SCRIPT `MOD2S1A`: MAXIMUM LIKELIHOOD ESTIMATION WITH NUMERICAL GRADIENT AND HESSIAN

Set basic R-options upfront and load all required R packages:

1. Load and describe data

This is our wage data set from script `mod1_2b`. We originally estimated a CLRM model based on these data via OLS. Now we'll do the same via MLE. Here we use `R` 's built-in `optim` function to find the MLE solution.

Since I saved the data in `R` format last time, I can call it in via the "load" command. To be specific, I saved it as: `save(data, file = "c:/Klaus/AAEC5126/R/data/wage1000.rda")`. This means that the object I saved was "data" (the actual data set), and the destination was workspace "wage1000" in my "c:/.../data/" folder. A workspace is almost like a special `R` folder within a windows folder. It can hold many objects of different type under a single name. Accordingly, the `load` command is as follows:

```
R> load("c:/klaus/AAEC5126/R/data/wage1000.rda")
```

Recall the variable definitions:

Table 1. Variable description

pos.	variable	description
1	wage	hourly wage (1995 dollars)
2	gender	(1= worker = female)
3	race	(1= worker = non-white)
4	union	(1 = worker = unionized)
5	education	years of education
6	experience	years of work experience
7	age	age in years

2. Define Components for Optimization

First, define all components that don't need to be updated during MLE optimization upfront to conserve on computing time.

```
R> names(data)[1]<-"wage"
R> names(data)[2]<-"gender"
R> names(data)[3]<-"race"
R> names(data)[4]<-"unionmember"
R> names(data)[5]<-"education"
R> names(data)[6]<-"experience"
R> names(data)[7]<-"age"
R> attach(data)
```

```
R> n<-nrow(data)
R> X<-cbind(rep(1,n),gender,race,unionmember,education,experience)
R> k<-ncol(X)
R> y<-matrix(log(wage))
R>
```

Next, we need to define the log-likelihood function to be fed into the optimization routine. This function (let's call it `CLRMllh`) will take a vector of parameters as input (call it `x`) and return the value of the sample log-likelihood. In fact, since `R` 's `optim` routine actually *minimizes* the llhf, we need to return the *negative* value of this function to achieve maximization.

Note that for the error variance σ^2, we'll send in a starting value for the std, then square this term within the opt. routine to keep it positive. In other words, we'll work with the variance during optimization, but the `optim` routine will return the standard deviation.
Note also it's important for the parameter vector to be the first function input.

```
R> CLRMllf<-function(x,y,X,n,k){
+
+ bm<-x[1:k]
+ sig2<-x[k+1]^2    #square to keep positive
+
+ llf<- -(n/2)*log(2*pi)-(n/2)*log(sig2)-((1/(2*sig2))*t((y-X%*%bm))%*%(y-X%*%bm))
+ # sample log-lh for the CLRM
+
+ # Note we could have also used R's built-in function "dnorm" to
+ # quickly evaluate the sample log-lh
+
+ return(-llf)
+ }
```

Now we need starting values (collected in vector `x0` for all of our model parameters, i.e. the slope coefficients and the error variance. We could use the OLS results, but that would make it "too easy" for the MLE optimization routine (since the OLS and MLE solutions are asymptotically equivalent). Instead, I will use "perturbed" OLS starters.

```
R> bols<-solve((t(X)) %*% X) %*% (t(X) %*% y)# compute OLS estimator
R> e<-y-X%*%bols # Get residuals.
R> SSR<-(t(e)%*%e)#sum of squared residuals - should be minimized
R> s2<-(t(e)%*%e)/(n-k) #get the regression error (estimated variance of "eps").
R> Vb<-s2[1,1]*solve((t(X))%*%X) # get the estimated variance-covariance matrix of bols
R> se=sqrt(diag(Vb)) # get the standard erros for your coefficients;
R> tval=bols/se # get your t-values.
R> x0<-0.7*c(bols,s2)
```

3. Optimization

Now we feed everything into the optimization routine and set the optimization parameters.

```
R> opt<-optim(x0,CLRMllf,gr=NULL,y,X,n,k,
+            control=list(maxit=1000,abstol=0.000000001,trace=1),hessian=TRUE)
```

The inputs are as follows:

TABLE 2. Optim inputs

element	description
x0	vector of starting values
CLRMllf	function to be optimized (user-defined)
gr=NULL	no analytical gradient function provided (required for some algorithms)
y,X,n,k	remaining arguments to be fed into user function
maxit=1000	maximum number of iterations allowed
abstol=0.0001	tolerance level on change in llf value for convergence
trace=1	show current LLH value for each 10 iterations (default) in console
hessian=TRUE	generate numerical Hessian matrix as part of output

The output (which we labeled `opt`) is a collection of the following objects:

`opt$par` vector of parameter estimates
`opt$value` value of the llf at convergence
`opt$counts` number of function and (if applicable) gradient evaluations
`opt$convergence` convergence code; "0"= successful convergence, "1"= maximum number of iterations reached, all other codes are bad news as well.
`opt$message` Any other info returned by `optim`
`opt$hessian` Symmetric Hessian matrix

Lets capture these elements. Note that the square root of the diagonal of the inverted Hessian returns the standard errors for all model parameters. Let's compare OLS and MLE results. Any discrepancy should vanish if you lower the `abstol` value in `optim`.

```
R> bm<-opt$par #this includes sigma
R> sig2<-bm[k+1]^2
R> H<-opt$hessian #already in negative form
R> sem<-sqrt(diag(solve(H)))
R> tm<- bm/sem
R> ttols<-data.frame(col1=
+   c("constant","gender","race","unionmember","education","experience"),
+                col2=bols,
+                col3=se,
+                col4=tval)
R> colnames(ttols)<-c("variable","estimate","s.e.","t")
R> ttmle<-data.frame(col1=
+   c("constant","gender","race","unionmember","education","experience","sigma"),
+                col2=bm,
+                col3=sem,
+                col4=tm)
R> colnames(ttmle)<-c("variable","estimate","s.e.","t")

R> ttolsx<- xtable(ttols,caption="OLS output")
R> digits(ttolsx)<-3   #decimals to be shown for each column
R> ttmlex<- xtable(ttmle,caption="MLE output")
```

```
R> digits(ttmlex)<-3
R> print(ttolsx,include.rownames=FALSE,
+ latex.environment="center", caption.placement="top",table.placement="!h")
```

TABLE 3. OLS output

variable	estimate	s.e.	t
constant	0.831	0.083	10.067
gender	-0.230	0.030	-7.630
race	-0.140	0.043	-3.274
unionmember	0.163	0.041	3.931
education	0.106	0.005	19.814
experience	0.013	0.001	10.055

```
R> print(ttmlex,include.rownames=FALSE,
+ latex.environment="center", caption.placement="top",table.placement="!h")
```

TABLE 4. MLE output

variable	estimate	s.e.	t
constant	0.823	0.082	9.994
gender	-0.228	0.030	-7.590
race	-0.139	0.043	-3.245
unionmember	0.165	0.041	3.996
education	0.106	0.005	19.962
experience	0.013	0.001	10.127
sigma	0.473	0.011	44.724

The estimated error variance for the OLS model is 0.225.
The estimated error variance for the MLE model is 0.224.
The value of the log-likelihood function at convergence is -670.894

```
R> proc.time()-tic
   user  system elapsed
   2.21    0.14    2.36
```