Computational Statistics. Analysis of Labor Force Participation Data using the Probit Model

Thierry Denoeux

2024-03-26

```
set.seed(20240321)
```

Question 1

```
library(Ecdat)
data(Participation)
fit<-glm(lfp~.,data=Participation,family=binomial("probit"),x=TRUE)
summary(fit)
```

```
##
## Call:
## glm(formula = lfp ~ ., family = binomial("probit"), data = Participation,
##     x = TRUE)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  6.36847    1.28939   4.939 7.85e-07 ***
## lnnlinc     -0.50258    0.12257  -4.100 4.12e-05 ***
## age         -0.31085    0.05424  -5.731 9.97e-09 ***
## educ         0.02041    0.01753   1.164    0.244
## nyc         -0.78454    0.10352  -7.579 3.50e-14 ***
## noc         -0.01348    0.04489  -0.300    0.764
## foreignyes   0.80434    0.11926   6.745 1.54e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1203.2  on 871  degrees of freedom
## Residual deviance: 1053.0  on 865  degrees of freedom
## AIC: 1067
##
## Number of Fisher Scoring iterations: 4
```

Question 2

Question 2a

Function `confint.glm` computes approximate confidence intervals for the coefficients using the profile likelihood:

```
CI1<-confint(fit)
```

```
## Waiting for profiling to be done...
```

```
print(CI1)
```

```
##                    2.5 %      97.5 %
## (Intercept)  3.84681410  8.96302455
## lnnlinc     -0.74938932 -0.26284474
## age         -0.41837345 -0.20459458
## educ        -0.01392008  0.05487422
## nyc         -0.98459566 -0.58923131
## noc         -0.10189367  0.07478181
## foreignyes   0.57255699  1.03848220
```

Question 2b

To compute normal-approximation confidence intervals, we first write a function that computes the conditional likelihood for this problem:

```
loglik <- function(beta,X,y){
  logPhi <- pnorm(X%*%beta,log.p=TRUE)
  log1mPhi <- pnorm(X%*%beta,lower.tail=FALSE,log.p=TRUE)
  return(sum(y*logPhi+ (1-y)*log1mPhi))
}
```

We then maximize this function using function `optim` and set `Hessian=TRUE` to retrieve the Hessian matrix at the maximum:

```
X<-fit$x
y<-fit$y
p<-ncol(X)
opt<-optim(rep(0,p),fn=loglik,X=X,y=y,method="BFGS",control=list(fnscale=-1),
           hessian=TRUE)
H<-opt$hessian
```

We can check that the coefficient estimates are identical to those computed by function `glm`:

```
print(cbind(fit$coefficients,opt$par))
```

```
##                     [,1]        [,2]
## (Intercept)  6.36846811  6.36939564
## lnnlinc     -0.50258264 -0.50267281
## age         -0.31085093 -0.31084954
## educ         0.02040503  0.02040759
## nyc         -0.78453977 -0.78452767
## noc         -0.01348039 -0.01347479
## foreignyes   0.80434322  0.80433556
```

The observed information matrix is the negative inverse of the Hessian at $\widehat{\beta}$. It is an estimate of the covariance matrix of the estimate. It allows us to compute approximate confidence intervals:

```
V <- - solve(H)
CI2<-cbind(opt$par-1.96*sqrt(diag(V)),opt$par+1.96*sqrt(diag(V)))
print(cbind(CI1,CI2))
```

```
##                   2.5 %      97.5 %
## (Intercept)  3.84681410  8.96302455  3.81219661  8.92659467
## lnnlinc     -0.74938932 -0.26284474 -0.74587088 -0.25947474
## age         -0.41837345 -0.20459458 -0.41771594 -0.20398314
## educ        -0.01392008  0.05487422 -0.01398144  0.05479662
## nyc         -0.98459566 -0.58923131 -0.98217777 -0.58687758
## noc         -0.10189367  0.07478181 -0.10178840  0.07483881
## foreignyes   0.57255699  1.03848220  0.57142804  1.03724308
```

The intervals computed by the two methods are very close.

Question 2c

We now use the case-based bootstrap and the percentile method to obtain approximate confidence intervals:

```
B<-10000
BETA <- matrix(0,B,p)
n <- nrow(X)
for(i in 1:B){
  ii <- sample(n,n,replace=TRUE)
  fit<-glm(lfp~.,data=Participation[ii,],family=binomial("probit"))
  BETA[i,]<-fit$coefficients
}
CI3<-matrix(0,p,2)
for(j in 1:p) CI3[j,]<-quantile(BETA[,j],probs=c(0.025,0.975))
print(cbind(CI1,CI2,CI3))
```

```
##                   2.5 %      97.5 %
## (Intercept)  3.84681410  8.96302455  3.81219661  8.92659467  4.0197331
## lnnlinc     -0.74938932 -0.26284474 -0.74587088 -0.25947474 -0.7480201
## age         -0.41837345 -0.20459458 -0.41771594 -0.20398314 -0.4237265
## educ        -0.01392008  0.05487422 -0.01398144  0.05479662 -0.0134171
## nyc         -0.98459566 -0.58923131 -0.98217777 -0.58687758 -1.0415222
## noc         -0.10189367  0.07478181 -0.10178840  0.07483881 -0.1058313
## foreignyes   0.57255699  1.03848220  0.57142804  1.03724308  0.5743703
##
## (Intercept)  9.00286049
## lnnlinc     -0.27587670
## age         -0.21122262
## educ         0.05565414
## nyc         -0.58754604
## noc          0.07470384
## foreignyes   1.05445883
```

Again, the obtained intervals are similar to those found by the other two methods.

Question 3

Question 3a

A first idea is to use the rejection sampling method, using the prior as proposal distribution. To generate from the multivariate normal distribution, we use function `mvrnorm` from package `MASS`. We try to generate a sample of size $N = 1000$.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:Ecdat':
##
##     SP500
```

```
beta0<-rep(0,p)
B0<-10*diag(p)
N<-1000
Beta<-matrix(0,N,p)
i<-0
j<-0
likh<-opt$value
while((i<N) & (j<10000)){
  j<-j+1
  beta_star<-mvrnorm(1,mu=beta0, Sigma=B0)
  u <- runif(1)
  if(u<exp(loglik(beta_star,X,y)-likh)){
    print(i)
    Beta[i,]<-beta_star
  }
}
print(c(i,j))
```

```
## [1]     0 10000
```

We can see that this method fails because all candidate values β^* are rejected. We need a more sophisticated method.

Question 3b

We will implement a Gibbs sampler. The idea is to generate, alternatively, $\boldsymbol{y}^*$ from its distribution given β and $\boldsymbol{y}$, and β from its distribution given $\boldsymbol{y}^*$ and $\boldsymbol{y}$.

Given β and $\boldsymbol{y}$, y_i^* has a truncated normal distribution $TN_{(-\infty,0)}(x_i^T\beta, 1)$ if $y_i = 0$ and $TN_{[0,+\infty)}(x_i^T\beta, 1)$ if $y_i = 1$.

Given $\boldsymbol{y}^*$, the estimation of β becomes a linear regression problem. The joint posterior pdf of β given $\boldsymbol{y}^*$ is

$$f(\beta|\boldsymbol{y}^*) \propto f(\boldsymbol{y}^*|\beta)f(\beta),$$

that is,

$$f(\beta|\boldsymbol{y}^*) \propto \exp\left[-\frac{1}{2}(\boldsymbol{y}^* - \boldsymbol{X}\beta)^T(\boldsymbol{y}^* - \boldsymbol{X}\beta)\right] \exp\left[-\frac{1}{2}(\beta - \beta_0)^T\boldsymbol{B}_0^{-1}(\beta - \beta_0)\right]$$

Developing and rearranging the terms in the exponential, we get

$$f(\beta|\boldsymbol{y}^*) \propto \exp\left\{-\frac{1}{2}\left[\beta^T(\boldsymbol{X}^T\boldsymbol{X}+\boldsymbol{B}_0^{-1})\beta - 2\beta^T(\boldsymbol{X}\boldsymbol{y}+\boldsymbol{B}_0^{-1}\beta_0)) + \sigma^{-2}\boldsymbol{y}^T\boldsymbol{y} + \beta_0^T\boldsymbol{B}_0^{-1}\beta_0\right]\right\}$$
$$\propto \exp\left\{-\frac{1}{2}(\beta-\overline{\beta})^T\boldsymbol{B}_1^{-1}(\beta-\overline{\beta})\right\},$$

with

$$\boldsymbol{B}_1 = (\boldsymbol{X}^T\boldsymbol{X}+\boldsymbol{B}_0^{-1})^{-1}$$

and

$$\overline{\beta} = \boldsymbol{B}_1(\boldsymbol{X}^T\boldsymbol{y}^* + \boldsymbol{B}_0^{-1}\beta_0).$$

To generate from the truncated normal distribution, we use function `rtruncnorm` from package `truncnorm`. We thus have the following algorithm:

```
gibbs_probit<- function(y,X,beta0,B0,D=1000,N=5000){
  n<-nrow(X)
  p<-ncol(X)
  B0inv<-solve(B0)
  B1 <- solve(t(X)%*%X + B0inv)
  beta<-matrix(0,nrow=D+N,ncol=p)
  ystar<-y
  # Initialization
  beta_old<-    mvrnorm(n=1,mu=beta0,Sigma=B0)
  for(t in 1:(D+N)){
    for(i in 1:n)
      if(y[i]==0)
        ystar[i]<- rtruncnorm(1, a=-Inf, b=0, mean = X[i,]%*%beta_old, sd = 1)
      else
        ystar[i]<- rtruncnorm(1, a=0, b=Inf, mean = X[i,]%*%beta_old, sd = 1)
    beta1<- B1 %*% (t(X)%*%ystar + B0inv%*%beta0)
    beta[t,]<- mvrnorm(n=1,mu=beta1,Sigma=B1)
    beta_old<-beta[t,]
  }
  return(beta[(D+1):(D+N),])
}
```

We apply this algorithm to our data with a burn-in period of length $D = 1000$ and a usable chain length $N = 5000$, and we compute the 95% credible interval.

```
D <- 1000
N <- 5000
library("truncnorm")
beta <- gibbs_probit(y,X,beta0,B0,D,N)
CI4 <- matrix(0,p,2)
for(j in 1:p) CI4[j,] <- quantile(beta[,j],probs=c(0.025,0.975))
print(cbind(CI1,CI4))
```

```
##                   2.5 %      97.5 %
## (Intercept)  3.84681410  8.96302455  3.15139159  7.7625716
## lnnlinc     -0.74938932 -0.26284474 -0.64010478 -0.1946832
## age         -0.41837345 -0.20459458 -0.41117349 -0.2000855
## educ        -0.01392008  0.05487422 -0.01477162  0.0537691
## nyc         -0.98459566 -0.58923131 -0.98395126 -0.5788565
## noc         -0.10189367  0.07478181 -0.10401105  0.0736785
## foreignyes   0.57255699  1.03848220  0.58302379  1.0546186
```