# ARE212- FINAL

*Anaya Hall*

*May 10, 2018*

**ARE212 Take-home Final: Exploring Measurement Error.**

In this assessment we explore what happens to the least squares estimator for varying degrees of measurement error in the left hand and right hand side variables. Assume the following population model:

$y_i^* = \beta_0 + \beta_1 \cdot x_{1i} + \beta_2 \cdot x_{2i} + \epsilon_i$

Assume: $E[\epsilon|x] = 0$ and $\epsilon$ is drawn from a normal distribution (0,1)

$x_{1i}$ and $x_{2i}$ drawn from uniform normal [-200, 200]

$\beta_0 = 1$, $\beta_1 =$ -0.75, $\beta_2 = 0.75$

Set seed to 22092008.

### 1. Generate random sample

```
# Generate population data:
# Set a seed
set.seed(22092008)

# Set the population size
N <- 1e2

# Generate the data for X and E
x1 = runif(n = N, min = -200, max = 200)
x2 = runif(n = N, min = -200, max = 200)
e = rnorm(n = N, mean = 0, sd = 1)

# Generate the y variables (in anticipataion of question #2)
y = 1 -0.75*x1 + 0.75*x2 + e

# Join the data together
pop_df <- as.data.frame(cbind(y, x1, x2))
```

### 2. Generate y's and estimate b's with OLS

Generate y variables (done above). Estimate the three $\beta$ coefficients using least squares. Calculate the difference between the true $\beta$ and the estimated coefficient for each of the three $\beta$ coefficients.

First, let's load our OLS functions from the last few problem sets:

```
# Function to convert tibble, data.frame, or tbl_df to matrix
to_matrix <- function(the_df, vars) {
  # Create a matrix from variables in var
```

```r
  new_mat <- the_df %>%
    #Select the columns given in 'vars'
    select_(.dots = vars) %>%
    # Convert to matrix
    as.matrix()
  # Return 'new_mat'
  return(new_mat)
}


b_ols <- function(y, X) {
  # Calculate beta hat
  beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
  # Return beta_hat
  return(beta_hat)
}

ols <- function(data, y_data, X_data, intercept = T, hetsked = F, H0 = 0, two_tail = T, alpha = 0.
  # Function setup ----
    # Require the 'dplyr' package
    require(dplyr)

  # Create dependent and independent variable matrices ----
    # y matrix
    y <- to_matrix (the_df = data, vars = y_data)
    # X matrix
    X <- to_matrix (the_df = data, vars = X_data)
      # If 'intercept' is TRUE, then add a column of ones
      if (intercept == T) {
      X <- cbind(1,X)
      colnames(X) <- c("intercept", X_data)
      }

  # Calculate b, y_hat, and residuals ----
    b <- solve(t(X) %*% X) %*% t(X) %*% y
    y_hat <- X %*% b
    e <- y - y_hat

    # Inverse of X'X
    XX <- t(X) %*% X
    XX_inv <- solve(t(X) %*% X)

    if (hetsked == T) {
      # For each row, calculate x_i' x_i e_i^2; then sum
     sigma_hat <- lapply(X = 1:n, FUN = function(i) {
      # Define x_i
      x_i <- matrix(as.vector(X[i,]), nrow = 1)
      # Return x_i' x_i e_i^2
      return(t(x_i) %*% x_i * e[i]^2)
      }) %>% Reduce(f = "+", x = .) }
```

```r
  if (hetsked == F) sigma_hat <- XX

# Useful -----
  n <- nrow(X) # number of observations
  k <- ncol(X) # number of independent variables
  dof <- n - k # degrees of freedom
  i <- rep(1,n) # column of ones for demeaning matrix
  A <- diag(i) - (1 / n) * i %*% t(i) # demeaning matrix
  y_star <- A %*% y # for SST
  X_star <- A %*% X # for SSM
  SST <- drop(t(y_star) %*% y_star)
  SSM <- drop(t(b) %*% t(X_star) %*% X_star %*% b)
  SSR <- drop(t(e) %*% e)

# Measures of fit and estimated variance ----
  R2uc <- drop((t(y_hat) %*% y_hat)/(t(y) %*% y)) # Uncentered R^2
  R2 <- 1 - SSR/SST # Uncentered R^2
  R2adj <- 1 - (n-1)/dof * (1 - R2) # Adjusted R^2
  AIC <- log(SSR/n) + 2*k/n # AIC
  SIC <- log(SSR/n) + k/n*log(n) # SIC
  s2 <- SSR/dof # s^2

# Measures of fit table ----
  mof_table_df <- data.frame(R2uc, R2, R2adj, SIC, AIC, SSR, s2)
  mof_table_col_names <- c("$R^2_\\text{uc}$", "$R^2$",
                           "$R^2_\\text{adj}$",
                           "SIC", "AIC", "SSR", "$s^2$")
  mof_table <-  mof_table_df %>% knitr::kable(
    row.names = F,
    col.names = mof_table_col_names,
    format.args = list(scientific = F, digits = 4),
    booktabs = T,
    escape = F
  )

# t-test----
  # Standard error
  se <- sqrt(s2 * diag(XX_inv %*% sigma_hat %*% XX_inv)) # Vector of _t_ statistics
  # Vector of _t_ statistics
  t_stats <- (b - H0) / se
  # Calculate the p-values
  if (two_tail == T) {
  p_values <- pt(q = abs(t_stats), df = dof, lower.tail = F) * 2
  } else {
    p_values <- pt(q = abs(t_stats), df = dof, lower.tail = F)
  }
  # Do we (fail to) reject?
  reject <- ifelse(p_values < alpha, reject <- "Reject", reject <- "Fail to Reject")
```

```
    # Nice table (data.frame) of results
    ttest_df <- data.frame(
      # The rows have the coef. names
      effect = rownames(b),
      # Estimated coefficients
      coef = as.vector(b) %>% round(3),
      # Standard errors
      std_error = as.vector(se) %>% round(4),
      # t statistics
      t_stat = as.vector(t_stats) %>% round(3),
      # p-values
      p_value = as.vector(p_values) %>% round(4),
      # reject null?
      significance = as.character(reject)
      )

    ttest_table <-  ttest_df %>% knitr::kable(
      col.names = c("", "Coef.", "S.E.", "t Stat", "p-Value", "Decision"),
      booktabs = T,
      format.args = list(scientific = F),
      escape = F,
      caption = "OLS Results"
    )

  # Data frame for exporting for y, y_hat, X, and e vectors ----
    export_df <- data.frame(y, y_hat, e, X) %>% tbl_df()
    colnames(export_df) <- c("y","y_hat","e",colnames(X))

  # Return ----
    return(list(n=n, dof=dof, b=b, se=se, vars=export_df, R2uc=R2uc,R2=R2,
                R2adj=R2adj, AIC=AIC, SIC=SIC, s2=s2, SST=SST, SSR=SSR,
                mof_table=mof_table, ttest=ttest_table))
}
```

Now ready to estimate!

```
X_mat <- c("x1", "x2")
q2 <- ols(data = pop_df, y_data = "y", X_data = X_mat, intercept = T)

q2$b %>% knitr::kable()
```

| | y |
|---|---|
| intercept | 0.9074893 |
| x1 | -0.7498911 |
| x2 | 0.7498942 |

```
diff_mat <- data_frame(
  diff0 = (q2$b[1] - 1),
  diff1 = (q2$b[2] - (-0.75)),
```

```
  diff2 = (q2$b[3] - (0.75)))

diff_mat %>% knitr::kable()
```

| diff0 | diff1 | diff2 |
|---|---|---|
| -0.0925107 | 0.0001089 | -0.0001058 |

### 3. Introducing MEASUREMENT ERROR on y

Now assume that $y_i$ is measured with error! It is in fact $y_i + r_i$, where $r_i$ is drawn from a random normal distribution with mean zero and variance $\sigma^2$. Using the $y_i$ from the previous step, add the measurement error $r_i$ to them and use this measured with error dependent variable as your outcome.

Estimate the three $\beta$ coefficients again using least squares. Do this for $\sigma^2 = [1\ 10\ 100]$.

```
sigma2 <- c(1,10,100)

# for (s in sigma2) {
#   # Generate r_i
#   r <- rnorm(N, 0, s)
#
#   # Add new error to y
#   pop_df %<>% mutate(y_err = y + r)
#
#   # Rerun OLS
#   ols(pop_df, "y_err", X_mat)$b %>% knitr::kable()
# }

# Generate r_i
r <- rnorm(N, 0, 1)
# Add new error to y
pop_df %<>% mutate(y_err = y + r)
# Rerun OLS
ols(pop_df, "y_err", X_mat)$b %>% knitr::kable()
```

| | y_err |
|---|---|
| intercept | 1.0596229 |
| x1 | -0.7495957 |
| x2 | 0.7487727 |

```
# Generate r_i
r <- rnorm(N, 0, 10)
# Add new error to y
pop_df %<>% mutate(y_err = y + r)
# Rerun OLS
ols(pop_df, "y_err", X_mat)$b %>% knitr::kable()
```

| | y_err |
|---|---|
| intercept | 2.2691807 |
| x1 | -0.7457138 |
| x2 | 0.7575671 |

```
# Generate r_i
r <- rnorm(N, 0, 100)
# Add new error to y
pop_df %<>% mutate(y_err = y + r)
# Rerun OLS
ols(pop_df, "y_err", X_mat)$b %>% knitr::kable()
```

| | y_err |
|---|---|
| intercept | 5.6064263 |
| x1 | -0.8522662 |
| x2 | 0.7731456 |

Worse coefficients!! ### 4. Measurement Error on x2 Now assume that your $x_{2i}$ is measured with error. You observe $x_2^* = x_{2i} + r_i$, where $r_i$ is drawn from a random normal distribution with mean zero and variance $\sigma^2$ (Just reuse the $r_i$ from the previous step.). Use the $y_i$ from the first step (the one measured without error) and replace $x_{2i}$ with $x_2^*$ in your estimation. Estimate the three $\beta$ coefficients using least squares on your data. Do this for $\sigma^2 = [1\ 10\ 100]$.

**5. Non-symmetric measurement error: POSITIVE**

Repeat step 4 exactly, only now assume that your measurement error is not symmetric, but always positive. Simply take the absolute value of $r_i$ (again using the $r_i$ from above) before generating your $x_2^*$. Only do this for $\sigma^2 = 100$.

**6. Non-symmetric measurement errror: NEGATIVE**

Repeat step 4 exactly, only now assume that your measurement error is not symmetric, but always negative. Simply take the absolute value of $r_i$ and multiply it times (-1) before generating your $x_2^*$ (again using the $r_i$ from above). Only do this for $\sigma^2 = 100$.

**SCALE UP!**

Repeat the above steps 10,000 times (set the seed only the first time, not each time) and calculate the bias for $\beta_0$, $\beta_1$ and $\beta_2$ for each setting and fill in the table below. What have you learned from this exercise that you did not know before? Was there anything surprising?

# simple table creation here

| Tables | Are | Are | Cool |
|---|---|---|---|
| col 3 is | right-aligned | right-aligned | $1600 |
| col 2 is | centered | centered | $12 |

| Tables | Are | Are | Cool |
|---|---|---|---|
| zebra stripes | are neat | are neat | $1 |