STAT230 HW 9
University of California, Berkeley

Thibault Doutre, Student ID 26980469

April 7, 2016

1 Lab 13

```
## # LAB 13

# Import data
c = as.matrix(read.table("rindcor.txt"))
c = c+t(c)
diag(c) = 1
c = as.data.frame(c)
names(c) = c("OCC","RACE","NOSIB","FARM","REGN",
                   "ADOLF","REL","YCIG","FEC","ED","EG")
row.names(c) = names(c)
c = as.matrix(c)
n = 1766

# Set X, Y and Z
idx = c(11,2:8,1)
idz = c(9,2:8,1)
idy = 10

# Perform regression
Z = c[idz,idz]*n
ZZ_inv = solve(Z)
ZX = c[idz,idx]*n
ZY = c[idz,idy]*n
beta_IVLS = solve(t(ZX) %*% ZZ_inv %*% ZX, t(ZX) %*% ZZ_inv %*% ZY)
beta_IVLS
```

```
##                 [,1]
## EG      0.14727108
## RACE   -0.07652004
## NOSIB  -0.21655055
## FARM   -0.02330998
## REGN   -0.10928188
## ADOLF  -0.14559186
## REL    -0.09242772
## YCIG   -0.12776409
## OCC     0.24837554

# Standard errors
XX = c[idx,idx]*n
YX = c[idy,idx]*n
YY = c[idy,idy]*n
e2 = YY + t(beta_IVLS) %*% XX %*% beta_IVLS - 2 * (YX %*% beta_IVLS)
sigma_hat = sqrt(as.numeric(e2)/(n-9))
sigma_hat^2

## [1] 0.6745362

cov_hat = (sigma_hat^2)*solve(t(ZX) %*% ZZ_inv %*% ZX)

se_hat = sqrt(diag(cov_hat))
se_hat

##         EG        RACE       NOSIB        FARM
         REGN
## 0.09253546 0.02488338 0.02109963 0.02181783
   0.02379165
##      ADOLF         REL        YCIG         OCC
## 0.02460040 0.02102358 0.02277556 0.02468539
```

The coefficients in Rindfuss et al. are slightly different, probably because of the rounding errors in the correlation matrix.

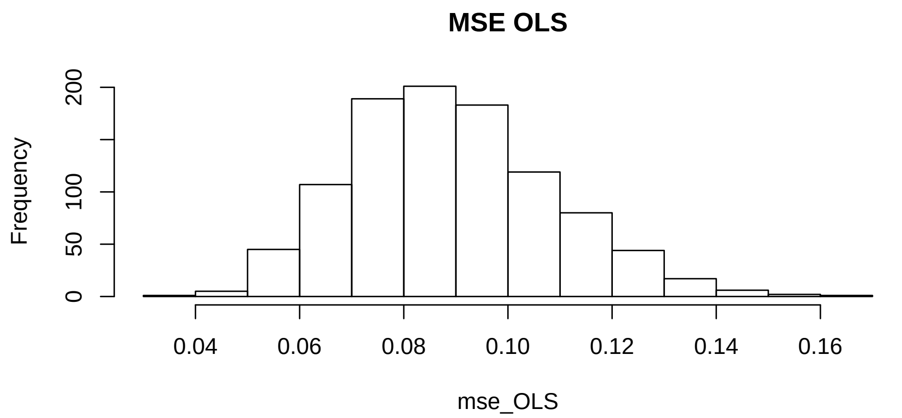
2 Simulation IVLS vs OLS

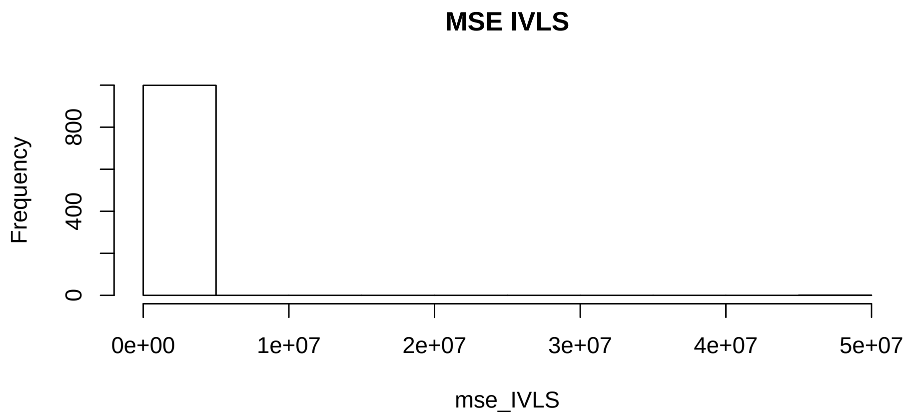
```
## # IVLS Simulation

# Function to generate delta and eps
generate_delta_eps = function(n){
  # Define parameters
  rho = 0.3
  mu1=0; s1=1; mu2=0; s2=1

  # Define X, Y and Z with the bivariate normal relationship
  X = rnorm(n)
  Z = rnorm(n)
  eps = sqrt(1-rho^2) * Z
  Y = rho * X + eps

  # Adjust means and variances
  Y = (Y-mean(Y))/sd(Y)*s2+mu2
  X = (X-mean(X))/sd(X)*s1+mu1

  # Adjust rho by transforming Y
  rho_hat = cor(X,Y)
  a = s1^4*(rho^2-1)
  b = 2*rho_hat*s1^3*s2*(rho^2-1)
  c = (rho^2-rho_hat^2)*s2^2*s1^2
  delta = b^2-4*a*c
  correction = (-b-sqrt(delta))/(2*a)
  Y=Y+correction*X

  # Adjust means and variances
  Y = (Y-mean(Y))/sd(Y)*s2+mu2
  X = (X-mean(X))/sd(X)*s1+mu1

  return(cbind(X,Y))
}

# Simulate B betas for IVLS and OLS
simulation_betas = function(n,C,B){
  beta_OLS_values = c()
```

```
  beta_IVLS_values = c()
  for (i in 1:B){
    # True beta
    beta = 1
    # Generate Z
    Z = rnorm(n)
    # Generate delta and eps
    delta_eps = generate_delta_eps(n)
    delta = delta_eps[,1]
    eps = delta_eps[,2]
    # Generate X
    X = C*Z+delta
    # Generate Y
    Y = X*beta+eps
    # Estimates of beta
    beta_OLS = solve(t(X)%*%X) %*% t(X) %*% Y
    beta_IVLS = solve(t(X)%*%Z %*% solve(t(Z)%*%Z) %*% t(Z)%*%X) %*%
      t(X)%*%Z %*% solve(t(Z)%*%Z) %*% t(Z)%*%Y
    beta_OLS_values =c(beta_OLS_values,beta_OLS)
    beta_IVLS_values=c(beta_IVLS_values,beta_IVLS)
  }
  list(OLS=beta_OLS_values,IVLS=beta_IVLS_values)
}

output_results = function(C,n,B){
  sim = simulation_betas(n,C,B)
  sim_OLS = sim$OLS
  sim_IVLS = sim$IVLS
  mse_OLS = (sim_OLS-1)^2
  mse_IVLS = (sim_IVLS-1)^2

  par(mfrow = c(2,1))
  hist(mse_OLS, main = "MSE OLS")
  hist(mse_IVLS, main = "MSE IVLS")
  par(mfrow = c(1,1))

  print('Summary OLS')
  print(summary(mse_OLS))
  print('Summary IVLS')
```

```
  print(summary(mse_IVLS))
}
```

```
B=1000
# Simumation 1
C=.1
n=10
output_results(C,n,B)
```

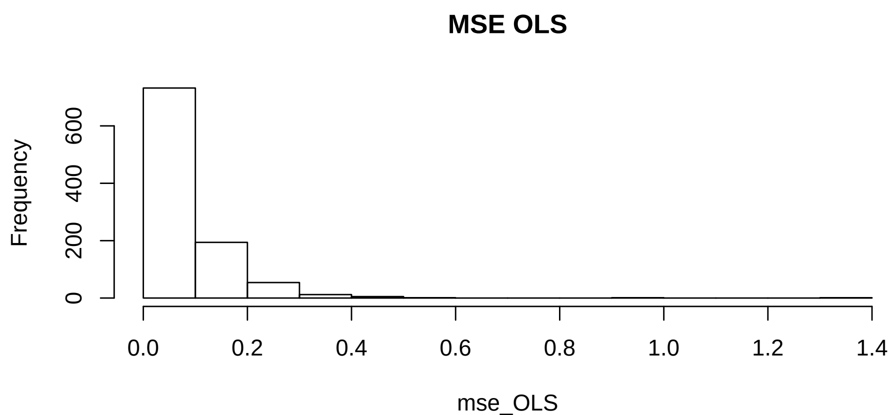


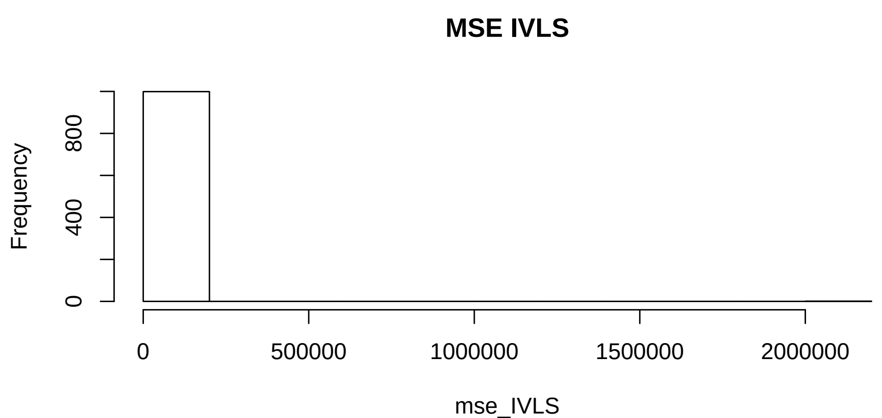


```
## [1] "Summary OLS"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## 0.03759 0.07574 0.08732 0.08924 0.10170 0.16220 
## [1] "Summary IVLS"
##     Min.  1st Qu.   Median     Mean  3rd Qu.     
   Max. 
##        0        0        1    48670        5 
   48050000 
```

OLS performs better here.

```
# Simumation 2
C=.5
n=10
output_results(C,n,B)
```

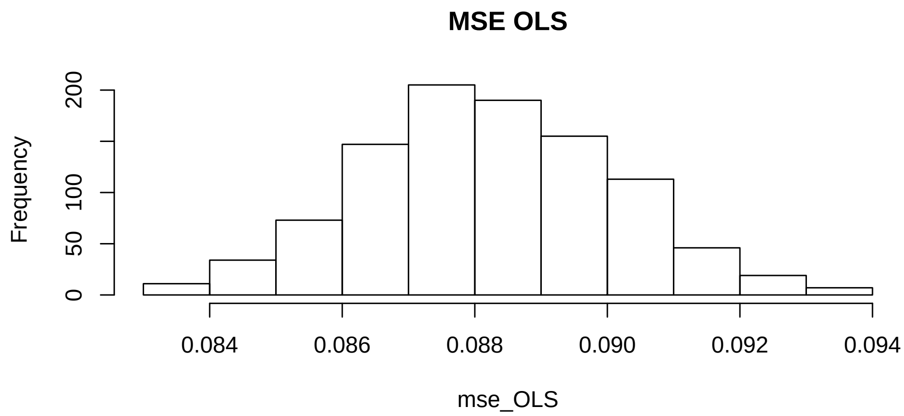


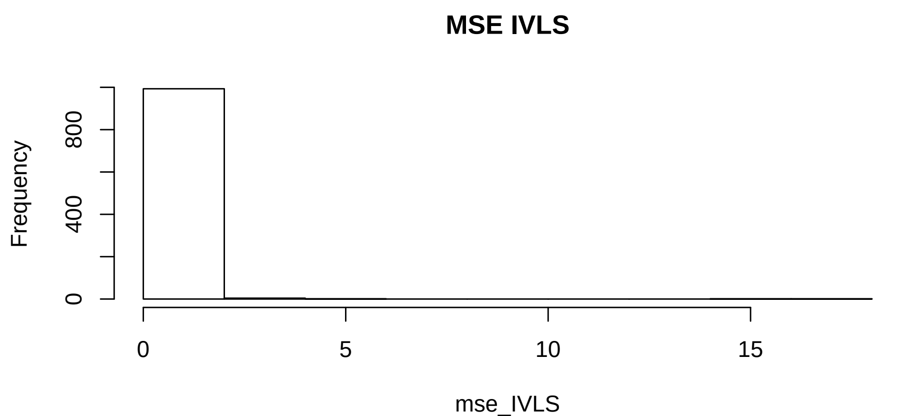



```
## [1] "Summary OLS"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.02284 0.05537 0.07795 0.10460 1.33500
## [1] "Summary IVLS"
##      Min.   1st Qu.    Median      Mean   3rd Qu.
      Max.
##       0.0       0.0       0.2    2153.0       0.9
  2028000.0
```

OLS performs better here.

```
# Simumation 3
C=.1
n=1000
output_results(C,n,B)
```



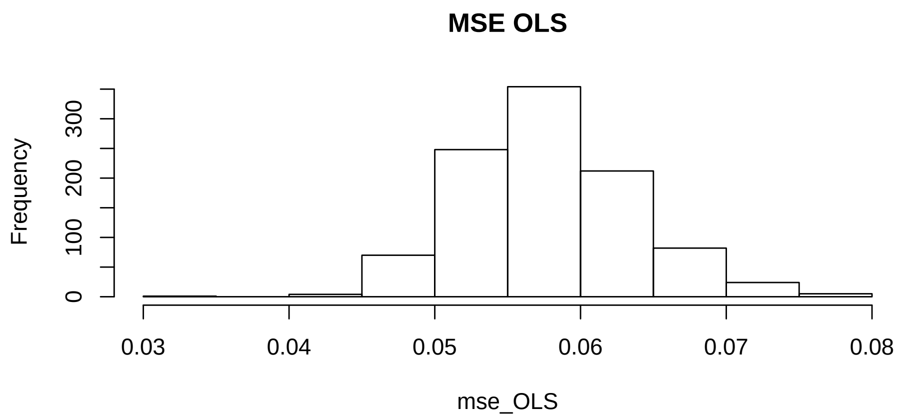


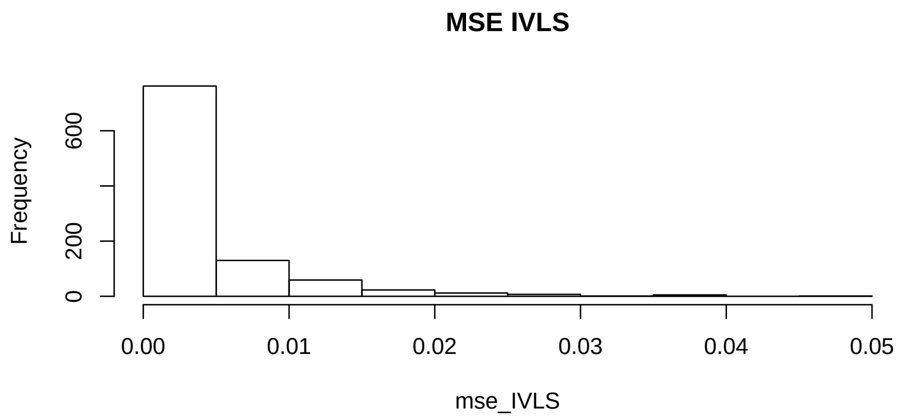

```
## [1] "Summary OLS"
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.08342 0.08690 0.08814 0.08821 0.08955 0.09387
## [1] "Summary IVLS"
##      Min.   1st Qu.    Median      Mean   3rd Qu.
   Max.
```

```
##    0.00000   0.01190   0.05777   0.17740   0.16030
   16.54000
```

OLS performs better here.

```
# Simumation 4
C=.5
n=1000
output_results(C,n,B)
```





```
## [1] "Summary OLS"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03487 0.05372 0.05750 0.05777 0.06144 0.07776
## [1] "Summary IVLS"
##      Min.   1st Qu.    Median      Mean   3rd Qu.
        Max.
## 1.000e-08 3.922e-04 1.600e-03 3.772e-03 4.813e-03
    4.592e-02
```

IVLS performs better here.

IVLS performs better as we increase both C and n. We have the following equation: (simultaneity bias)ˆ2 + OLS variance < (small-sample bias)ˆ2 + IVLS variance. Thus, sometimes OLS has a smaller mean squared error than IVLS (low C or low n).