SCRIPT `MOD1_2B`: APPLICATION: WAGE REGRESSION

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

1. Load and describe data

I created a folder `c:/Klaus/AAEC5126/R/data` where I will keep all data sets to be used by `R` for this course. Please do the same within your folder environment. The data for this exercise, "wage1000.txt" was originally an Excel file. I openend it in Excel, then saved it as a "tab-delimited" text (.txt) file. I also removed the variable headers, which was necessary for Matlab, but is optional for `R` , since `R` is able to read in labels.

These are wage data (1000 obs.'s) from the US Bureau of Census Current Population survey, March 1995. The underlying population is the employed labor force, age 18-65. The variables are as follows:

(1) hourly wage
(2) female (1= worker = female)
(3) non-white (1= worker = non-white)
(4) union (1 = worker = unionized)
(5) education (years of education)
(6) experience (years of work experience)
(7) age

Once in .txt format, the data can now be loaded in with the `read.table` command. I also save it immediately in `R` format to facilitate later use.

```
R> data<- read.table('c:/Klaus/AAEC5126/R/data/wage1000.txt',
 sep="\t", header=FALSE)
R> save(data, file = "c:/Klaus/AAEC5126/R/data/wage1000.rda")
```

As you can see, I call my data set generically "data". `sep="\t"` tells `R` that the variables are tab-delimited. `header=FALSE` tells `R` that there were no variable names loaded in. `R` now treats "data" as a `data.frame` object. This format is usually best for storing and working with data.

For a few quick checks on the properties of your data, load it into `R` (copy the line above into `R` and press "Enter", so you don't have to sweave the entire file). Then type the following commands into `R` , followed by the Enter key:

`names(data)`: This returns variable names. `R` assigned generic ones since you didn't provide any.
`dim(data)`: This returns the number of rows and columns.
`str(data)`: This shows the format for each variable ("numeric" (= continuous), "integer", "string", etc)
`describe(data)`: For a quick table of descriptives for each variable

Let's assign names to each variable, "attach" the data.frame to the search path (so we can work with each variable individually), and create a table of decriptives.

```
R> names(data)[1]<-"wage"
R> names(data)[2]<-"female"
R> names(data)[3]<-"nonwhite"
R> names(data)[4]<-"unionmember"
R> names(data)[5]<-"education"
R> names(data)[6]<-"experience"
R> names(data)[7]<-"age"
R> attach(data)
R> tt<-data.frame(col1=c("wage","female","nonwhite","unionmember","education","experience","age"),
 col2=c(mean(wage),mean(female),mean(nonwhite),mean(unionmember),mean(education),
        mean(experience),mean(age)),
 col3=c(sd(wage),sd(female),sd(nonwhite),sd(unionmember),sd(education),
        sd(experience),sd(age)),
 col4=c(min(wage),min(female),min(nonwhite),min(unionmember),min(education),
        min(experience),min(age)),
 col5=c(max(wage),max(female),max(nonwhite),max(unionmember),max(education),
        max(experience),max(age)))
R> colnames(tt)<-c("variable","mean","std","min","max")
```

TABLE 1. summary statistics

variable	mean	std	min	max
wage	12.82	8.24	0.84	64.08
female	0.49	0.50	0.00	1.00
nonwhite	0.15	0.35	0.00	1.00
unionmember	0.16	0.37	0.00	1.00
education	13.18	2.86	0.00	20.00
experience	19.23	11.83	0.00	56.00
age	38.42	11.70	18.00	65.00

2. Rank Condition

Define an $\mathbf{X}$ matrix and check if it has full rank.

```
R> n<-nrow(data)
R> X<-cbind(rep(1,n),female,nonwhite,unionmember,education,experience,age)
R> k<-ncol(X)
R> rX<-qr(X)$rank
```

The rank of $\mathbf{X}$ is 6. Since $\mathbf{X}$ contains 7 columns, this is implies that it is "rank-deficient" - i.e. plagued by "perfect collinearity" amongst some variables or linear combinations thereof. This causes trouble in estimation, for example by taking the inverse. Try `solve(t(X)%*%X)` in `R` and see what happens...

Let's check the sample correlation:

```
R>  cor(X[,2:k])
```

```
             female nonwhite unionmember education
female       1.00000  0.02444   -0.073060 -0.022251
nonwhite     0.02444  1.00000    0.076912 -0.081809
unionmember -0.07306  0.07691    1.000000 -0.009443
education   -0.02225 -0.08181   -0.009443  1.000000
experience  -0.02781 -0.04055    0.178271 -0.166528
age         -0.03357 -0.06104    0.177954  0.076505
            experience      age
female        -0.02781 -0.03357
nonwhite      -0.04055 -0.06104
unionmember    0.17827  0.17795
education     -0.16653  0.07650
experience     1.00000  0.97041
age            0.97041  1.00000
```

Clearly, age and experience are highly correlated, but we would need perfect correlation of "1" for a rank violation. There must be an underlying linear relationship amongst these regressors. Idea:

```
R> int<-education + experience
R> cor(int,age)
[1] 1
```

We found the problem - age perfectly collinear to the sum of education and experience (in fact, it is computed as $age = education + experience + 6$). We need to drop one of these three variables to make $\mathbf{X}$ full rank. Let's drop age.

```
R> X<-X[,1:(k-1)]
R> k<-ncol(X)
```

3. Least Squares and the Sum of Squared Residuals

Let's run OLS and capture the results.

```
R> y<-matrix(wage)
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> tt<-data.frame(col1=c("constant","female","nonwhite","unionmember","education","experience"),
                  col2=bols,
                  col3=se,
                  col4=tval)
R> colnames(tt)<-c("variable","estimate","s.e.","t")

R> ttx<- xtable(tt,caption="OLS output")
R> digits(ttx)<-3   #decimals to be shown for each column
R> print(ttx,include.rownames=FALSE,
 latex.environment="center", caption.placement="top",table.placement="!h")
```

TABLE 2. OLS output

variable	estimate	s.e.	t
constant	-8.579	1.161	-7.388
female	-3.099	0.424	-7.313
nonwhite	-1.607	0.603	-2.664
unionmember	0.821	0.583	1.408
education	1.498	0.075	19.948
experience	0.170	0.018	9.197

The SSR for this model is 44283.64. Any other coefficient vector than the one generated by OLS produces a larger SSR value, if our underlying model assumptions are correct. You will show this in your first problem set.

Alternative model without constant, but male and female indicator:

```
R> X<-cbind(1-female,female,nonwhite,unionmember,education,experience)
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> tt<-data.frame(col1=c("male","female","nonwhite","unionmember","education","experience"),
                  col2=bols,
                  col3=se,
                  col4=tval)
R> colnames(tt)<-c("variable","estimate","s.e.","t")

R> ttx<- xtable(tt,caption="OLS output")
R> digits(ttx)<-3   #decimals to be shown for each column
R> print(ttx,include.rownames=FALSE,
 latex.environment="center", caption.placement="top",table.placement="!h")
```

TABLE 3. OLS output

variable	estimate	s.e.	t
male	-8.579	1.161	-7.388
female	-11.677	1.150	-10.152
nonwhite	-1.607	0.603	-2.664
unionmember	0.821	0.583	1.408
education	1.498	0.075	19.948
experience	0.170	0.018	9.197

The SSR for this model is 44283.64.

4. Orthogonality and Projection

Let's verify that $\mathbf{X}'\mathbf{e} = 0$ for our OLS model, and that the residuals sum to zero:

```
R> sum(e)
[1] -1.356e-11
R> t(X)%*%e
                    [,1]
             2.528e-12
female      -1.602e-11
nonwhite    -2.810e-12
unionmember -1.263e-11
education   -1.167e-10
experience  -2.072e-10
```

Close enough - these small discrepancies are due to computational rounding errors.

```
R> proc.time()-tic
   user  system elapsed
   3.27    0.19    3.46
```