées pour faire apparaître un effet de taille. Comment peut-on alors interpréter les axes principaux de la question 5 ?

```
taille.pca <- PCA(deca, graph = T, scale.unit = F)
```

```
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
## Warning: ggrepel: 6 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
taille.desc <- dimdesc(taille.pca, axes = c(1,2), proba = 0.05)
taille.desc$Dim.1$quanti
```

```
##        correlation         p.value
## c1500 0.9993506388 6.526966095e-58
## c400  0.4032299240 8.945953729e-03
```

```
taille.desc$Dim.2$quanti
```

```
##        correlation         p.value
## javel 0.9619837766 1.369499988e-23
## poids 0.5038617740 7.835507265e-04
## disq  0.3837681440 1.324971777e-02
```

On peut modifier les données en ne centrant pas et en ne les réduisant pas. Un effet de taille apparait alors. L'axe principale est expliqué "uniquement" par le 1500 mètres. Le deuxième axe principal par le javelot.