

```

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"
[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-alpha for simplicity.
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 the CD model.

```

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 standard error (SE).
```

```
          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 # Gives the confidence interval on
```

```
[1] 0.6722 0.7878
```

```
coef(modelCD)
```

```
lambda  alpha
0.01026 0.27003
```

Combine all estimates and their confidence intervals into data frames with intelligent row names

```
estCD <- data.frame(lambda = coef(modelCD)["lambda"], alpha = coef(modelCD)["alpha"], beta = beta, gamma = NA)
```

```
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%"], beta = betaCICD[1], gamma = NA)
```

```
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%"], beta = betaCICD[2], gamma = NA)
```

```
row.names(upperCD) <- "+ 95% CI"
```

Now create the data for a table.

```
dataCD <- rbind(lowerCD, estCD, upperCD); dataCD
```

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

```
colnames(dataCD) <- c("$\\lambda$", "$\\alpha$", "$\\beta$", "$\\gamma$")
```

```
tableCD <- xtable(dataCD)
```

Warning: provided 3 variables to replace 1 variables

```
print(xtable(dataCD), floating=FALSE)
```

	λ	α	β	γ
- 95% CI	0.01	0.21	0.67	
Cobb-Douglas	0.01	0.27	0.73	
+ 95% CI	0.01	0.33	0.79	

```
# According to http://cran.r-project.org/web/packages/xtable/vignettes/xtableGallery.pdf, Section 3.1, I should
# be able to use the "sanitize.text.function" parameter to allow markup in column headers. But this next
# line is not working at the present time. --MKH, 18 Jan 2012.
# print(tableCD, sanitize.text.function = function(x){x})
```

2 Cobb-Douglas With Q

```
# 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

# Runs a non-linear least squares fit to the data.
# modelCDQ <- nls(iGDP ~ exp(lambda*iYear) * iCapStk^alpha * iLabor^beta * iQ^(1.0 - alpha - beta),
#               start=(list(lambda=lambdaGuess,alpha=alphaGuess,beta=betaGuess)),
#               data=dataTable)

# Reparameterize to ensure that we meet the constraint that alpha + beta + gamma = 1.0.
# 0 < a < 1
# 0 < b < 1
# alpha = min(a, b)
# beta = b - a
# gamma = 1 - max(a, b)

lambdaGuess <- 0.0 # guessing lambda = 0 means there is no technological progress.
alphaGuess <- 0.2 # a typical value for alpha
betaGuess <- 0.8 # a typical value for beta
modelCDq <- nls(iGDP ~ exp(lambda*iYear) *
                iCapStk^min(a,b) * iLabor^abs(b-a) *
                iQ^(1.0 - max(a,b)),
                algorithm = "port",
                start = list(lambda=lambdaGuess, a=alphaGuess, b=alphaGuess+betaGuess),
                lower = list(lambda=-Inf, a=0, b=0),
                upper = list(lambda=Inf, a=1, b=1),
                data = dataTable)

aicCDq <- AIC(modelCDq, k=2); aicCDq # Checks validity of the model. AIC stands for Akaike's Information Crite
[1] -161.1

summaryCDq <- summary(modelCDq); summaryCDq # Gives the nls summary table

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

# Provides confidence intervals on lambda, a, and b. But, we need CIs on alpha and beta.
ciCDq <- confint(modelCDq, level = 0.95); ciCDq

Waiting for profiling to be done...

      2.5%  97.5%
lambda 0.00885 0.0127
a      0.18580 0.3283
b      0.82175    NA

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

# Report results with SE
beta.est <- deltaMethod(modelCDq, "b-a"); beta.est # Reports results for beta, because beta = b - a.

      Estimate      SE
b - a    0.7157 0.05916

gamma.est <- deltaMethod(modelCDq, "1-b"); gamma.est # Reports results for gamma, because gamma = 1 - b.

      Estimate      SE
1 - b    0.0211 0.07647

# Now calculate confidence intervals.
dofCDq <- summaryCDq$df[2]; dofCDq # Gives the degrees of freedom for the model.

[1] 29

tvalCDq <- qt(0.975, df = dofCDq); tvalCDq

[1] 2.045

betaCICDq <- with(beta.est, Estimate + c(-1.0, 1.0) * tvalCDq * SE); betaCICDq # Gives the confidence interval

[1] 0.5947 0.8367

gammaCICDq <- with(gamma.est, Estimate + c(-1.0, 1.0) * tvalCDq * SE); gammaCICDq # Gives the confidence interval

[1] -0.1353 0.1775

# Combine all estimates and their confidence intervals into data frames with intelligent row names
estCDq <- data.frame(lambda = lambda, alpha = alpha, beta = beta, gamma = gamma); estCDq

      lambda  alpha  beta  gamma
1 0.01049 0.2632 0.7157 0.0211

#row.names(estCDq) <- "Cobb-Douglas with q: $y = e^{\lambda t}k^{\alpha}l^{\beta}q^{\gamma}$"
row.names(estCDq) <- "CobbDouglas with q"
# The [1] subscripts pick off the lower confidence interval
lowerCDq <- data.frame(lambda = ciCDq["lambda", "2.5%"], alpha = ciCDq["a", "2.5%"], beta = betaCICDq[1], gamma = gammaCICDq[1])
row.names(lowerCDq) <- "- 95% CI"
# The [2] subscripts pick off the upper confidence interval
upperCDq <- data.frame(lambda = ciCDq["lambda", "97.5%"], alpha = ciCDq["a", "97.5%"], beta = betaCICDq[2], gamma = gammaCICDq[2])
row.names(upperCDq) <- "+ 95% CI"

# Now create the data for a table.
dataCDq <- rbind(lowerCDq, estCDq, upperCDq); dataCDq

```

```

          lambda  alpha  beta  gamma
- 95% CI      0.00885 0.1858 0.5947 -0.1353
CobbDouglas with q 0.01049 0.2632 0.7157 0.0211
+ 95% CI      0.01270 0.3283 0.8367 0.1775

colnames(dataCDq) <- c("$\\lambda$", "$\\alpha$", "$\\beta$", "$\\gamma$")
tableCDq <- xtable(dataCDq)

```

```
dataAll <- rbind(dataCD, dataCDq); dataAll
```

```

          $\\lambda$ $\\alpha$ $\\beta$ $\\gamma$
- 95% CI      0.008862    0.2124    0.6722      NA
Cobb-Douglas  0.010256    0.2700    0.7300      NA
+ 95% CI      0.011645    0.3279    0.7878      NA
- 95% CI1     0.008850    0.1858    0.5947    -0.1353
CobbDouglas with q 0.010486    0.2632    0.7157    0.0211
+ 95% CI1     0.012700    0.3283    0.8367    0.1775

```

```
print(xtable(dataCDq), floating=FALSE)
```

	λ	α	β	γ
- 95% CI	0.01	0.19	0.59	-0.14
CobbDouglas with q	0.01	0.26	0.72	0.02
+ 95% CI	0.01	0.33	0.84	0.18

```

# According to http://cran.r-project.org/web/packages/xtable/vignettes/xtableGallery.pdf, Section 3.1, I should
# be able to use the "sanitize.text.function" parameter to allow markup in column headers. But this next
# line is not working at the present time. --MKH, 18 Jan 2012.
# print(tableCDq, sanitize.text.function = function(x){x})

```

```
print(xtable(dataAll, floating=FALSE))
```

	λ	α	β	γ
- 95% CI	0.01	0.21	0.67	
Cobb-Douglas	0.01	0.27	0.73	
+ 95% CI	0.01	0.33	0.79	
- 95% CI1	0.01	0.19	0.59	-0.14
CobbDouglas with q	0.01	0.26	0.72	0.02
+ 95% CI1	0.01	0.33	0.84	0.18

3 Cobb-Douglas With X

```

# 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

# Runs a non-linear least squares fit to the data.
modelCDX <- nls(iGDP ~ exp(lambda*iYear) * iCapStk^alpha * iLabor^beta * iX^(1.0 - alpha - beta),
               start=(list(lambda=lambdaGuess,alpha=alphaGuess,beta=betaGuess)),
               data=dataTable)

# Gives the nls summary table
summary(modelCDX)

```

```

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

Parameters:
      Estimate Std. Error t value Pr(>|t|)
lambda  0.01071    0.00105   10.16 4.6e-11
alpha    0.25689    0.03666    7.01 1.0e-07
beta     0.70011    0.05916   11.83 1.3e-12

Residual standard error: 0.018 on 29 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 2.88e-06

confint(modelCDX, level = 0.95)

Waiting for profiling to be done...

      2.5%    97.5%
lambda 0.008559 0.01287
alpha  0.182137 0.33181
beta   0.578423 0.82111

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

      lambda    alpha    beta    gamma
0.01071 0.25689 0.70011 0.04300

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

[1] -161.4

```

4 Cobb-Douglas With U

An issue arises in this example because constraining 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 - alpha - beta),
  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.

4.1 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 + betaGuess),
  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), gamma=1-max(a,b)))

  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.)

5 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 portion of the function
lambda_EGuess <- 0.008 #assuming no technical progress on the energy portion of the function

# Runs a non-linear least squares fit to the data with constraints
modelCESQ <- nls(iGDP ~ ((1-zeta) * (exp(lambda_L*iYear) * iLabor^beta * iCapStk^(1-beta)))^phi

```



```

+ zeta*(exp(lambda_E*iYear) * iQ)^phi)^(1/phi),
algorithm = "port",
control = nls.control(maxiter = 500, tol = 1e-06, minFactor = 1/1024,
printEval = FALSE, warnOnly = FALSE),
start = list(phi=phiGuess, beta=betaGuess, zeta=zetaGuess, lambda_L=lambda_LGuess,
lambda_E=lambda_EGuess),
lower = list(phi=-Inf, beta=0, zeta=0, lambda_L=-Inf, lambda_E=-Inf),
upper = list(phi=0, beta=1, zeta=1, lambda_L=Inf, lambda_E=Inf),
data=dataTable)

# Gives the nls summary table
summary(modelCESQ)

```

Formula: $iGDP \sim ((1 - zeta) * (\exp(\lambda_L * iYear) * iLabor^\beta * iCapStk^{(1 - \beta)})^\phi + zeta * (\exp(\lambda_E * iYear) * iQ)^\phi)^{1/\phi}$

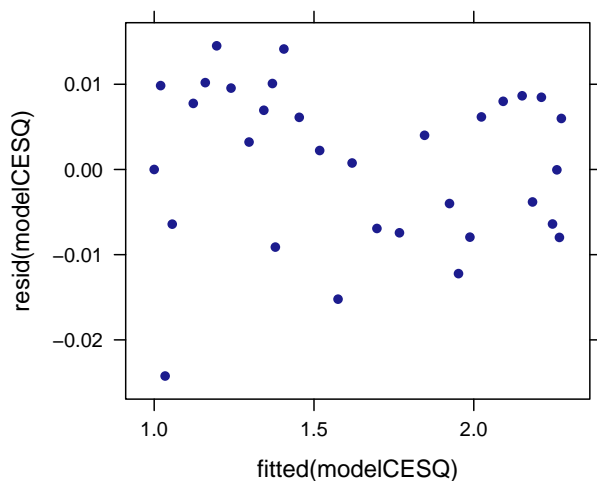
Parameters:

	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

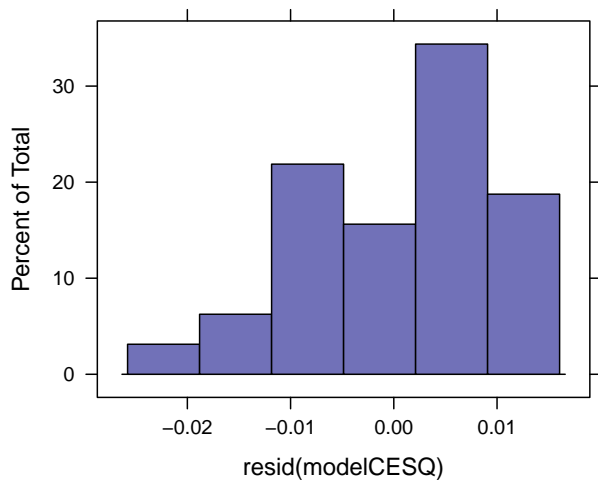
Residual standard error: 0.00993 on 27 degrees of freedom

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

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



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



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

