NIPALS

Cristian Preda, Vincent Vandewalle

2/5/2019

Simulation des données

```
library(MASS)
set.seed(1234)
mu=c(1,2,4,3)
n=100
p=4
sigma=matrix(c(0.7,0,1.3,0.5,
               0,1.2,-0.3,-0.1,
               1.3,-0.3,3.1,1.3,
               0.5,-0.1,1.3,0.6),
             nrow=p, ncol=p)
X=mvrnorm(n,mu,sigma)
cat("Quelques observations des données simulées:")
```

```
Quelques observations des données simulées:
```

```
head(X)
```

```
          [,1]      [,2]     [,3]     [,4]
[1,] 2.0957051 2.1947399 6.153399 3.736707
[2,] 0.8964736 1.5269139 3.523403 2.520930
[3,] 0.2470616 2.2806159 2.099409 2.143699
[4,] 2.9775327 0.9590023 7.978456 4.696517
[5,] 0.6589963 1.1838370 3.190021 2.593942
[6,] 0.8601072 2.2579124 3.120676 2.401118
```

On vérfie que les données sont "bien" simulées:

```
cat("les moyennes :", round(apply(X,2,"mean"),2))
```

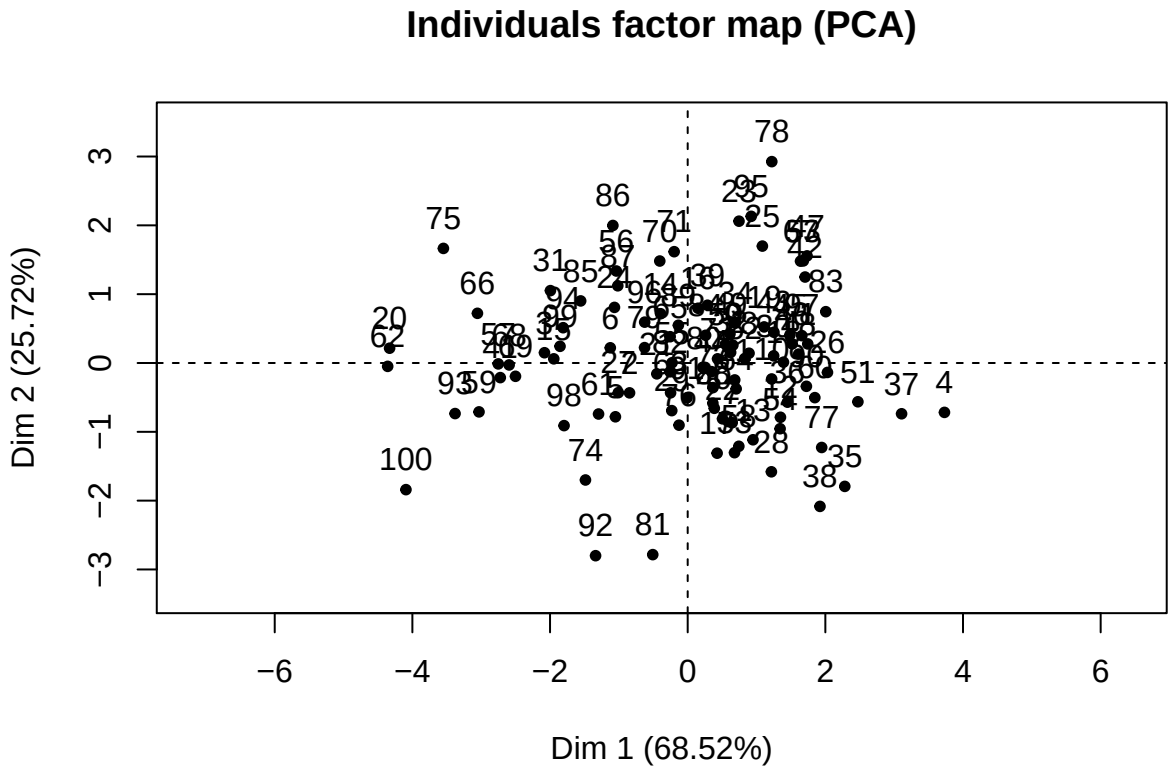
```
les moyennes : 1.17 2.01 4.27 3.09
```

```
print(round(cov(X),2))
```

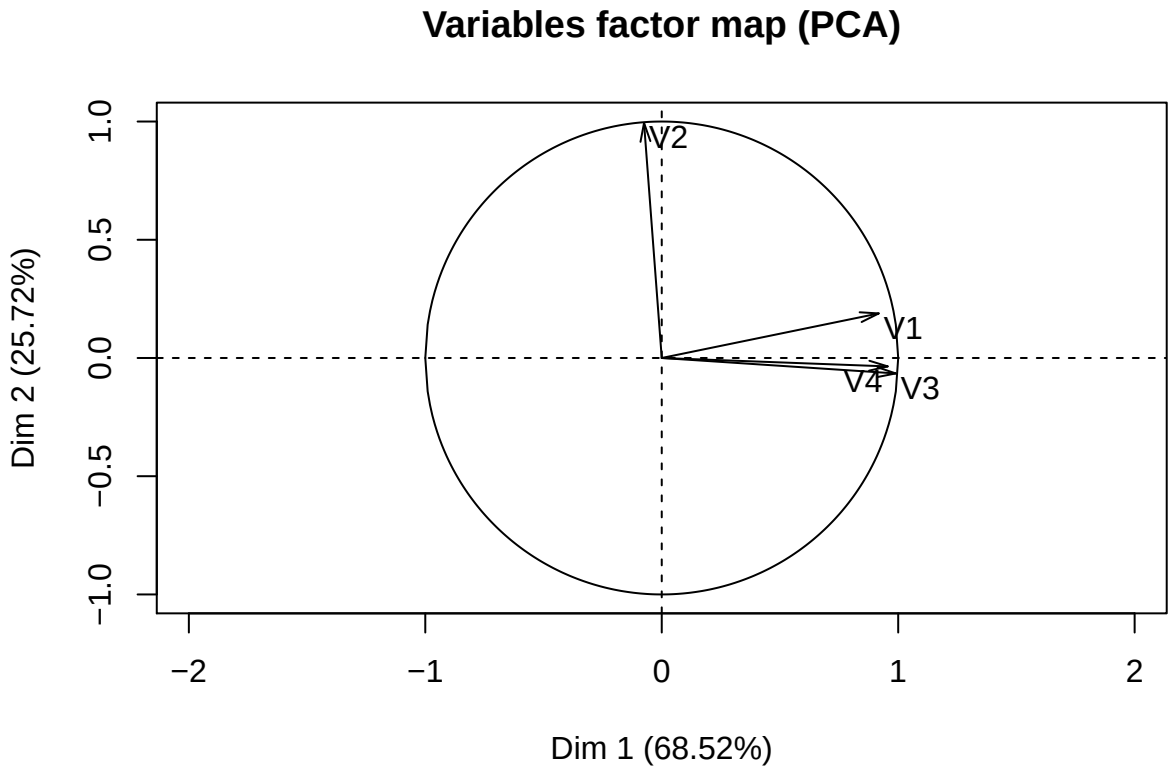
```
     [,1]  [,2]  [,3]  [,4]
[1,] 0.67  0.08  1.27  0.49
[2,] 0.08  1.25 -0.27 -0.07
[3,] 1.27 -0.27  3.19  1.32
[4,] 0.49 -0.07  1.32  0.60
```

ACP normée sur les données simulés.

```
library(FactoMineR)
acp=PCA(X, scale.unit=TRUE, graph=FALSE)
#graphe des individus
plot(acp,choix="ind", axes=c(1,2))
```




```
#plan des variables
plot(acp,choix="var", axes=c(1,2))
```




```
#composantes principales : les 6 premiers individus
print(head(acp$ind$coord))
```

```
      Dim.1      Dim.2        Dim.3       Dim.4
1  1.744297  0.2774305 -0.180939655  0.07417452
2 -0.844467 -0.4340644 -0.317415690  0.12253830
3 -2.080672  0.1491917  0.007667117  0.04016068
4  3.730728 -0.7178369 -0.305401251 -0.19033768
5 -1.050154 -0.7815551 -0.113669128 -0.01946425
6 -1.124327  0.2208457 -0.292807030  0.09684624
```

```
#facteurs principaux
print(acp$svd$V)
```

```
            [,1]        [,2]       [,3]        [,4]
[1,]  0.55440697  0.18578678 -0.7590821 -0.28620007
[2,] -0.04520317  0.97985412  0.1740390  0.08690798
[3,]  0.59771542 -0.06432375  0.1231065  0.78958439
[4,]  0.57734380 -0.03509465  0.6151012 -0.53581061
```

```
#valeurs propres
print(acp$eig)
```

```
       eigenvalue percentage of variance cumulative percentage of variance
comp 1 2.74085999             68.5214999                           68.52150
comp 2 1.02895575             25.7238937                           94.24539
comp 3 0.20873995              5.2184987                           99.46389
comp 4 0.02144431              0.5361078                          100.00000
```

NIPALS sans traitement des données manquantes

```
NIPALS=function(X,h=2,iter=100)
  {
  #renvoie les composantes principales (CP), les facteurs principaux (FP) et les données reconstituées
  n = nrow(X)
  p=ncol(X)
  #centrer et réduire matrice X
  m = apply(X,2,mean) #calcul des moyennes
  s = apply(X,2,sd)*sqrt((n-1)/n) #calcul des écart-types
  Xr = (X-rep(1,n)%*%t(m))/(rep(1,n)%*%t(s))

  #on reserve la place pour:
  CP=matrix(0,nrow=n,ncol=h)    # les composantes principales
  FP=matrix(0,ncol=h, nrow=p)   # les facteurs principaux
  Xrec=matrix(0,nrow=n,ncol=p)  # les données reconstituées

  #deroulement de l'algorithme:
  for (i in 1:h)
    {
    #voir pages 30-32 du cours
    r=calcul_cp_fp(Xr,iter) # fonction qui calcule la 1ere comp. princ et 1er fact. principal
    CP[,i]=r$cp
    FP[,i]=r$fp
    Xr=Xr-(r$cp)%*%t(r$fp)
    }

# Reconstitution des données avec h composantes
```

```
   Xrec = CP%*%t(FP)
   Xrec = Xrec*(rep(1,n)%*%t(s)) + rep(1,n)%*%t(m)
  return (list(CP=CP, FP=FP, rec=Xrec))
}

# la fonction qui calcule CP_1 et FP_1
calcul_cp_fp=function(X,iter)
  {
  cp=X[,1]
  fp=rep(0,ncol(X))
  for(i in 1:iter)
    {
    fp = t(X)%*%cp
    #on normalize fp:
    fp=fp/sqrt(sum(fp^2))
    cp=X%*%fp
    }
  return (list(cp=cp, fp=fp))
}
```

Application de NIPALS aux données simulées

```
cat("Voici ce qu'on obtient avec h=4 composantes. A comparer avec ce qui est donné par FactoMineR dans l
```

```
Voici ce qu'on obtient avec h=4 composantes. A comparer avec ce qui est donné par FactoMineR dans l'obj
```

```
res = NIPALS(X,h=ncol(X))
cat("Les facteurs principaux:")
```

```
Les facteurs principaux:
```

```
print(res$FP)
```

```
            [,1]        [,2]       [,3]        [,4]
[1,]  0.55440697  0.18578678  0.7590821  0.28620007
[2,] -0.04520317  0.97985412 -0.1740390 -0.08690798
[3,]  0.59771542 -0.06432375 -0.1231065 -0.78958439
[4,]  0.57734380 -0.03509465 -0.6151012  0.53581061
```

```
print(acp$svd$V)
```

```
            [,1]        [,2]       [,3]        [,4]
[1,]  0.55440697  0.18578678 -0.7590821 -0.28620007
[2,] -0.04520317  0.97985412  0.1740390  0.08690798
[3,]  0.59771542 -0.06432375  0.1231065  0.78958439
[4,]  0.57734380 -0.03509465  0.6151012 -0.53581061
```

```
cat("Les composantes principales")
```

```
Les composantes principales
```

```
print(head(res$CP))
```

```
          [,1]       [,2]         [,3]        [,4]
[1,]  1.744297  0.2774305  0.180939655 -0.07417452
[2,] -0.844467 -0.4340644  0.317415690 -0.12253830
```

```
[3,] -2.080672  0.1491917 -0.007667117 -0.04016068
[4,]  3.730728 -0.7178369  0.305401251  0.19033768
[5,] -1.050154 -0.7815551  0.113669128  0.01946425
[6,] -1.124327  0.2208457  0.292807030 -0.09684624
```

```
print(head(acp$ind$coord))
```

```
      Dim.1      Dim.2        Dim.3       Dim.4
1  1.744297  0.2774305 -0.180939655  0.07417452
2 -0.844467 -0.4340644 -0.317415690  0.12253830
3 -2.080672  0.1491917  0.007667117  0.04016068
4  3.730728 -0.7178369 -0.305401251 -0.19033768
5 -1.050154 -0.7815551 -0.113669128 -0.01946425
6 -1.124327  0.2208457 -0.292807030  0.09684624
```

La reconstitution complete des données (toutes les composantes principles)

```
cat("La reconstitution des données avec toutes les composantes principales")
```

```
La reconstitution des données avec toutes les composantes principales
```

```
res = NIPALS(X,h=ncol(X))
print(head(res$rec))
```

```
          [,1]      [,2]     [,3]     [,4]
[1,] 2.0957051 2.1947399 6.153399 3.736707
[2,] 0.8964736 1.5269139 3.523403 2.520930
[3,] 0.2470616 2.2806159 2.099409 2.143699
[4,] 2.9775327 0.9590023 7.978456 4.696517
[5,] 0.6589963 1.1838370 3.190021 2.593942
[6,] 0.8601072 2.2579124 3.120676 2.401118
```

```
print(head(X))
```

```
          [,1]      [,2]     [,3]     [,4]
[1,] 2.0957051 2.1947399 6.153399 3.736707
[2,] 0.8964736 1.5269139 3.523403 2.520930
[3,] 0.2470616 2.2806159 2.099409 2.143699
[4,] 2.9775327 0.9590023 7.978456 4.696517
[5,] 0.6589963 1.1838370 3.190021 2.593942
[6,] 0.8601072 2.2579124 3.120676 2.401118
```

Approximation des données avec quelques composantes (ici h=2)

Voici la reconstitution des données avec juste deux composantes :

```
res = NIPALS(X,h=2)
print(head(res$rec))
```

```
          [,1]     [,2]     [,3]     [,4]
[1,] 2.0014656 2.222630 6.088915 3.852965
[2,] 0.7293914 1.576574 3.420921 2.721751
[3,] 0.2611132 2.275243 2.041389 2.156632
[4,] 2.7451794 1.036614 8.312288 4.762611
[5,] 0.5844492 1.207751 3.242192 2.639732
[6,] 0.7022176 2.305289 3.048855 2.579692
```

```
print(head(X))
```

```
          [,1]      [,2]     [,3]     [,4]
[1,] 2.0957051 2.1947399 6.153399 3.736707
[2,] 0.8964736 1.5269139 3.523403 2.520930
[3,] 0.2470616 2.2806159 2.099409 2.143699
[4,] 2.9775327 0.9590023 7.978456 4.696517
[5,] 0.6589963 1.1838370 3.190021 2.593942
[6,] 0.8601072 2.2579124 3.120676 2.401118
```

NIPALS avec données manquantes.

L'algorithme précédent est adapté aux cas où il y a données manquantes. Les points à modifier sont au niveau du :

- calcul des moyennes (m) et écart-types (s)
- calcul des composantes et facteurs dans la fonction calcul_cp_fp

On ré-écris donc ces fonctions en le renommant : *NIPALS_dm* et *calcul_cp_fp_dm*

```
NIPALS_dm=function(X,h=2,iter=100)
  {
  #renvoie les composantes principales (CP), les facteurs principaux (FP) et les données reconstituées
  n = nrow(X)
  p=ncol(X)
  #centrer et réduire matrice X
  #calcul des moyennes
  m = apply(X,2,function(x){return(mean(x, na.rm=TRUE))})
  #calcul des écart-types
  s = apply(X,2,function(x){return(sd(x, na.rm=TRUE))})*sqrt((n-1)/n)
  Xr = (X-rep(1,n)%*%t(m))/(rep(1,n)%*%t(s))

  #on reserve la place pour:
  CP=matrix(0,nrow=n,ncol=h)    # les composantes principales
  FP=matrix(0,ncol=h, nrow=p)   # les facteurs principaux
  Xrec=matrix(0,nrow=n,ncol=p)  # les données reconstituées

  #deroulement de l'algorithme:
  for (i in 1:h)
    {
    #voir pages 30-32 du cours
    r=calcul_cp_fp_dm(Xr,iter) # fonction qui calcule la 1ere comp. princ et 1er fact. principal
    CP[,i]=r$cp
    FP[,i]=r$fp
    Xr=Xr-(r$cp)%*%t(r$fp)
    }

# Reconstitution des données avec h composantes
   Xrec = CP%*%t(FP)
   Xrec = Xrec*(rep(1,n)%*%t(s)) + rep(1,n)%*%t(m)
  return (list(CP=CP, FP=FP, rec=Xrec))
}

# la fonction qui calcule CP_1 et FP_1 avec données manqunates
```

```
calcul_cp_fp_dm=function(X,iter)
  {
  cp=X[,1]
  fp=rep(0,ncol(X))
  for(i in 1:iter)
    {
    for(j in 1:ncol(X)) {fp[j] = sum(X[,j]*cp, na.rm=TRUE)}
    #on normalize fp
    fp=fp/sqrt(sum(fp^2))
    cp=apply(X, 1,function(x){return(sum(x*fp, na.rm=TRUE))}) #vectoriser le calcul
    }
  return (list(cp=cp, fp=fp))
}
```

On vérifie que la version modifiée *NIPALS_dm* donne les memes resultats que *NIPALS* lorsqu'il n'y a pas données manquantes.

```
res_dm = NIPALS_dm(X,h=ncol(X))
cat("Les facteurs principaux:")
```

```
Les facteurs principaux:
```

```
print(res_dm$FP)
```

```
            [,1]        [,2]       [,3]        [,4]
[1,]  0.55440697  0.18578678  0.7590821  0.28620007
[2,] -0.04520317  0.97985412 -0.1740390 -0.08690798
[3,]  0.59771542 -0.06432375 -0.1231065 -0.78958439
[4,]  0.57734380 -0.03509465 -0.6151012  0.53581061
```

```
cat("Les composantes principales")
```

```
Les composantes principales
```

```
print(head(res_dm$CP))
```

```
          [,1]       [,2]         [,3]        [,4]
[1,]  1.744297  0.2774305  0.180939655 -0.07417452
[2,] -0.844467 -0.4340644  0.317415690 -0.12253830
[3,] -2.080672  0.1491917 -0.007667117 -0.04016068
[4,]  3.730728 -0.7178369  0.305401251  0.19033768
[5,] -1.050154 -0.7815551  0.113669128  0.01946425
[6,] -1.124327  0.2208457  0.292807030 -0.09684624
```

```
cat("Les données reconstituées")
```

```
Les données reconstituées
```

```
print(head(res_dm$rec))
```

```
          [,1]      [,2]     [,3]     [,4]
[1,] 2.0957051 2.1947399 6.153399 3.736707
[2,] 0.8964736 1.5269139 3.523403 2.520930
[3,] 0.2470616 2.2806159 2.099409 2.143699
[4,] 2.9775327 0.9590023 7.978456 4.696517
[5,] 0.6589963 1.1838370 3.190021 2.593942
[6,] 0.8601072 2.2579124 3.120676 2.401118
```

Parfait!

Simulation des données manqunates sur la matrice X

```
#pourcentage des données manqunates
pm = 0.1
#generation des valeurs manquantes
# on se rappelle n=100, p=4.
Xm = X
#indices des valeurs manquantes
im = which(runif(n*p) < pm)
Xm[which(runif(n*p) < pm)]=NA
summary(Xm)
```

```
       V1                V2                V3                V4
 Min.   :-1.0321   Min.   :-0.9992   Min.   :-0.6178   Min.   :1.014
 1st Qu.: 0.6748   1st Qu.: 1.3269   1st Qu.: 3.1978   1st Qu.:2.646
 Median : 1.4196   Median : 2.0012   Median : 4.6292   Median :3.203
 Mean   : 1.2012   Mean   : 2.0055   Mean   : 4.3014   Mean   :3.098
 3rd Qu.: 1.8524   3rd Qu.: 2.6112   3rd Qu.: 5.5697   3rd Qu.:3.601
 Max.   : 2.9775   Max.   : 5.1038   Max.   : 8.0416   Max.   :5.206
 NA's   :9         NA's   :7         NA's   :6         NA's   :15
```

```
cat("Voici les valeurs qui ont été déclarées manquantes :")
```

```
Voici les valeurs qui ont été déclarées manquantes :
```

```
print(X[im])
```

```
 [1]  1.5539468  1.1377092  1.9785295  2.1827555 -0.4140870  2.2881670
 [7]  0.8686446  1.3197100  1.4562753  2.2806159  2.2579124  2.5755878
[13]  4.0492694  2.2178541  2.1559718  3.5867321  1.5410720  1.5421984
[19]  0.4872301  2.3674007  4.2508310  2.0994089  4.7570332  6.1647522
[25]  5.4603127  5.3898212  6.5416000  2.7420120  3.8845037  2.2131857
[31]  2.5939416  3.6013689  3.7980385  3.2030856  3.6213698  3.5619919
[37]  3.5558357  1.4177402  2.9780713  3.1812048
```

Imputation des valeurs manquantes avec NIPALS :

```
imp_nip = NIPALS_dm(Xm) # par defaut, on utilise donc juste deux comp (h=2)
cat("Voici les valeurs estimées par NIPALS pour les données manqunates :")
```

```
Voici les valeurs estimées par NIPALS pour les données manqunates :
```

```
print(imp_nip$rec[im])
```

```
 [1] 1.4637529 1.1982610 1.8005454 1.9796091 0.6662561 2.1611444 1.0453665
 [8] 1.2423298 1.1760366 2.2785626 2.2958204 1.9057493 4.2088656 2.2298189
[15] 2.1475382 3.5006879 1.5436491 1.5859594 0.5457333 2.4251449 4.2519734
[22] 1.9605387 4.6940470 5.7290255 4.7467176 5.2908515 6.3774020 2.8457060
[29] 4.3009323 2.2079245 2.8043320 3.3722589 3.7793711 3.2258540 3.6957376
[36] 3.6133975 3.5092467 1.2719192 2.9821672 3.4503615
```

Cela a l'air pas mal!