Replication Code

June 21, 2005

Here, we provide code to reproduce the results in the paper. The R code below will run if pasted into the R command line. Results at present will print to the screen since sending the result to a text file requires a path that is typically computer specific. Be advised that the program can take several hours to run on a standard computer depending on processor speed. Also note that due to a mistake on our part the seeds here may not exactly match those used in the text. This doesn't change the results much, but does mean that they don't match exactly.

1 Model Comparison

1.1 Varying α Monte Carlo

```
#GLS Routine
cochrane.orcutt <- function(mod, ...){
     UseMethod("cochrane.orcott")
     }


cochrane.orcutt.lm <- function(mod){
     X <- model.matrix(mod)
     y <- model.response(model.frame(mod))
     e <- residuals(mod)
     n <- length(e)
     names <- colnames(X)
     rho <- sum(e[1:(n-1)]*e[2:n])/sum(e^2)
     y <- y[2:n] - rho * y[1:(n-1)]
     X <- X[2:n,] - rho * X[1:(n-1),]
     mod <- lm(y ~ X - 1)
     result <- list()
     result$coefficients <- coef(mod)
     names(result$coefficients) <- names
     summary <- summary(mod, corr = F)
     result$cov <- (summary$sigma^2) * summary$cov.unscaled
     dimnames(result$cov) <- list(names, names)
     result$sigma <- summary$sigma
     result$rho <- rho
```

```
     class(result) <- 'cochrane.orcutt'
     result
     }
prais.winsten <- function(mod, ...){
     UseMethod("prais.winsten")
     }


prais.winsten.lm <- function(mod){
     X <- model.matrix(mod)
     y <- model.response(model.frame(mod))
     e <- residuals(mod)
     n <- length(e)
     names <- colnames(X)
     rho <- sum(e[1:(n-1)]*e[2:n])/sum(e^2)
     y <- c(y[1] * (1 - rho^2)^0.5, y[2:n] - rho * y[1:(n-1)])
     X <- rbind(X[1,] * (1 - rho^2)^0.5, X[2:n,] - rho * X[1:(n-1),])
     mod <- lm(y ~ X - 1)
     result <- list()
     result$coefficients <- coef(mod)
     names(result$coefficients) <- names
     summary <- summary(mod, corr = F)
     result$cov <- (summary$sigma^2) * summary$cov.unscaled
     dimnames(result$cov) <- list(names, names)
     result$sigma <- summary$sigma
     result$rho <- rho
     class(result) <- 'prais.winsten'
     result
     }

####################Monte Carlo Starts Here####################

#Number of Iterations
k <- 1000
# Number of Cases
n <- 100
#Autoregressive Parameters
rho1 <- 0.95
rho2 <- 0.75
set.seed(831047)

#Beta Value to Vary Model Fit
b <- 0.50
#Alpha value
a <- list(0.0, 0.10, 0.20, 0.50)
```

```
for (a in a){

cat("#################Parameter Values############################\n")
cat("Beta: ", b, " Alpha: ", a, "\n")
cat("Rho_1: ", rho1, " Rho_2: ", rho2, "\n")
cat("############################################################\n")
long.beta <- b/(1-a)


#Lag Operator
tslag<-function(y,d=1){
n <- length(y)
c(rep(NA,d),y)[1:n]
}

#Storage Matrices
beta.na1 <- matrix(NA,k,1)
stder.na1 <- matrix(NA,k,1)
beta.na2 <- matrix(NA,k,1)
stder.na2 <- matrix(NA,k,1)
beta.na3 <- matrix(NA,k,1)
stder.na3 <- matrix(NA,k,1)
arma.beta <- matrix(NA,k,1)
arma.ar <- matrix(NA,k,1)
arma.cvg <- matrix(NA,k,1)
corc.beta <- matrix(NA,k,1)
corc.rho <- matrix(NA,k,1)
beta.ols <- matrix(NA,k,1)
prais.beta <- matrix(NA,k,1)
prais.rho <- matrix(NA,k,1)
dldv.alpha1 <- matrix(NA,k,1)
dldv.alpha2 <- matrix(NA,k,1)
dldv.beta <- matrix(NA,k,1)
long.est <- matrix(NA,k,1)
dldv.long.est <- matrix(NA,k,1)

j <- 1
#Start Monte Carlo Here
for (j in j:k){
# Generate AR(1) process
#X DGP
e <- rnorm(n)
x <- double(n)
x[1] <- rnorm(1)
for(i in 2:n) {
```

```
x[i] <- rho1 * x[i-1] + e[i]
}

#Y DGP Error Process
e1 <- rnorm(n)
e2 <- double(n)
e2[1] <- rnorm(1)
for(i in 2:n) {
e2[i] <- rho2 * e2[i-1] + e1[i]
}

# Y DGP
y <- double(n)
y[1] <- rnorm(1)
for(i in 2:n){
y[i] <- a * y[i-1] + b * x[i] + e2[i]
}

#Data frame
lagy <- tslag(y)
simldv <- data.frame(y, x, lagy)
attach(simldv)

#Data Frame 2
lag2y <- tslag(lagy)
simdldv <- data.frame(y,x,lagy,lag2y)

# estimate LDV w/ OLS
ldv <- summary(lm(y ~ lagy + x, data = simldv))

beta.na1[j] <-ldv$coef[1,1]
stder.na1[j] <-ldv$coef[1,2]
beta.na2[j] <-ldv$coef[2,1]
stder.na2[j] <-ldv$coef[2,2]
beta.na3[j] <-ldv$coef[3,1]
stder.na3[j] <-ldv$coef[3,2]
long.est[j] <- ldv$coef[3,1]/(1-ldv$coef[2,1])

#Estimate with Two Lags
dldv <- summary(lm(y~lagy + lag2y + x, simdldv))
dldv.alpha1[j] <-dldv$coef[2,1]
dldv.alpha2[j] <-dldv$coef[3,1]
dldv.beta[j] <-dldv$coef[4,1]
dldv.long.est[j] <- dldv$coef[4,1]/(1-dldv$coef[2,1])
```

```
#ARMA via MLE
arma <- arima(y, order = c(1,0,0), xreg = x, method = c("ML"))
d <- as.matrix(coef(arma))
arma.beta[j] <- d[3,1]
arma.ar[j] <- d[1,1]
arma.cvg[j] <- arma$code
rm(d)

#Corc, Prais and OLS
start <- lm(y ~ x, data = simldv)
p <- as.matrix(coef(start))
beta.ols[j] <- p[2,1]
corc <- cochrane.orcutt.lm(start)
m <- as.matrix(coef(corc))
corc.beta[j] <- m[2,1]
corc.rho[j] <- corc$rho

prais <- prais.winsten.lm(start)
h <- as.matrix(coef(prais))
prais.beta[j] <- h[2,1]
prais.rho[j] <- prais$rho
rm(m,h)

j <- j+1
}

par3 <-matrix(b,k,1)
beta.par <- beta.na3
arma.par <- arma.beta
corc.par <- corc.beta
ols.par <- beta.ols
prais.par <- prais.beta
dldv.par <- dldv.beta

cat("Results for", n, "Cases","\n")

#RMSE
rmse <- sqrt(mean((beta.par-par3)^2))
cat("RMSE LDV: " , rmse,"\n")

rmse <- sqrt(mean((arma.par-par3)^2))
cat("RMSE ARMA: " , rmse,"\n")

rmse <- sqrt(mean((corc.par-par3)^2))
cat("RMSE Corc: " , rmse,"\n")
```

```
rmse <- sqrt(mean((ols.par-par3)^2))
cat("RMSE OLS: " , rmse,"\n")

rmse <- sqrt(mean((prais.par-par3)^2))
cat("RMSE Prais: " , rmse,"\n")

rmse <- sqrt(mean((dldv.par-par3)^2))
cat("RMSE DLDV: " , rmse,"\n")


#Bias in Beta

#X
cat("Mean of X W/ LDV: ", mean(beta.na3), "\n")
cat("Mean of X W/ ARMA ", mean(arma.par), "\n")
cat("Mean of X W/ GLS: ", mean(corc.par), "\n")
cat("Mean of X W/ OLS: ", mean(ols.par), "\n")
cat("Mean of X W/ Prais: ", mean(prais.par), "\n")
cat("Mean of X W/ DLDV: ", mean(dldv.par), "\n")
cat("Mean of Long Run Mult: ", mean(long.est), "\n")

bias.ldv <- (mean(beta.na3)-b)
bias.arma <- (mean(arma.par)-b)
bias.corc <- (mean(corc.par)-b)
bias.ols <- (mean(ols.par)-b)
bias.prais <- (mean(prais.par)-b)
bias.dldv <- (mean(dldv.par)-b)

cat("Bias in LDV: ", bias.ldv, "\n")
cat("Bias in ARMA: ", bias.arma, "\n")
cat("Bias in Corc: ", bias.corc, "\n")
cat("Bias in OLS: ", bias.ols, "\n")
cat("Bias in Prais: ", bias.prais, "\n")
cat("Bias in DLDV: ", bias.dldv, "\n")

per.ldv <- (1-((b-(bias.ldv))/b))*100
per.arma <- (1-((b-(bias.arma))/b))*100
per.corc <- (1-((b-(bias.corc))/b))*100
per.ols <- (1-((b-(bias.ols))/b))*100
per.prais <- (1-((b-(bias.prais))/b))*100
per.dldv <- (1-((b-(bias.dldv))/b))*100

cat("Percent of Bias LDV:", per.ldv, "\n")
cat("Percent of Bias ARMA:", per.arma, "\n")
```

```
cat("Percent of Bias Corc:", per.corc, "\n")
cat("Percent of Bias OLS:", per.ols, "\n")
cat("Percent of Bias Prais:", per.prais, "\n")
cat("Percent of Bias DLDV:", per.dldv, "\n")

#Long Run Bias
long.bias.ldv <- (mean(long.est)-long.beta)
long.bias.arma <- (mean(arma.par)-long.beta)
long.bias.corc <- (mean(corc.par)-long.beta)
long.bias.ols <- (mean(ols.par)-long.beta)
long.bias.prais <- (mean(prais.par)-long.beta)
long.bias.dldv <- (mean(dldv.long.est)-long.beta)

cat("True Long Run Effect: ", long.beta, "\n")
cat("Long Bias in LDV: ", long.bias.ldv, "\n")
cat("Long Bias in ARMA: ", long.bias.arma, "\n")
cat("Long Bias in Corc: ", long.bias.corc, "\n")
cat("Long Bias in OLS: ", long.bias.ols, "\n")
cat("Long Bias in Prais: ", long.bias.prais, "\n")
cat("Long Bias in DLDV: ", long.bias.dldv, "\n")

long.per.ldv <- (1-((long.beta-(long.bias.ldv))/long.beta))*100
long.per.arma <- (1-((long.beta-(long.bias.arma))/long.beta))*100
long.per.corc <- (1-((long.beta-(long.bias.corc))/long.beta))*100
long.per.ols <- (1-((long.beta-(long.bias.ols))/long.beta))*100
long.per.prais <- (1-((long.beta-(long.bias.prais))/long.beta))*100
long.per.dldv <- (1-((long.beta-(long.bias.dldv))/long.beta))*100

cat("Percent of Long Bias LDV:", long.per.ldv, "\n")
cat("Percent of Long Bias ARMA:", long.per.arma, "\n")
cat("Percent of Long Bias Corc:", long.per.corc, "\n")
cat("Percent of Long Bias OLS:", long.per.ols, "\n")
cat("Percent of Long Bias Prais:", long.per.prais, "\n")
cat("Percent of Long Bias DLDV:", long.per.dldv, "\n")

}
```

1.2 Varying ϕ Monte Carlo

```
#GLS Routine
cochrane.orcutt <- function(mod, ...){
     UseMethod("cochrane.orcott")
     }


cochrane.orcutt.lm <- function(mod){
```

```
     X <- model.matrix(mod)
     y <- model.response(model.frame(mod))
     e <- residuals(mod)
     n <- length(e)
     names <- colnames(X)
     rho <- sum(e[1:(n-1)]*e[2:n])/sum(e^2)
     y <- y[2:n] - rho * y[1:(n-1)]
     X <- X[2:n,] - rho * X[1:(n-1),]
     mod <- lm(y ~ X - 1)
     result <- list()
     result$coefficients <- coef(mod)
     names(result$coefficients) <- names
     summary <- summary(mod, corr = F)
     result$cov <- (summary$sigma^2) * summary$cov.unscaled
     dimnames(result$cov) <- list(names, names)
     result$sigma <- summary$sigma
     result$rho <- rho
     class(result) <- 'cochrane.orcutt'
     result
     }

prais.winsten <- function(mod, ...){
     UseMethod("prais.winsten")
     }


prais.winsten.lm <- function(mod){
     X <- model.matrix(mod)
     y <- model.response(model.frame(mod))
     e <- residuals(mod)
     n <- length(e)
     names <- colnames(X)
     rho <- sum(e[1:(n-1)]*e[2:n])/sum(e^2)
     y <- c(y[1] * (1 - rho^2)^0.5, y[2:n] - rho * y[1:(n-1)])
     X <- rbind(X[1,] * (1 - rho^2)^0.5, X[2:n,] - rho * X[1:(n-1),])
     mod <- lm(y ~ X - 1)
     result <- list()
     result$coefficients <- coef(mod)
     names(result$coefficients) <- names
     summary <- summary(mod, corr = F)
     result$cov <- (summary$sigma^2) * summary$cov.unscaled
     dimnames(result$cov) <- list(names, names)
     result$sigma <- summary$sigma
     result$rho <- rho
     class(result) <- 'prais.winsten'
```

```
     result
     }


####################Monte Carlo Starts Here#####################

#Number of Iterations
k <- 1000
# Number of Cases
n <- 100
#Autoregressive Parameters
rho1 <- 0.95
rho2 <- list(0.0, 0.10, 0.20, 0.50)

#Beta Value to Vary Model Fit
b <- 0.50
#Alpha value
a <- 0.75
long.beta <- b/(1-a)
set.seed(831047)

for (rho2 in rho2){

cat("#################Parameter Values############################\n")
cat("Beta : ", b, " Alpha: ", a, "\n")
cat("Rho_1: ", rho1, " Rho_2: ", rho2, "\n")
cat("#############################################################\n")

#Lag Operator
tslag<-function(y,d=1){
n <- length(y)
c(rep(NA,d),y)[1:n]
}

#Storage Matrices
beta.na1 <- matrix(NA,k,1)
stder.na1 <- matrix(NA,k,1)
beta.na2 <- matrix(NA,k,1)
stder.na2 <- matrix(NA,k,1)
beta.na3 <- matrix(NA,k,1)
stder.na3 <- matrix(NA,k,1)
arma.beta <- matrix(NA,k,1)
arma.ar <- matrix(NA,k,1)
arma.cvg <- matrix(NA,k,1)
corc.beta <- matrix(NA,k,1)
```

```
corc.rho <- matrix(NA,k,1)
beta.ols <- matrix(NA,k,1)
prais.beta <- matrix(NA,k,1)
prais.rho <- matrix(NA,k,1)
dldv.alpha1 <- matrix(NA,k,1)
dldv.alpha2 <- matrix(NA,k,1)
dldv.beta <- matrix(NA,k,1)
long.est <- matrix(NA,k,1)
dldv.long.est <- matrix(NA,k,1)

j <- 1
#Start Monte Carlo Here
for (j in j:k){

# Generate AR(1) process
#X DGP
e <- rnorm(n)
x <- double(n)
x[1] <- rnorm(1)
for(i in 2:n) {
x[i] <- rho1 * x[i-1] + e[i]
}

#Y DGP Error Process
e1 <- rnorm(n)
e2 <- double(n)
e2[1] <- rnorm(1)
for(i in 2:n) {
e2[i] <- rho2 * e2[i-1] + e1[i]
}

# Y DGP
y <- double(n)
y[1] <- rnorm(1)
for(i in 2:n){
y[i] <- a * y[i-1] + b * x[i] + e2[i]
}

#Data frame
lagy <- tslag(y)
simldv <- data.frame(y, x, lagy)
attach(simldv)

#Data Frame 2
lag2y <- tslag(lagy)
```

```
simdldv <- data.frame(y,x,lagy,lag2y)

# estimate LDV w/ OLS
ldv <- summary(lm(y ~ lagy + x, data = simldv))

beta.na1[j] <-ldv$coef[1,1]
stder.na1[j] <-ldv$coef[1,2]
beta.na2[j] <-ldv$coef[2,1]
stder.na2[j] <-ldv$coef[2,2]
beta.na3[j] <-ldv$coef[3,1]
stder.na3[j] <-ldv$coef[3,2]
long.est[j] <- ldv$coef[3,1]/(1-ldv$coef[2,1])

#Estimate with Two Lags
dldv <- summary(lm(y~lagy + lag2y + x, simdldv))
dldv.alpha1[j] <-dldv$coef[2,1]
dldv.alpha2[j] <-dldv$coef[3,1]
dldv.beta[j] <-dldv$coef[4,1]
dldv.long.est[j] <- dldv$coef[4,1]/(1-dldv$coef[2,1])


#ARMA via MLE
arma <- arima(y, order = c(1,0,0), xreg = x, method = c("ML"))
d <- as.matrix(coef(arma))
arma.beta[j] <- d[3,1]
arma.ar[j] <- d[1,1]
arma.cvg[j] <- arma$code
rm(d)

#Corc, Prais, and OLS
start <- lm(y ~ x, data = simldv)
p <- as.matrix(coef(start))
beta.ols[j] <- p[2,1]
corc <- cochrane.orcutt.lm(start)
m <- as.matrix(coef(corc))
corc.beta[j] <- m[2,1]
corc.rho[j] <- corc$rho

prais <- prais.winsten.lm(start)
h <- as.matrix(coef(prais))
prais.beta[j] <- h[2,1]
prais.rho[j] <- prais$rho
rm(m,h)

j <- j+1
```

```
}

par3 <-matrix(b,k,1)
beta.par <- beta.na3
arma.par <- arma.beta
corc.par <- corc.beta
ols.par <- beta.ols
prais.par <- prais.beta
dldv.par <- dldv.beta

cat("Results for", n, "Cases","\n")

#RMSE
rmse <- sqrt(mean((beta.par-par3)^2))
cat("RMSE LDV: " , rmse,"\n")

rmse <- sqrt(mean((arma.par-par3)^2))
cat("RMSE ARMA: " , rmse,"\n")

rmse <- sqrt(mean((corc.par-par3)^2))
cat("RMSE Corc: " , rmse,"\n")

rmse <- sqrt(mean((ols.par-par3)^2))
cat("RMSE OLS: " , rmse,"\n")

rmse <- sqrt(mean((prais.par-par3)^2))
cat("RMSE Prais: " , rmse,"\n")

rmse <- sqrt(mean((dldv.par-par3)^2))
cat("RMSE DLDV: " , rmse,"\n")

#Bias in Beta

#X
cat("Mean of X W/ LDV: ", mean(beta.na3), "\n")
cat("Mean of X W/ ARMA ", mean(arma.par), "\n")
cat("Mean of X W/ Corc: ", mean(corc.par), "\n")
cat("Mean of X W/ OLS: ", mean(ols.par), "\n")
cat("Mean of X W/ Prais: ", mean(prais.par), "\n")
cat("Mean of X W/ DLDV: ", mean(dldv.par), "\n")
cat("Mean of Long Run Mult: ", mean(long.est), "\n")

bias.ldv <- (mean(beta.na3)-b)
bias.arma <- (mean(arma.par)-b)
bias.corc <- (mean(corc.par)-b)
```

```
bias.ols <- (mean(ols.par)-b)
bias.prais <- (mean(prais.par)-b)
bias.dldv <- (mean(dldv.par)-b)

cat("Bias in LDV: ", bias.ldv, "\n")
cat("Bias in ARMA: ", bias.arma, "\n")
cat("Bias in Corc: ", bias.corc, "\n")
cat("Bias in OLS: ", bias.ols, "\n")
cat("Bias in Prais: ", bias.prais, "\n")
cat("Bias in DLDV: ", bias.dldv, "\n")

per.ldv <- (1-((b-(bias.ldv))/b))*100
per.arma <- (1-((b-(bias.arma))/b))*100
per.corc <- (1-((b-(bias.corc))/b))*100
per.ols <- (1-((b-(bias.ols))/b))*100
per.prais <- (1-((b-(bias.prais))/b))*100
per.dldv <- (1-((b-(bias.dldv))/b))*100

cat("Percent of Bias LDV:", per.ldv, "\n")
cat("Percent of Bias ARMA:", per.arma, "\n")
cat("Percent of Bias Corc:", per.corc, "\n")
cat("Percent of Bias OLS:", per.ols, "\n")
cat("Percent of Bias Prais:", per.prais, "\n")
cat("Percent of Bias DLDV:", per.dldv, "\n")

#Long Run Bias
long.bias.ldv <- (mean(long.est)-long.beta)
long.bias.arma <- (mean(arma.par)-long.beta)
long.bias.corc <- (mean(corc.par)-long.beta)
long.bias.ols <- (mean(ols.par)-long.beta)
long.bias.prais <- (mean(prais.par)-long.beta)
long.bias.dldv <- (mean(dldv.long.est)-long.beta)

cat("True Long Run Effect: ", long.beta, "\n")
cat("Long Bias in LDV: ", long.bias.ldv, "\n")
cat("Long Bias in ARMA: ", long.bias.arma, "\n")
cat("Long Bias in Corc: ", long.bias.corc, "\n")
cat("Long Bias in OLS: ", long.bias.ols, "\n")
cat("Long Bias in Prais: ", long.bias.prais, "\n")
cat("Long Bias in DLDV: ", long.bias.dldv, "\n")

long.per.ldv <- (1-((long.beta-(long.bias.ldv))/long.beta))*100
long.per.arma <- (1-((long.beta-(long.bias.arma))/long.beta))*100
long.per.corc <- (1-((long.beta-(long.bias.corc))/long.beta))*100
long.per.ols <- (1-((long.beta-(long.bias.ols))/long.beta))*100
```

```
long.per.prais <- (1-((long.beta-(long.bias.prais))/long.beta))*100
long.per.dldv <- (1-((long.beta-(long.bias.dldv))/long.beta))*100

cat("Percent of Long Bias LDV:", long.per.ldv, "\n")
cat("Percent of Long Bias ARMA:", long.per.arma, "\n")
cat("Percent of Long Bias Corc:", long.per.corc, "\n")
cat("Percent of Long Bias OLS:", long.per.ols, "\n")
cat("Percent of Long Bias Prais:", long.per.prais, "\n")
cat("Percent of Long Bias DLDV:", long.per.dldv, "\n")
}
```

2 In-Depth Monte Carlo Code

```
library(lmtest)

nn <- c(25,50,75,100,250,500,1000)
rr <- c(.65,.75,.85,.95)
set.seed(836491047)
rho2 <- list(0.00, 0.10, 0.20, 0.50)

  ##Define Lag Operator
  tslag <-function(y,d=1){
    n <- length(y)
    c(rep(NA,d),y)[1:n]
  }

for (rho2 in rho2){
cat("##################Parameter Values##############################\n")
cat("Rho_2: ", rho2, "\n")
cat("#################################################################\n")

bbias <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(bbias) <- nn
colnames(bbias) <- rr

abias <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(abias) <- nn
colnames(abias) <- rr

bbias.per <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(bbias.per) <- nn
colnames(bbias.per) <- rr

abias.per <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(abias.per) <- nn
```

```
colnames(abias.per) <- rr

over.con <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(over.con) <- nn
colnames(over.con) <- rr

rej.rate <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(rej.rate) <- nn
colnames(rej.rate) <- rr

r.sqd <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(r.sqd) <- nn
colnames(r.sqd) <- rr

long.per.ldv <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(long.per.ldv) <- nn
colnames(long.per.ldv) <- rr

long.ldv <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(long.ldv) <- nn
colnames(long.ldv) <- rr

bg.per <- matrix(NA,nrow=length(nn),ncol=length(rr))
rownames(bg.per) <- nn
colnames(bg.per) <- rr

for (ii in 1:length(nn)) {

  ## Number of Experiments
  k <- 1000
  ## Number of Cases
  n <- nn[ii]
  ##Parmeter Values
  b<- 0.50
  a <- 0.75
  long.beta <- b/(1-a)
  ##Define Empty Matrices
  beta.na2  <- matrix(NA,k,1)
  stder.na2 <- matrix(NA,k,1)
  beta.na3  <- matrix(NA,k,1)
  stder.na3 <- matrix(NA,k,1)
  r.sqd.2 <- matrix(NA,k,1)
  long.est <- matrix(NA,k,1)
  ac.test <- matrix(NA,k,1)
```

```
for (jj in 1:length(rr)) {
  rho1 <- rr[jj]

  ##Set Counter and Start Monte Carlo Loop
  j <- 1
  while (j < k + 1){

    ##X DGP
    ## Generate AR(1) process
    e <- rnorm(n)
    x <- double(n)
    x[1] <- rnorm(1)
    for(i in 2:n) {
      x[i] <- rho1 * x[i-1] + e[i]
    }
    ##Y DGP Error Process
    e1 <- rnorm(n)
    e2 <- double(n)
    e2[1] <- rnorm(1)
    for(i in 2:n) {
      e2[i] <- rho2 * e2[i-1] + e1[i]
    }
    ## Y DGP
    ##Parameter Value For Lagged Y
    y <- double(n)
    y[1] <- rnorm(1)
    for(i in 2:n){
      y[i] <- a * y[i-1] + b * x[i] + e2[i]
    }
    ##Generate Data Frame
    lagy <- tslag(y)
    simldv <- data.frame(y, x, lagy)

    ## Estimate LDV
    ldv <- summary(lm(y ~ lagy + x, data = simldv))
    beta.na2[j]  <-ldv$coef[2,1]
    stder.na2[j] <-ldv$coef[2,2]
    beta.na3[j]  <-ldv$coef[3,1]
    stder.na3[j] <-ldv$coef[3,2]
    long.est[j] <- ldv$coef[3,1]/(1-ldv$coef[2,1])
    r.sqd.2[j] <- ldv$adj.r.squared
    bg <- bgtest(ldv)
    ac.test[j] <- as.numeric(bg$p.value <= 0.05)
    j <- j+1
  }
```

```
    ##Average Bias
    bias.b2 <- (mean(beta.na2)-a)
    bias.b3 <- (mean(beta.na3)-b)

    abias[ii,jj] <- bias.b2
    bbias[ii,jj] <- bias.b3

    ##Percentage of Bias
    per.alpha <- (1-((a -(bias.b2))/a))*100
    per.beta  <- (1-((b-(bias.b3))/b))*100

    abias.per[ii,jj] <- per.alpha
    bbias.per[ii,jj] <- per.beta

    ##Rejection Rate
    rejrate2 <- ifelse(beta.na3/stder.na3>1.645, 0, 1)
    rej.rate[ii,jj] <- mean(rejrate2)

    ##Confidence Measure
    con.beta  <- 100 * sqrt(mean((beta.na3-mean(beta.na3))^2))/sqrt(mean(stder.na3^2))
    over.con[ii,jj] <- con.beta
    ##Model Fit
    r.sqd[ii,jj] <- mean(r.sqd.2)
    #Long Run Multiplier
    #cat("Mean of Long Run Mult: ", mean(long.est), "\n")
    long.ldv[ii,jj] <- (mean(long.est)-long.beta)
    long.per.ldv[ii,jj] <- (1-((long.beta-(long.ldv[ii,jj]))/long.beta))*100
    #BG Test
    bg.per[ii,jj] <- mean(ac.test)
      }
}

cat("Bias in Alpha \n")
print(abias)
cat("Bias in Beta \n")
print(bbias)
cat("Bias in Alpha as a Percentage\n")
print(abias.per)
cat("Bias in Beta as a Percentage\n")
print(bbias.per)
cat("Rejection Rate For Beta\n")
print(rej.rate)
cat("Measure of Confidence for Beta\n")
print(over.con)
```

```
cat("Model Fit: R-Sqd\n")
print(r.sqd)
cat("True Long Run Effect: ", long.beta, "\n")
cat("Long-Run Multiplier Bias\n")
print(long.ldv)
cat("Long-Run Multiplier Bias as a %\n")
print(long.per.ldv)
cat("Percentage of Auto-Correlated Model Residuals\n")
print(bg.per)
}
```