Codes associated with Pool Testing Study

Arvind Raghavendran

August 2, 2021

Contents

1 Installing the packages

```
library(R1magic) #For Non-Linear penalized optimization
library(pROC) #Generating the ROC curves and AUC values
library(crossdes) #Generating the BIBDs
```

2 Declaring the functions

2.1 MakeBinary

MakeBinary(v) converts a vector **v** into a binary vector, i.e., zeroes remain as zeroes, and any non-zero value is returned as 1. Intuitively speaking, **x** was a disease score vector for the patients, then MakeBinary(**v**) only tells us which patient is positive/negative. This is the basis for our experimental design.

```
MakeBinary = function(v){
  n = length(v)
  for (i in 1:n){
    if (v[i]>0){
      v[i] = 1
    }
    else{
      v[i] = 0
    }
  }
  return(v)
}
```

2.2 VectorEstimate

VectorEstimate(M,y,T,p,lambda) is the function that actually determines the sparse vector. Given the test vector **y**, pool matrix M, basis T, an initial guess p and a penalty coefficient λ(lambda), we use the predefined function SolveL1 in the R1magic package [**?**] that does L1 non-linear penalized optimization to recover the sparse vector x that satisfies the system of equations $M\mathbf{x} = \mathbf{y}$. In reality, the patient sample vector is sparse since the positivity rate of COVID-19 is low, atmost 5%, so the assumption of a sparse **x** is justified.

```
VectorEstimate = function(M,y,T1,p,lambda=0.1){
  l1 = solveL1(M,y,T1,p,lambda)
  x_ = l1$estimate
  return(x_)
}
```

2.3 First1

First1(**x**,n) divides the vector **x** into n sub-vectors. It returns $i-1$, where i is the first subvector that contains a non-zero entry.

```
First_1 = function(x,n){
  c = which(x==1)[1]
  m = c %/% n
  m1 = c/n
  if (m1==m){
    return(max(m-1,0))
  }
  else{
    return(m)
  }
}
```

3 Defining the Matrices

3.1 Initial Matrices

Finding the design parameters

```
n = 10
while(n<=20){
  v = 3
  k = 2
  while(v<n){
    while(k<v){
      l = n*k*(k-1)/(v*(v-1))

      if (round(l,1)==l){
            n1 = l*(v-1)
            n2 = l*v*(v-1)
            r1 = n1%%(k-1)
            r2 = n2%%(k*(k-1))
            r = (l*(v-1))/(k-1)

            if(r1==0 & r2==0 & v<=8){
                cat("v=",v,"b=",n,"k=",k,"\n",sep=" ")
            }
        }
      k = k+1
    }
    k = 2
    v = v+1
  }
  n = n+1
}
```

Declaring the matrices

```
B101 = (isGYD(find.BIB(5,10,2),type=FALSE))$`Row incidence matrix of d`
B102 = (isGYD(find.BIB(5,10,3),type=FALSE))$`Row incidence matrix of d`
B103 = (isGYD(find.BIB(5,10,4),type=FALSE))$`Row incidence matrix of d`
B104 = (isGYD(find.BIB(6,10,3),type=FALSE))$`Row incidence matrix of d`
B121 = (isGYD(find.BIB(3,12,2),type=FALSE))$`Row incidence matrix of d`
B122 = (isGYD(find.BIB(4,12,2),type=FALSE))$`Row incidence matrix of d`
B123 = (isGYD(find.BIB(4,12,3),type=FALSE))$`Row incidence matrix of d`
B124 = (isGYD(find.BIB(6,12,3),type=FALSE))$`Row incidence matrix of d`
B125 = (isGYD(find.BIB(6,12,4),type=FALSE))$`Row incidence matrix of d`
B126 = (isGYD(find.BIB(6,12,5),type=FALSE))$`Row incidence matrix of d`
B141 = (isGYD(find.BIB(7,14,3),type=FALSE))$`Row incidence matrix of d`
B142 = (isGYD(find.BIB(7,14,4),type=FALSE))$`Row incidence matrix of d`
B143 = (isGYD(find.BIB(7,14,6),type=FALSE))$`Row incidence matrix of d`
B144 = (isGYD(find.BIB(8,14,4),type=FALSE))$`Row incidence matrix of d`
B151 = (isGYD(find.BIB(3,15,2),type=FALSE))$`Row incidence matrix of d`
B152 = (isGYD(find.BIB(5,15,2),type=FALSE))$`Row incidence matrix of d`
B153 = (isGYD(find.BIB(5,15,3),type=FALSE))$`Row incidence matrix of d`
B154 = (isGYD(find.BIB(5,15,4),type=FALSE))$`Row incidence matrix of d`
B155 = (isGYD(find.BIB(6,15,2),type=FALSE))$`Row incidence matrix of d`
B156 = (isGYD(find.BIB(6,15,4),type=FALSE))$`Row incidence matrix of d`
B161 = (isGYD(find.BIB(4,16,3),type=FALSE))$`Row incidence matrix of d`
B162 = (isGYD(find.BIB(8,16,7),type=FALSE))$`Row incidence matrix of d`
B181 = (isGYD(find.BIB(3,18,2),type=FALSE))$`Row incidence matrix of d`
B182 = (isGYD(find.BIB(4,18,2),type=FALSE))$`Row incidence matrix of d`
B183 = (isGYD(find.BIB(6,18,2),type=FALSE))$`Row incidence matrix of d`
B184 = (isGYD(find.BIB(6,18,3),type=FALSE))$`Row incidence matrix of d`
```

```
B185 = (isGYD(find.BIB(6,18,4),type=FALSE))$`Row incidence matrix of d`
B186 = (isGYD(find.BIB(6,18,5),type=FALSE))$`Row incidence matrix of d`
B201 = (isGYD(find.BIB(4,20,3),type=FALSE))$`Row incidence matrix of d`
B202 = (isGYD(find.BIB(5,20,2),type=FALSE))$`Row incidence matrix of d`
B203 = (isGYD(find.BIB(5,20,3),type=FALSE))$`Row incidence matrix of d`
B204 = (isGYD(find.BIB(5,20,4),type=FALSE))$`Row incidence matrix of d`
B205 = (isGYD(find.BIB(6,20,3),type=FALSE))$`Row incidence matrix of d`
```

3.2 With practical constraints

Finding the design parameters

```
n = 1
while(n<=100){
  v = 3
  k = 2
  while(v<n){
    while(k<v){
      l = n*k*(k-1)/(v*(v-1))

      if (round(l,1)==l){
            n1 = l*(v-1)
            n2 = l*v*(v-1)
            r1 = n1%%(k-1)
            r2 = n2%%(k*(k-1))
            r = (l*(v-1))/(k-1)

            if(r1==0 & r2==0 & v %in% c(4,8,16,32) & k<=4 & r<=7){
                cat("v=",v,"b=",n,"k=",k,"r=",r,sep=" ")
                print("\n")
            }
        }
      k = k+1
    }
    k = 2
    v = v+1
  }
  n = n+1
}
```

Declaring the matrices

```
B61 = (isGYD(find.BIB(4,6,2),type = FALSE))$`Row incidence matrix of d`
B81 = (isGYD(find.BIB(4,8,3),type = FALSE))$`Row incidence matrix of d`
B122 = (isGYD(find.BIB(4,12,2),type=FALSE))$`Row incidence matrix of d`
B144 = (isGYD(find.BIB(8,14,4),type=FALSE))$`Row incidence matrix of d`
B206 = (isGYD(find.BIB(16,20,4),type=FALSE))$`Row incidence matrix of d`
B241 = (isGYD(find.BIB(16,24,2),type=FALSE))$`Row incidence matrix of d`
B242 = (isGYD(find.BIB(16,24,4),type=FALSE))$`Row incidence matrix of d`
B281 = (isGYD(find.BIB(8,28,2),type=FALSE))$`Row incidence matrix of d`
B282 = (isGYD(find.BIB(16,28,4),type=FALSE))$`Row incidence matrix of d`
B321 = (isGYD(find.BIB(16,32,3),type=FALSE))$`Row incidence matrix of d`
B481 = (isGYD(find.BIB(16,48,2),type=FALSE))$`Row incidence matrix of d`
```

3.3 *Tapestry*

```
Create_Tapestry_M = function(){

  M = matrix(0,16,40)

  r1  = c(2,13,14,18,34,35,39)
  r2  = c(2,5,11,17,24,27,35)
  r3  = c(6,8,9,12,23,30,32)
  r4  = c(1,3,5,15,21,25,29,32)
  r5  = c(6,10,14,17,21,31,36)
  r6  = c(4,12,18,20,24,36,37)
  r7  = c(8,10,25,27,38,39,40)
  r8  = c(11,13,19,22,23,36)
  r9  = c(1,16,22,28,31,32,38)
  r10 = c(1,4,8,19,26,31,37)
  r11 = c(3,10,16,17,25,33,34,37,40)
  r12 = c(2,7,15,16,20,23,29)
  r13 = c(3,7,12,13,21,26,27,28)
  r14 = c(14,15,24,26,30,33,38)
  r15 = c(5,9,19,20,28,34,35,40)
  r16 = c(4,6,7,9,22,29,33,39)

  index_list = list(r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16)

  for (i in 1:16){
    for (j in index_list[i]){
      M[i,j] = 1
    }
  }

  return(M)
}
Tapestry = Create_Tapestry_M()
```

4 Generating the data

Parameters in code below are for display, and can be manipulated depending upon the parameters of the design.

4.1 Setting the parameters

```
b = 6
N = 10000*b
M = B61
pr = 0.01
```

4.2 Generating the ROC

```
x0 = matrix(0,N,1)
infected = N*pr
c = sample.int(N,infected)
x0[c] = 1
ind = First_1(x0,b)
x = x0[((b*ind)+1):(b*(ind+1))]
y = M %*% x
y = MakeBinary(y)
T1 = diag(b)
p = matrix(0,b,1)
```

```
auc_check_vec = c()
for (i in 0:1000){
  l = 7*i/1000
  x_ = VectorEstimate(M,y,T1,p,l)
  df = data.frame("Predictor"=x,"Responses"=x_)
  a = auc(df$Predictor,df$Responses,percent=TRUE,direction="<",levels=c(0,1))
  if (a>=85){
    auc_check_vec = append(auc_check_vec,l)
  }
}

#Checking AUC for the whole 100000 by using lambda as max of auc_check_vec WLOG
if (is.null(auc_check_vec)){
  print(50)
  stop("No lambda exists")
}
l = min(auc_check_vec)
x_01 = VectorEstimate(M,y,T1,p,l)
index_vec = 0:9999
index_vec = index_vec[! index_vec %in% ind]

for (i in index_vec){
  x = x0[((b*i)+1):(b*(i+1))]
  y = M %*% x
  y = MakeBinary(y)
  x_ = VectorEstimate(M,y,T1,p,l)
  x_01 = append(x_01,x_)
}

x_0 = c()
if (ind==0){
  x_0 = x_01
}else{
  x_0 = x_01[(b+1):(b*(ind+1))]
  x_0 = append(x_0,x_01[1:b])
  x_0 = append(x_0,x_01[(b*(ind+1)+1):N])
}

DF = data.frame("Predictor"=x0,"Responses"=x_0)
plot.roc(r,auc.polygon=TRUE,auc.polygon.col="red",
         max.auc.polygon=TRUE,max.auc.polygon.col="green",main="ROC",
         print.auc=TRUE,percent=TRUE,direction="<",levels=c(0,1))
```

4.3 95% CI around AUC and best sensitivity-specificity achieved by the curve

```
v = c()
vse = c()
vsp = c()
for (l in 1:50){
x0 = matrix(0,N,1)
infected = N*pr
c = sample.int(N,infected)
x0[c] = 1
ind = First_1(x0,b)
x = x0[((b*ind)+1):(b*(ind+1))]
y = M %*% x
y = MakeBinary(y)
T1 = diag(b)
```

```
p = matrix(0,b,1)

auc_check_vec = c()
for (i in 0:1000){
  l = 7*i/1000
  x_ = VectorEstimate(M,y,T1,p,l)
  df = data.frame("Predictor"=x,"Responses"=x_)
  a = auc(df$Predictor,df$Responses,percent=TRUE,direction="<",levels=c(0,1))
  if (a>=85){
    auc_check_vec = append(auc_check_vec,l)
  }
}

#Checking AUC for the whole 100000 by using lambda as max of auc_check_vec WLOG
if (is.null(auc_check_vec)){
  print("No lambda exists")
  v = append(v,50)
  vse = append(vse,50)
  vsp = append(vsp,50)
  next
}
l = min(auc_check_vec)
x_01 = VectorEstimate(M,y,T1,p,l)
index_vec = 0:9999
index_vec = index_vec[! index_vec %in% ind]

for (i in index_vec){
  x = x0[((b*i)+1):(b*(i+1))]
  y = M %*% x
  y = MakeBinary(y)
  x_ = VectorEstimate(M,y,T1,p,l)
  x_01 = append(x_01,x_)
}

x_0 = c()
if (ind==0){
  x_0 = x_01
}else{
  x_0 = x_01[(b+1):(b*(ind+1))]
  x_0 = append(x_0,x_01[1:b])
  x_0 = append(x_0,x_01[(b*(ind+1)+1):N])
}

DF = data.frame("Predictor"=x0,"Responses"=x_0)
r = roc(DF$Predictor,DF$Responses,percent=TRUE,direction="<",levels=c(0,1))
v = append(v,round(auc(r),5))
d = coords(r,x="local maximas")
gmvec=c()
for (i in 1:nrow(d)){
  gm = sqrt(d[i,"sensitivity"]*d[i,"specificity"])
  gmvec = append(gmvec,gm)
}
z = which.max(gmvec)
vse = append(vse,d[z,"sensitivity"])
vsp = append(vsp,d[z,"specificity"])
}
table(v)
mean(v)
c(mean(vse),mean(vsp))
```