Chapter 7 Heteroscedastic Tobit Model Code

The code for estimating a heteroscedastic Tobit model in R and Stata is provided in this document, along with an annotated demonstration of it using the reading accuracy scores example from chapters 6 and 7.

Code in R:

Functions and their usage

The log-likelihood function `htobit` receives a vector of starting values *h*, dependent variable *y*, an indicator variable *t* distinguishing censored from uncensored observations, a vector of predictors *x* for the location submodel, and a vector of predictors *z* for the dispersion submodel.

```
# This is the log-likelihood function:
htobit <- function(h, y, t, x, z)
{
hx = x%*%h[1: length(x[1,])]
mu = hx
gz = z%*%h[length(x[1,])+1: length(z[1,])]
s = exp(gz)
   loglik = t*log(dnorm(y,mean = mu,sd = s)) + (1-t)*log(1 - pnorm(y,mean =
   mu,sd = s))
   -sum(loglik, na.rm = TRUE)
}
```

The next set of commands illustrates a two-step estimation procedure that we recommend for estimating these models. First, we assume that commands have been issued in R that construct or identify the vectors and/or variables `start`, `xdata`, `ydata`, `tdata`, and `zdata`. Then these vectors are passed along with the function `htobit` to `optim` using the Nelder-Mead algorithm for estimation. The parameter estimates from this step are substituted for `start` in the second step, which re-estimates the model using the BFGS algorithm.

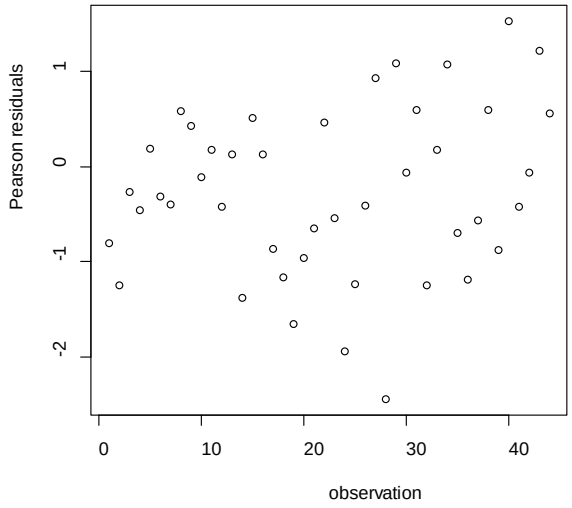
```
# The maximum likelihood estimation proceeds in two stages. The first is a
   “rough” but reliable estimator:
> betaopt0 <- optim(start, htobit, hessian = T, x = xdata, y = ydata, t =
   tdata, z = zdata, method = "Nelder-Mead")
# We use the parameter estimates from betaopt0 as new starting values, and
   then re-estimate the model with a more fine-tuned method:
> start <- betaopt0$par
> betaopt1 <- optim(start, htobit, hessian = T, x = xdata, y = ydata, t =
   tdata, z = zdata, method = "BFGS")
```

Reading accuracy score example

```
# Assuming that you’ve already loaded up the dyslexic3 data file and
attached it.
> library(MASS)
# This is the log-likelihood function:
htobit <- function(h, y, t, x, z)
{
hx = x%*%h[1: length(x[1,])]
mu = hx
gz = z%*%h[length(x[1,])+1: length(z[1,])]
s = exp(gz)
   loglik = t*log(dnorm(y,mean = mu,sd = s)) + (1-t)*log(1 - pnorm(y,mean =
   mu,sd = s))
   -sum(loglik, na.rm = TRUE)
}
# Next, we need to declare that the upper censoring limit, tau, is 1:
```

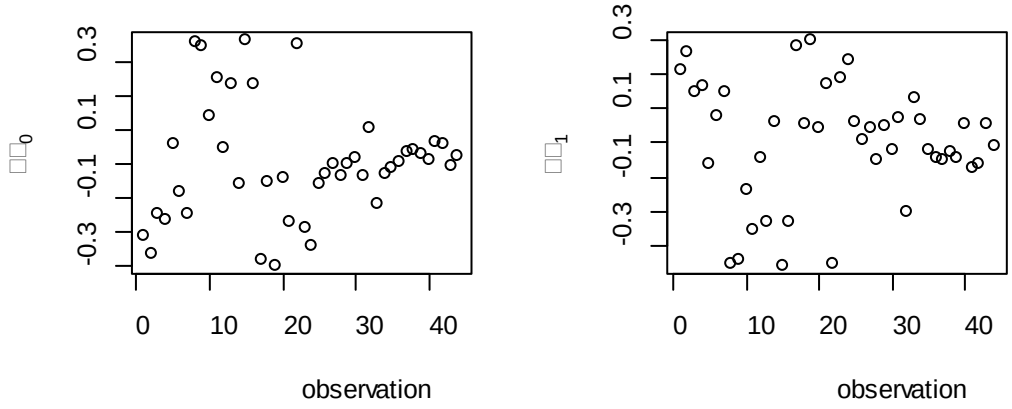
```
> tau = 1;
# Then we need to declare the DV:
> ydata <- score;
# We create an indicator variable, tdata, that is 1 for uncensored
   observations and 0 for censored observations:
> tdata <- c(rep(0,length(ydata)));
> for (i in 1:length(ydata)) if(ydata[i] < tau) tdata[i] = 1;
# Then we specify the variables included in the location submodel:
> const <- rep(1,length(ydata));
> dziq <- dys*ziq;
> xdata <- cbind(const,dys,ziq,dziq);
# And finally, we specify the variables included in the dispersion
   submodel:
> zdata <- cbind(const, dys);
# We provide starting values from the last censReg model for this example
   from chapter 7, with the parameter for dys in the dispersion submodel
   being given the starting value of 0.1:
> start <- c(0.90919296, -0.3128252, 0.09426998, -0.10886289, -
   2.0622075,0.1)
# The maximum likelihood estimation first step:
> betaopt0 <- optim(start, htobit, hessian = T, x = xdata, y = ydata, t =
   tdata, z = zdata, method = "Nelder-Mead")
# We use the parameter estimates from betaopt0 as new starting values, and
   then re-estimate the model with a more fine-tuned method:
> start <- betaopt0$par
> betaopt1 <- optim(start, htobit, hessian = T, x = xdata, y = ydata, t =
   tdata, z = zdata, method = "BFGS")
# Here is the log-likelihood:
> betaopt1$value
[1] -19.69492
# And here are the parameter estimates and significance tests:
> outopt1 <- rbind(estim <- betaopt1$par, serr <-
   sqrt(diag(solve(betaopt1$hessian))), zstat <- estim/serr, prob <- 2*(1-
   pnorm(abs(zstat)))); row.names(outopt1) <- c("estim", "serr", "zstat",
   "prob"); outopt1
             [,1]          [,2]       [,3]        [,4]          [,5]
estim  0.93953587 -3.431894e-01 0.11713668 -0.13171602 -1.580260e+00
serr   0.05505395  5.789106e-02 0.06306718  0.06535104  2.289887e-01
zstat 17.06573066 -5.928194e+00 1.85733192 -2.01551512 -6.901041e+00
prob   0.00000000  3.062853e-09 0.06326395  0.04385070  5.162315e-12
               [,6]
estim -1.218278e+00
serr   2.806271e-01
zstat -4.341270e+00
prob   1.416617e-05
#
# Finally, add ziq to the dispersion submodel:
zdata <- cbind(const, dys, ziq);
start <- c(0.93953587, -0.3431894, 0.11713668, -0.13171602, -1.580260, -
  1.218278,0.1)
betaopt0 <- optim(start, htobit, hessian = T, x = xdata, y = ydata, t =
tdata, z = zdata, method = "Nelder-Mead")
> start <- betaopt0$par
> betaopt1 <- optim(start, htobit, hessian = T, x = xdata, y = ydata, t =
tdata, z = zdata, method = "BFGS")
> betaopt1$value
[1] -21.40103
> outopt1 <- rbind(estim <- betaopt1$par, serr <-
sqrt(diag(solve(betaopt1$hessian))), zstat <- estim/serr, prob <- 2*(1-
pnorm(abs(zstat)))); row.names(outopt1) <- c("estim", "serr", "zstat",
"prob"); outopt1
```

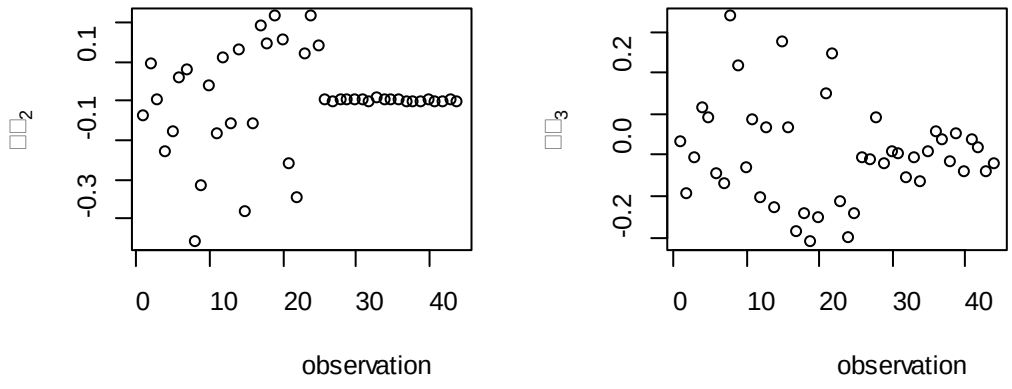
```
            [,1]          [,2]       [,3]        [,4]          [,5]
estim  0.90785214 -3.137709e-01 0.11933325 -0.14112463 -1.486007e+00
serr   0.06526118  6.628411e-02 0.05850790  0.06003806  2.373530e-01
zstat 13.91105824 -4.733727e+00 2.03960934 -2.35058626 -6.260749e+00
prob   0.00000000  2.204341e-06 0.04138925  0.01874386  3.831335e-10
               [,6]        [,7]
estim -1.670181e+00 -0.47437253
serr   3.577230e-01  0.25037692
zstat -4.668923e+00 -1.89463362
prob   3.027824e-06  0.05814096
#
# Let's get residuals for this model.
# We have to build them from scratch.
# To begin, generate Pearson residuals,
# which are (y_i – mu_i)/sigma_i:
pmu <- betaopt1$par[1] + betaopt1$par[2]*dys + betaopt1$par[3]*ziq +
betaopt1$par[4]*dziq
psig <- exp(betaopt1$par[5] + betaopt1$par[6]*dys + betaopt1$par[7]*ziq)
betaopt1pres <- (score - pmu)/psig
plot(betaopt1pres, xlab = "observation", ylab = "Pearson residuals")
```



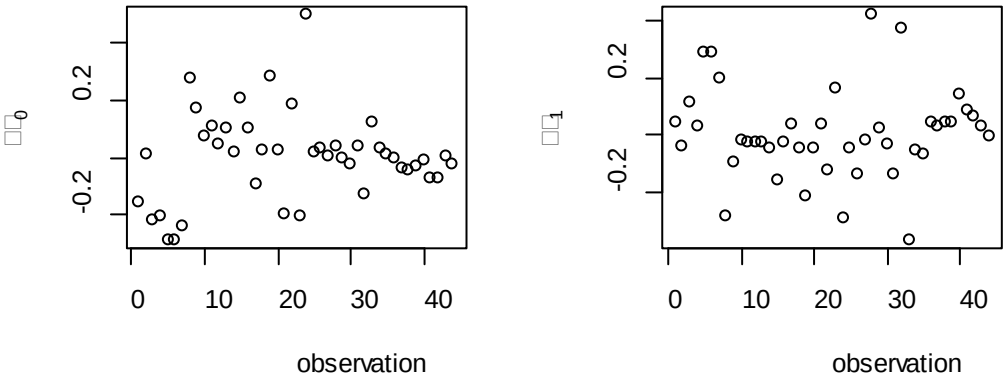

```
#
# Now for the dfbetas.
# Set up an index variable and combine it with the data:
ind <- seq(nrow(tob))
indys <- cbind(ind, tob, dziq)
# Set up an array:
cmat <- array(rep(0,308),dim = c(44,7))
# Provide starting values:
start <- betaopt1$par
serr1 <- sqrt(diag(solve(betaopt1$hessian)))
# Run leave-one-out models and collect the coefficients into the array
for (i in 1:nrow(indys)) {
ydatai <- ydata[which(indys$ind != i)];
xdatai <- xdata[which(indys$ind != i),];
tdatai <- tdata[which(indys$ind != i)];
zdatai <- zdata[which(indys$ind != i),];
betaopt0i <- optim(start, htobit, hessian = T, x = xdatai, y = ydatai, t =
tdatai, z = zdatai, method = "Nelder-Mead")
starti <- betaopt0i$par
betaopt1i <- optim(starti, htobit, hessian = T, x = xdatai, y = ydatai, t =
tdatai, z = zdatai, method = "BFGS")
```

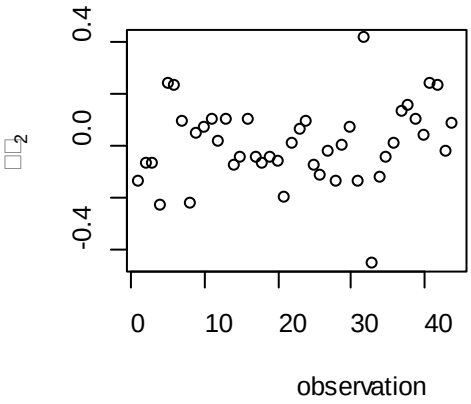
```
cmat[i,] <- (betaopt1$par-betaopt1i$par)/serr1
}
#
# Where are the extreme observations?  Here are the location submodel
plots:
par(mfrow = c(2,2))
plot(indys$ind,cmat[,1], ylab = expression(Delta * beta[0]), xlab =
"observation")
plot(indys$ind,cmat[,2], ylab = expression(Delta * beta[1]), xlab =
"observation")
plot(indys$ind,cmat[,3], ylab = expression(Delta * beta[2]), xlab =
"observation")
plot(indys$ind,cmat[,4], ylab = expression(Delta * beta[3]), xlab =
"observation")
#
```





```
# Now for the dispersion submodel plots:
par(mfrow = c(2,2))
plot(indys$ind,cmat[,5], ylab = expression(Delta * delta[0]), xlab =
"observation")
plot(indys$ind,cmat[,6], ylab = expression(Delta * delta[1]), xlab =
"observation")
plot(indys$ind,cmat[,7], ylab = expression(Delta * delta[2]), xlab =
"observation")
```





Code in Stata:

Functions and their usage

The log-likelihood program `htobit` receives a dependent variable *y*, a threshold value *q* distinguishing censored from uncensored observations, a vector of predictors *Xb* for the location submodel, and a vector of predictors *Xd* for the dispersion submodel.

```
capture program drop htobit
program define htobit
args lnf Xb Xd
tempvar phi mu
quietly {
gen double `phi' = exp(`Xd')
gen double `mu' = `Xb'
replace `lnf' = ln(normalden($ML_y,`mu',`phi')) if $ML_y < q
replace `lnf' = ln(1 - normal(($ML_y-`mu')/`phi')) if $ML_y == q
  }
end
```

Reading accuracy score example

```
// This is the log-likelihood function:
capture program drop htobit
program define htobit
args lnf Xb Xd
tempvar phi mu
quietly {
gen double `phi' = exp(`Xd')
gen double `mu' = `Xb'
replace `lnf' = ln(normalden($ML_y,`mu',`phi')) if $ML_y < 1
replace `lnf' = ln(1 - normal(($ML_y-`mu')/`phi')) if $ML_y == 1
  }
end
//
// Verifying that it reproduces the homoscedastic Tobit model:
ml model lf htobit (muvar: score = dys##c.ziq) (phivar: )
ml search
ml max
                                                  Number of obs   =         44
                                                  Wald chi2(3)    =      72.68
Log likelihood =  10.006221                       Prob > chi2     =     0.0000
------------------------------------------------------------------------------
       score |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
muvar        |
       1.dys |  -.3128434   .0492802    -6.35   0.000    -.4094308   -.2162559
         ziq |   .0942714   .0375864     2.51   0.012     .0206033    .1679394
             |
   dys#c.ziq |
          1  |  -.1088507   .0518848    -2.10   0.036    -.2105431   -.0071583
             |
       _cons |   .9091898   .0321065    28.32   0.000     .8462622    .9721174
-------------+----------------------------------------------------------------
phivar       |
       _cons |  -2.062086   .1354239   -15.23   0.000    -2.327512    -1.79666
------------------------------------------------------------------------------
//
// Now for the heteroscedastic model:
ml model lf htobit (muvar: score = dys##c.ziq) (phivar: dys ziq)
ml search
ml max
                                                  Number of obs   =         44
                                                  Wald chi2(3)    =      79.97
Log likelihood =  21.401029                       Prob > chi2     =     0.0000
------------------------------------------------------------------------------
       score |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
muvar        |
       1.dys |  -.3137731   .0662831    -4.73   0.000    -.4436857   -.1838606
         ziq |     .11933   .0585077     2.04   0.041     .0046569     .234003
             |
   dys#c.ziq |
          1  |  -.1411213   .0600377    -2.35   0.019    -.2587931   -.0234495
             |
       _cons |   .9078543   .0652602    13.91   0.000     .7799467    1.035762
-------------+----------------------------------------------------------------
phivar       |
         dys |  -1.670175   .3577231    -4.67   0.000    -2.371299   -.9690504
         ziq |  -.4743698   .2503781    -1.89   0.058    -.9651019    .0163623
       _cons |  -1.486012   .2373533    -6.26   0.000    -1.951216   -1.020808
------------------------------------------------------------------------------
```