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

```
fileName <- "data/USData.txt"

# Read the data file as a table with a header.
## dataTable is a poor name.  Name it after contents, unless this is the
## start of a script where this will be provided at the command line.
dataTable <- read.table(fileName, header = TRUE)

# Identifies the header names associated with dataTable
names(dataTable)

 [1] "Year"
 [2] "GDP.Millionsofreal2005USdollars."
 [3] "Labour.Millionsofhoursworked."
 [4] "CapitalStock.Millionsofreal2005USdollars."
 [5] "Thermalenergy.TJ."
 [6] "Exergy.TJ."
 [7] "UsefulWork.TJ."
 [8] "iYear"
 [9] "iGDP"
```

*Corresponding author

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

```
[10] "iLabor"
[11] "iCapStk"
[12] "iQ"
[13] "iX"
[14] "iU"
```

1. Cobb-Douglas Without Energy

```
# Establish guess values for alpha and lambda.
lambdaGuess <- 0.0 # guessing lambda = 0 means there is no technological progress.
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-alph
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); aicCD

[1] -163


summaryCD <- summary(modelCD) # Gives the nls summary table.
print(summaryCD)


Formula: iGDP ~ exp(lambda * iYear) * iCapStk^alpha * iLabor^(1 - alpha)

Parameters:
       Estimate Std. Error t value Pr(>|t|)
lambda 0.010256   0.000682   15.03  1.7e-15
alpha  0.270030   0.028311    9.54  1.4e-10

Residual standard error: 0.0178 on 30 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 3.81e-07
```

```
ciCD <- confint(modelCD, level = 0.95); ciCD # Displays confidence intervals for t

Waiting for profiling to be done...

           2.5%   97.5%
lambda 0.008862 0.01164
alpha  0.212405 0.32785


# 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 s

          Estimate      SE
1 - alpha     0.73 0.02831

# Now calculate a confidence interval on beta
dofCD <- summaryCD$df[2]; dofCD # Gives the degrees of freedom for the model.

[1] 30

tvalCD <- qt(0.975, df = dofCD); tvalCD

[1] 2.042

betaCICD <- with(beta.est, Estimate + c(-1.0, 1.0) * tvalCD * SE); betaCICD # Give

[1] 0.6722 0.7878


coef(modelCD)

 lambda   alpha
0.01026 0.27003
```

```
# Combine all estimates and their confidence intervals into data frames with intel
estCD <- data.frame(lambda = coef(modelCD)["lambda"], alpha = coef(modelCD)["alpha
row.names(estCD) <- "Cobb-Douglas"
#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", "97.5
row.names(upperCD) <- "+ 95% CI"

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

               lambda  alpha   beta gamma
+ 95% CI     0.011645 0.3279 0.7878    NA
Cobb-Douglas 0.010256 0.2700 0.7300     0
- 95% CI     0.008862 0.2124 0.6722    NA

colnames(dataCD) <- c("$\\lambda$", "$\\alpha$", "$\\beta$", "$\\gamma$")
tableCD <- xtable(dataCD, caption="U.S. Cobb-Douglas, 1980-2011", digit = c(4, 4,
```

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

Table 1: U.S. Cobb-Douglas, 1980-2011

	\$\lambda\$	\$\alpha\$	\$\beta\$	\$\gamma\$
+ 95% CI	0.0116	0.33	0.79	
Cobb-Douglas	0.0103	0.27	0.73	0.0
- 95% CI	0.0089	0.21	0.67	

```
# 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

```
###############################
# 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
#
# dataTable is the table to be used for this curve fit
# iEnergy is the name of the column to be used for energy.
#   This may be any of heat (iQ), exergy (iX), or useful work (iU).
##
cobbDouglasEnergy <- function(dataTable, iEnergy){
  # 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)
energyName = names(iEnergy) #grabs the name of the 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 = 0.95)
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(0.975, 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("Cobb-Douglas with ", energyName, 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)
}
```

2.1. Cobb-Douglas with Q

```
dataCDe <- cobbDouglasEnergy(dataTable, dataTable["iQ"])

[1] -161.1

Formula: iGDP ~ exp(lambda * iYear) * iCapStk^min(a, b) * iLabor^abs(b -
    a) * iEnergy^(1 - max(a, b))

Parameters:
```

```
       Estimate Std. Error t value Pr(>|t|)
lambda  0.01049    0.00108    9.67  1.4e-10
a       0.26322    0.03795    6.94  1.3e-07
b       0.97890    0.07647   12.80  1.9e-13

Residual standard error: 0.0181 on 29 degrees of freedom

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

Waiting for profiling to be done...

          2.5%  97.5%
lambda 0.00885 0.0127
a      0.18580 0.3283
b      0.82175     NA
[1] 29
[1] 2.045
[1] 0.5947 0.8367
[1] -0.1353  0.1775
   lambda  alpha   beta  gamma
1 0.01049 0.2632 0.7157 0.0211
                      lambda  alpha   beta   gamma
+ 95% CI             0.01270 0.3283 0.8367  0.1775
Cobb-Douglas with iQ 0.01049 0.2632 0.7157  0.0211
- 95% CI             0.00885 0.1858 0.5947 -0.1353

tableCDe <- xtable(dataCDe, caption="U.S. 1980-2011.", digit = c(4, 4, 2, 2, 3))
#estCDe <- dataCDe[2]
#dataAll <- rbind(estCD, estCDe); dataAll
#tableAll <- xtable(dataAll, caption="U.S. 1980-2011, All Models.", digit = c(4, 4
```

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

```
# 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
```

Table 2: U.S. 1980-2011.

	lambda	alpha	beta	gamma
+ 95% CI	0.0127	0.33	0.84	0.177
Cobb-Douglas with iQ	0.0105	0.26	0.72	0.021
- 95% CI	0.0089	0.19	0.59	-0.135

```
# 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

```
dataCDx <- cobbDouglasEnergy(dataTable, dataTable["iX"])

[1] -161.4

Formula: iGDP ~ exp(lambda * iYear) * iCapStk^min(a, b) * iLabor^abs(b -
    a) * iEnergy^(1 - max(a, b))

Parameters:
       Estimate Std. Error t value Pr(>|t|)
lambda  0.01071    0.00105   10.16  4.6e-11
a       0.25689    0.03666    7.01  1.0e-07
b       0.95700    0.07467   12.82  1.8e-13

Residual standard error: 0.018 on 29 degrees of freedom

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

Waiting for profiling to be done...

           2.5%   97.5%
lambda 0.008902 0.01287
a      0.182137 0.32622
b      0.803506      NA
```

```
[1] 29
[1] 2.045
[1] 0.5791 0.8211
[1] -0.1097  0.1957
   lambda  alpha   beta gamma
1 0.01071 0.2569 0.7001 0.043
                     lambda  alpha   beta   gamma
+ 95% CI             0.012867 0.3262 0.8211  0.1957
Cobb-Douglas with iX 0.010714 0.2569 0.7001  0.0430
- 95% CI             0.008902 0.1821 0.5791 -0.1097

tableCDx <- xtable(dataCDx, caption="U.S. 1980-2011.", digit = c(4, 4, 2, 2, 3))
```

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

Table 3: U.S. 1980-2011.

	lambda	alpha	beta	gamma
+ 95% CI	0.0129	0.33	0.82	0.196
Cobb-Douglas with iX	0.0107	0.26	0.70	0.043
- 95% CI	0.0089	0.18	0.58	-0.110

2.3. Cobb-Douglas With U

```
dataCDu <- cobbDouglasEnergy(dataTable, dataTable["iU"])

Error:  Missing value or an infinity produced when evaluating the
model

tableCDu <- xtable(dataCDu, caption="U.S. 1980-2011.", digit = c(4, 4, 2, 2, 3))

Error:  object 'dataCDu' not found
```

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

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

An issue arrises in this example because contraining α and β is not sufficient to guarantee that γ is properly constrained.

```
# Establish guess values for alpha, beta, and lambda.
lambdaGuess <- 0.0 # guessing lambda = 0 means there is no technological progress.
alphaGuess <- 0.2 # a typical value for alpha
betaGuess <- 0.6 # a typical value for beta
gammaGuess <- 0.01 # a nice low value

# Runs a non-linear least squares fit to the data with constraints
modelCDU <- nls(iGDP ~ exp(lambda*iYear) * iCapStk^alpha * iLabor^beta * iU^(1.0 -
                algorithm="port",
                start = list(lambda=lambdaGuess, alpha=alphaGuess, beta=betaGuess)
                lower = list(lambda=-Inf, alpha=0, beta=0),
                upper = list(lambda=Inf, alpha=1, beta=1),
                data=dataTable)

# Gives the nls summary table
summary(modelCDU)


Formula: iGDP ~ exp(lambda * iYear) * iCapStk^alpha * iLabor^beta * iU^(1 -
    alpha - beta)

Parameters:
       Estimate Std. Error t value Pr(>|t|)
lambda  0.00920    0.00129    7.11  1.2e-06
alpha   0.32389    0.07187    4.51  0.00027
beta    0.71425    0.07641    9.35  2.5e-08

Residual standard error: 0.00994 on 18 degrees of freedom

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

```
  (11 observations deleted due to missingness)

confint(modelCDU, level = 0.95)

Waiting for profiling to be done...

          2.5%   97.5%
lambda 0.00649 0.01192
alpha  0.17289 0.47457
beta   0.55373 0.87481
```

```
# Checks validity of the model. AIC stands for Akaike's Information Criterion
AIC(modelCDU, k=2)

[1] -129.3
```

The problem here is that $\hat{\gamma} < 0$.

```
# Calculate gamma and report it.
alpha <- coef(modelCDU)["alpha"]
beta <- coef(modelCDU)["beta"]
gamma <- as.numeric(1.0 - alpha - beta)
c(coef(modelCDU), gamma=gamma)

   lambda     alpha      beta     gamma
 0.009203  0.323887  0.714248 -0.038135
```

Our $\hat{\gamma}$ is not much below 0. Let's compute the standard error.

```
require(car)  # the methods in alr3 have been deprecated.  Use car instead.
gamma.est <- deltaMethod(modelCDU, "1-alpha-beta"); gamma.est

                 Estimate      SE
1 - alpha - beta -0.03813 0.03055

# crude CI (using 2 as a rough estimate for critical value)
with(gamma.est, Estimate + c(-1,1) * 2 * SE)

[1] -0.09924  0.02297
```

So no real cause for concern: our data don't convince us that the real γ is different from 0.

2.4. Forcing $\gamma \geq 0$

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

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

```
modelCDUforced <- nls(iGDP ~ exp(lambda*iYear) *
                        iCapStk^min(a,b) * iLabor^abs(b-a) *
                        iU^(1.0 - max(a,b)),
                      algorithm="port",
                      start = list(lambda=lambdaGuess, a=alphaGuess, b=alphaGuess
                      lower = list(lambda=-Inf, a=0, b=0),
                      upper = list(lambda=Inf, a=1, b=1),
                      data=dataTable)

coef(summary(modelCDUforced))

       Estimate Std. Error t value  Pr(>|t|)
lambda 0.009299   0.001349   6.893 1.908e-06
a      0.318805   0.074953   4.253 4.780e-04
b      1.000000   0.031938  31.310 3.765e-17

with( as.data.frame(t(coef(modelCDUforced))), c(alpha=min(a,b), beta=abs(b-a), gam

 alpha   beta  gamma
0.3188 0.6812 0.0000
```

But the naive delta method to calculate significance information fails because R doesn't know how to calculate the derivatives for the minimum and maximum functions. So we need to be clever, using the fact that we know now that $a < b$:

```
# alpha = a
deltaMethod( modelCDUforced, "a")
```

```
  Estimate      SE
a   0.3188 0.07495

# beta = b - a
deltaMethod( modelCDUforced, "b-a")

      Estimate      SE
b - a   0.6812 0.07972

# gamma = 1-b
deltaMethod( modelCDUforced, "1-b")

      Estimate      SE
1 - b        0 0.03194
```

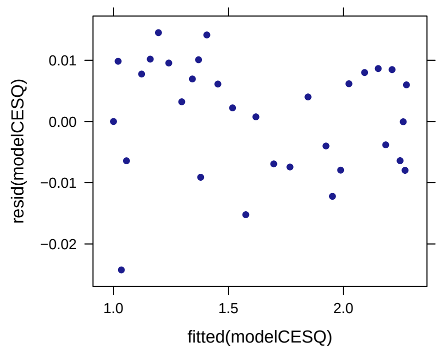
So this seems to give us what we want for this case: We have parameter estimates and standard errors subject to all of our constraints.

It may be that we can avoid using min and max and just use a, $b - a$ and $1 - b$. If that works, then this generalizes fairly easily to any number of parameters that must be bounded by 0 and 1 and sum to 1. (Else we have to "sort" the dummy parameters first, which is OK but makes the coding a bit uglier.)

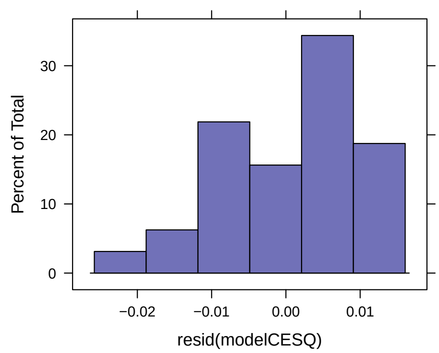
3. CES With Q

```
# Establish guess values for alpha, beta, and lambda.
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 portic
lambda_EGuess <- 0.008 #assuming no technical progress on the energy portion of th

# Runs a non-linear least squares fit to the data with constraints
modelCESQ <- nls(iGDP ~ ((1-zeta) * (exp(lambda_L*iYear) * iLabor^beta * iCapStk^(
                         + zeta*(exp(lambda_E*iYear) * iQ)^phi)^(1/phi),
                 algorithm = "port",
                 control = nls.control(maxiter = 500, tol = 1e-06, minFactor = 1/1
```

```
                                    printEval = FALSE, warnOnly = FALSE),
                start = list(phi=phiGuess, beta=betaGuess, zeta=zetaGuess, lambda
                             lambda_E=lambda_EGuess),
                lower = list(phi=-Inf, beta=0, zeta=0, lambda_L=-Inf, lambda_E=-I
                upper = list(phi=0, beta=1, zeta=1, lambda_L=Inf, lambda_E=Inf),
                data=dataTable)

# Gives the nls summary table
summary(modelCESQ)


Formula: iGDP ~ ((1 - zeta) * (exp(lambda_L * iYear) * iLabor^beta * iCapStk^(1 -
    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)

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




```
histogram( ~resid(modelCESQ) )
```




```
qqmath( ~resid(modelCESQ) )
```