Multinomial Predictive Belief Functions

Thierry Denoeux

2023-08-25

Preparation

We need the following packages:

```
library(DescTools)
library(evclust)
library(ibelief)
library(lpSolve)
```

Question 1

Let us consider the following probability vector:

```
p <- c(0.3,0.2,0.1,0.4)
```

We generate $q = 1000$ samples from a multinomial distribution with parameters $n = 100$ and p. (This simulates 1000 repetitions of an experiment that consists in drawing 100 balls with replacement from an urn containing balls of different colors, in proportions contained in vector p).

```
set.seed(20230822)
q<-1000
N<-rmultinom(q,100,p)
```

For each of these 1000 samples, we compute the Goodman 95% simultaneous confidence intervals using function `multinomCI` from package `DescTools`, and we count how many times these intervals contain the true probabilities:

```
k<-0
for(i in 1:q){
  int<-MultinomCI(N[,i],conf.level = 0.95,method="goodman")
  if(all((int[,2]<=p)&(int[,3]>=p))) k<-k+1
}
print(k/q)
```

```
## [1] 0.959
```

The coverage rate is close to the nominal value. We can compute a 95% confidence interval on the coverage rate as follows:

```
print(prop.test(k,q)$conf.int)
```

```
## [1] 0.9442836 0.9700623
## attr(,"conf.level")
## [1] 0.95
```

Question 2

We will use function `makeF` from package `evclust`, which gives all the subsets of a frame of K elements, in binary form. For instance, with $K = 3$, we get

```
Foc <- makeF(3,"full")
print(Foc)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    1    0    0
## [3,]    0    1    0
## [4,]    1    1    0
## [5,]    0    0    1
## [6,]    1    0    1
## [7,]    0    1    1
## [8,]    1    1    1
```

In this matrix, each row corresponds to a subset of a frame $\{1, 2, 3\}$ with three elements. The first row corresponds the empty set, the second and third row are, respectively, the singletons $\{1\}$ and $\{2\}$, the fourth row is the pair $\{1, 2\}$, etc. We can remark that there is a symmetry in this table: the last row corresponds to the complement of the first one, row 7 is the complement of row 2, etc.

Let us consider a particular sample, and the corresponding confidence region:

```
K<-3
N<-c(20,30,50)
int<-MultinomCI(N,conf.level=0.95,method="goodman")
```

We can compute the quantities

$$\sum_{\xi_k \in A} P^-(\xi_k) \quad \text{and} \quad \sum_{\xi_k \in A} P^+(\xi_k)$$

for all subsets A as follows:

```
  SPm<-Foc%*%int[,2]
  SPp<-Foc%*%int[,3]
```

The lower and upper probabilities $P^-(A)$ and $P^+(A)$ can then be computed for any subset A as

```
  Pm<-pmax(SPm,1-SPp[2^K:1])
  Pp<-pmin(SPp,1-SPm[2^K:1])
  print(cbind(Pm,Pp))
```

```
##           [,1]      [,2]
## [1,] 0.0000000 0.0000000
## [2,] 0.1217246 0.3107983
## [3,] 0.2036004 0.4180816
## [4,] 0.3835903 0.6164097
## [5,] 0.3835903 0.6164097
## [6,] 0.5819184 0.7963996
## [7,] 0.6892017 0.8782754
## [8,] 1.0000000 1.0000000
```

We know that, for $K = 2$ and $K = 3$, the lower envelope P^- is a belief function, and the upper envelope P^+ is the corresponding plausibility function. We can use function `beltom` in package `ibelief` to compute the corresponding mass function:

```
m<-as.vector(beltom(Pm))
```

Let us print the mass, belief and pausibility functions together with the focal sets:

```
print(cbind(Foc,m,Pm,Pp),2)
```

```
##                 m
## [1,] 0 0 0 0.000 0.00 0.00
## [2,] 1 0 0 0.122 0.12 0.31
## [3,] 0 1 0 0.204 0.20 0.42
## [4,] 1 1 0 0.058 0.38 0.62
## [5,] 0 0 1 0.384 0.38 0.62
## [6,] 1 0 1 0.077 0.58 0.80
## [7,] 0 1 1 0.102 0.69 0.88
## [8,] 1 1 1 0.054 1.00 1.00
```

We observe that the masses are all positive, which confirms that P^- is indeed a belief function. Finally, we can write the following generic function that computes m, Bel and Pl for any sample N:

```
multinom_pbf1<-function(N,level=0.95,method_CI="goodman"){
  K<-length(N)
  if(K>3) stop("K must be <=3")
  int<-MultinomCI(N,conf.level=level,method=method_CI)
  Foc<-makeF(K,type="full")
  SPm<-Foc%*%int[,2]
  SPp<-Foc%*%int[,3]
  Pm<-pmax(SPm,1-SPp[2^K:1])
  Pp<-pmin(SPp,1-SPm[2^K:1])
  m<-as.vector(beltom(Pm))
  return(list(m=m,Bel=Pm,Pl=Pp))
}
```

Question 3

Let us now check that the predictive belief function Bel computed from simultaneous confidence intervals is less committed than the probability distribution P_X of X, for at least $100(1-\alpha)\%$ of the samples, i.e.,

$$P(Bel \leq P_X) \geq 1-\alpha.$$

We start with a probability vector p, and compute the corresponding probability measure P_X:

```
p <- c(0.2,0.3,0.5)
P_X <- Foc%*%p
```

We generate 1000 samples from a multinomial random variable N with parameters $n = 100$ and p:

```
q<-1000
N<-rmultinom(q,100,p)
```

We then count how many times Bel is less committed than P_X:

```
k<-0
for(i in 1:q){
  pbf<-multinom_pbf1(as.vector(N[,i]))
  if(all(pbf$Bel<=P_X)) k<-k+1
}
print(k/q)
```

```
## [1] 0.952
```

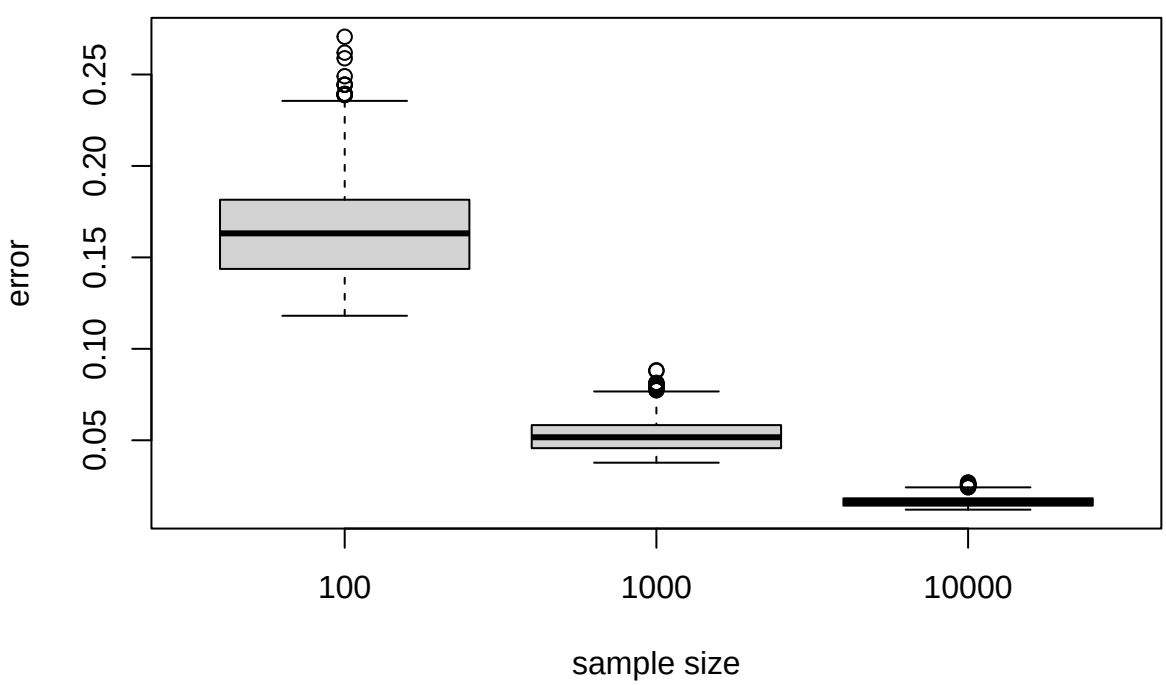
This is very close to the nominal value 0.95.

Finally, let us check the convergence property,

$$\forall A \subseteq \mathcal{X}, Bel(A) \xrightarrow{P} P_X(A) \text{ as } n \to \infty.$$

For this, we will generate samples with $n = 100$, $n = 1000$ and $n = 10^4$, and plot the distributions of errors $\max_A |Bel(A) - P_X(A)|$:

```r
# n=100
N<-rmultinom(q,100,p)
err1<-rep(0,q)
for(i in 1:q){
  pbf<-multinom_pbf1(as.vector(N[,i]))
  err1[i]<-max(abs(P_X-pbf$Bel))
}
# n=1000
N<-rmultinom(q,1000,p)
err2<-rep(0,q)
for(i in 1:q){
  pbf<-multinom_pbf1(as.vector(N[,i]))
  err2[i]<-max(abs(P_X-pbf$Bel))
}
# n=10000
N<-rmultinom(q,10000,p)
err3<-rep(0,q)
for(i in 1:q){
  pbf<-multinom_pbf1(as.vector(N[,i]))
  err3[i]<-max(abs(P_X-pbf$Bel))
}
boxplot(err1,err2,err3,names=c(100,1000,10000),xlab="sample size",ylab="error")
```



Question 4

When $K > 3$, P^- is not, in general, a belief function. We need to find a belief function Bel verifying $Bel \leq P^-$, but still as committed as possible. This can be achieved by solving the following linear optimization problem:

$$\max_m J(m) = \sum_{A \subseteq \mathcal{X}} Bel(A)$$

under the constraints:

$$Bel(A) \leq P^-(A), \quad \forall A \subset \mathcal{X},$$
$$\sum_{A \subseteq \mathcal{X}} m(A) = 1,$$
$$m(A) \geq 0, \quad \forall A \subseteq \mathcal{X}.$$

The objective function can be written as

$$J(m) = \sum_{B \subseteq \mathcal{X}} 2^{K-|B|} m(B)$$

and the constraints $Bel(B) \leq P^-(A)$ can be written as

$$\sum_{B \subseteq A} m(B) \leq P^-(A).$$

Here, we notice that the vector containing all the numbers $Bel(B)$ for $B \subseteq \mathcal{X}$ can be obtained by multiplying the vector of masses by a matrix BfrM of size $2^K \times 2^K$:

$$Bel = \mathsf{BfrM} \cdot m.$$

The general term of matrix BfrM is

$$\mathsf{BfrM}(A, B) = \begin{cases} 1 & \text{if } B \subseteq A \\ 0 & \text{otherwise.} \end{cases}$$

An efficient way to compute this matrix recursively is described in the following paper:

Philippe Smets, The application of the matrix calculus to belief functions, *International Journal of Approximate Reasoning* 31(1–2):1-30, 2002.

This method can be implemented in R as follows (here, $K = 3$):

```
BfrM<-1
for(i in 1:K) BfrM<-rbind(cbind(BfrM,matrix(0,2^(i-1),2^(i-1))),cbind(BfrM,BfrM))
print(BfrM)
```

```
##      BfrM
## [1,]    1 0 0 0 0 0 0 0
## [2,]    1 1 0 0 0 0 0 0
## [3,]    1 0 1 0 0 0 0 0
## [4,]    1 1 1 1 0 0 0 0
## [5,]    1 0 0 0 1 0 0 0
## [6,]    1 1 0 0 1 1 0 0
## [7,]    1 0 1 0 1 0 1 0
## [8,]    1 1 1 1 1 1 1 1
```

For linear optimization, we can use function `lp` in package `lpSolve`. Here is a more general function for computing a predictive belief function, for any value of K:

```r
multinom_pbf<-function(N,level=0.95,method_CI="goodman"){
  # Implements the method to compute a predictive belief function in the paper:
  # T. Denoeux. Constructing Belief Functions from Sample Data Using Multinomial
  # Confidence Regions. Int. J. of Approx. Reasoning 42(3):228-252, 2006.
  K<-length(N)
  M<-2^K-1
  int<-MultinomCI(N,conf.level=level,method=method_CI)
  Foc<-makeF(K,type="full")
  SPm<-Foc%*%int[,2]
  SPp<-Foc%*%int[,3]
  Pm<-pmax(SPm,1-SPp[2^K:1])
  Pp<-pmin(SPp,1-SPm[2^K:1])
  if(K<=3){ # Pm is a belief function, Pp is the corresponding plausibility function
    m<-beltom(Pm)
    Bel<-Pm
    Pl<-Pp
    Jopt<-NULL
  } else{ # Find Bel approximating Pm by linear programming
    BfrM<-1
    for(i in 1:K) BfrM<-rbind(cbind(BfrM,matrix(0,2^(i-1),2^(i-1))),cbind(BfrM,BfrM))
    BfrM <- BfrM[-1,-1] # We remove the row and column corresponding to the empty set
    card<-rowSums(Foc)[-1]
    f<-2^(K-card)
    C<-rbind(BfrM,diag(M))
    const.dir<-c(rep("<=",M-1),"=",rep(">=",M))
    const.rhs<-c(Pm[-1],rep(0,M))
    opt<-lp("max",f,C,const.dir,const.rhs)
    m<-c(0,opt$solution)
    Bel<-as.vector(mtobel(m))
    Pl<-as.vector(mtopl(m))
    Jopt<-opt$objval
  }
  return(list(Pm=Pm,Pp=Pp,m=m,Bel=Bel,Pl=Pl,J=Jopt,Foc=Foc))
}
```

Let us test this function with the psychiatric patients dataset:

```r
N<-c(91, 49, 37, 43)
pbf<-multinom_pbf(N,method_CI="goodman")
print(cbind(pbf$Pm,pbf$Bel))
```

```
##            [,1]      [,2]
##  [1,] 0.0000000 0.0000000
##  [2,] 0.3342025 0.3342025
##  [3,] 0.1608587 0.1608587
##  [4,] 0.4950612 0.4950612
##  [5,] 0.1145513 0.1145513
##  [6,] 0.4487538 0.4487538
##  [7,] 0.2754101 0.2754101
##  [8,] 0.7297642 0.7297642
##  [9,] 0.1374690 0.1374690
## [10,] 0.4716715 0.4716715
## [11,] 0.2983278 0.2983278
## [12,] 0.7598879 0.7598879
```

```
## [13,] 0.2520204 0.2520204
## [14,] 0.7001126 0.5916319
## [15,] 0.5021668 0.4128791
## [16,] 1.0000000 1.0000000
```

```
print(pbf$J)
```

```
## [1] 6.482489
```

We observe that the solution is not exactly the same as in the paper. Indeed, a linear optimization problem generally has many equivalent solutions. The value of the objective fucntion is the same as that reported in the paper: the two solutions are equivalent.

Question 5

Let us consider a consider a vector of probabilities with $K = 5$, as well as the corresponding probability measure p_X:

```
K<-5
p<-c(0.2,0.3,0.15,0.1,0.25)
Foc<-makeF(K,type="full")
P_X<-Foc%*%p
```

As before, we generate $q = 1000$ samples from a multinomial random variable N with parameters $n = 100$ and p:

```
q<-1000
N<-rmultinom(q,100,p)
```

We then count how many times Bel is less committed than P_X:

```
k<-0
for(i in 1:q){
  pbf<-multinom_pbf(as.vector(N[,i]))
  if(all(pbf$Bel<=P_X)) k<-k+1
}
print(k/q)
```

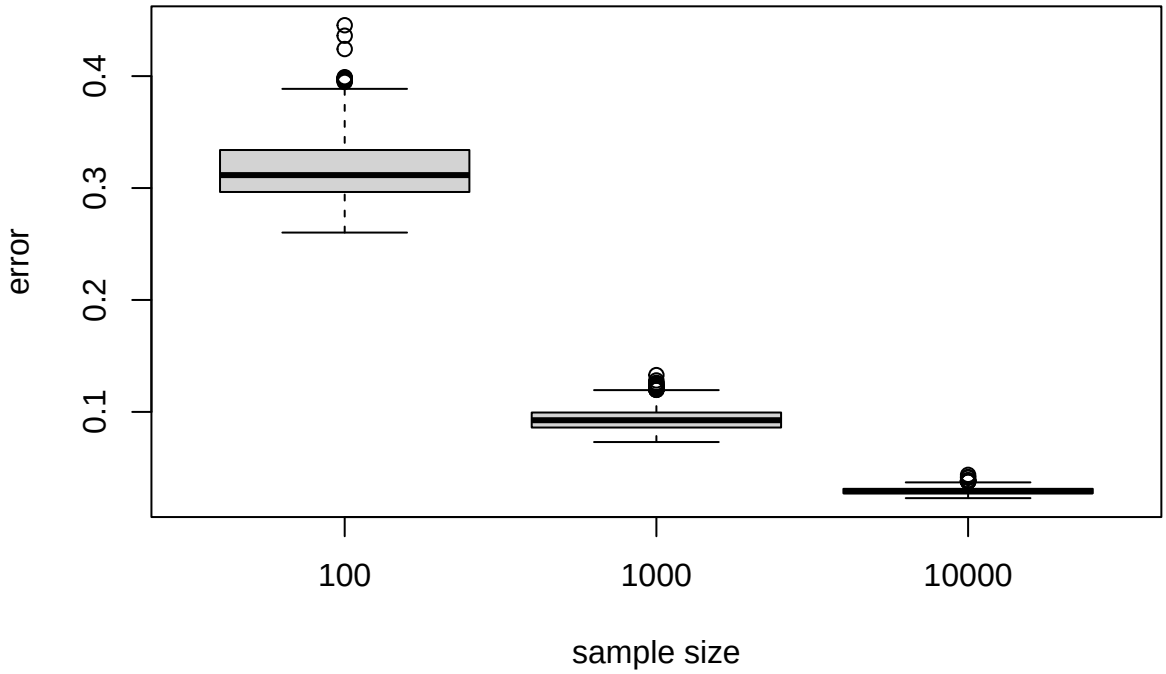
```
## [1] 0.955
```

This is again very close to the nominal value 0.95.

To verify the convergence property experimentally, we again proceed as before: we generate samples with $n = 100$, $n = 1000$ and $n = 10^4$, and plot the distributions of errors $\max_A |Bel(A) - P_X(A)|$:

```
# n=100
N<-rmultinom(q,100,p)
err1<-rep(0,q)
for(i in 1:q){
  pbf<-multinom_pbf(as.vector(N[,i]))
  err1[i]<-max(abs(P_X-pbf$Bel))
}
# n=1000
N<-rmultinom(q,1000,p)
err2<-rep(0,q)
for(i in 1:q){
  pbf<-multinom_pbf(as.vector(N[,i]))
  err2[i]<-max(abs(P_X-pbf$Bel))
```

```
}
# n=10000
N<-rmultinom(q,10000,p)
err3<-rep(0,q)
for(i in 1:q){
  pbf<-multinom_pbf(as.vector(N[,i]))
  err3[i]<-max(abs(P_X-pbf$Bel))
}
boxplot(err1,err2,err3,names=c(100,1000,10000),xlab="sample size",ylab="error")
```



Question 6

We start by writing a generic function that computes the lower, upper and pignitic expected utilities given a mass function and a utility matrix:

```
expectations <- function(m,Foc,U){
  nfoc<-length(m) # number of focal sets
  K<-ncol(Foc) # number of states
  r<-nrow(U) # number of acts
  Umin<-matrix(0,r,nfoc)
  Umax<-matrix(0,r,nfoc)
  Umean<-matrix(0,r,nfoc)
  for(i in 1:nfoc)
    if(any(Foc[i,]==1))
      for(j in 1:r){
        Umin[j,i]<-min(U[j,which(Foc[i,]==1)])
        Umax[j,i]<-max(U[j,which(Foc[i,]==1)])
        Umean[j,i]<-mean(U[j,which(Foc[i,]==1)])
    }
  Emin<-Umin%*%m
  Emax<-Umax%*%m
```

```
  Ebet<-Umean%*%m
  return(list(Emin=Emin,Emax=Emax,Ebet=Ebet))
}
```

Let us now use this function with the data of the exercise:

```
U<-matrix(c(100,50,30,30,20,
            300,500,200,100,-100,
            50,100,200,100, 50,
            0,10,20,50,20,
            -20,-50,100,150,200),byrow=TRUE,5,5)
N<-c(20, 5, 10, 40, 25)
pbf<-multinom_pbf(N)
expec<-expectations(pbf$m,pbf$Foc,U)
print(cbind(expec$Emin,expec$Emax,expec$Ebet))
```

```
##          [,1]      [,2]      [,3]
## [1,] 34.79128  58.79924  43.44726
## [2,] 59.83758 208.76560 132.84956
## [3,] 71.93945 116.66258  90.26489
## [4,] 19.68725  36.03509  26.80730
## [5,] 68.99501 143.14606 107.78461
```

We can see that the conservative strategy (maximizing the lower expected utility) prescribes to choose a red ball; in contrast, according to the optimistic and pignistic strategies, we should choose a black ball.