```
#Ethan Grant
#pset2

#problem 1
#i
setwd('C:/Users/Ethan/Desktop/Columbia/Senior_Fall/Advanced_Econometrics/pset_2')
data = read.table('nerlov.dat', header=FALSE)
head(data)

#ii
#computes new vars by taking log of columns
log_TC = log(data[,1])
intercept = rep(1, length(log_TC))
log_Q = log(data[,2])
log_PL = log(data[,3])
log_PK = log(data[,5])
log_PF = log(data[,4])
n = length(log_TC)
k = ncol(X)

#iii
#binds variables into matrix
X = cbind(intercept, log_Q, log_PL, log_PK, log_PF)
beta_hat = solve(t(X)%*%X)%*%t(X)%*%log_TC #solves for beta_hat
print(beta_hat)
X0 = cbind(log_Q, log_PL, log_PK, log_PF)
#finds rsquareds
log_TC_hat = X%*%beta_hat
#residual maker
M1 = diag(n) - intercept%*%solve(t(intercept)%*%intercept)%*%t(intercept)
beta_demean = solve(t(X0)%*%M1%*%X0)%*%t(X0)%*%(M1%*%log_TC)

#computes demeand y hat
y_hat_demean = (M1%*%X0)%*%beta_demean
log_TC_demean = M1%*%log_TC
#computes r squared and then adjusted r squared
centered_r = t(y_hat_demean)%*%(y_hat_demean)/(t(log_TC_demean)%*%(log_TC_demean))
adjusted_r = 1-(1-centered_r)*(n-1)/(n-k-1)
print(centered_r)
print(adjusted_r)


#iv
residual = log_TC - X%*%beta_hat
#calculate sample varriance unbiased
sigma_2 = sum(residual^2)/(n-k)
#calculates v homoskedastic and v white
v_homo = sigma_2*(solve(t(X)%*%X/n))
print(v_homo)
#creates empty sigma matrix and fills with residuals
residual_sqrd = residual^2
sigma = matrix(0, nrow=n, ncol=n)
for (i in 1:n) {
  sigma[i,i] <- residual_sqrd[i]
}
v_white = (solve(t(X)%*%X/n)%*%t(X)%*%sigma%*%X%*%solve(t(X)%*%X/n))*(1/n)
print(v_white)

#viii
#computing the Wald stat
R = matrix(c(0, 0,1,1,1), nrow=1,ncol=k)
q= c(1)
z = R%*%beta_hat-q
sigma = solve(R%*%v_white%*%t(R))
wald = t(z)*sigma*z
print(wald)

#ix
if(wald<=qchisq(0.95, df=1)){
  print('dont reject null')
}

#x
simple_X = matrix(c(constant, X[,2], X[,3], X[,4]-X[,3], X[,5]-X[,3]), n)
betas_simple = solve(t(simple_X)%*%simple_X)%*%t(simple_X)%*%log_TC
residual_simple = log_TC-simple_X%*%betas_simple
sigma_simple = matrix(0, nrow=n, ncol=n)
for(i in 1:n){
  sigma_simple[i,i] <- residual_simple[i]^2
}
v_white_simple = (solve(t(simple_X)%*%simple_X/n)%*%t(simple_X)%*%sigma_simple%*%simple_X%*%solve(t(simple_X)%*%simple_X/n))*(1/n)
print(v_white_simple)

#xi
t_test = (betas_simple[3]-q)/sqrt(v_white_simple[3,3])
print(t_test)
if (abs(t_test) < abs(qt(0.05/2, df=n-k))) {
  print('dont reject null')
}

#xii
#test for greater than
if (t_test < qt(0.95, df=n-k)) {
  print('dont reject null')
}

#test for less than
if (t_test > qt(0.05, df=n-k)) {
  print('dont reject null')
}

#xiii
c = beta_hat[3]*beta_hat[4]*beta_hat[5]
q = c(0)
G = matrix(c(0, 0, beta_hat[4]*beta_hat[5], beta_hat[3]*beta_hat[5], beta_hat[3]*beta_hat[4]), 1, k)
H_func = c - q
```

```
W_func = t(H_func)%*%solve(G%*%v_white%*%t(G))%*%H_func
print(W_func)

# xiv.
if (W_func < qchisq(0.95, df=1)) {
  print('Don\'t reject null hypothesis.')
}



#Problem 2
#iv

white_error_demo <- function(x_pct, B, N){

  #creates data and error
  x = matrix(rnorm(N*B, 0, 1), nrow=B, ncol=N)
  u = matrix(rnorm(N*B, 0, 1), nrow=B, ncol=N)

  #generates epsilon and y
  epsilon = exp(x_pct*x)*u
  y = x+epsilon

  #build structs to hold data
  beta = numeric(B)
  t_homo = numeric(B)
  t_white = numeric(B)

  #compute betas and test for reach b in B
  for (b in 1:length(B)){
    beta[b] = solve(t(x[b,])%*%x[b,])%*%x[b,]%*%y[b,]

    #get homoskedastic SE
    Q = t(x[b,])%*%x[b,]*1/N
    v_homo = exp(8)*solve(Q)

    #build white SE
    sigma = (1/N*sum((epsilon[b,]*x[b,])^2))
    v_white = solve(Q)%*%sigma%*%solve(Q)

  }
  t_white = (beta-1)/sqrt(v_white)
  t_homo = (beta-1)/sqrt(v_homo)
  num_reject_white = length(t_white[abs(t_white) > abs(qt(0.05/2, df=N-1))])
  num_reject_homo = length(t_homo[abs(t_homo) > abs(qt(0.05/2, df=N-1))])

  #report results
  cat(cat("white rejection: ", num_reject_white/B), "\n")
  cat(cat("homo rejection: ", num_reject_homo/B), "\n")
}

white_error_demo(1, 1000, 100)
white_error_demo(2, 1000, 1000)
white_error_demo(.1, 1000, 1000)
white_error_demo(0, 1000, 1000)
```