Empirical Analysis of the Role of Energy in Economic Growth

Caleb Reese[a], Lucas Timmer[a], Matthew Kuperus Heun[a,*]

[a]*Engineering Department, Calvin College, Grand Rapids, MI 49546, USA*

Abstract

*********** Add abstract ***********

Keywords: economic growth, energy, cobb-douglas, CES, LINEX

Caleb, put your LaTeX code here.

1. Cobb-Douglas Without Energy

```
#################################################
# Calculates parameter estimates and confidence intervals
# for the Cobb-Douglas production function given a country.
#
# countryName is a string containint the 2-letter abbreviation for the country, e.
#
# returns a vector of data for the Cobb-Douglas model.
# First item is the +95% CI on all parameters
# Second item contains the parameter estimates
# Third item is the -95% CI on all parameters
# Each row has names: lambda, alpha, beta, gamma, corresponding to the parameters
##
cobbDouglasData <- function(countryName){
```

*Corresponding author

Email address: mkh2@calvin.edu, tel: +1 (616) 526-6663, fax: +1 (616) 526-6501 (Matthew Kuperus Heun)

```
# Load the data that we need.
dataTable <- loadData(countryName)

# Establish guess values for alpha and lambda.
lambdaGuess <- 0.0 # guessing lambda = 0 means there is no technological progres
alphaGuess <- 0.3 # a typical value for alpha, the coefficient on capital stock

# Runs a non-linear least squares fit to the data. We've replaced beta with 1-al
modelCD <- nls(iGDP ~ exp(lambda*iYear) * iCapStk^alpha * iLabor^(1 - alpha),
               start=(list(lambda=lambdaGuess,alpha=alphaGuess)),
               data=dataTable)

# Checks validity of the model. AIC stands for Akaike's Information Criterion.
aicCD  <- AIC(modelCD, k=2)
#print(aicCD)

summaryCD <- summary(modelCD) # Gives the nls summary table.
#print(summaryCD)
ciCD <- confint(modelCD, level = ciLevel); ciCD # Displays confidence intervals

# Calculate beta and its confidence interval and report it.
alpha <- as.numeric(coef(modelCD)["alpha"])
beta <- 1.0 - alpha
beta.est <- deltaMethod(modelCD, "1 - alpha"); beta.est # Estimates beta and its
# Now calculate a confidence interval on beta
dofCD <- summaryCD$df[2]; dofCD # Gives the degrees of freedom for the model.
tvalCD <- qt(ciHalfLevel, df = dofCD); tvalCD
betaCICD <- with(beta.est, Estimate + c(-1.0, 1.0) * tvalCD * SE); betaCICD # Gi

#print(coef(modelCD))

# Combine all estimates and their confidence intervals into data frames with int
estCD <- data.frame(lambda = coef(modelCD)["lambda"], alpha = coef(modelCD)["alp
#print(estCD)
row.names(estCD) <- "CD"
#row.names(estCD) <- "Cobb-Douglas: $y = e^{\\lambda t}k^{\\alpha}l^{\\beta}$"
# The [1] subscripts pick off the lower confidence interval
```

```
  lowerCD <- data.frame(lambda = ciCD["lambda","2.5%"], alpha = ciCD["alpha", "2.5
  row.names(lowerCD) <- "- 95% CI"
  # The [2] subscripts pick off the lower confidence interval
  upperCD <- data.frame(lambda = ciCD["lambda","97.5%"], alpha = ciCD["alpha", "97
  row.names(upperCD) <- "+ 95% CI"

  # Now create the data for a table.
  dataCD <- rbind(upperCD, estCD, lowerCD)
  #print(dataCD)
  return(dataCD)
}

####################################
# Creates a LaTeX printable table from the data. This function first calls cobbDou
#
# countryName is a string containint the 2-letter abbreviation for the country, e.
#
# returns a printable LaTeX table from xtable.
##
cobbDouglasTable <- function(countryName){
  dataCD <- cobbDouglasData(countryName)
  colnames(dataCD) <- c("$\\lambda$", "$\\alpha$", "$\\beta$", "$\\gamma$")
  tableCD <- xtable(dataCD, caption=paste(countryName, " Cobb-Douglas, 1980-2011",
  #print(tableCD)
  return(tableCD)
}
```

```
tablesCD <- lapply(countries, cobbDouglasTable)

Waiting for profiling to be done...
Error:  could not find function "deltaMethod"
```

```
print(tablesCD[["US"]], caption.placement="top")

Error:  error in evaluating the argument 'x' in selecting a method
for function 'print':  Error:  object 'tablesCD' not found
```

```
print(tablesCD[["ZA"]], caption.placement="top")
```

```
Error:  error in evaluating the argument 'x' in selecting a method
for function 'print':  Error:  object 'tablesCD' not found
```

```
# According to http://cran.r-project.org/web/packages/xtable/vignettes/xtableGalle
# be able to use the "sanitize.text.function" parameter to allow markup in column
# line is not working at the present time. --MKH, 18 Jan 2012.
# print(tableCD, caption.placement="top", sanitize.text.function = function(x){x})
```

2. Cobb-Douglas With Energy

We can force α, β, and γ to be in $[0,1]$ by a reparameterization:

$$a \in [0,1], b \in [0,1], \alpha = \min(a,b), \beta = |b-a|, \gamma = 1 - \max(a,b)$$

```
###############################
# This function fits the Cobb Douglas production function including energy to hist
# The functional form is
#
# iGDP = exp(lambda * itime) * iCapStk^alpha * iLabor^beta * iEnergy^gamma
#
# iYear: time indexed to 0.0 at the first year [years since beginning of data set]
# iGDP = time series GDP data indexed to 1.0 at the first year [-]
# iCapStk = time series capital stock data indexed to 1.0 at the first year [-]
# iLabor = time series labor data indexed to 1.0 at the first year [-]
# iEnergy = time series energy data indexed to 1.0 at the first year [-].
#      This may be any of heat (iQ), exergy (iX), or useful work (iU).
#
# returns a data object containing three rows and 5 columns.
#   col 1: lambda
#   col 2: alpha
#   col 3: beta
#   col 4: gamma
#   row 1: -95% CI for each parameter
#   row 2: estimate for each parameter
#   row 3: +95% CI for each parameter
```

```
#
# countryName is a string containint the 2-letter abbreviation for the country, e.
# energyColumnName is the name of the column to be used for energy.
#   This may be any of heat ("iQ"), exergy ("iX"), or useful work ("iU").
# energyString is the string to be used in table captions reprsenting the type of
##
cobbDouglasEnergyData <- function(countryName, energyString){

  energyColumnName  <- paste("i", energyString, sep="")
  #print(energyColumnName)

  # Load the data that we need.
  dataTable <- loadData(countryName)

  # Reparameterize to ensure that we meet the constraints:
  # * alpha + beta + gamma = 1.0.
  # * alpha, beta, and gamma are all between 0.0 and 1.0.
  # To do this, we reparameterize as
  # * 0 < a < 1
  # * 0 < b < 1
  # * alpha = min(a, b)
  # * beta = b - a
  # * gamma = 1 - max(a, b)
  iEnergy <- dataTable[energyColumnName] #grabs the desired energy column
  lambdaGuess <- 0.0 # guessing lambda = 0 means there is no technological progres
  alphaGuess <- 0.3 # a typical value for alpha
  betaGuess <- 0.7 # a typical value for beta
  modelCDe <- nls(iGDP ~ exp(lambda*iYear) *
                    iCapStk^min(a,b) * iLabor^abs(b-a) *
                    iEnergy^(1.0 - max(a,b)),
                  algorithm = "port",
                  start = list(lambda=lambdaGuess, a=alphaGuess, b=alphaGuess+beta
                  lower = list(lambda=-Inf, a=0, b=0),
                  upper = list(lambda=Inf, a=1, b=1),
                  data = dataTable)
  aicCDe <- AIC(modelCDe, k=2) # Checks validity of the model. AIC stands for Akai
  #print(aicCDe)
```

```
summaryCDe <- summary(modelCDe) # Gives the nls summary table
#print(summaryCDe)

# Provides confidence intervals on lambda, a, and b. But, we need CIs on alpha a
ciCDe <- confint(modelCDe, level = ciLevel)
#print(ciCDe)

a <- as.numeric(coef(modelCDe)["a"])
b <- as.numeric(coef(modelCDe)["b"])
lambda <- as.numeric(coef(modelCDe)["lambda"])
alpha <- a
beta <- b - a
gamma <- 1.0 - alpha - beta

# Report results with SE
beta.est <- deltaMethod(modelCDe, "b-a") # Reports results for beta, because bet
gamma.est <- deltaMethod(modelCDe, "1-b") # Reports results for gamma, because g

# Now calculate confidence intervals.
dofCDe <- summaryCDe$df[2] # Gives the degrees of freedom for the model.
#print(dofCDe)
tvalCDe <- qt(ciHalfLevel, df = dofCDe)
#print(tvalCDe)
betaCICDe <- with(beta.est, Estimate + c(-1.0, 1.0) * tvalCDe * SE) # Gives the
#print(betaCICDe)
gammaCICDe <- with(gamma.est, Estimate + c(-1.0, 1.0) * tvalCDe * SE) # Gives th
#print(gammaCICDe)

# Combine all estimates and their confidence intervals into data frames with int
estCDe <- data.frame(lambda = lambda, alpha = alpha, beta = beta, gamma = gamma)
#print(estCDe);
#row.names(estCDe) <- "Cobb-Douglas with e: $y = e^{\\lambda t}k^{\\alpha}l^{\\b
row.names(estCDe) <- paste("C-D with ", energyString, sep="")
# The [1] subscripts pick off the lower confidence interval
lowerCDe <- data.frame(lambda = ciCDe["lambda","2.5%"], alpha = ciCDe["a", "2.5%
row.names(lowerCDe) <- "- 95% CI"
# The [2] subscripts pick off the lower confidence interval
```

```
  upperCDe <- data.frame(lambda = ciCDe["lambda","97.5%"], alpha = ciCDe["a", "97.
  row.names(upperCDe) <- "+ 95% CI"

  # Now create the data for a table.
  dataCDe <- rbind(upperCDe, estCDe, lowerCDe)
  #print(dataCDe)
  return(dataCDe)
}

cobbDouglasEnergyTable <- function(countryName, energyType){
  dataCDe <- cobbDouglasEnergyData(countryName, energyType)
  tableCDe <- xtable(dataCDe, caption=paste(countryName, ", 1980-2011.", sep=""),
  return(tableCDe)
}
```

2.1. Cobb-Douglas with Q

```
# Note that the anlaysis of ZA is taking a long time here. Need to figure out why.
CDqTables <- lapply(countries, cobbDouglasEnergyTable, energyType="Q")

Waiting for profiling to be done...
Error:  could not find function "deltaMethod"
```

```
print(CDqTables[["US"]], caption.placement="top")

Error:  error in evaluating the argument 'x' in selecting a method
for function 'print':  Error:  object 'CDqTables' not found

print(CDqTables[["ZA"]], caption.placement="top")

Error:  error in evaluating the argument 'x' in selecting a method
for function 'print':  Error:  object 'CDqTables' not found

# According to http://cran.r-project.org/web/packages/xtable/vignettes/xtableGalle
# be able to use the "sanitize.text.function" parameter to allow markup in column
# line is not working at the present time. --MKH, 18 Jan 2012.
```

```
# print(tableCDe, sanitize.text.function = function(x){x})

#print(tableAll, caption.placement="top")
```

2.2. Cobb-Douglas With X

```
energyColumn <- "iX"; energyString <- "x"
dataCDx <- cobbDouglasEnergy(countryName, energyColumn, energyString)

Error:  could not find function "cobbDouglasEnergy"

tableCDx <- xtable(dataCDx, caption=paste(countryName, ", 1980-2011.", sep=""), di

Error:  could not find function "xtable"
```

```
print(tableCDx, caption.placement="top")

Error:  error in evaluating the argument 'x' in selecting a method
for function 'print':  Error:  object 'tableCDx' not found
```

2.3. Cobb-Douglas With U

```
energyColumn <- "iU"; energySTring <- "u"
dataCDu <- cobbDouglasEnergy(countryName, energyColumn, energyString)

Error:  could not find function "cobbDouglasEnergy"

tableCDu <- xtable(dataCDu, caption=paste(countryName, ", 1980-2011.", sep=""), di

Error:  could not find function "xtable"
```

```
print(tableCDu, caption.placement="top")

Error:  error in evaluating the argument 'x' in selecting a method
for function 'print':  Error:  object 'tableCDu' not found
```

3. CES With Q

```
ces <- function(countryName, energyColumnName, energyString){

  # Load the data that we need.
  dataTable <- loadData(countryName)

  # Establish guess values for phi beta, zeta, lambda_L and lambda_E.
  phiGuess <- -20
  betaGuess <- 0.5 # a typical value for beta (exponent on labor)
  zetaGuess <- 0.0004 # a small value
  lambda_LGuess <- 0.007 #assuming no technical progress on the labor-capital port
  lambda_EGuess <- 0.008 #assuming no technical progress on the energy portion of

  # Runs a non-linear least squares fit to the data with constraints
  modelCES <- nls(iGDP ~ ((1-zeta) * (exp(lambda_L*iYear) * iCapStk^(1-beta) * iLa
                         + zeta*(exp(lambda_E*iYear) * iQ)^phi)^(1/phi),
                 algorithm = "port",
                 control = nls.control(maxiter = 500, tol = 1e-06, minFactor = 1
                                       printEval = FALSE, warnOnly = FALSE),
                 start = list(phi=phiGuess, beta=betaGuess, zeta=zetaGuess, lamb
                              lambda_E=lambda_EGuess),
                 lower = list(phi=-Inf, beta=0, zeta=0, lambda_L=-Inf, lambda_E=
                 upper = list(phi=0, beta=1, zeta=1, lambda_L=Inf, lambda_E=Inf)
                 data=dataTable)

  aicCES <- AIC(modelCES, k=2) # Checks validity of the model. AIC stands for Akai
  print(aicCES)

  # Gives the nls summary table
  summaryCES <- summary(modelCES) # Gives the nls summary table
  print(summaryCES)

  # Provides confidence intervals on phi, beta, zeta, lambda_L, and lambda_E. But,
  ciCES <- confint(modelCES, level = ciLevel)
  print(ciCES)
```

```
# Get the estimate for alpha
beta <- as.numeric(coef(modelCES)["beta"])
alpha <- 1.0 - beta
alpha.est <- deltaMethod(modelCES, "1 - beta") # Estimates alpha and its standar
print(alpha.est)

# Now calculate a confidence interval on alpha
dofCES <- summaryCES$df[2]
print(dofCES) # Gives the degrees of freedom for the model.
tvalCES <- qt(ciHalfLevel, df = dofCES); tvalCES
# Get confidence intervals for each parameter in the model
alphaCICES <- with(alpha.est, Estimate + c(-1.0, 1.0) * tvalCES * SE) # CI on al
print(alphaCICES)

# Assemble the data into data frames for the table.
estCES <- data.frame(phi = coef(modelCES)["phi"], alpha = alpha,
                     beta = coef(modelCES)["beta"], zeta = coef(modelCES)["zeta'
                     lambda_L = coef(modelCES)["lambda_L"], lambda_E = coef(mode
row.names(estCES) <- paste("CES with ", energyString, sep="")
#print(estCES)
# The [1] subscripts pick off the lower confidence interval
lowerCES <- data.frame(phi = ciCES["phi","2.5%"], alpha = alphaCICES[1],
                       beta = ciCES["beta", "2.5%"], zeta = ciCES["zeta", "2.5%'
                       lambda_L = ciCES["lambda_L", "2.5%"], lambda_E = ciCES["l
row.names(lowerCES) <- "- 95% CI"
# The [2] subscripts pick off the lower confidence interval
upperCES <- data.frame(phi = ciCES["phi","97.5%"], alpha = alphaCICES[2],
                       beta = ciCES["beta", "97.5%"], zeta = ciCES["zeta", "97.5
                       lambda_L = ciCES["lambda_L", "97.5%"], lambda_E = ciCES['
row.names(upperCES) <- "+ 95% CI"

# Now create the data for a table.
dataCES <- rbind(upperCES, estCES, lowerCES)
print(dataCES)
return(dataCES)

#xyplot( resid(modelCESQ) ~ fitted(modelCESQ) )
```

```
  #histogram( ~resid(modelCESQ) )
  #qqmath( ~resid(modelCESQ) )
}
```

```
countryName <- "US"
energyColumn <- "iQ"; energyString <- "q"
dataCESq <- ces(countryName, energyColumn, energyString)

[1] -197.8

Formula: iGDP ~ ((1 - zeta) * (exp(lambda_L * iYear) * iCapStk^(1 - beta) *
    iLabor^beta)^phi + zeta * (exp(lambda_E * iYear) * iQ)^phi)^(1/phi)

Parameters:
          Estimate Std. Error t value Pr(>|t|)
phi      -2.22e+01   1.50e+01   -1.48   0.1512
beta      5.82e-01   5.20e-02   11.19  1.2e-11
zeta      3.51e-04   1.38e-03    0.25   0.8014
lambda_L  7.62e-03   8.30e-04    9.18  8.5e-10
lambda_E  8.05e-03   2.84e-03    2.84   0.0085

Residual standard error: 0.00993 on 27 degrees of freedom

Algorithm "port", convergence message: relative convergence (4)

Waiting for profiling to be done...

              2.5%     97.5%
phi             NA -4.621867
beta      0.462998  0.656248
zeta            NA        NA
lambda_L  0.006024  0.008997
lambda_E -0.002867  0.012398

Error:  could not find function "deltaMethod"

tableCESq <- xtable(dataCESq, caption=paste(countryName, ", 1980-2011.", sep=""),

Error:  could not find function "xtable"
```

```
print(tableCESq, caption.placement="top")

Error:  error in evaluating the argument 'x' in selecting a method
for function 'print':  Error:  object 'tableCESq' not found
```