Problem Set #2

Anaya Hall & Christian Miller

Part 1: Theory

Part 2: Applied: Returns to Scale in Electricity Supply

First, load all OLS functions created in Problem Set #1.

```r
# THIS IS ED'S FUNCTION --> REPLACE WITH CGM's

# 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)
}
```

Question 1:

Read the data into R. Print out the data. Read it. Plot the series and make sure your data are read in correctly. Make sure your data are sorted by size (kwh). [Hint: Check for obvious typos in the data and if you find any fix them!]

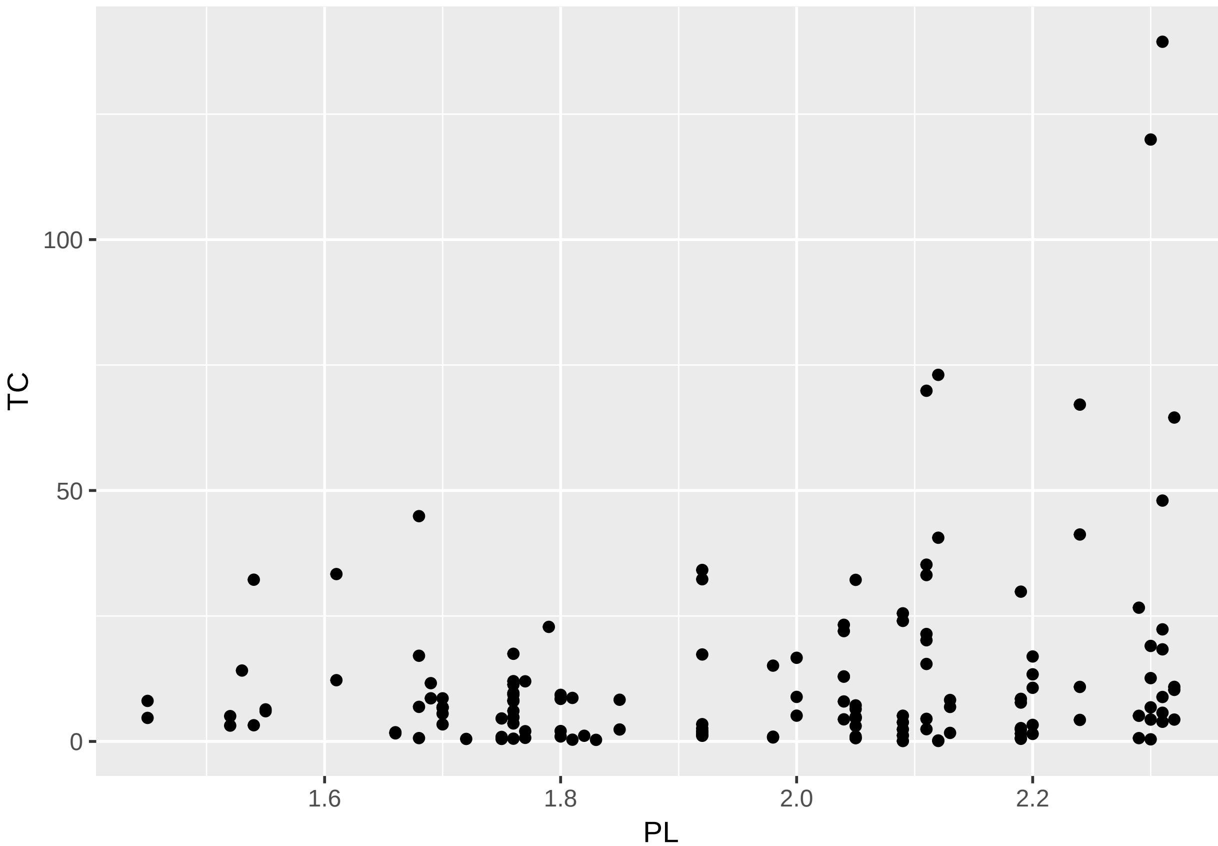
```r
nerlove <- readxl::read_excel("nerlove.xls", col_names=TRUE)

# Fix typo in 13th row (missing a decimal!)
# DO THIS MORE ELEGANTLY!
nerlove[13, "PL"] <- 1.81

nerlove
```

```
## # A tibble: 145 x 5
##        TC     Q    PL    PF    PK
##     <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 0.0820  2.00  2.09  17.9   183
##  2 0.661   3.00  2.05  35.1   174
##  3 0.990   4.00  2.05  35.1   171
##  4 0.315   4.00  1.83  32.2   166
##  5 0.197   5.00  2.12  28.6   233
##  6 0.0980  9.00  2.12  28.6   195
##  7 0.949  11.0   1.98  35.5   206
##  8 0.675  13.0   2.05  35.1   150
##  9 0.525  13.0   2.19  29.1   155
## 10 0.501  22.0   1.72  15.0   188
## # ... with 135 more rows
```

```r
nerlove %>%
  ggplot(aes(x=PL, y=TC)) + geom_point()
```



Question 2:

Replicate regression I (page 176) in the paper.

Looks like this-ish: log(TC - PF) = B0 + B1(Q) + B2(Log(PL)0-log(PF)) + B2(log(PK)-log(PF))