Problem Set #2

Anaya Hall & Christian Miller

Part 1: Theory

Part 2: Applied: Returns to Scale in Electricity Supply

First, load our OLS function created in Problem Set #1. We're including a built in t-test this time around.

```
ols <- function(data, y_data, X_data, intercept = T, H0 = 0, two_tail = T, alpha = 0.05) {
  # Function setup ----
    # Require the 'dplyr' package
    require(dplyr)
    # 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)
    }

  # 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

  # 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)
```

```r
  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 <- as.vector(sqrt(s2 * diag(solve(t(X) %*% X))))
  # 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(3),
    # 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)
    )
```

```r
  ttest_table <-  ttest_df %>% knitr::kable(
    booktabs = T,
    format.args = list(scientific = F),
    escape = F
  )

# 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, 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))
}
```

We'll also need functions to an F-test for this Problem Set.

```r
# joint test function (from Ed's notes)
F_test <- function(data, y_var, X_vars) {
  # Turn data into matrices
  y <- to_matrix(data, y_var)
  X <- to_matrix(data, X_vars)
  # Add intercept
  X <- cbind(1, X)
  # Name the new column "intercept"
  colnames(X) <- c("intercept", X_vars)
  # Calculate n and k for degrees of freedom
  n <- nrow(X)
  k <- ncol(X)
  # J is k-1
  J <- k - 1
  # Create the R matrix: bind a column of zeros
  # onto a J-by-J identity matrix
  R <- cbind(0, diag(J))

  # Estimate coefficients
  b <- b_ols(data, y_var, X_vars)
  # Calculate OLS residuals
  e <- y - X %*% b
  # Calculate s^2
  s2 <- (t(e) %*% e) / (n-k)
  # Force s2 to numeric
  s2 %<>% as.numeric()

  # Create the inner matrix R(X'X)^(-1)R'
  RXXR <- R %*% solve(t(X) %*% X) %*% t(R)
  # Calculate the F stat
  f_stat <- t(R %*% b) %*% solve(RXXR) %*% (R %*% b) / J / s2
  # Calculate the p-value
  p_value <- pf(q = f_stat, df1 = J, df2 = n-k, lower.tail = F)
  # Create a data.frame of the f stat. and p-value
  results <- data.frame(
    f_stat = f_stat %>% as.vector(),
```

```
    p_value = p_value %>% as.vector())
  return(results)

}
```

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!]

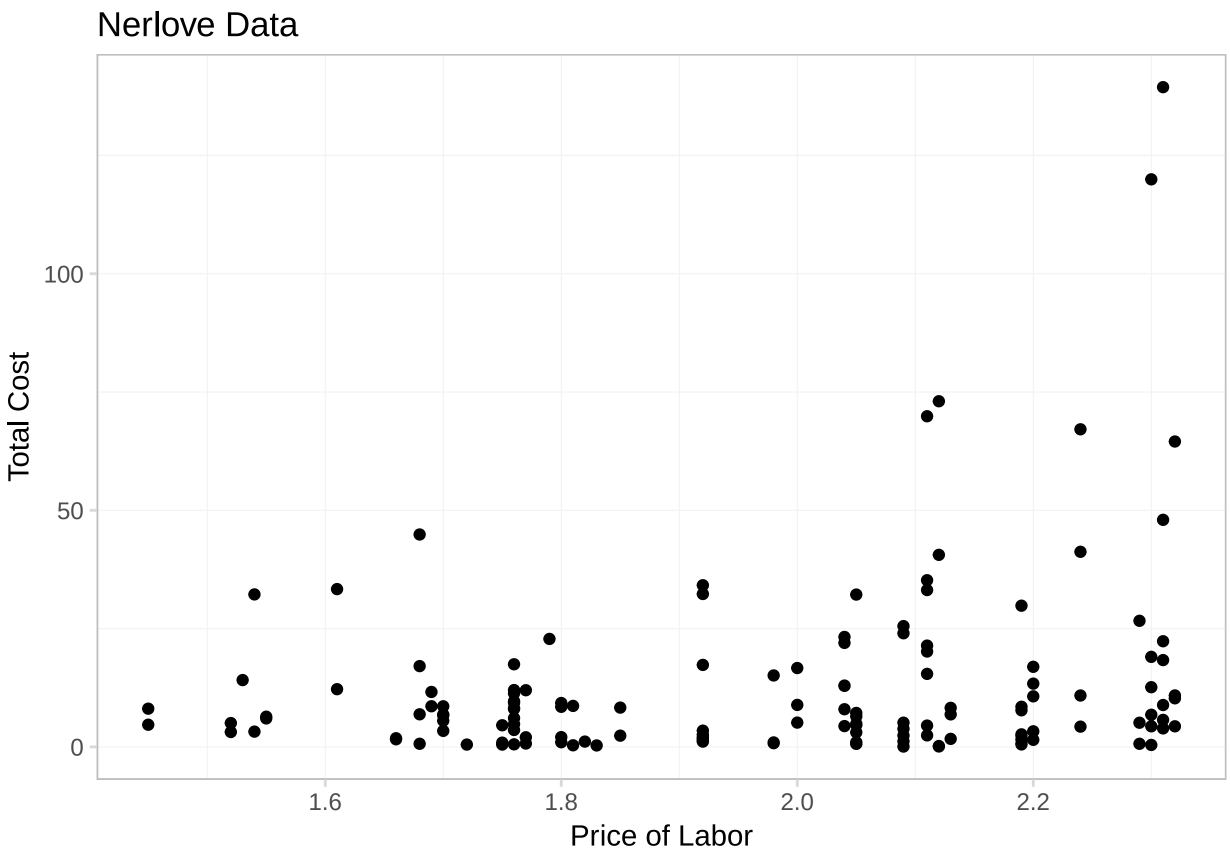
```
nerlove <- readxl::read_excel("nerlove.xls", col_names=T)

# 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.  2.09  17.9  183.
##  2 0.661     3.  2.05  35.1  174.
##  3 0.990     4.  2.05  35.1  171.
##  4 0.315     4.  1.83  32.2  166.
##  5 0.197     5.  2.12  28.6  233.
##  6 0.0980    9.  2.12  28.6  195.
##  7 0.949    11.  1.98  35.5  206.
##  8 0.675    13.  2.05  35.1  150.
##  9 0.525    13.  2.19  29.1  155.
## 10 0.501    22.  1.72  15.0  188.
## # ... with 135 more rows
```

```
ggplot(nerlove, aes(x=PL, y=TC)) +
  geom_point() +
  labs(title="Nerlove Data", x="Price of Labor", y="Total Cost") +
  theme_are
```




```
### plot rest of independent vars too.
```

Question 2:

Replicate regression I (page 176) in the paper.

Looks like this-ish: $log(TC) - log(P_F) = \beta_0 + \beta_1 Q + \beta_2 \Big(log(P_L) - log(P_F)\Big) + \beta_3 \Big(log(P_K) - log(P_F)\Big)$

Where:
$\beta_1 = \frac{1}{r}$, $\beta_2 = \frac{a_L}{r}$, $\beta_3 = \frac{a_K}{r}$ P_L = wage rate,P_K = "price" of capital, P_F = price of fuel
TC = total production cost, Q = output (measured in kWh)

```
# Create log variables
nerlove %<>% mutate(
  TClog = log10(TC),
  Qlog = log10(Q),
  PLlog = log10(PL),
  PKlog = log10(PK),
  PFlog = log10(PF)
)

# Create PF scaled variables
nerlove %<>% mutate(
  TCscaled = TClog - PFlog,
  PLscaled = PLlog - PFlog,
  PKscaled = PKlog - PFlog
)
```

```r
reg_I <- ols(data = nerlove,y_data = "TClog",
             X_data = c("Qlog","PLscaled","PKscaled"),
             intercept = T, H0 = 0, alpha = 0.05)

reg_I$ttest
```

effect	coef	std_error	t_stat	p_value	significance
intercept	-0.621	0.386	-1.608	0.1100	Fail to Reject
Qlog	0.720	0.018	41.099	0.0000	Reject
PLscaled	0.154	0.206	0.751	0.4538	Fail to Reject
PKscaled	-0.603	0.192	-3.145	0.0020	Reject

```r
reg_I$mof_table
```

R^2_{uc}	R^2	R^2_{adj}	SIC	AIC	SSR	s^2
0.9698	0.925	0.9234	-3.424	-3.506	4.119	0.02921