

```
#include <gsl/gsl_randist.h>
#include <gsl/gsl_cdf.h>
#include <gsl/gsl_math.h>
#include <arrays.h>
#include <mle.h>

class TOBIT : public MLE
{
public:
    TOBIT(vector& Y, matrix& X): MLE(Y, X){}
    TOBIT(double* y, double* x, int n, int k): MLE(y, x, n, k){}
    unsigned numobs_nonlimit(){return n1;}

private:
    unsigned n1, n0;
    double Y1bar, sY1;

protected:
    virtual void setname(){name=(char *)"TOBIT";}
    class array1dim<bool> censored;

    double lnL();
    void dlnL(vector& derivative);
    void initialvalues();
    void calculatecov();
};
```

```

void TOBIT::initialvalues()
{
    int i;
    double *y;
    bool *c;
    kparam++;
    b.resize(kparam);
    covar.resize(kparam,kparam);
    censored.setsize(n);
    Y1bar=0.0;
    n0=0;
    for(i=n,y=Y.ptr(), c=censored.ptr(); i; i--, y++, c++) if(*c = (*y == 0.0)) n0++; else Y1bar
+= *y;
    n1 = n-n0;
    Y1bar /= n1;
    sY1 = 0.0;
    for(i=n,y=Y.ptr(), c=censored.ptr(); i; i--, y++, c++) if(!*c) sY1 += (*y * *y);
    sY1 /= (n-1);
    b.zero();
    b(klisted) = 1.0 / sY;
}

```

```

double TOBIT::lnL()
{
    double l, zi, *y=Y.ptr(), *x=X.ptr(), *p=b.ptr()+klisted;
    bool *c=censored.ptr();
    int i, j;
    l=0.0;
    for(i=n; i; i--, y++, c++){

```

```

    zi = *c ? 0.0 : *p * *y;
    for(j=klisted, p-=klisted;j<, x++, p++) zi -= *p * *x;
    l += *c ? log ( gsl_cdf_ugaussian_P(zi) ) : -0.5 * zi * zi;
}
l -= 0.5 * n1 * log(2.0*M_PI);
l += n1 * log(*p);
return l;
}

void TOBIT::dlnL(vector& derivative){
    int i, j, j1;
    double zi, *d=derivative.ptr()+klisted, *y=Y.ptr(), *p=b.ptr()+klisted, *x=X.ptr();
    bool *c=censored.ptr();
    derivative.zero();
    *d = n1 / (*p);
    for(i=0;i<n;i++, c++, y++) {
        d-=klisted;
        zi = *c ? 0.0 : *p * *y;
        for(j1=0, p-=klisted; j1<klisted; j1++,p++,x++) zi -= *p * *x;
        if(*c) zi = -gsl_ran_ugaussian_pdf(zi) / gsl_cdf_ugaussian_P(zi);
        for(j=0, x-=klisted;j<klisted;j++, d++,x++) *d += zi * *x;
        if(!*c) *d -= zi * *y;
    }
}

void TOBIT::calculatecova()
{
    LikelyhoodRatio=lnL();
    int i, j, l, j1;

```

```

double sigma=1.0/b(klisted), sigma2=sigma*sigma;
matrix Score(n,kparam), J(kparam,kparam), ara(kparam,kparam);
double *s=Score.ptr(), zi, *y=Y.ptr(), *p=b.ptr()+klisted, *x=X.ptr();
bool *c=censored.ptr();
Score.zero();
for(i=0;i<n;i++, c++, y++, s++) {
    zi = *c ? 0.0 : *p * *y;
    for(j1=0, p-=klisted; j1<klisted; j1++,p++,x++) zi -= *p * *x;
    x-=klisted;
    if(*c){
        for(j=0;j<klisted;j++,x++, s++) *s += -gsl_ran_ugaussian_pdf(zi) /
gsl_cdf_ugaussian_P(zi) * *x;
    } else{
        for(j=0;j<klisted;j++,x++,s++) *s += zi * *x;
        *s += 1.0 / *p - zi * *y;
    }
}
MtM(Score,cova);
cova.inv();
for(i=0;i<kparam;i++) for(j=0;j<kparam;j++) J(i,j) = (i==j) ? ((i<klisted) ? sigma : -
sigma2) : ((j==klisted) ? -sigma2*b(i) : 0.0);
ara=J;
ara*cova;
J.transpose();
ara*J;
cova=ara;
for(j=0;j<klisted;j++) b(j) *= sigma;
b(klisted)=sigma;
}

```