Problem Set #2

Anaya Hall

Part 1: Theory

Part 2: Applied

Returns to Scale in Electricity Supply

```
# Function to convert tibble, data.frame, or tbl_df to matrix
to_matrix <- function(the_df, vars) {
  # Create a matrix from variables in var
  new_mat <- the_df %>%
    # Select the columns given in 'vars'
    select_(.dots = vars) %>%
    # Convert to matrix
    as.matrix()
  # Return 'new_mat'
  return(new_mat)
}
# Function for OLS coefficient estimates
b_ols <- function(data, y_var, X_vars, intercept = T) {
  # Require the 'dplyr' package
  require(dplyr)
  # Create the y matrix
  y <- to_matrix(the_df = data, vars = y_var)
  # Create the X matrix
  X <- to_matrix(the_df = data, vars = X_vars)
  # If 'intercept' is TRUE, then add a column of ones
  if (intercept == T) {
    # Bind a column of ones to X
    X <- cbind(1, X)
    # Name the new column "intercept"
    colnames(X) <- c("intercept", X_vars)
  }
  # Calculate beta hat
  beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
  # Return beta_hat
  return(beta_hat)
}
# Function that demeans the columns of Z
demeaner <- function(N) {
  # Create an N-by-1 column of 1s
  i <- matrix(data = 1, nrow = N)
  # Create the demeaning matrix
  A <- diag(N) - (1/N) * i %*% t(i)
  # Return A
  return(A)
}
```

```r
# Function to return OLS residuals
resid_ols <- function(data, y_var, X_vars, intercept = T) {
  # Require the 'dplyr' package
  require(dplyr)
  # Create the y matrix
  y <- to_matrix(the_df = data, vars = y_var)
  # Create the X matrix
  X <- to_matrix(the_df = data, vars = X_vars)
  # If 'intercept' is TRUE, then add a column of ones
  if (intercept == T) {
    # Bind a column of ones to X
    X <- cbind(1, X)
    # Name the new column "intercept"
    colnames(X) <- c("intercept", X_vars)
  }
  # Calculate the sample size, n
  n <- nrow(X)
  # Calculate the residuals
  resids <- (diag(n) - X %*% solve(t(X) %*% X) %*% t(X)) %*% y
  # Return 'resids'
  return(resids)
}
# Create function for r-squared, aic, and sic
r2_ic <- function(data, y_var, X_vars, intercept = T) {
  # Create y and X matrices
  y <- to_matrix(data, vars = y_var)
  X <- to_matrix(data, vars = X_vars)
  # Add intercept column to X
  X <- cbind(1, X)
  # Find N and K (dimensions of X)
  N <- nrow(X)
  K <- ncol(X)
  # Calculate the OLS residuals
  e <- resid_ols(data, y_var, X_vars, intercept)
  # Calculate sum of squared errors
  sse <- t(e) %*% e
  # Calculate the y_star (demeaned y)
  y_star <- demeaner(N) %*% y
  # Calculate r-squared values
  r2_uc  <- 1 - sse / (t(y) %*% y)
  r2     <- 1 - sse / (t(y_star) %*% y_star)
  r2_adj <- 1 - (N-1) / (N-K) * (1 - r2)
  # Calculate sic and aic
  sic <- log(sse / N) + K / N * log(N)
  aic <- log(sse / N) + 2 * K / N
  # Calculate s2
  s2 <- sse / (N - K)
  # Create a data.frame
  tmp_df <- data.frame(r2_uc, r2, r2_adj, sic, aic, sse, s2)
  names(tmp_df) <- c("r2_uc", "r2", "r2_adj", "sic", "aic",
    "sse", "s2")
  return(tmp_df)
}
```

```r
# Function to export a nice table
r2_table <- function(org_df, scientific = T) {
  # Column names for the output of r2_ic
  r2_ic_names <- c("$R^2_\\text{uc}$", "$R^2$", "$R^2_\\text{adj}$",
    "SIC", "AIC", "SSE", "$s^2$")
  new_table <- org_df %>% knitr::kable(
    row.names = F,
    col.names = r2_ic_names,
    digits = 4,
    format.args = list(scientific = scientific)
    )
  return(new_table)
}
```

Including Plots