Chapter 1

Production Function and Regression Methods Using R

As an example of econometric and statistical methods, production function estimation has several useful features as it brings together economic theory and statistical methods. This chapter will exploit those features to introduce (i) elementary econometric methods of thinking about economics, (ii) some basic statistical tools, and (iii) some basics of the R software system. We assume that the reader already has some familiarity with all three topics. The URLs (http://www.r-project.org/, http://wiki.r-project.org/rwiki/doku.php, and http://gking.harvard.edu/zelig/) are suitable for understanding the basics of R.

Note that R is an object-oriented language. Command lines are interpreted one by one or in sets defined by curly braces. This book includes several collections of command lines, called snippets, mostly identified by `#Rc.s.n`, where c is chapter number, s is section number, and n is snippet number. All command lines are typeset in a distinct typewriter (Courier) font. R uses the # symbol to represent the start of a "comment" or explanatory note. R ignores everything after the # symbol. We ask the reader to try to understand R commands in snippets. Do please copy and paste them into your own R-GUI (graphical user interface) or RStudio.

Since R is free, one can download the latest version of R and packages on a home computer. This book offers a "hands-on" experience with learning R and econometrics. The reader is encouraged to copy and paste the R code to re-create most of the book's results and graphs at home. Our snippets include several # comments to indicate what a nearby command does. The book's website allows downloading the snippet codes. Most snippets are human-readable text files named `Rc.s.n.txt`, where c=chapter number, s=section number. See https://www.worldscientific.com/worldscibooks/10.1142/6895. We recommend that readers open the snippets into RStudio (https://www.rstudio.com) or an MSWord document, try to understand each line of code by slightly changing it. You will need a lot of cuts and paste if you use MSWord. RStudio allows running parts of code by highlighting it, and clicking on the Run button. The snippets are templates to be modified, perhaps extended, and improved to apply to some other data/issue. Hence, some typing is unavoidable. Many of my students seem to enjoy the challenge of trying to modify/improve my snippets. The snippets in this chapter dealing with the break-up of the Bell Telephone monopoly in 1982 provide hands-on experience with microeconometric tools and collinearity-type issues.

1.1. R and Microeconometric Preliminaries

The theory of the firm from microeconomics includes a range of concepts related to the production of a physical object (e.g., corn) or a service output provided by a firm or an entrepreneur. We describe the production process as a production function, usually involving at least two inputs, capital (denoted by K) and labor (denoted by L). Since inputs and the output may have several components, it is customary to use "quantity" index numbers to represent their aggregates. The "value-added" index of output denoted by y refers to all enhancements of a product or service made by a producer before selling it to customers. A production function is defined as

$$y = f(K, L), \tag{1.1.1}$$

a single-valued mathematical function. The famous Cobb–Douglas production function: $y = f(K, L) = AK^{\alpha}L^{\beta}$, where A, α, and β are parameters (numbers), is discussed later in Eq. (1.1.4).

We assume that the reader has already visited (http://www.r-project.org/) and clicked on CRAN in the left column. Then, choose a mirror, look for the box "Download and Install R", pick the computer type, e.g., Windows (8 and later), and click on it. Assuming windows

PC, next click on "base", Click on "R-3.6.1-win.exe" (or a similar name) and successfully set up R. Now click on the R icon and get R-GUI, with R-Console and a command prompt ">" showing. If not, go back to the above website and get more detailed directions for getting started. The following snippet cleans up R from previous use and gives a date–time stamp, so one knows exactly when the particular run was made:

```
#R1.1.1 Recommended Start of a session of R software
objects() # this command lists all objects already
#lurking in the memory of R
rm(list=ls()) #rm means remove them all to clean up old stuff

print(paste("Following executed on", date()))
#date−time stamp
```

1.1.1. *Data on metals production available in R*

Our hands-on example considers production of metal by 27 regional US firms belonging to the Standard Industrial Classification Code (SIC) 33. Our data are readily available in R with the following command snippet:

```
#R1.1.2 Read production function data already in R into the
#memory
library(Ecdat) #pulls into the current memory of R
# Ecdat and fBasics packages
help(Metal)# gives further info about the dataset Metal,
#sources etc.
data(Metal)#pulls the Metal data into memory of R
names(Metal)#provides the names of series in various
#columns
summary(Metal) #provides descriptive stats of the data
met=as.matrix(Metal)#we create an object called met
Ly=log(met[,1])#pull first column of met, take log, define Ly
LL=log(met[,2])#pull second col. of met, take log, define LL
LK=log(met[,3])#pull third col. of met, take log, define LK
datmtx=cbind(Ly,LK, LL) #bind columns into a matrix
fBasics::basicStats(datmtx)
matplot(cbind(Ly, LK, LL), typ="b", pch=c("y","K","L"),
main="Metals Data Description", xlab="Observation Number",
ylab="Logs of output, capital, and labor")
```

The command `fBasics::basicStats(datmtx)` containing the double colon (::) asks R to pull the R-package called "fBasics" and then use the command "basicStats(.)" function, avoiding the command `library(fBasics)`. However, (::) will not work if "fBasics" is not installed. The package and function names routinely exploit the fact that R is case-sensitive. For example, lower case c followed by upper case S in "basicStats" help separate the two words in its name. The command `matplot(cbind(Ly, LK, LL),` refers to plotting a matrix (mtx) created by binding the indicated three columns into a matrix by the R function "cbind(.)". The options within a command are separated by commas. The plot-type choice "b" refers to both line and print characters listed as a vector c(.,.) of characters distinguished by straight (not fancy) quotes: `typ="b", pch=c("y","K","L")`. The plotting option "main" refers to the main header, whereas the axis labels are "xlab, ylab".

1.1.2. *Descriptive statistics using R*

One should always study the basic descriptive statistics of available data before fitting any model. See the output in Table 1.1, where the abbreviations in the first column are standard: nobs=number of observations=T in our example, NAs=number of missing values. 50% data are both below and above the median. The first quartile has 25% data below it and 75% above it. The third quartile has 75% below it and 25% above it.

SE (mean) in Table 1.1 refers to the standard error of the sample mean, which is the standard deviation of the sampling distribution. Table 1.1 reports SE of the mean. Think of the sample mean as a random variable and consider the mean of all possible samples of size T as the sample space. The sample mean is a random variable defined on this sample space, in the sense that we compute all possible samples and then all possible sample means. This is the sampling distribution of the mean. By the central limit theorem (CLT) this distribution is Normal. The standard deviation of this sampling distribution is the standard error, SE. The descriptive statistics in R include the LCL and UCL as the lower and upper confidence limits, respectively, of a 95% confidence interval on the mean. It also reports skewness and kurtosis, which are also defined in snippet #R1.1.3 and described in the following.

Table 1.1. Basic descriptive statistics, metals data.

Description	Ly	LK	LL
Nobs	27	27	27
NAs	0	0	0
Minimum	6.384941	5.634754	4.919981
Maximum	9.195142	9.546066	7.355532
1st Quartile	6.935645	6.736932	5.360868
3rd Quartile	7.928982	8.063829	6.233483
Mean	7.443631	7.445922	5.763652
Median	7.378996	7.436605	5.560335
Sum	200.978045	201.03991	155.6186
SE (mean)	0.146484	0.186384	0.126293
LCL (mean)	7.142529	7.062804	5.504052
UCL (mean)	7.744733	7.829041	6.023252
Variance	0.579354	0.937957	0.430651
Std dev	0.761153	0.968482	0.65624
Skewness	0.613768	0.231518	0.701836
Kurtosis	−0.522245	−0.677601	−0.49553

1.1.3. *Writing skewness and kurtosis functions in R*

Let population central moments of order j be denoted by $\mu_j = \sum_{i=1}^{T}(x_i - \bar{x})^j$. Skewness measures the nature of asymmetry in the data. Positively skewed density has outcomes above the mean more likely than outcomes below the mean. Negatively skewed is similarly defined. Kurtosis measures the degree to which extreme outcomes in the tails of a distribution are likely. Pearson's measures of skewness and kurtosis are $\text{skew}_\text{p} = (\mu_3)^2/(\mu_2)^3$ and $\text{kurt}_\text{p} = \mu_4/(\mu_2)^2$. Since Pearson's formula does not distinguish between positive and negative skewness, R package "fBasics" uses a more informative signed square root, $\text{skew} = (\mu_3)/\sigma^3$, where $\mu_2 = \sigma^2$.

The Normal distribution has zero skewness: $\text{skew}_\text{p} = 0 = \text{skew}$. Since the kurtosis of the Normal is 3, R package "fBasics" subtracts 3 to define "excess kurtosis" as: $\text{kurt} = [\mu_4/(\mu_2)^2 - 3]$. If $\text{kurt}_P > 3$, i.e., if $\text{kurt} > 0$, the distribution is more peaked than Normal near the mode (maximum frequency) of the distribution and has thicker (fatter) tails.

In Table 1.1, all variables have $\text{kurt} < 0$, suggesting that their probability distributions are flatter than the Normal. The sample estimates of skewness and kurtosis use sample moments defined by the usual unbiased estimates. Snippet #R1.1.3 illustrates how to write R functions (algorithms) for computing skewness and kurtosis, whose definitions involve third and fourth powers. Download the R package "moments". Issuing commands `moments::skewness` and `moments::kurtosis` on the R console prints to screen more sophisticated versions of these algorithms. Packaged functions often minimize computing time, but can be hard to understand.

```
#R1.1.3 New Functions in R to compute skewness and kurtosis
 myskew <- function(x) {
 m3 <- mean((x-mean(x)) ^3) #mean is already a function in R
 skew <- m3/(sd(x) ^3) #std dev in R is given by sd
 list(skew=skew) #typical output of a function
 }  #functions begin and end with curly braces

 mykurt <- function(x) {
 m4 <- mean((x-mean(x)) ^4)
 kurt <- m4/(sd(x) ^4)-3
 list(kurt=kurt)}
#example x=c(1,2,3,4,20); myskew(x);mykurt(x)
#library(fBasics);basicStats(x)
```

Since high powers induce higher variability, sample estimates of skewness and kurtosis are subject to a rather large sampling variation, especially in small samples. Hence, small sample estimates of skewness and kurtosis can be unreliable.

Section 1.14 provides some hints on analyzing a table of basic descriptive statistics similar to Table 1.1. It helps detect hidden data or coding errors. One should study the table carefully to see if the data, as understood by the computer program, are exactly what the researcher has in mind. The number of observations (nobs) should be equal for each series, and missing data (NAs) should not be too many. The table also reports the 95% confidence interval (LCL and UCL) for the mean of each series. One should check whether these ranges make sense. If the median is too far from the mean, it indicates skewness. One should make sure that the skewness and kurtosis are not caused by outliers or data errors. If the means and standard deviations of different regressors are too different from each other, a certain rescaling of data may be needed as discussed in Section 1.1.4.

1.1.4. *Measurement units and numerically reliable β*

If standard deviations of different regressors are too different from each other, it can be because the units of measurement of different regressors are misaligned. It is advisable to have regressors with numerically similar regressors in units. But this may not always be possible since some variables like interest rates move in narrow bands while others like GDP are large. Still, it is possible to measure GDP in billions or tens of billions to make units closer to other variables (like interest rates) in the model. The numerical reliability of regression coefficients and standard errors is enhanced if the units are not too dissimilar. The graphics in Section 1.1.5 will not look right if the levels and units are too dissimilar. The adjustment for levels is adding a constant, and adjustment for scale is usually the multiplication of a column of data by 10^x for some suitable x. Good software programs like R are not too sensitive to units because they use robust versions of matrix inverse and other sophisticated tools.

1.1.5. *Basic graphics in R*

The command `plot(.)` of R is very powerful for all kinds of plots, and we suggest the reader complete many examples seen upon issuing the command `?plot`. Recall our discussion of the command `matplot(.)` in the snippet #R1.1.2, where there are only 27 observations. In general, it is important that the matrix argument of the R command "matplot" has columns with similar magnitudes (Fig. 1.1). Otherwise, "matplot" will not be able to select proper scaling for the axes.

Consider a three-dimensional (3D) space having the usual x and y axes for K and L of metals data (say) with the third (height) dimension along the new z-axis for the value-added output of production denoted by y. We want the plot to look like a surface, not a set of $n = 27$ tall pins. An R function called `pillar3D` in the R package "generalCorr" fattens each data point to create some extra points near each observed y providing suitable x, y coordinates as the base of our pillars. See Fig. 1.2 for an example of a pillar chart.

```
#R1.1.4 pillar plots representing 3D data
library(Ecdat); data(Metal); library(generalCorr)
y=Metal$va; L=Metal$labor; K=Metal$capital
pillar3D(L,K,y, drape=FALSE,
```

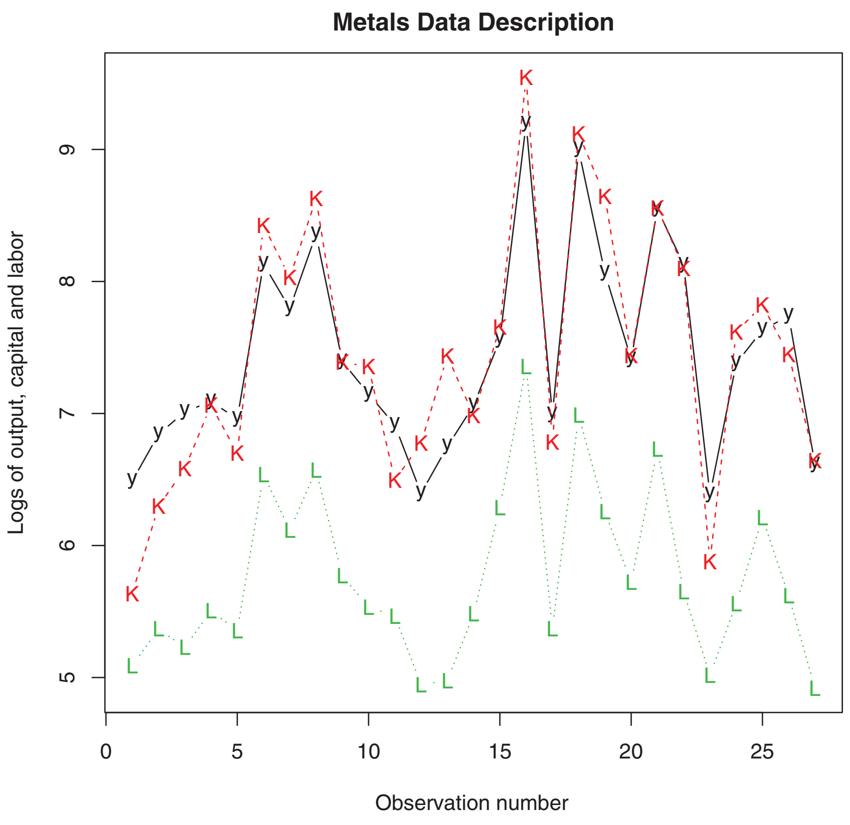

Fig. 1.1. Metals data description.

```
xlab="Labor", ylab="Capital", zlab="Value added",
mymain="Pillar Chart for Metals Production")
library(car);Ly=log(y);LK=log(K);LL=log(L)
scatter3d(LL,Ly,LK, fit="quadratic")
```

The last two lines of the code above create a scatterplot with a quadratic surface representing a fitted production function. The distance between a point and the plane represents residuals of the regression. Of course, some y points depicted in dark shade are below and others are above the surface. The 3D graph created by the "car" package can be re-oriented by moving the mouse on it. Unfortunately, one cannot save its high-quality copy for illustrating here. However, a reader can view it on local R.

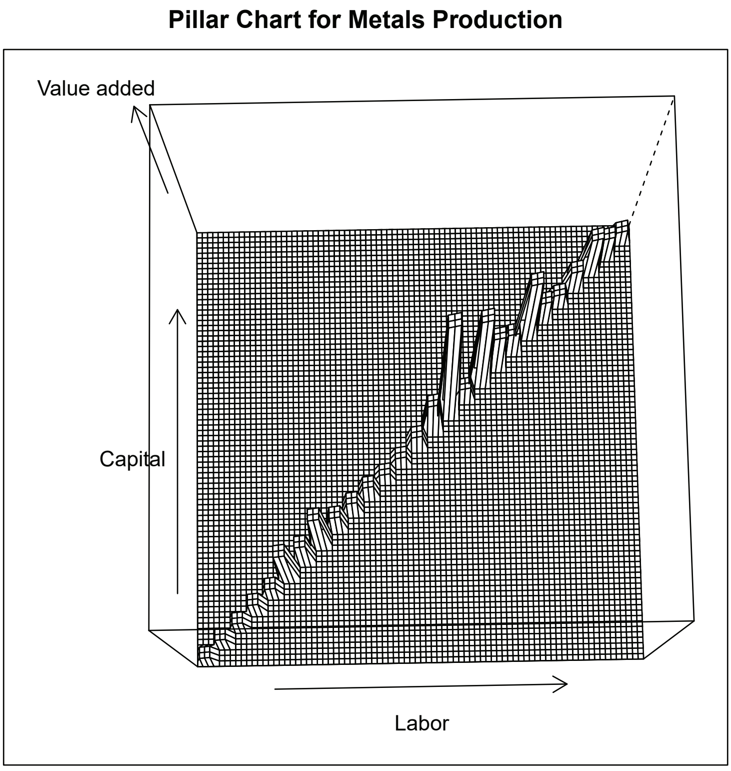

Fig. 1.2. Metals data pillar chart.

1.1.6. *The isoquant*

A 3D surface representing all potential combinations of inputs and outputs is the geometric representation of a mathematical production "function". The isoquant is defined as the level curve (contour line) where the level of output is fixed along the curve and all input combinations give the same (iso) quantity (quant) of output. Its shape is usually a rectangular hyperbola (reverse J shape, similar to indifference curves). The reader can consult almost any Economics text for examples of these plots. We cite Ferguson (1971) throughout in what follows, because his coverage is comprehensive, although our notation is different from his and our focus is on empirical estimation using R software. The isoquants for the homogeneous Cobb–Douglas function are all parallel to each other, and not particularly interesting. Vinod

(J1972a) was perhaps the first paper to plot them for an important real-world economic example, with the use of multiplicative non-homogeneous (MNH) production function. We plot empirical isoquants later in Fig. 1.8.

1.1.7. *Total productivity of an input*

Given the production function $f(K, L)$, holding one input fixed, $K = K_0$, the total productivity of the other input (labor) is simply obtained by plugging in different values of L in the analytical functional form f. We have

$$\mathrm{TP_L} = f(L, K_0) \quad \text{and} \quad \mathrm{TP_K} = f(L_0, K). \tag{1.1.2}$$

Wherever possible, we try to use self-explanatory notation, TP for total productivity, L for labor, K for capital, etc.

1.1.8. *The marginal productivity (MP) of an input*

The marginal productivity of one input (L or K) is given by the partial derivative of f with respect to the input. The partial derivative can be unconditionally available for all values of the other input. Depending on the functional form, the partial derivative can be a function of the level of the other input. In that case, the marginal productivity can be evaluated only at specified values of the other input similar to L_0 and K_0 above. In general, we have the following marginal productivities:

$$\mathrm{MP}_L = (\partial f/\partial L) \text{ at } K = K_0 \quad \text{and} \quad \mathrm{MP}_K = (\partial f/\partial K) \text{ at } L = L_0. \tag{1.1.3}$$

1.1.9. *Slope of the isoquant and MRTS*

Let Δi denote a finite change in a magnitude of i, which is used to approximate the differential (or partial differential) di, for $i = K, L$. We depict capital K along the vertical axis and L along the horizontal axis in the isoquant map. The slope of the isoquant between two adjacent points on it is the negative of the ratio of input changes $[-\Delta K/\Delta L]$. Since marginal products are related to change in inputs, the slope of the isoquant also equals the ratio (MP_L/MP_K). It is also known as the marginal rate of technical substitution (MRTS_{LK}).

A Cobb–Douglas production function is sometimes given as $y = f(K, L) = AK^{\alpha}L^{(1-\alpha)}$ in Economics texts and is called linearly homogeneous. More generally, it is stated as

$$y = AK^{\alpha}L^{\beta}, \tag{1.1.4}$$

which is a nonlinear functional form. Given data on y, K, and L, as 27×1 vectors, we estimate the production function (1.1.4), which amounts to estimating the unknown parameters A, α, and β, called (regression) coefficients. We should not expect a perfect fit, even if we are willing to make some assumptions needed by statistical regression theory. Most such assumptions are regarding the lack of fit or error ε defined by

$$y = AK^{\alpha}L^{\beta} + \varepsilon. \tag{1.1.5}$$

For example, statisticians often assume that errors are zero on an average: $E(\varepsilon) = 0$, where E denotes the expectation operator. The parameters of (1.1.5) can be estimated by calling the function "nls" of R software, where nls stands for nonlinear least-squares illustrated in a snippet #R1.2.1 in the next section. The "nls" minimizes the error sum of squares "locally" even if the function is highly nonlinear. Unfortunately, most available nonlinear minimization algorithms can only find the local minimum. A further disadvantage of nonlinear estimation is that the user has to provide starting values for the unknown parameters A, α, and β. We recommend using OLS estimates of a linearized version of the model as starting values. In general, a whole range of distinct starting values should be attempted, so that the algorithm does not provide wrong estimates of the coefficients while missing the global minimum. We shall see that estimates of α and β from the nonlinear and the following "linear in logs" models do not agree in relative magnitudes.

Until recently, nonlinear estimation was not available in standard software. Hence, estimation of the parameters α and β of the Cobb–Douglas used to proceed after making it "linear in parameters" by using the log transform of both sides of (1.1.4). We have

$$\log y = \log A + \alpha \log K + \beta \log L. \tag{1.1.6}$$

Note that it is customary to insert the error term after taking the log transform as in (1.1.6) and then to estimate the parameters by linear regression.

Now, consider "metals" data, where we have cross-sectional observations on y, K, and L. We define the marginal elasticity of input as the percentage change in output from a 1% change in the input. The percent change (or proportionate change) is conveniently given by partials of (1.1.6) involving logs.

$$\begin{aligned} \mathrm{ME}_L &= (\partial \log y / \partial \log L) \text{ at } K = K_0, \quad \text{and} \\ \mathrm{ME}_K &= (\partial \log y / \partial \log K) \text{ at } L = L_0. \end{aligned} \tag{1.1.7}$$

1.1.10. *Scale elasticity as the returns to scale parameter*

Economists are generally interested in the scale elasticity (SCE) of a production function. The SCE is defined as the scale change, often measured by the percent change in output from a simultaneous 1% change in all inputs. Note that SCE is an important measure of the efficiency of production. If SCE = 1, the production process is subject to constant returns to scale. If SCE >1, the production process is said to be particularly efficient and exhibits "economies of scale". A possible implication of this is that larger production facilities will be more efficient than small-scale operations. The parametric measure of scale elasticity is available from the functional form of the production function f. We have for the Cobb–Douglas:

$$\mathrm{SCE} = \mathrm{ME}_K + \mathrm{ME}_L = \alpha + \beta. \tag{1.1.8}$$

Thus, marginal elasticities are more useful for some purposes than marginal productivities. Since $(\partial \log y/\partial y) = (1/\mathrm{y})$ holds, there is a simple definitional identity between marginal elasticity and marginal productivity: $\mathrm{ME}_L = (\partial \log y/\partial \log L) = (L/y)(\partial y/\partial L) = (L/y)\mathrm{MP}_L$. Hence,

$$\begin{aligned} \mathrm{MP}_K &= (y/K)\mathrm{ME}_K = y\alpha/K, \quad \text{and} \\ \mathrm{MP}_L &= (y/L)\mathrm{ME}_L = (y\beta/L). \end{aligned} \tag{1.1.9}$$

The following standard derivations are provided for convenience and can be skipped by most readers:

$$\begin{aligned} y &= AK^{\alpha}L^{\beta}, \\ \mathrm{MP}_K &= \partial y/\partial K = \alpha AK^{\alpha-1}L^{\beta}, \\ \mathrm{MP}_L &= \partial y/\partial L = \beta AK^{\alpha}L^{\beta-1}, \end{aligned}$$

$$\begin{aligned} &\log y = \log A + \alpha \log K + \beta \log L, \\ &f(x) = \ln g(x) \text{ implies that } f'(x) = g'(x)/g(x), \end{aligned}$$

$$\begin{aligned} \mathrm{ME}_L &= (\partial \log y/\partial \log L) = (L/y)(\partial y/\partial L) = (L/y)\mathrm{MP}_L \\ &= L/y\beta AK^{\alpha}L^{\beta-1} = (L/AK^{\alpha}L^{\beta}) * \beta AK^{\alpha}L^{\beta-1} \\ &= \beta AK^{\alpha}L^{\beta-1}L/AK^{\alpha}L^{\beta} = \beta. \end{aligned}$$

Then,

$$\begin{aligned} \mathrm{MP}_L &= \mathrm{ME}_L(y/L) = y\beta/L, \\ \mathrm{ME}_K &= (\partial \log y/\partial \log K) = (K/y)(\partial y/\partial K) = (K/y)\mathrm{MP}_K \\ &= (K\alpha AK^{\alpha-1}L^{\beta})/y = \alpha, \\ \mathrm{ME}_K &= (K/y)\mathrm{MP}_K \text{ implies that } \mathrm{MP}_K = \mathrm{ME}_K(y/K) = \alpha y/K. \end{aligned}$$

1.1.11. *Elasticity of substitution*

Similar to the scale elasticity, economists are interested in elasticity of substitution (EOS), which measures the ease with which one input can be substituted by another. It is viewed as a property of the functional form. The Cobb–Douglas form implies that EOS $= 1$, suggesting considerable ease of substitution. Recall the MRTS $=$ $(\mathrm{MP}_L/\mathrm{MP}_K)$ is the slope of the isoquant. If z denotes the input ratio (K/L), the EOS $= d(\mathrm{MRTS})/\mathrm{d}z$, the derivative of the MRTS with respect to the input ratio.

It is more convenient to define EOS by first defining a few quantities involving second-order partial derivatives of the production function f denoted by two subscripts on f and re-defining the marginal productivities as $\mathrm{MP}_K = \mathrm{f}_K$, $\mathrm{MP}_L = \mathrm{f}_L$.

The following "radical" (denoted by D) involving second-order partials of the production function mixed with the products of two first-order partials plays an important role in the microeconomic theory:

$$\mathrm{D} = 2f_{LK}f_Lf_K - (f_L)^2f_{KK} - (f_K)^2f_{LL}, \tag{1.1.10}$$

$$\mathrm{EOS} = f_Lf_K(f_LL + f_KK)/(LKD). \tag{1.1.11}$$

The important point to note is that the quantities of interest to economists are measured by the parameters α and β of the functional form of the production function $f(K, L)$.

The Cobb–Douglas form is suitable for the estimation by a linear regression model if we insert model errors only after the log transformation (1.1.6). After adding the error term, we have the following linear regression model:

$$\log(y) = \log A + \alpha \log K + \beta \log L + \varepsilon. \tag{1.1.12}$$

It is useful to discuss the linear regression model in more general terms in the next subsection for future reference. Note that EOS is not empirically estimable for the Cobb–Douglas form, but rather always fixed at 1. If the substitution might not be constant at unity, one should not rely on the Cobb–Douglas functional form.

1.1.12. *Typical steps in empirical work*

The production function example illustrates typical steps in empirical work in social sciences.

(i) First, specify the suitable functional form of a relationship among the variables of interest, such as the production function.
(ii) Next, collect suitable proxy data variables for outputs and inputs (y, K, and L). Use methods of descriptive statistics to understand the data with an eye on possibly hidden data and measurement errors.
(iii) Now, choose the equation to be estimated. If the researcher truly believes that the data follow the nonlinear Cobb–Douglas form, he should estimate the nonlinear (1.1.5). When we use (1.1.12) instead of the nonlinear form, we are making a compromise for the convenience of statistical estimation. This illustrates how the estimation of parameters often involves some compromises.

(iv) Using the parameters and data, one finally estimates the quantities of economic interest such as MEs, SCE, MPs, and EOS.

1.2. Preliminary Regression Theory: Results Using R

The reader is encouraged to be familiar with the following standard statistical notation, including matrix algebra. Consider the standard linear regression model with y as output or dependent variable, x_1 to x_p as inputs or regressor variables, ε as the error term. The intercept β_0 is the point on y axis where the line of regression meets. Denote by β_1 to β_p the regression slope coefficients. The reader must learn the regression model in matrix notation. In my experience, students learn better by working with small examples while making all matrices concrete (almost tangible) with the help of R software. For example, the R command for row–column matrix multiplication of conformable matrices is `A%*%B`.

Let us write the model in matrix notation as

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon = X\beta + \varepsilon. \tag{1.2.1}$$

Assuming that there are T observations, X is the $T \times (p+1)$ matrix of data on all regressor variables including the first column of ones to represent the intercept β_0, y is a $T \times 1$ vector of data on the dependent variable, β is a $(p+1) \times 1$ vector of regression coefficients, and ε is a $T \times 1$ vector of unknown true errors.

Some authors incorporate the intercept into the column vector of coefficients and treat β as $p \times 1$ vector without loss of generality. Also, any nonlinear model, which is linear in parameters, can be accommodated in (1.2.1) by having the nonlinear functions (e.g., logs, powers) as regressors in the X matrix. For example, the Cobb–Douglas production function (1.1.12) is linear in parameters, where the variables are replaced by their log transforms and $p = 2$. A polynomial of order p is also linear in parameters by choosing the columns of X to have various powers of the regressor.

Standard statistical theory of linear regression assumes the following semi-parametric (when normality is not assumed) probabilistic structure for y and errors:

$$y = X\beta + \varepsilon, \quad E(\varepsilon) = 0, \quad E(\varepsilon\varepsilon') = \sigma^2 I_T, \tag{1.2.2}$$

where I_T is a $T \times T$ identity matrix, suggesting constant variances and zero covariances among errors. The variances are along the diagonal and parametrize the heteroscedasticity (if any), and covariances are off-diagonal and reveal autocorrelated errors (if any). Thus, (1.2.2) assumes that the errors do not have any autocorrelation or heteroscedasticity problems.

From (1.2.2) verify that $E(y) = X\beta$ is also the conditional mean $E(y|X)$. If we further assume that (a) the X matrix of regressors has full column rank, (b) all columns of X are uncorrelated with errors, and (c) that y (and ε) are multivariate Normal, $y \sim N(\mu\ ,\sigma^2 I)$, then we can do standard statistical testing and inference. Without the (normality) distributional assumption, the usual t-tests and F tests on coefficients and overall model are not available. In particular, let the tth error ε_t have the following Normal density:

$$f(\varepsilon_t) = (2\pi)^{-1/2}\sigma^{-1}\exp\{-(\varepsilon_t'\varepsilon_t)/2\sigma^2\}, \tag{1.2.3}$$

where ε_t is the tth element of the vector $(y - X\beta)$ and is known only when β is known. Hence, the density $f(\varepsilon_t)$ is a function of data on y, X at time t, *given* specified numerical values of β and σ^2. The density is usually written as $f(y_t, X_t|\beta,\ \sigma^2)$, where we assume that the numerical values of all items after the vertical bar are known. The joint density for all T observations is the product of such densities for all t from $t = 1$ to $t = T$:

$$f_{\text{joint}}(y, X|\beta, \sigma^2) = \prod_{t=1}^{T} f(\varepsilon_t), \tag{1.2.4}$$

where $f(\varepsilon_t)$ is defined in (1.2.3) and does depend on β through $(y - X\beta)$.

The idea of the likelihood function is a great contribution of Fisher. It is obtained by rewriting (re-interpreting) the same joint density function $f_{\text{joint}}(\cdot)$ as $f_{lkhd}(\beta, \sigma^2|y, X)$. Its log is called the log-likelihood (LL) function. It is useful when the object is to find the unknown parameters from the given data. For the regression model (1.2.2), we are not assuming the form of the density of ε. However, we know the mean and variance–covariance matrix. A quasi-log-likelihood function can be defined from the assumptions of (1.2.2). Both the quasi and the usual LL by expanding the product in (1.2.4) after using (1.2.3) are given by

$$\text{LL} = -(T/2)\log(2\pi\sigma^2) + \varepsilon'\varepsilon, \tag{1.2.5}$$

where

$$\varepsilon'\varepsilon = (y - X\beta)'(y - X\beta) = y'y - 2\beta'X'y + \beta'X'X\beta. \tag{1.2.6}$$

It is well known that the ordinary least squares (OLS) estimator of the parameter vector β is obtained by minimizing the error sum of squares without any reference to a likelihood function. We can directly minimize $\varepsilon'\varepsilon$ with respect to β. The matrix derivative of $2\beta'X'y$ with respect to β is $2X'y$, and the derivative of the quadratic form $\beta'X'X\beta$ is $2X'X\beta$. Set the derivative equal to zero to get the first-order condition (FOC) given by $-2X'y + 2X'X\beta = 0$. Now, cancel the "2" and solve for β. Finally, we have the OLS estimator b as the solution:

$$b = (X'X)^{-1}X'y. \tag{1.2.7}$$

The *score vector* is an important concept in more advanced statistical theory based on the FOC for maximization of log (quasi) likelihood. In fact, the score vector is defined as the derivative of the LL function written as

$$X'(y - X\beta) = X'\varepsilon. \tag{1.2.8}$$

The second-order condition for a maximum from calculus is that the matrix of second-order partial derivatives should be negative definite. The maximum likelihood (ML) estimator of the β vector satisfies both conditions.

Now, we describe the OLS solution (1.2.7) from the regression theory by using R. Let us use the metals production function discussed in Section 1.1.1, without loss of generality. The vector b is created in R software by the command `lm(.)` for the linear model. R is an object-oriented language in the sense that it readily creates objects in the memory of the computer. Snippet #R1.2.1 lists the R commands for multiple regression.

```
#R1.2.1 Linear regression Metals data already in R
#Task 1 get the data in computer memory
library(Ecdat); data(Metal)
```

```
met=as.matrix(Metal) #we want to give our own names to
#variables
#Task 2 transform the data
Ly=log(met[,1])#pull first column of met, take log, define Ly
LL=log(met[,2])#pull second col. of met, take log, define LL
LK=log(met[,3])#pull third col. of met, take log, define LK
# Task 3 understand data with descriptive stats done above
# Task 4 regress Ly on LK and LL see Eq. (1.1.6)
reg1 = lm(Ly~LK+LL); reg1; names(reg1)
#Type help(lm) for details
#reg1 is an object in R created by fitting the regression
#We see coefficients b by using the command reg1$coef
#(Intercept)=1.1706, coeff of LK=0.3757, coeff of LL =0.6030
summary(reg1)# this gives full details omitted for brevity.
#begin nonlinear estimation of Eq. (1.1.5)
y=met[,1]; L=met[,2]; K=met[,3]
regnon=nls(y~A*(K^alp)*(L^bet),
start=list(A=1.17,alp=0.4,bet=0.6)) #input start values
summary(regnon) #NonLinear coefficient estimates
```

1.2.1. *Regression as an object "reg1" in R*

In Task 4 of snippet #R1.2.1, we use the `lm(.)` command to create an object called reg1 by the R command: reg1 $=$ lm$(Ly \sim LK + LL)$, where the dependent variable Ly appears on the left side of "$\sim$" and the regressors are listed on the right-side separated by a "+" symbol. The reader can get a great deal of information about how "lm" works by typing the command "?lm". One can view the contents of the user-created R object on the screen by simply typing its name on the R console. An extremely powerful feature of R, further described in the next section, is the availability of the dollar symbols to extract named objects from existing R objects for further manipulation.

1.2.2. *Accessing objects within an R object by using the dollar symbol*

The object "reg1" created in Task 4 above now contains all kinds of results of the regression. Typing the command "names(reg1)" one can find out the names of all 12 objects within the object reg1. For example, there are objects for "coefficients", "residuals", and "fitted values". Merely typing reg1 does not list the results of all 12 objects. The programmer has chosen to display the calling formula and coefficients. The command "reg1\$coefficients" will extract only the coefficients. Note that R is smart and knows that users do not like to type a lot if they do not have to. Accordingly, R recognizes the name as long as we type enough characters to identify the name uniquely. For example, "reg1\$c" is not enough, but reg1\$co is enough for coefficients. It also accepts subscripts for extraction. For example, "reg1\$co[2]" will extract only the second coefficient.

The command "summary(reg1)" at the end of the snippet provides a standard output of regression. If instead of printing the output, one wants to use it for further analysis or reporting, one uses the command: sureg1=summary(reg1). This creates an object called sureg1, and `names(sureg1)` will list the names of 11 objects within sureg1. All of these are accessible by using the dollar symbol convention of R. For example, sureg1\$fstatistic has three items: the value of the F statistic, numerator, and denominator degrees of freedom. `sureg1$coe` extracts a matrix with columns for (i) coefficient estimate, (ii) Standard Error, (iii) Student's t value, and (iv) the p-value for that coefficient. See Table 1.2 for an example using Metals data.

Table 1.2. Cobb–Douglas production function estimation for metals.

	Estimate	Std. error	t value	Pr(>|t|)
(Intercept)	1.1706	0.3268	3.58	0.0015
log K	0.3757	0.0853	4.40	0.0002
log L	0.6030	0.1260	4.79	0.0001
A	2.7361	0.9212	2.970	0.006662
α	0.5509	0.1331	4.138	0.000371
β	0.4036	0.1667	2.422	0.023372

Notes: Equation (1.1.6) in the upper panel and (1.1.5) in the lower. Residual standard error: 0.1884 on 24 degrees of freedom (df). Multiple R^2 0.9435. Adjusted R^2: 0.9388, F-statistic: 200.2 on 2 and 24 DF. p-value: 1.067e-15. Lower panel for nonlinear fit, which is achieved by Residual standard error: 467.3 on 24 DF and convergence tolerance: 2.636e-07 after only four iterations. The nonlinear estimate is robust after several starting values. It is strange that the relative sizes of estimated α and β are opposite in the two panels.

1.3. Deeper Regression Theory: Diagonals of the Hat Matrix

The fitted values of the regression model are given in matrix notation by Xb. If we replace b by the expression in (1.2.7), we have

$$\hat{y} = \text{fitted}(y) = Xb = X(X'X)^{-1}X'y = Hy, \tag{1.3.1}$$

which defines the hat matrix $H = X(X'X)^{-1}X'$. Any matrix is a linear mapping or transformation. When we do the matrix multiplication Hy, we imagine *placing a hat* on y to get the vector of fitted y values. The algebraic properties of H are interesting. It is symmetric (transpose of H equals H) and idempotent (squaring of H gives back H). The matrix expression Hy is imagined as the linear map H operating on a vector y.

The diagonal elements of the hat matrix, denoted by h_{tt}, represent the leverage, which shows the sensitivity of the regression line to tth individual observation. The leverage is very important in studying the regression fit. **Scaled Leverage** is defined as $h_{tt}/(1 - h_{tt})$. One limitation of H is that it focuses only on the "influence" of the X matrix on the regression line while ignoring that of the dependent variable y.

```
#R1.3.1 Regression fitted values and residuals, descriptive
#stats. Note that fitted is a useful function (operator) in R
# Run this only after all commands of R1.2.1 are run
fitted(reg1); summary(fitted(reg1)) #descriptive stats
resid(reg1); summary(resid(reg1))
```

Similar to the hat matrix M (=identity minus the hat matrix) matrix is defined in (1.3.2) with similar properties (idempotent and symmetric). Both are called projection matrices. The matrix projection M operating on y gives the residuals of linear regression. The $T \times 1$ vector of regression residuals measures the lack-of-fit and is given by

$$r = y - \hat{y} = (I - X(X'X)^{-1}X')y = My, \tag{1.3.2}$$

which play an important role in statistical diagnostics and inference. These are readily fitted and plotted in various ways in modern computing environments. The residual variance is defined from the residual sum of squares (SSR) as

$$s^2 = \text{SSR}/(T-p) = (T-p)^{-1}\sum_{t=1}^{T}(r_t)^2, \tag{1.3.3}$$

where the denominator $(T-p)$ represents the degrees of freedom (df). Sometimes, we use the notation p to represent the number of genuine regressors (exclusive of the column of ones for the intercept), implying that df $= T - p - 1$. Note that s^2 is the unbiased estimate of σ^2, in the sense that $E(s^2) = \sigma^2$. Also, $MX = (X - X) = 0$, a matrix of zeros. Using this property and substituting (1.2.2) into (1.3.2) gives a rather intriguing matrix relation between the true unknown errors and observable residuals:

$$r = M(X\beta + \varepsilon) = M\varepsilon, \tag{1.3.4}$$

which shows that the observed residual vector r is a linear combination of ε true errors. Hence, if we want to test the properties of true errors, observable r provides only an imperfect approximation.

Further statistical theory is concerned with the sampling distribution of r residuals over all possible samples, assuming that true errors ε are normally distributed in an imagined population of regressions. It can be shown that the standard error (standard deviation of the sampling distribution) is given by

$$\text{SE}(r_t) = s\surd(1 - h_{tt}), \tag{1.3.5}$$

where s is the standard error of the entire regression given by $s = \surd(s^2)$.

We need the tth diagonal term h_{tt} of the hat matrix for our standardized (studentized) residual defined as

$$t_t = \text{szd}(r_t) = r_t/\text{SE}(r_t), \tag{1.3.6}$$

where SE is the standard error (standard deviation) of the sampling distribution of residuals defined over all possible regressions. Clearly, szd(r_t) is a Wald-type statistic approximately distributed as Student's t random variable. This means we can conduct a t-test of the null hypothesis that the residual is zero in the population.

Using advanced matrix algebra results regarding the eigenvalue–eigenvector (spectral) decomposition of X, Cook and Weisberg (1982) prove that if h_{tt} is large, three things happen:

(1) X_t (representing the tth observation) is away from the mean;
(2) X_t is in the direction of the eigenvector associated with the small eigenvalue; and
(3) Low "power" of the t-test (1.3.6) used for testing the statistical significance of the tth residual. The power of a test represents the probability of rejecting the null hypothesis when it should be rejected.

The R software is designed to provide insights using such properties.

```
#R1.3.2 Finding residuals and diagonals of the hat matrix
# Run this only after all commands of R1.2.1 are run
r1=resid(reg1) #gives the vector of residuals
tail(sort.int(r1)) #print observation Number and only
#the largest 6 residuals near the tail of series
hatvalues(reg1)# diagonals of hat matrix H defined in (1.3.1)
fitted(reg1) #gives the vector of fitted values Eq. (1.3.1)
```

1.4. Discussion of Four Diagnostic Plots by R

Statisticians are much interested in the behavior of residuals to understand the quality of the model. Accordingly, a great many powerful tools are available to the users of R to do all kinds of tests about the quality of the model and all kinds of hypotheses of interest to applied statisticians.

A useful command in R produces a sequence of *four* diagnostic plots. The user must hit the enter key to see them, one by one, after entering the command "plot(reg1)" of the snippet #R1.4.1. They are illustrated in Fig. 1.3.

```
#R1.4.1  Diagnostic Plots for reg1
# Run this only after all commands of R1.2.1 are run
reg1 = lm(Ly~LK+LL); summary(reg1);
influence.measures(reg1)
# dfb.1_    dfb.LL   dfb.LK   dffit cov.r  cook.d   hat inf
# Last column with * suggests statistical significance.
# Then, that observation needs further look (possible outlier)
plot(reg1);#help(plot.lm)
#plot(reg1, which=4) #plots Cook's distance vs obs. number
#plot(reg1, which=6) #Cook's distance vs scaled leverage
hv=hatvalues(reg1);hv;plot(hv, typ="l")
cd=cooks.distance(reg1);cd;plot(cd, typ="l")
```

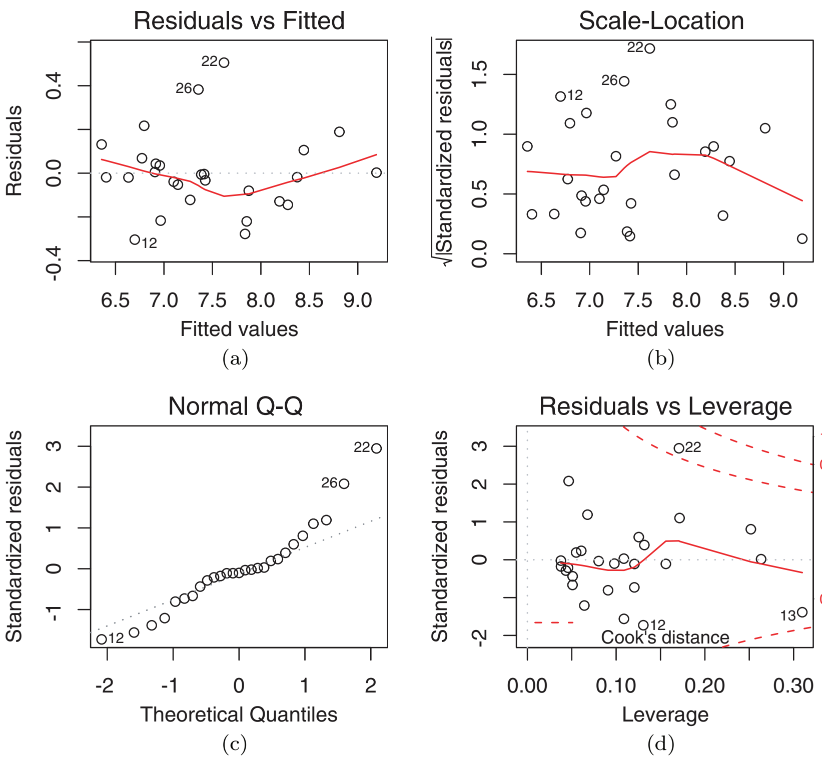

Fig. 1.3. Four diagnostic plots provided by R.

(1) The first plot (Fig. 1.3(a)) shows residuals for the 27 observations and draws a curve for fitted values. We might need a nonlinear model if the fitted values curve is nonlinear. This plot identifies observation numbers 12, 22, and 26 as potential outliers.
(2) The second plot (Fig. 1.3(c)) has quantiles of the suitable normal density on the horizontal axis and quantiles of the standardized residuals on the vertical axis. Such plots are called (Q–Q) plots. One looks for the points lying along the 45° line as drawn in the plot. If the points stay reasonably close to the 45° line, one can conclude that the residuals are approximately normally distributed. R has identified points 12, 22, and 26 as somewhat farther from the 45° line. Too many such points imply a poor regression.
(3) The third plot (Fig. 1.3(b)) is called "scale location plot" with the square root of standardized residuals of Eq. (1.3.6) on the vertical

axis and fitted values on the horizontal axis. The vertical axis standardization gives a better picture as it reduces the skewness. A good model has a near-horizontal line with a random spread of points around it. A trending line with widening (or shrinking) spread around it suggests a poor model. Again, data points 12, 22, and 26 are identified as deserving further attention.

(4) The fourth plot (Fig. 1.3(d) produced by the "plot(reg1)" command in R) is called "residuals versus leverage". It exposes the sensitivity of the regression line to all individual observations. For example, the standardized residual for observation number 22 is outside one of the limits shown as the inner dashed line (marked 0.5). The horizontal axis has the leverage of each point measured by the size of its Cook's distance defined later in Eq. (1.4.3).

Two additional plots are available but need explicit commands. `plot(reg1, which=4)` plots Cook's distance against the observation number, and `plot(reg1, which=6)` plots Cook's distance against scaled leverage $h_{tt}/(1-h_{tt})$. If some points are well beyond Cook's distance lines in the latter plot, we know that their presence is shifting the regression slope.

The influence function is defined for any given dataset. It attempts to measure the influence (or effect) of a quantity on the parameter estimates. Let b denote the $p \times 1$ vector estimating the parameter vector β. Let b_A denote some alternative estimator of β. Hence, the influence vector is also a p by 1 vector defined as

$$I^{nf} = b_A - b. \tag{1.4.1}$$

The weighted distance is defined as a quadratic form in the influence vector as

$$D_A(M, c) = (1/c)(I^{nf\prime} M I^{nf}), \tag{1.4.2}$$

where M is any weight matrix chosen for the quadratic form.

Cook's distance statistic from 1977 (assuming notation p includes parameters in β including the intercept) is defined as

$$\text{Cook}D = [ps^2]^{-1} \sum_{i=1}^{n} [\hat{y}_{i(k)} - y_i]^2, \tag{1.4.3}$$

where p is the number of regressors, s^2 is residual variance, and $\hat{y}_{i(k)}$ denotes the fitted value of ith observation when kth observation is completely omitted from the regression, and the entire regression is based on $T-1$ observations only. It can be viewed as a special case of the quadratic form distance $D_A(M,c)$ when the weight matrix $M = X'X$ and $c = ps^2$.

It can be proved that

$$\text{Cook}D = (1/p)(t_t)^2[h_{tt}/(1-h_{tt})], \tag{1.4.4}$$

where (t_t) are the studentized residuals of (1.3.6), and h_{tt} are diagonals of the hat matrix $X(X'X)^{-1}X'$. The beauty of this result is that we can compute Cook$D$ without actually running $T-1$ regressions, while omitting one observation at a time. The tedious regression runs can be avoided. From (1.4.4) it is obvious that Cook$D$ is large when $(t_i)^2$ is large (large and statistically significant residual), and/or when the scaled leverage $[h_{tt}/(1-h_{tt})]$ is large.

In Fig. 1.3, 22nd observation has a high Cook$D$ statistic, arising from a large residual and/or high leverage. High leverage points have a great deal of influence (or leverage) on the slope and location of the fitted OLS regression line. It also means that omitting that observation will *significantly* change the fit.

However, this does not necessarily mean that we should omit that observation, if the point is a legitimate observation from the underlying production process. See Belsley *et al.* (1980) and Cook and Weisberg (1982) for further details regarding Cook's distance and the concept of leverage of one observation on the regression line.

There is considerable literature on robust regression, which down weights the outlier observations. The command `MASS::rlm(.)` implements several versions of robust regression. R package "msme" has a function `irls()` for iteratively reweighted least squares algorithms, where the weights are a function of unknown parameters, and they are updated with each iteration. Haughton (2021) cites a Venezuela example where outliers severely distort macroeconometric inference.

1.5. Testing Constant Returns and 3D Scatter Plots

An important issue in the study of production functions is a statistical test for constant returns to scale. In the Cobb–Douglas case, assume the metals data is in the memory of R software. Following commands can test the hypothesis that two regression coefficients add up to unity. Recall that SCE $= \alpha + \beta$ is defined in (1.1.8), where α is the coefficient of $\log K$ and β is the coefficient of log L in (1.1.12). Constant returns to scale mean that SCE is 1. The null hypothesis is that SCE $= 1$. It is a linear hypothesis on the coefficients of linear regression (1.1.12). The R package "car" is designed for hypothesis testing, among other things. In snippet #R1.5.1, reg1 is the R object containing a linear (regression) model. The command `linear.hypothesis(reg1, "1*LK+1*LL=1")` states that the right-hand side of the null is 1 (SCE $= 1$) and that we are simply adding the coefficients of LK and LL (i.e., with weights 1 each). The software uses the usual F test defined from the residual sum of squares (RSS) of two regressions: restricted denoted by rstRSS and unrestricted denoted by unRSS. The test statistic is

$$F(n_r, T-p) = \frac{[rstRSS - unRSS]/n_r}{unRSS/(T-p)}, \qquad (1.5.1)$$

where the degrees of freedom *df* of the numerator are n_r, the number of restrictions, and the *df* of the denominator are $(T-p)$, those of the unrestricted model. For our Metals example $n_r = 1$, rstRSS $=$ 0.85574, unRSS $= 0.85163$, and the F statistic from (1.5.1) has Numerator $= (0.85574 - 0.85163)$, Denominator $= 0.85163/24$, and $F(n_r, T-p) = 0.1158249$.

In the old days, we used to have to look up the F table under appropriate degrees of freedom for the numerator and denominator to see if the observed test statistic exceeds the tabulated value and reject the null if it does so. The R software avoids looking up F tables and gives the p-value (probability of F random variable being as extreme or more extreme than the observed test statistic). If we are using a 5% test, we simply compare the p-value with 0.05. The car package reports a p-value of 0.7366, which is much larger than 0.05. Hence, we fail to reject (accept) the null hypothesis of constant returns to scale in Metals production.

```
#R1.5.1 Testing constant returns to scale
# Run this snipett only after all commands of R1.2.1 are run
library(car) #invoke the needed package of R
reg1=lm(Ly~LK+LL)
summary(reg1)
```

```
linear.hypothesis(reg1, "1*LK+1*LL=1")
# p-value of the F test is 0.7366 > 0.05
#this suggests constant returns to scale
```

Having done the regression fits, let us now take a second look at the original Metals production data and use it to discuss how the Cobb–Douglas production function is estimated and plotted as a 3D surface in the log domain. The snippet #R1.5.2 explains how the plots are produced in R. Since we have 27 observations, the snippet labels them with letters *a* to z for the first 26, and labels the last observation by the number 1.

We produce two figures here using a package called "scatterplot3d". Figure 1.4 is for the data in original units where the height dimension has the output, and the two inputs are along the base. Figure 1.5 has log units on all axes so that the Cobb–Douglas production function is a linear surface. That surface is superimposed on the data.

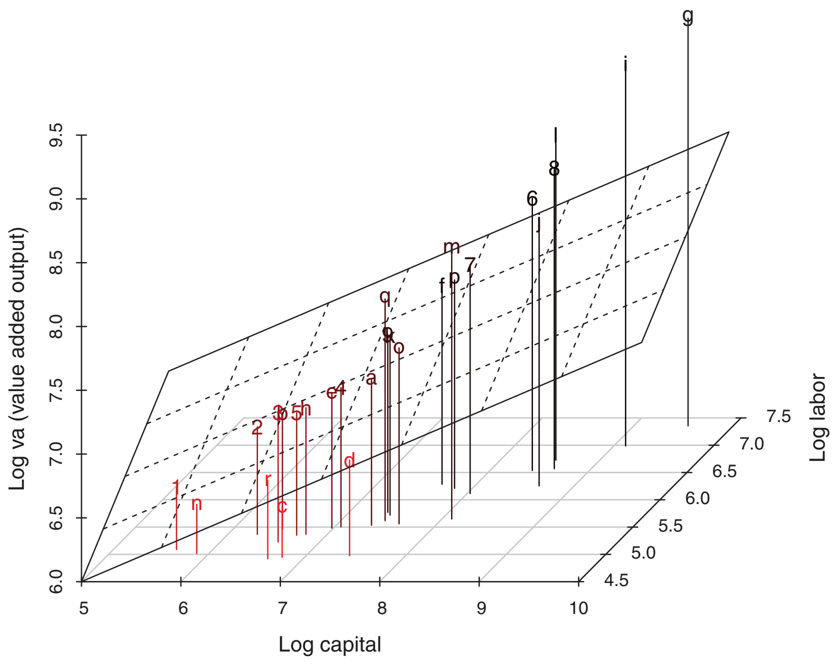

Fig. 1.4. Metals production scatterplot (original units).

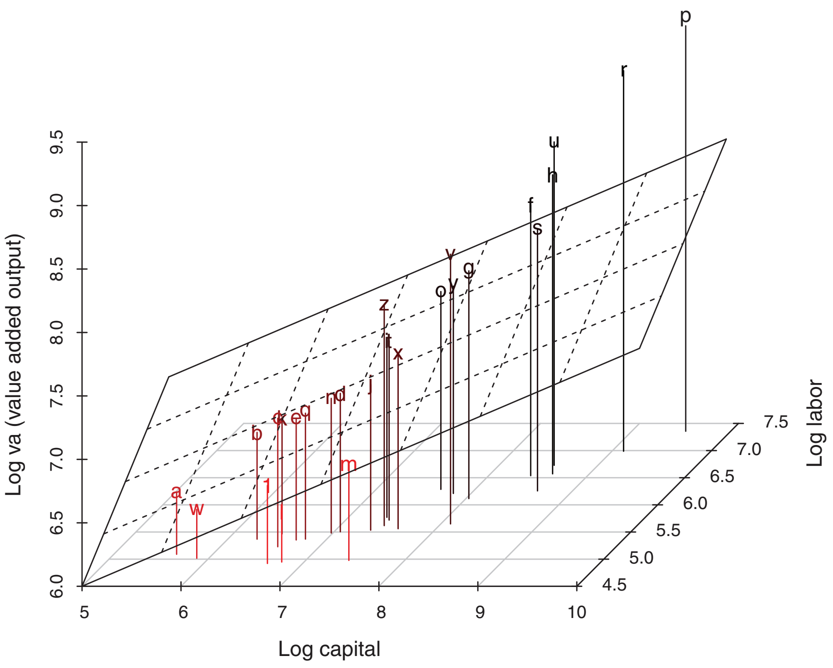

Fig. 1.5. (Log transformed units) Metals: Cobb–Douglas production function.

```
#R1.5.2 for Metals 3D plots and Cobb-Douglas surface
#superimposed
library(scatterplot3d)
library(Ecdat); data(Metal); names(Metal);
met=as.matrix(Metal)
attach(Metal)
length(labor)
#this is 27, we need one character identifiers for 27 obs.s
mynumlet=c(as.character(1:9),letters[1:18])
print(mynumlet)
# print character pch=(1:9,letters a,b,...r) id of observations
Met=data.frame(capital,labor,va)
#the input has to be a data.frame for scatterplot3d function
m3d <- scatterplot3d(Met, type="h", highlight.3d=TRUE,
angle=55, scale.y=0.7, pch=mynumlet,
main="Metals Production Scatterplot (original units)",
box=F,zlab="Value added output")
#my.lm <- lm(va ~ capital+ labor)  this is commented out
#This is not Cobb-Douglas, does not make sense to use linear
```

```
#regression here.
met=as.matrix(Metal)
Ly=log(met[,1])
LL=log(met[,2])
LK=log(met[,3])
Metlog=data.frame(LK, LL, Ly)
m3dL<- scatterplot3d(Metlog, type="h", highlight.3d=TRUE,
angle=55, scale.y=0.7, pch=mynumlet,
main="Metals: Cobb-Douglas Production Function",
box=F, zlab="Log va (value added output)", xlab="Log capital",
ylab="Log labor")
my.lm2 <- lm(Ly ~ LK+LL)
m3d$plane3d(my.lm2, lty.box = "solid")
```

1.6. Homothetic Production and Cost Functions

A function $f(K, L)$ of two variables is homogeneous of degree SCE if

$$\text{Homog(deg} = \text{SCE)} : f(kK, kL) = k^{\text{SCE}} f(K, L). \tag{1.6.1}$$

We can verify that the constant returns to scale Cobb–Douglas production function, $y = f(K, L) = AK^{\alpha}L^{(1-\alpha)}$, is homogeneous of degree 1 as follows. $f(kK, kL) = Ak^{\alpha}K^{\alpha}\ k^{(1-\alpha)}\ L^{(1-\alpha)} = kf(K, L)$, where the power of k is unity. That is, the returns to scale parameter SCE equals 1.

Ferguson (1971, p. 94) lists eight properties of homogeneous functions, including the Euler theorem stating that

$$y = f_K K + f_L L, \quad \text{if } SCE = 1, \tag{1.6.2}$$

where subscripts denote partial derivatives. Consider an aggregate production function for the entire economy. Since wages represent the price of labor, it is customary to use notation w for input prices. The marginal productivity (MP) theory of distribution refers to the aggregate production function and states that ith input should be paid the price (w_i) equal to its contribution to the output, measured by its MP defined in (1.1.3). If $w_i = MP_i = f_i$, for $i = K, L$, then (1.6.2) states that the output is completely (and perhaps fairly) exhausted with no leftovers or shortages, provided SCE=1. Under increasing (diminishing) returns to scale, SCE $>$ 1(SCE $<$ 1), the aggregate output will be in surplus (or short supply), which would be hard to interpret. These interpretation problems vanish when we fit a production function to smaller entities of the economy.

The elasticity of substitution (EOS) defined in (1.1.11) is a complicated expression involving second-order partials. For homogeneous production functions, it simplifies to

$$\mathrm{EOS}_{\mathrm{hom}} = f_K f_L/[y f_{KL}], \quad \text{where } f_{KL} = [\partial^2 f/\partial K \partial L]. \qquad (1.6.3)$$

One of the reasons why older economics texts before the 1960s relied on homogeneous production functions was this simplification. If f represents the aggregate production function for the whole economy, then the Euler theorem allows the output y of the economy to be exhausted if the price of input (wages) equals marginal productivities of each input. This was a neat result in the minds of the earlier generation of economists.

1.6.0.1. *The constant elasticity of substitution production function*

The Constant Elasticity of Substitution (CES) functional form is derived by solving a differential equation, Ferguson (1971, p. 103), which equates the EOS to a constant. After considerable calculus and algebra, we have

$$y = \gamma[\delta K^{-\rho} + (1-\delta)L^{-\rho}]^{-\nu/\rho}, \qquad (1.6.4)$$

where γ measures the efficiency, δ measures input intensity, ν measures SCE, and ρ measures substitution. See Henningsens's (2011) R package "micEconCES" for estimating CES functions.

1.6.0.2. *Hidden restrictions of homogeneous/homethetic functional forms*

Recall that for the purpose of fitting, we often consider a logarithmic transformation, log $f(K, L)$. After the log transformations, we simplify the nonlinear Cobb–Douglas function into a function which is linear in parameters. The log transformation is monotonically increasing. In general, any homothetic function is a monotonically increasing transformation of a homogeneous function. Economics' texts discuss homothetic functions at length, but are not concerned about statistical estimation. Students do not always recognize the following hidden restrictions:

(i) All isoquants of a homothetic production function are forced to be parallel to each other.
(ii) The marginal rates of technical substitution (similar to MRTS_{LK}) are forced to be constant along any ray (straight line in the NE direction) from the origin.
(iii) The scale elasticity for homothetic functions depends only on the output level.
(iv) The EOS=1 for the Cobb–Douglas and EOS$=1/(1+\rho)$ for the CES.

In empirical work, homogeneous functional forms imply these four hidden restrictions on production technology. Since it is unlikely that the researcher knows the production technology enough to presuppose them, we recommend using non-homogeneous forms discussed in Section 1.8. These forms involve higher-order terms (squares and cross-products of logs).

1.6.0.3. *Cost function Lagrangian for minimization of costs*

If we assume that the input prices w_K and w_L are fixed, the total cost of using any input combination is the cost function: $c(y,w) = w_K K + w_L L$. If the left-hand side of the cost function is fixed, we construct an iso-cost or fixed cost line. If we have K and L along the two axes, let $K_{\max}$ denote the largest quantity of K, which can be purchased with the entire cost budget. Similarly, let $L_{\max}$ denote the largest quantity of L, which can be purchased with the entire cost budget. The iso-cost line is a straight line connecting points $K_{\max}$ and $L_{\max}$ along the respective axes.

The arguments of a typical cost function, $c(y,w) = w_K K + w_L L$, remind us that it is a function of output $y = f(K,L)$, and a vector w of input prices. Any manufacturer can be assumed to want to choose an optimal input combination to minimize her total cost, given a specified constant level of output y, denoted by y^*. This is a usual constrained minimization problem, solved by writing the Lagrangian minimand as

$$L_{\text{gr}} = w_K K + w_L L + \lambda[f(K,L) - y^*]. \tag{1.6.5}$$

1.6.0.4. *Cost minimizing solution*

Now, let us minimize the cost function by differentiating the L_{gr} function with respect to the inputs K and L. We have $(\partial L_{\text{gr}}/\partial K) = w_K + \lambda\, f_K = 0$, and $(\partial L_{\text{gr}}/\partial L) = w_L + \lambda\, f_L = 0$. Solving these FOC, we have $w_i = \lambda\, f_i$, where $i = K$, L.

Since λ is common in two equations $w_i = \lambda f_i$, we can eliminate λ and obtain the FOC for a minimum as the following relation:

$$(f_i/f_j) = w_i/w_j, \quad \text{where } i, j = K, L \text{ and } i \neq j. \tag{1.6.6}$$

The interpretation of FOC is that the marginal rate of technical substitution (MRTS) between any two inputs equals the price ratio between that pair of inputs. Another way of stating FOC is that (f_i/w_i), the ratio of marginal productivity to input price is constant for all inputs.

1.6.1. *Euler theorem and duality theorem*

Substituting (1.6.6) into the cost function gives $c(y, w) = \lambda f_K K + \lambda f_L L$. The Euler theorem applied to homogeneous functions of degree 1 says that $y = f_K K + f_L L$. Then, the cost function becomes simply: $c(y, w) = \lambda y$. Written this way, it is obvious that λ can be interpreted as the marginal cost, since $(\partial c/\partial y) = \lambda$.

Recall that the scale elasticity $\text{SCE} = \alpha + \beta$, for the Cobb–Douglas $f(K, L) = AK^{\alpha}L^{\beta}$. It exhibits increasing returns to scale if $\text{SCE} > 1$, and diminishing returns to scale if $\text{SCE} < 1$. Now, the Euler theorem for the case when SCE need not be unity will imply

$$y = \text{SCE}[f_K K + f_L L]. \tag{1.6.7}$$

Upon substituting (1.6.7) into the long-run cost function assuming fixed input prices yields the well-known result that the scale elasticity

$$\text{SCE} = \text{AC}/\text{MC}, \tag{1.6.8}$$

where MC is the marginal cost, and AC is the average cost. For a derivation of this equation using the Cobb–Douglas function, see Ferguson (1971, pp. 163–165) and Section 1.7.1. Ferguson also considers flexible input prices. Denote the price elasticity of jth input as θ_j and further denote their weighted average (weight equals total expenditure on that input) as θ. Now, the left-hand side of Eq. (1.6.8) is replaced by SCE$/(1 + 1/\theta)$. Thus, the result is entirely analogous under flexible input prices.

1.6.1.1. *Expansion path*

The locus of cost-minimizing points is called the expansion path. It is the path along which MRTS equals the corresponding input price ratio. It represents typical input combinations chosen by the optimizing manufacturer. If the production function is homogeneous of any degree, the expansion path is a positively sloped ray from the origin. The expansion path bends toward the axis represented by an input; if that input must be increased relatively more as the overall output, y increases. In empirical work, the observed data points are often assumed to be from the expansion path.

If the price of output is not known, the (manufacturing) firm maximizes output on the one hand and minimizes its costs on the other hand. Hence, the cost function is regarded as the dual of the production function in the theory of the firm. Of course, if the output price is known and fixed, quantity y times price p gives the revenue, $R_{\mathrm{ev}} = py$. Then, net profits can be calculated from revenue minus costs.

1.6.1.2. *Duality theorem*

Minimizing the cost of producing a given level of output or maximizing the output attainable for a given budget to be spent on inputs yield exactly the same optimal input combination.

1.6.2. ***Profit maximizing solutions***

Now let us relax the assumption of fixed output price and allow a demand function: $y = h(p)$, whereby y the output demand increases if price p decreases. A downward-sloping demand curve means that we expect $(\mathrm{d}y/\mathrm{d}p) < 0$. The quantity sold and bought must be equal in equilibrium, meaning that $y = f(K, L)$ from production function and demand function $y = h(p)$ must be equal to each other. The elasticity of demand is defined by

$$\gamma_{\mathrm{d}} = (\mathrm{d}\log y/d\log p) = (p/y)(\mathrm{d}y/\mathrm{d}p). \tag{1.6.9}$$

We remind the reader that this elasticity is simply the slope (regression) coefficient if we regress log y on log p. Now, by simple rearrangement of (1.6.9), we have

$$(\gamma_{\mathrm{d}} y/p) = \mathrm{d}y/\mathrm{d}p, \quad \text{or} \quad \mathrm{d}p/\mathrm{d}y = p/(y\gamma_{\mathrm{d}}). \tag{1.6.10}$$

Even if price p is allowed to vary, the firm's revenue will still be py, and the cost will be $C = Kw_K + Lw_L$. Profit (= revenue − cost) is defined by

$$\Pi = py - C = h^{-1}(y)y - C. \tag{1.6.11}$$

Now, applying calculus methods to maximize (1.6.11) gives the following FOC, assuming $i = K, L$:

$$\begin{aligned}(\mathrm{d}\Pi/\mathrm{d}i) &= p(\mathrm{d}y/\mathrm{d}i) + y(\mathrm{d}p/\mathrm{d}y)(\mathrm{d}y/\mathrm{d}i) - \mathrm{d}C/\mathrm{d}i,\\ &= pf_i + yp/(y\gamma_{\mathrm{d}})f_i - w_i.\\ &= pf_i + (p/\gamma_{\mathrm{d}})f_i - w_i.\end{aligned} \tag{1.6.12}$$

Hence, FOC says that $(p + p/\gamma_{\mathrm{d}})f_i = w_i$. Again, it can be restated to require that (f_i/w_i) equals the constant $(p + p/\gamma_d)^{-1}$ for all inputs. Alternatively, we can write for $i, j = K, L$ and $i \neq j$, $(f_i/f_j) = w_i/w_j$, where the constant cancels.

For greater generality, we can let input prices w_i change in response to the employment of input quantities. Then, we must replace the input price w_i by its revised version:

$$w_{i,\mathrm{rev}} = w_i[1 + (\mathrm{d}\log w_i)/(\mathrm{d}\log i)], \tag{1.6.13}$$

where we are acknowledging that the elasticity of demand for the input need not be unity. Then, the relation $(f_i/f_j) = w_i/w_j$ becomes $(f_i/f_j) = w_{i,\mathrm{rev}}/w_{j,\mathrm{rev}}$. If suitable price and quantity data are available, we can regress the log of input price on the log of input quantities. The slope of this regression can estimate $(\mathrm{d}\log w_i)/(\mathrm{d}\log i)$; its reciprocal is the input demand elasticity for each input i. The point is that substitution of regression coefficient in the expression for $w_{i,\mathrm{rev}}$ makes (1.6.13) empirically estimable, even though few authors have ventured such empirical studies.

Symmetric Normalized Quadratic (SNQ) Profit functions are studied in R package called micEcon.

1.6.3. *Elasticity of total cost wrt output*

The relation SCE = AC/MC of (1.6.8) has some further implications. The total cost function is $c(y, w)$. By definition, the elasticity of total cost with respect to output y is given by

$$(\mathrm{d}\log c/\mathrm{d}\log y) = (y/c)(\mathrm{d}c/\mathrm{d}y) = \mathrm{MC}/\mathrm{AC}, \tag{1.6.14}$$

where we have simply used the definitions MC $= \mathrm{d}c/\mathrm{d}y$ and AC $= c/y$. This shows that the reciprocal of the scale elasticity is the elasticity of total cost wrt output, or percent change in total cost as output increases by 1%.

1.7. Miscellaneous Microeconomic Topics

1.7.1. *Analytic input demand function for the Cobb–Douglas form*

The input demand equations can be determined from the production function by using the FOC. In particular, for the Cobb–Douglas functional form $y = AL^{\alpha}K^{\beta}$, it is easy to verify that the demand for labor input is given by (Ferguson, 1971, p. 164)

$$L = [A^{-1}y\beta^{-\beta}\alpha^{\beta}(w_L)^{-\beta}(w_K)^{\beta}]^{1/(\alpha+\beta)}. \tag{1.7.1}$$

This expression makes intuitive sense, since it says that the quantity demanded of labor increases when the price of labor decreases (since $\beta > 0$), or when output y increases, or when the price of capital increases. The demand for capital input is similarly given by

$$K = [A^{-1}y\beta^{\alpha}\alpha^{-\alpha}(w_L)^{\alpha}(w_K)^{-\alpha}]^{1/(\alpha+\beta)}. \tag{1.7.2}$$

The dual cost function is also known analytically from the production function as

$$C = w_K[(\alpha+\beta)/\beta][X^{\alpha}Y]^{1/(\alpha+\beta)}, \tag{1.7.3}$$

where $X = [(\beta w_L)/(\alpha w_K)]$, $Y = (y/A)$. From (1.7.3) one can verify (after some algebra) that the scale elasticity $(\alpha + \beta)$ indeed equals AC/MC in the Cobb–Douglas case.

1.7.2. *Separability in the presence of three or more inputs*

Separable technologies are of interest when there are more than two inputs. Inputs x_i and x_j are separable from a third input x_k in a production function $f(x)$, where x is a vector of the three inputs if the derivative of MRTS $= (f_i/f_j)$ with respect to x_k is zero. In a later example, we implement this for $f(K, L, G)$ and find that the correlation coefficient between MRTS_{KL} and input G is near zero, implying that the third input G is separable.

1.7.3. *Two or more outputs as joint outputs*

Assume we have two outputs: y_j, $j = 1, 2$. If we can distinguish the amount of x_i needed to produce output y_j for $j = 1$ and 2, it is called *allocable input*, x_i. For example, if the passenger and freight are jointly produced by a railroad, the bogies are an allocable input. But the railroad track itself is shared and is *non-allocable.* If wool and mutton are jointly produced, we cannot easily determine the amount of feed that went for wool or mutton. There is a joint production of corn for human food, animal feed, and ethanol fuel, especially if there are subtle distinctions among corn varieties causing distinct cost profiles.

In general, we have *joint* production where the output of one y_1 depends on how much y_2 is produced.

1.7.4. *Economies of scope*

Recall that if economies of scale are present, a larger firm can produce more cheaply than a smaller firm. A similar concept is economies of scope, applicable to joint production. An interesting issue is: Is it more efficient to produce the distinct outputs together or separately (in individual plants, say)? Let C_j denote the cost of producing y_j for $j = 1, 2$ separately in individual plants dedicated to the separate outputs. Let $C(y_1, y_2)$ represent the cost of producing them jointly in the same plant. Economies of scope exist if there are synergies in the production process, which is if we have sub-additivity defined by

$$C(y_1, y_2) < C_1 + C_2. \tag{1.7.4}$$

If scope economies are present, the policymakers are justified in allowing only one firm to produce different outputs, often under one roof. A typical firm produces all kinds of goods and services in the real world, and the data on costs under joint production and completely separate production are hard to get.

1.7.4.1. *Bell system example on economies of scope*

Empirical proof of sub-additivity is difficult. Consider a near-monopoly company like AT&T, which provided both y_1 = local and y_2 = long-distance telecommunication service before the 1980s. Equation (1.7.4) is based on an empirical experiment where cost C_j of service is known if AT&T produced only one of the services and not the other. The data on exclusive costs C_1 and C_2 simply do not exist, unless AT&T is broken up into two parts exclusively devoted to the two services. There was considerable intermingling between the two services, because they used much of the same network equipment and labor. Cost accountants have ways of fair allocation of total costs to individual components. However, since microeconomics is not cost accounting, these allocations say nothing about what the cost might be if only one service is provided.

1.7.5. *Productivity and efficiency comparisons of firms or countries*

There is considerable literature on tools based on estimated production surfaces allowing one to compare the productive efficiency of various entities, including firms or countries. Stochastic Frontier Analysis (SFA) and Data Envelopment Analysis (DEA) are the primary tools. Bloom *et al.* (2013) study the question "does management matter?" using Indian data. A survey by Sickles *et al.* (2019) also evaluates related R packages.

1.8. Non-homogeneous Production Functions

Vinod (J1972a) argued that the homogeneity of production functions is an unnecessarily restrictive assumption and that there is no need to assume constant elasticity of substitution (EOS) as in (1.6.4).

He proposed a simple multiplicative non-homogeneous (MNH) production function by adding a cross-product (interaction) term,

$$\log y = a_0 + a_1 \log K + a_2 \log L + a_3(\log K)(\log L). \tag{1.8.1}$$

This generalizes the Cobb–Douglas function, because when $a_3 = 0$, it coincides with the Cobb–Douglas form. Its marginal elasticities $(d\log y/d\log i)$ for $i = K, L$ are not constant. For $i = K$, $\mathrm{ME}_K = a_1 + a_3 \log L$. Since the data on labor L varies, ME_K will vary from one observation to another. Similarly, marginal productivities also vary from one observation to another. From the definition of EOS in (1.1.11), it is clear that the formula contains marginal productivities. Hence, it is easy to verify that (1.8.1) is a more general *variable* EOS (VES) production function than the complicated CES of (1.6.4). Jagpal *et al.* (J1982d) apply MNH to annual domestic advertising and sales of Lydia E. Pinkham Medicine Company in thousands of dollars (1907–1960), a part of the R package called "mAr". The MNH form (1.8.1) has the additional advantage that $\mathrm{EOS} = (\mathrm{ME}_K + \mathrm{ME}_L)/(\mathrm{ME}_K + \mathrm{ME}_L + 2a_3)$, so that in the particular case, when $a_3 = 0$, $\mathrm{EOS} = 1$ for the Cobb–Douglas.

1.8.1. ***Three-input production function for widgets***

Bell System was a near-monopoly providing telephone service in the USA before it was broken up in 1982. Its manufacturing arm was called Western Electric Company (WECo), an engineering-oriented entity. They did not traditionally keep records of capital and labor inputs in the distinct sense the terms are used by economists. Vinod (J1972a) develops a time series data regarding the production of certain widgets called *sealed contacts* at a WECo facility over a 59-month period. Readers familiar with creating data files can skip the following subsection. In my teaching experience, there are always some students who are extremely frustrated by this step in R. Perhaps some of these hints might be useful.

1.8.1.1. *Creating a text file of data on your C drive for reading in R*

One must first learn to create a file folder on one's Windows computer by double-clicking first on the "My Computer" icon and then on

"Local Disk (C:)" then on "New" and then give a name like "data" for the directory folder for all your data files (say). Now, check your C drive to make sure that there is a folder named data.

Our next task is to place the following data in the just-created "data" folder. The following Western Electric Co. data has 61 lines of monthly numbers and one header line containing the names of variables. You are free to change the names, but remember that the header line must have exactly as many items as the number of series (columns of numbers), and the names should be short with no blanks. Names are case-sensitive, and blank spaces or symbols or characters like /, ^, :, *, @, &, %, $, underscore, etc., all should be avoided in naming your variables. This avoids much grief later. Also, the data lines should have no additional comments or extraneous material such as source, the meaning of variable names, etc. All these materials should be moved to the top. One can always skip any number (5, say) lines at the top of the file by using the "read.table(..,skip=5.)" function with the argument skip. If data are missing, do not leave blanks but enter a symbol, "NA", which is recognized by R.

The file named "WeCodata.csv" is a comma-separated-values file (csv) containing the needed data. It can be directly imported into your R by using the following code:

```
ur="https://faculty.fordham.edu/vinod/wecodata.csv"
wec=read.table(file=ur, header=T, sep=",")
head(wec,2) #initial data values
#no. idno   L     K  Engg TotExp     Y
#1   3      72  0.106 0.338  0.444 0.019
#2   4      112 0.106 0.529  0.636 0.049
tail(wec,2)# last two data values
#no. idno   L     K  Engg TotExp     Y
#58  363   133 3.456 2.902  7.013 8.456
#59  367   157 3.456 2.917  7.022 5.767
```

We use variable names L=labor, K=capital, G=engineering input (Engg), and Y=output. No data on units is available.

1.8.1.2. *Reading data and MNH estimation in R*

The function "read.table" of R reads data into R. Type help(read.table) in R to know more about the wide variety of options available for this important function. For example, it reads comma-delimited "csv" files created from the Excel workbook and illustrated later in the snippet #R1.11.1 with sep=",". Since R also allows you to skip the first few lines (skip=4 in our example), we suggest keeping important but preliminary information about data. Do not count the header line among the lines skipped.

The first few lines of code in the snippet #R1.8.1 illustrate using the "read.table" command with line skipping and descriptive stats. Keep an eye on missing data or NAs.

Vinod (J1972a) fits a three-input MNH production function: $y = f(K, L, G)$, where G is the engineering input, similar to (1.8.1). Note that we need three cross-product terms in place of the one in (1.8.1). Vinod reports the estimation of marginal elasticities and scale elasticity. For brevity, we omit the R graphs of marginal elasticities of K, L, and G. Their sum representing SCE is included as Fig. 1.6 plots the scale elasticity. The snippet #R1.8.1 implements the MNH estimation, including the code for the creation of Figs. 1.6–1.10 by using the plot function of R.

```
#R1.8.1  Read data and run multiplicative non-homogeneous
#production function
rm(list=ls()) #rm means remove, ls() means every object.
# the above command cleans out memory for a fresh start
url="https://faculty.fordham.edu/vinod/WECOData.csv"
weco=read.csv(url)
names(weco) #list names as understood by R
library(fBasics)# call package for descriptive stats
basicStats(weco)
attach(weco)#Allow access to variables by header names
length(Y)#should be 59
#Take natural log of each variable and define
#cross products of the natural logs of 3 inputs.
lY=log(Y);lK=log(K);G=Engg;lG=log(G);lL=log(L)
lKL=lK*lL
lKG=lK*lG
lLG=lL*lG
lK2=lK^2#The next three regressors are needed for
#         the translog production function regression
lG2=lG^2
lL2=lL^2
reg=lm(lY~lK+lL+lG+lKL+lKG+lLG)#MNH specification
```

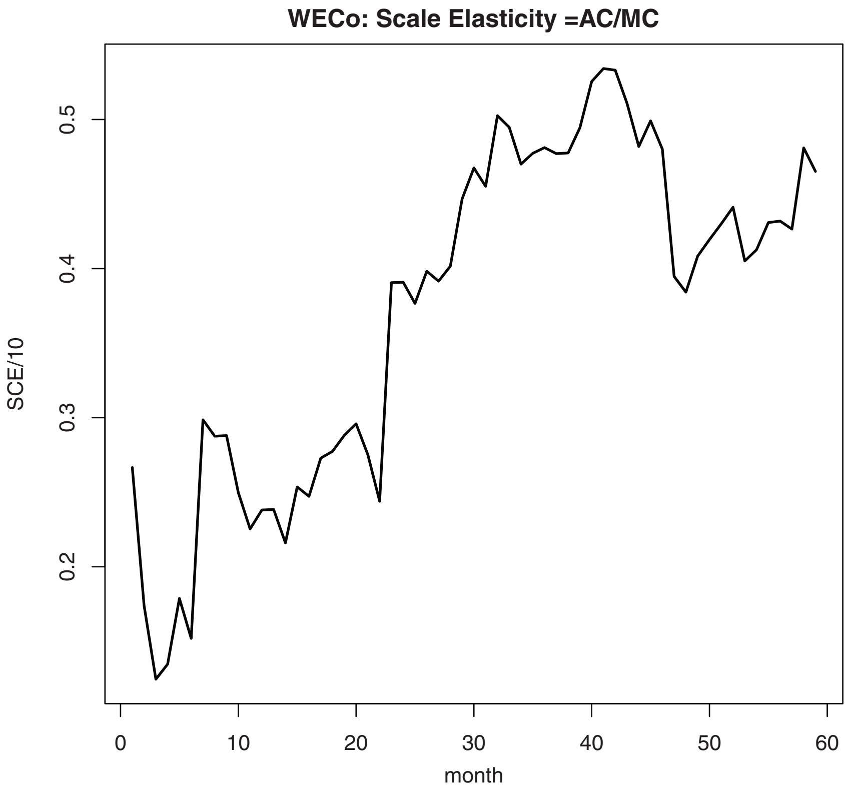

Fig. 1.6. WECo: Scale elasticity = AC/MC.

```
summary(reg)#prints results
a0=reg$co[1]#Extract The intercept
a1=reg$co[2]#The coefficient for log capital
a2=reg$co[3]#The coefficient for log labor
a3=reg$co[4]#The coefficient for log engineering
a4=reg$co[5]#The coef for interaction term (log K)(log L)
a5=reg$co[6]#The coef for interaction (log K)(log G)
a6=reg$co[7]#The coef for interaction (log G)(log L)
# following are vectors not constants.
MEK=a1+a4*lL+a5*lG  #marginal elasticity of capital
MEL=a2+a4*lK+a6*lG  #ME for labor
MEG=a3+a5*lK+a6*lL #ME for engineering
SCE=(MEK+MEL+MEG) #scale elasticity
inv.SCE=(MEK+MEL+MEG)^(-1)
```

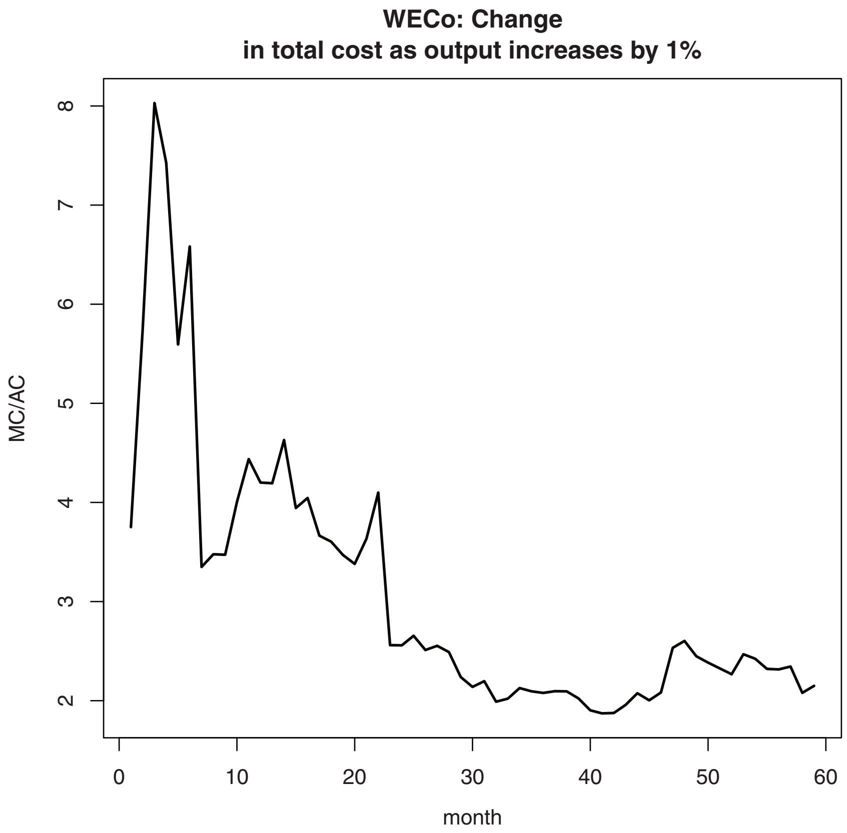

Fig. 1.7. WECo: Change in total cost as output increases by 1%.

```
#The following commands generate the charts
plot(MEK, type="l", xlab="month",
main="WECo: Marginal Elasticity of Capital")
plot(MEL, type="l", xlab="month",
main="WECo: Marginal Elasticity of Labor")
plot(MEG, type="l", xlab="month",
main="WECo: Marginal Elasticity of Engineering")
plot(SCE/10, type="l", xlab="month",
main="WECo: Scale Elasticity =AC/MC",lwd=2)
plot(inv.SCE*10, type="l", xlab="month", main="WECo: Change
in total cost as output increases by 1%", ylab="MC/AC",lwd=2)
mrtsLK=(MEL/MEK)*(K/L)#marginal rate of tech substitution
#between L and K
```

```
cor(G, mrtsLK)# if correlation is low, then engineering input
#is separable. corr=0.02236146 is indeed low See Section 1.7.2
```

It is clear from the elasticity graphs that an empirical study of marginal elasticities is fairly simple with the use of MNH production functions of the type (1.8.1). Marginal elasticity of capital for months 32 and 33 is low when the machinery is idle and high for months 12 and 48 when the machinery is heavily utilized. Engineering has a high degree of marginal elasticity, which leads us to look into the relative cost of the input to see if it is cost-effective to add more engineers. We also note from the plot of the reciprocal of the scale elasticity (having MC/AC on the vertical axis) that after initial learning accomplished in about three months, the total cost is indeed declining as the number of items produced increases. This is what is sometimes called "learning by doing". The snippet also shows how to check for separability. We find that the engineering input is separable since the MRTS of L and K is not correlated with the engineering input.

1.8.2. *Isoquant plotting for a Bell System production function*

In Section 1.1.6, dealing with isoquants, we mentioned that Vinod (J1972a) estimates an aggregate production for the Bell System. Now, we describe the contour function in R for producing isoquants. We define a general function in the snippet #R1.8.2 which is general enough for the Cobb–Douglas, MNH, or trans-log functional forms. The input to the function "pfcontour(.)" (production function contour) is flexible. If the data are in the levels of y, K, and L, the user of the function should use the argument level $= T$ (the default). If the data are in logs, the user has to set level $= F$. If there is an additional regressor besides K and L for technology (say), then it is set as the z variable. If there is no such regressor, the default is $z = 0$, and it is ignored. All inputs must be of length T. The output is the regression fit object. The number of evaluations is set at the default value of 50, which can be changed by $n50 = 10$ (say), if the user wants only 10 evaluations. The reader might find it instructive how the "switch" command in R helps us to switch between three functional forms (Fig. 1.8 illustrates MNH).

```
#R1.8.2 New R function to do production function contours
pfcontour=function(y,K, L, level=T, z=0, type=c("Cobb-Douglas",
"MNH", "TransLog"),n50=50) {
#fit regression and draw contours using clines package
Ly=y; LK=K; LL=L
if (level) Ly=log(y)
T=length(y)
if(level) LK=log(K);
LK2=LK^2
if (level) LL=log(L);
LL2=LL^2; LKLL=LK*LL
#print(c(T,length(z)))
if(length(z)==T){
reg1=switch(type,
        "Cobb-Douglas" = lm(Ly~LK+LL+z),
        "MNH" = lm(Ly~LK+LL+LKLL+z),
        "TransLog" = lm(Ly~LK+LL+LKLL+LK2+LL2+z))
print(reg1)
}#end if length(z)
```

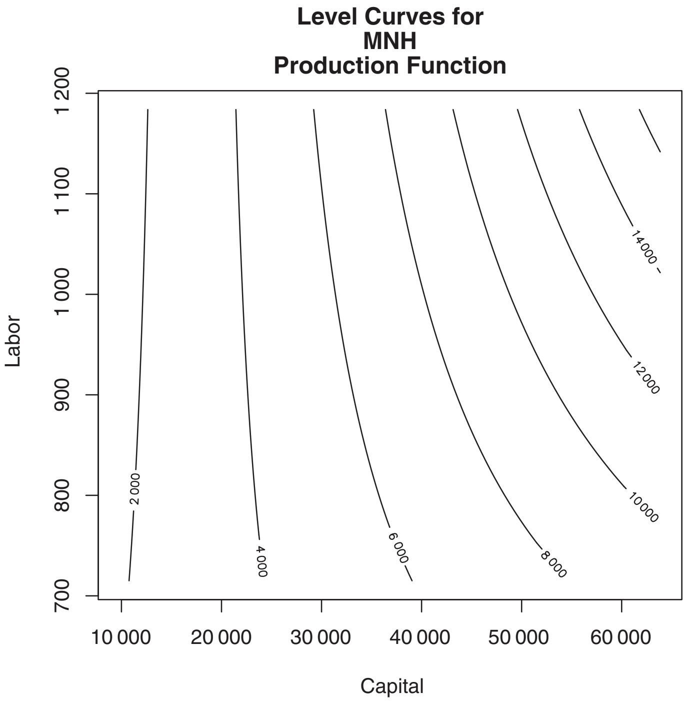

Fig. 1.8. Level curves of MNH production function.

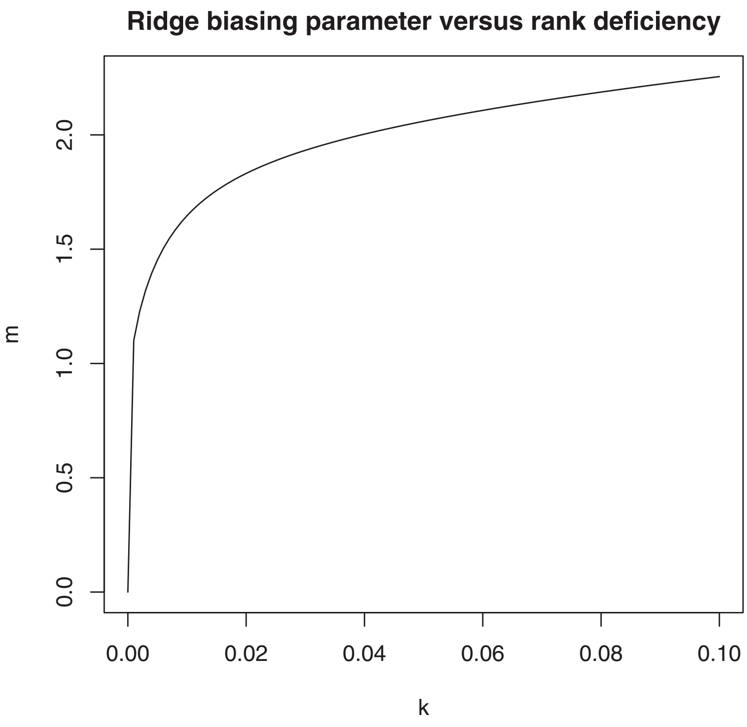

Fig. 1.9. Ridge biasing parameter k versus rank deficiency m.

```
if(length(z)==1){
reg1=switch(type,
        "Cobb-Douglas" = lm(Ly~LK+LL),
        "MNH" = lm(Ly~LK+LL+LKLL),
        "TransLog" = lm(Ly~LK+LL+LKLL+LK2+LL2))
print(reg1)
}#end if length(z)
Lymtx=matrix(NA,n50,n50)
a=as.numeric(reg1$coe)
x=rep(NA,n50)
y=rep(NA,n50)
rangeLK=(max(LK)-min(LK))/n50
rangeLL=(max(LL)-min(LL))/ n50
for (i in 1:n50){Lk=min(LK)+rangeLK*i;x[i]=Lk
for (j in 1:n50){Ll=min(LL)+rangeLL*j;y[j]=Ll
#
if (length(z)==T){
Lymtx[i,j]= switch(type,
```

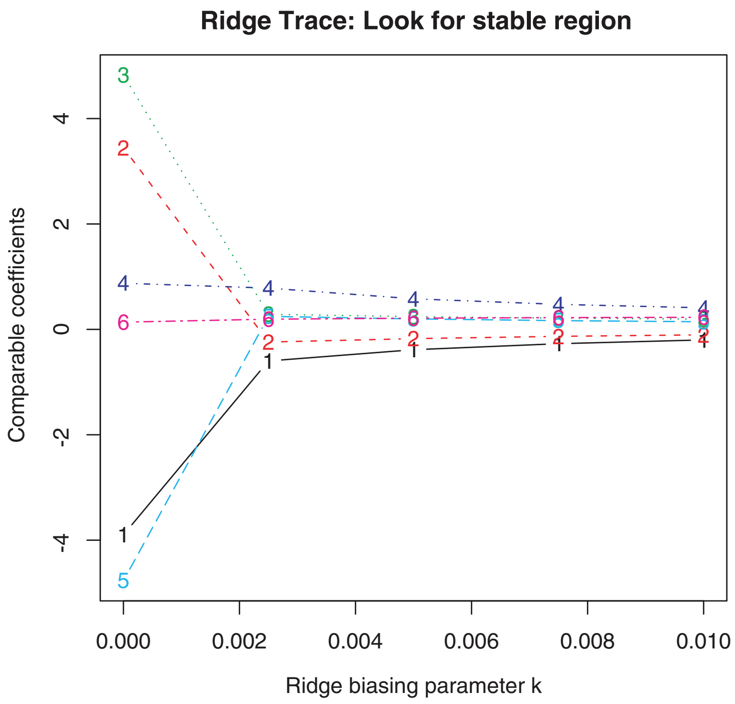

Fig. 1.10. Ridge Trace: Look for a stable region.

```
"Cobb-Douglas" =a[1]+a[2]*Lk+a[3]*Ll+a[4]*z,
"MNH"=a[1]+a[2]*Lk+a[3]*Ll+a[4]*Lk*Ll+a[5]*z,
"TransLog" =a[1]+a[2]*Lk+a[3]*Ll+a[4]*Lk*Ll+a[5]*Lk^2+
a[6]*Ll ^2+a[7]*z
) #end switch simple parenthesis
}# end if length z
#
if (length(z)==1){
Lymtx[i,j]= switch(type,
"Cobb-Douglas" =a[1]+a[2]*Lk+a[3]*Ll,
"MNH"=a[1]+a[2]*Lk+a[3]*Ll+a[4]*Lk*Ll,
"TransLog" =a[1]+a[2]*Lk+a[3]*Ll+a[4]*Lk*Ll+a[5]*Lk^2+a[6]*
Ll^2
) #end switch simple parenthesis
}# end if length z
}#end j loop
}#end i loop
#print(head(Lymtx))
contour(exp(x),exp(y),exp(Lymtx),
main=paste(c("Level Curves for Production Function",type),
sep=" "), xlab="capital", ylab="labor",lwd=2)
#
list(Lymtx=Lymtx, reg1=reg1)
#these lists are output of the function.
#In Lymtx=Lymtx the Lymtx on left of equality is
# the output name released outside the function and the same
# Lymtx on right side of (=) in the name inside this function
}######END of the function
#Now read data.
#use hints from subsection 1.8.1.1
bell=read.table(file="c:/data/belldata.csv", header=T,
sep=",")
#sometimes use file extension to be csv
#csv are comma separated files created from excel worksheet,
#need sep to be a comma
summary(bell); attach(bell)
pfc= pfcontour(y,k,lab,level=T, type="MNH",n50=50,z=0)

#above line creates an object called pfc holding the output
#from the function pfcontour. This way, R will not print
#the entire Lymtx and reg1 and clutter the screen
#It will create and show Figure on the screen
#file menu of R allows one to save the figure in many formats.
```

1.9. Collinearity Problem, Singular Value Decomposition and Ridge Regression

Recall the regression model of (1.2.2) where the error covariance matrix is well behaved (satisfies no autocorrelation or heteroscedasticity). Let us write it as

$$y = X\beta + \varepsilon, \quad E(\varepsilon) = 0, \quad E(\varepsilon\varepsilon') = \sigma^2 I, \tag{1.9.1}$$

in matrix notation, where y is $T \times 1$, X is $T \times p, \beta$ is $p \times 1$, and ε is $T \times 1$. Recall from (1.2.7) that $b = (X'X)^{-1}X'y$ is the OLS estimator minimizing error sum of squares $\varepsilon'\varepsilon$. This formula contains the inverse of $X'X$. The notion of collinearity addresses the numerical reliability of this inverse. and of all quantities depending on it (e.g., coefficients).

1.9.1. *What is collinearity?*

Exact multicollinearity occurs when the inverse of $X'X$ does not exist, preventing the computation of regression coefficients. In ordinary arithmetic, the ratio $(c/0)$ where c is any number does not exist, since it involves division by zero. In matrix algebra, zero can be (i) a matrix of all zeros or (ii) a matrix whose determinant is zero.

Artificial Data: We imagine a data of two regressors: x1 having numbers 1:5 and x2 = (84, 88, 92, 96, 100). The x2 values are obtained by re-centering and rescaling x1. Here, the second regressor x2 contains no additional information than x1, since it is a simple transformation of x1. It is intuitively obvious that regressions using such x2 should be discouraged. Ragnar Frisch called this the (multi)collinearity problem as he imagined the two parallel vectors x1 and x2 as spanning a single line instead of a two-dimensional plane. The X matrix will have three columns, with the first column of all ones for the intercept. The following snippet verifies that the determinant of $X'X$ is $-5.163795\text{e} - 11$, where there are 10 zeros before the first nonzero digit, -5 appears, i.e., the determinant is very close to zero. Theoretically, it should be exactly zero, but due to errors in representations of numbers in a computer, it is not so. In any case, if one attempts to use exactly collinear regressors, good software like R report that regression coefficient estimates are "NA", or missing. Then, there are no serious misleading consequences of collinearity.

```
#R1.9.1  Exact Multicollinearity Artificial example.
x1=1:5  #define x1
x2=scale(x1,center=-20,scale=1/4) #re-center and rescale to
#get x2 as  (84, 88, 92, 96, 100)
ane=rep(1,5)  #a column vector of repeated ones
X=cbind(ane,x1,x2) #create the X matrix by binding 3 columns
colnames(X)[3]="x2"
XTX=t(X) %*% X    # matrix multiply transpose of X and X
det(XTX); #-5.163795e-11 is near zero
solve(XTX) #inverse of matrix is called solve in R, it does
#not exist
y=c(2,13,6,18,19) # artificial data for dependent variable
reg1=lm(y~x1+x2); reg1 #Note that R correctly states that
#the regression coefficient for x2 is NA or not available.
```

In the artificial example of the snippet #R1.9.1, X is $T \times p$ matrix with $T = 5$ and $p = 3$. Eigenvalues and eigenvectors of a $p \times p$ square matrix $X_{tx} = X'X$ is an important matrix algebra concept relevant for a deeper study of collinearity. A nonzero $p \times 1$ column vector g is an eigenvector (also called right eigenvector, right characteristic vector) of X_{tx} if there exists a number λ such that $X_{tx}g = \lambda g$. Then, λ is called the eigenvalue of X_{tx}. The characteristic equation of a $p \times p$ matrix X_{tx} is a degree-p polynomial equation that equates the following determinant to zero:

$$\det(X_{tx} - \lambda I_p) = 0, \tag{1.9.2}$$

where I_p is the identity matrix of dimension $p \times p$. Since a degree-p polynomial has p roots, there are p eigenvalues λ_i and corresponding eigenvectors denoted by g_i for $i = 1, 2, \ldots, p$. In our modern computing environment, instead of solving degree-p polynomial, it is easier to think of the eigenvalue–eigenvector decomposition defined by the following relation:

$$X_{tx} = G\Lambda G', \tag{1.9.3}$$

where $G = \{g_1, g_2, \ldots, g_p\}$ is a $p \times p$ orthogonal matrix of eigenvectors satisfying the (orthogonality) property that its inverse equals its transpose: $G' = G^{-1}$, and where Λ is a diagonal matrix of eigenvalues. In our example, they are: (4.253246e + 04, 7.543651e + 00, 1.139337e − 15) seen to be in decreasing order of magnitude, that is

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p. \tag{1.9.4}$$

1.9.1.1. *Rank deficiency m*

The snippet #R1.9.2 uses the R package called "fEcofin" so that we can use several intuitive command names unavailable in standard R. For example, we use "inv" for matrix inverse instead of "solve" and "rk" for rank. The rank of X_{tx} is only $(p-1)$ or 2 even though X_{tx} has three columns. Thus, there is rank-deficiency $m = 1$. This value of m is used in principal components regression (PCR) discussed later. Whenever $m > 0$, regressing y on X is not a good idea due to the exact collinearity of columns of X. Vinod (J1976a) refers to m as multicollinearity allowance.

```
#R1.9.2  Eigenvalue Decomposition Eq. (1.9.3) Verified in R
#Be sure to run entire snippet #R1.9.1 before running this
library(fEcofin)
evd=eigen(XTX) #R object called evd now has
#eigenvalues-vectors of XTX
G=evd$vec #extract the matrix G
LAM=evd$val;LAM #extract eigenvalues in decreasing order of
#magnitude
#diag(LAM) constructs a matrix with diagonals as eigenvalues
ev.decompose= G %*% diag(LAM) %*% t(G) #implements Eq. (1.9.3)
XTX-ev.decompose #Verify: This matrix has near-zero elements
rk(XTX)# Rank is 2 even though it has 3 columns
inv(XTX)# is impossible
```

1.9.1.2. *Near collinearity*

Now, we tweak the example of snippet #R1.9.1 and add very small constants $e = (0.0272, 0.0241, 0.0433, 0.0442, 0.0293)$ to define x2n = x2+e. Instead of being the exact transform of x1, the x2n is only close to such transform, implying that we have "near collinearity". The latter is more pernicious since it is hard to detect. In the following snippet, the determinant is no longer zero, the matrix inverse of $X'X$ does exist, but the estimated coefficients are huge. The intercept is 17489.4 (with standard error =24327.2), coefficient of x1 is 878.6 (se=1216.7), and the coefficient of nearly collinear x2n is now -218.5 (se=304.0). Of course, all coefficients are statistically insignificant.

```
#R1.9.3 Illustrate near collinearity, Artificial Example
#Be sure to run entire snippet #R1.9.1 before running this
e=c(0.0272, 0.0241, 0.0433, 0.0442, 0.0293) #define the
perturb for x2
x2n=x2+e  #notation n for near collinearity case
Xn=cbind(ane,x1,x2n) #create the Xn matrix by binding 3
#columns
colnames(Xn)[3]="x2n"
XTXn=t(Xn) %*% Xn    # matrix multiply transpose of X and X
det(XTXn); #not zero  0.01485495
solve(XTXn) #inverse does exist in near-collinear case
y=c(2,13,6,18,19) # artificial data for dependent variable
reg2=lm(y~x1+x2n) #near collinear x2n is second regressor
summary(reg2) #Note that OLS gives unreliable coefficients
#the regr coef are now huge with huge standard errors
```

```
library(car)#need this to call the vif function
vif(reg2)#539196.6 & 539196.6 are huge VIF
#VIF =variance inflation factor, Eq. (1.9.5)
```

1.9.2. *Consequences of near collinearity*

(i) OLS regression coefficient estimates are very large in numerical magnitude compared to their true values: $b'b > \beta'\beta$. In other words, the OLS vector b is *too long* (Euclidean distance from the origin) compared to β. This is verified by a simulation of snippet #R1.9.4, and explains why a shrinkage solution is needed. However, exactly how to shrink requires further theory discussed in the sequel. Another way of stating this consequence is that the average squared Euclidean distance between the OLS estimate b and the true vector β is large. After developing some definitions and relevant algebra, it will be shown in Remark 1 later that the mean squared error or MSE(b) $\to \infty$. Then, it will be clear why near collinearity can lead to extremely unreliable OLS estimates b.

(ii) Standard errors are inflated. Let $(R_i)^2$ denote the multiple R^2 when ith regressor is regressed on all remaining regressors in the model. If the remaining regressors are orthogonal (perpendicular, opposite of collinear) to the ith regressor, then $R_i = 0$. Variance inflation factor (VIF) for the variance of ith regressor is defined as

$$\text{VIF}_i = 1/[1-(R_i)^2]. \tag{1.9.5}$$

It measures the inflation in the variance (squared standard error). In the above snippet, we found that collinearity is a serious problem by noting that VIF values for the two regressors are huge, implying that variances are being inflated.

(iii) The third consequence of collinearity, first observed by Vinod and Ullah (B1981), is that regression coefficients become too sensitive to very small perturbations (beyond significant digits) in the data. For example, let x1 data be integers. Let our perturbations be done by adding a random constant in the range $[-0.49, 0.49]$, so the rounding to the nearest integer will give the original integer back. The third consequence then is that in such an example, the regression coefficients change too much. Although this consequence is also a matter of numerical reliability of computed inverse of a matrix, a certain interesting theory is available for the perturbation problem, and it will be discussed in the next subsection.

```
#R1.9.4 Verify that OLS b is too long
# b'b > > beta'beta
#Run entire snippet #R1.9.1 before running this one!
# Generally, true beta is unknown, except in simulations
x1=1:5;x2=scale(x1,center=-20,scale=1/4)
ya=1+2*x1+3*x2 #by definition if y is this way, beta1=1,
#beta2=2 and beta3=3
beta=c(1,2,3); t(beta)%*%beta  #True length is 14
e=c(0.0272, 0.0241, 0.0433, 0.0442, 0.0293) #define the perturb
#for x2
x2n=x2+e  #notation x2n for near-collinearity case
reg3=lm(ya~x1+x2n); summary(reg3)
t(reg3$coef)%*%reg3$coef  #length b'b is huge 58277 compared
#to 14. Hence, shrinkage makes sense
set.seed(239); perturb.x1=runif(5,min=-0.49, max=0.49)
perturb.x2=runif(5,min=-0.49, max=0.5);x1p=x1+perturb.x1
x2np=x2n+perturb.x2
reg4=lm(ya~x1p+x2np) # coefficients change a lot
t(reg4$coef)%*%reg4$coef  #length b'b 13789.41 not equal to
#58277
```

1.9.2.1. *Collinearity and ill-conditioned matrices*

The condition number is a concept from numerical analysis branch of algebra and is defined as

$$K^{\#} = (\lambda_1/\lambda_p)^{1/2} = \max(\text{singular value})/\min(\text{singular value}). \tag{1.9.6}$$

Note that $K^{\#}$ of $X'X$ is infinitely large when $\lambda_p = 0$, or the matrix is singular (non-invertible). The reader may wonder why the definition (1.9.6) has λ_1 in the numerator, since for singularity, only λ_p matters. The answer is that the presence of λ_1 makes the ratio not sensitive to units of measurement of various columns of X. If columns of X are perpendicular (orthogonal) to each other, $\lambda_1 = \lambda_p = 1$ and $K^{\#}$ is unity, this is the opposite of collinearity. Thus, $K^{\#} \in [1, \infty)$. It is proved in numerical analysis literature that when $K^{\#}$ is large, the effect of small perturbation in X can be large for OLS coefficients (Vinod and Ullah, B1981, p. 128). It is generally agreed that one should avoid ill-conditioned matrices, but the magnitude of $K^{\#}$ to determine when a particular matrix is seriously ill-conditioned depends on the subject matter under study.

1.9.2.2. *Rule of thumb for a large condition number*

In my experience with economic data, if $K^{\#} > 10p$, (p = number of regressors), it is safe to conclude that ill-conditioning is severe enough to require remedial action.

1.9.2.3. *Collinearity as a non-problem*

A reviewer for Fox and Monette's (1992) paper on collinearity diagnostics stated that collinearity is in the nature of the problem, and "railing against collinearity is rather like complaining about not being able to fly [due to gravity] by flapping your arms". It appears that the reviewer was unfamiliar with the ill-conditioning literature. Fox and Monette explain that identifying specific sources of imprecision is important, and that collecting additional data is like "abandoning arm flapping and trying an airplane". In social science applications, better specification or abandoning OLS in favor of ridge regression can be useful alternatives, as shown in Vinod (J1976a). Further discussion of why collinearity is a real problem is found in Remark 1.

1.9.3. *Regression theory using the singular value decomposition*

In the numerical example, we have seen that when one regressor x2 or x2n is obtained from re-centering and rescaling x1, we have collinearity. However, the eigenvalue–eigenvector decomposition of (1.9.3) applied to $X'X$ (not X). Hence, it cannot be substituted in our model: $y = X\beta + \varepsilon$, directly. Vinod and Ullah (B1981) argue that the singular value decomposition (SVD) is a better decomposition tool applied to X itself, and provides a deeper understanding of the regressor data. The R software command "svdx=svd(X)" creates an object called svdx, which contains three matrices representing a decomposition of X into three matrices as

$$X = U\Lambda^{1/2}G', \tag{1.9.7}$$

where U is a $T \times p$ matrix, similar to X itself. It satisfies $U'U = I$. Note, however, that UU' is NOT equal to the identity matrix. Hence, this U is not an orthogonal matrix. The geometric interpretation of U is that it contains standardized sample principal coordinates of X. Given the multidimensional scatter of all data, one places an ellipsoid around it. The first *principal axis* has the greatest spread, as illustrated in Vinod and Ullah (B1981). The subsequent principal axes are perpendicular to the previous ones and have the greatest spread sequentially. The matrices Λ and G in (1.9.7) are exactly the same as in (1.9.3). The middle diagonal matrix $\Lambda^{1/2}$ contains the so-called "singular values" of X, which are square roots of eigenvalues. If one of the singular values is zero, then the matrix $X'X$ is singular. A matrix is said to be singular if it cannot be inverted and/or its determinant is zero and/or one or more of its eigenvalues are zero. The eigenvectors in the columns of G, g_i are direction cosine vectors (cosines of direction angles) which orient the ith principal axis of X with respect to the given original axes of the X data. See snippet #R 1.9.5.

```
#R1.9.5 Verify that SVD does decompose X, find condition
#number
#Be sure to run entire snippet #R1.9.1 before running
#this one!
svdx=svd(X);X; svdx$u %*% diag(svdx$d) %*% t(svdx$v)
#verify that we got the original X matrix back after the SVD
cond.no=svdx$d[1]/svdx$d[ncol(X)] #=8.134754e+17>>10p, p=3
```

Substituting SVD in (1.9.1), we have two alternate specifications whereby the unknown parameter vectors (to be estimated) might be denoted either by γ or by α:

$$y = X\beta + \varepsilon = U\Lambda^{1/2}G'\beta + \varepsilon = U\Lambda^{1/2}\gamma + \varepsilon = U\alpha + \varepsilon, \qquad (1.9.8)$$

where we have used notation γ for $G'\beta$ and $\alpha = \Lambda^{1/2}\ G'\beta$. The notation change is not a "material" change, and yet it reveals some important insights. According to the last equation, the matrix U now has the p columns for p transformed regressors. Since $U'U = I$, only the *columns* of U when they are viewed as regressors are orthogonal. It may seem that this has removed collinearity completely. It can be shown that the problem of collinearity does not go away by any change of notation. Collinearity will reveal its ugly head when we try to estimate the β vector even from the reliably available estimates of the α vector (due to the presence of the G matrix).

1.9.3.1. *MSE(b) and extreme unreliability of OLS under near collinearity*

How good is an estimator depends on how close it is to true values! If b is a scalar (1×1 matrix), the closeness of b to β is measured by the absolute value of the error in estimating it $|(b-\beta)|$. Since our b is a $p \times 1$ vector (not a scalar), we need a slightly general concept of closeness. It is customary to use the Euclidean distance between b and true value β, which is $||\mathrm{b}-\beta||$, or the sum of squared errors. The sum of squared estimation errors of all elements of the vector b is $||b-\beta|| = (b-\beta)'(b-\beta)$, which is a positive number (scalar).

When we allow for the fact that b is a random variable, the distance concept must allow for all possible values of b over the "sampling distribution" of b. Then, we will have millions of distances between all possible b and true β. It is customary to summarize them by their average of squared errors, called mean squared error or MSE. The average is given by expected values. Hence, the mean or average of squared errors is given by applying the expectation operator. We have MSE $= E(b-\beta)'(b-\beta)$, a scalar. Actually, it turns out that MSE evaluations are easier to discuss if we define the following $p \times p$ matrix (derived from the "outer product", not the scalar MSE):

$$\mathrm{MtxMSE}(b) = E(b-\beta)(b-\beta)', \tag{1.9.9}$$

where the transpose is on the second vector, unlike in the scalar MSE. Since the scalar MSE is simply the trace(MtxMSE), or the sum of diagonals of MtxMSE, we lose nothing by adopting the matrix viewpoint. Now, we try to evaluate the MSE matrix from the properties of OLS, such as the variance-covariance matrix, $V(b)$. Note that we are treating b as a random vector from millions of possible estimates of β based on millions of possible values of the random errors ε. The density of b is called its sampling distribution. The estimation error vector is

$$\begin{aligned} b-\beta &= (X'X)^{-1}X'y - \beta = (X'X)^{-1}X'[X\beta+\varepsilon] - \beta \\ &= (X'X)^{-1}X'\varepsilon, \end{aligned} \tag{1.9.10}$$

where we cancel β with $(X'X)^{-1}X'X\beta$. Thus, we have stated the estimation error for b in terms of the model error ε, whose properties are $E(\varepsilon) = 0$ and $E(\varepsilon\varepsilon') = \sigma^2\ I_T$. Assuming that the regressors X are pre-determined or non-stochastic, the expectation operator E applied to terms involving X simply passes through, implying that $E(X) = X$ holds. Substituting (1.9.10) into (1.9.9), we have

$$V(b) = \text{MtxMSE}(b) = E(X'X)^{-1}X'\varepsilon\varepsilon'X(X'X)^{-1} = \sigma^2(X'X)^{-1}, \tag{1.9.11}$$

where we have used the fact that the symmetric matrix $X'X = [X'X]'$ and its inverse is also symmetric. The familiar standard errors of regression coefficients are simply the square roots of the diagonals of $V(b)$. The VIF of (1.9.5) actually inflates $V(b)$.

For greater generality, let b^* be some *biased* estimator of β so that $E(b^*)$ does not equal β. The estimation error vector is: $b^* - \beta = b^* - \beta + Eb^* - Eb^* = (b^* - Eb^*) + (Eb^* - \beta) = P + B$, which defines two parts of the $p \times 1$ error vector. In the particular case when b^* is an unbiased estimator of β, our $B \equiv 0$. Hence, part B can be described as the bias vector. Substituting the two parts in (1.9.9), we have the following relation among $p \times p$ matrices:

$$\begin{aligned}\text{MtxMSE}(b^*) = E(P+B)(P+B)' &= E(PP') \\ &+ E(PB') + E(BP') + E(BB').\end{aligned} \tag{1.9.12}$$

Now, evaluating the MtxMSE boils down to evaluating the four terms. The first part P has the interesting property that $E(P) = E(b^* - Eb^*) = Eb^* - Eb^* = 0$ (by definition of expectations). The expectation of the second part B is not a random variable at all. However, $E(B) = E(Eb^* - \beta)$ is still the bias vector and remains nonzero for our biased estimator b^*.

Consider the first of the four terms in (1.9.12), $E(PP')$. This equals $E(b^* - Eb^*)(b^* - Eb^*)'$, which, by definition, is the variance–covariance matrix of b^* or $V(b^*)$. Recall from elementary statistics the result that when x is a random variable vector with $E(x) = 0$ and c is a constant, $E(cx) = 0$. We apply this result to show that the second and third term expectations in (1.9.12) are zero, except that there, the c is the bias vector and x is the estimation error $(b^* - Eb^*)$ having zero expectation. Now, the last positive term of the four terms remains. The expectation of the bias part B is not a random variable at all. $E(B) = E(Eb^* - \beta) = B$ is still the bias vector and is nonzero for any biased estimator b^*. Finally, we have

$$\text{MtxMSE}(b^*) = V(b^*) + (BB'). \tag{1.9.13}$$

The advantage of considering the MtxMSE instead of the scalar MSE is that it allows us to look at the individual components more closely, instead of lumping all regression coefficients in one. The scalar MSE is the sum of diagonals (trace) of MtxMSE and is seen to have two parts, variance and squared bias. Since the OLS estimator b is unbiased, the scalar $\text{MSE}(b) = \sigma^2 \text{ trace}[(X'X)^{-1}]$ from (1.9.11). Now, let us apply the eigenvalue–eigenvector decomposition of (1.9.3) to this expression and write $X'X = G\Lambda G'$. According to a well-known rule of matrix algebra, $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$. Now, recall that G is orthogonal, so its inverse equals its transpose and Λ is a diagonal matrix, so its inverse is simply Λ^{-1}. Thus, we write

$$\begin{aligned}\text{MSE}(b) &= \sigma^2\text{tr}[(X'X)^{-1}] = \sigma^2\text{tr}[G\Lambda^{-1}G'] = \sigma^2\text{tr}[\Lambda^{-1}G'G]\\ &= \sigma^2\text{tr}[\Lambda^{-1}G'G] = \sigma^2\sum_{i=1}^{p}(\lambda_i)^{-1},\end{aligned} \tag{1.9.14}$$

where tr denotes the trace and where the third equality uses the matrix algebra result that $\text{tr}(ABC) = \text{tr}(BCA)$, and the last equality uses the property of our G matrix that $G'G = I$. The point is that the MSE is proportional to the *sum* of reciprocals of eigenvalues of $X'X$ and is greatly inflated when even one eigenvalue is near zero.

1.9.3.2. *REMARK 1 (extreme unreliability of OLS)*

Under (near) collinearity, the OLS coefficient vector b can be an extremely unreliable estimator of the true unknown parameter vector β. $\text{MSE}(b)$ measures the mathematical expectation of the Euclidean distance between b and true β. If there are two competing estimators, b and b^*, we would generally prefer the one with the smaller scalar MSE. Near collinearity means that the $(X'X)$ matrix is almost singular, or that its smallest eigenvalue is very close to zero, $\lambda_p \to 0$. The reciprocal of 0 is infinity, hence substituting this into (1.9.15) implies that $\text{MSE}(b) \to \infty$. To repeat, near-infinite distance from the true vector is another way of stating the extreme unreliability of individual coefficient estimates in b, since individual coefficients are unreliable, and any forecasts based on such estimates are also unreliable.

1.9.3.3. *Forecasting and collinearity*

In some applications, we are interested in forecasting more than in individual parameters. Then we should focus on the $T \times 1$ vector of forecast errors defined by $(Xb - X\beta)$. The overall forecast error is given by the sum of squared forecast errors, as follows

$$||Xb - X\beta|| = (Xb - X\beta)'(Xb - X\beta) = (b - \beta)'(X'X)(b - \beta). \tag{1.9.15}$$

Many authors define weighted MSE = WMSE = $(b - \beta)'W(b - \beta)$, where W is a positive semi-definite matrix (its smallest eigenvalue is zero or positive). Then the sum of the squared forecast error of (1.9.15) is WMSE with $W = (X'X)$ as weights.

1.10. Near Collinearity Solutions by Coefficient Shrinkage

The simplest solution to near-collinearity is to *omit* offending regressors from the model. This amounts to forcing their coefficients to be zero, a form of extreme down-weighting. A less extreme approach is to shrink the coefficients toward zero, without necessarily making them zero. This section discusses a sophisticated shrinkage solution to the collinearity problem, called ridge regression. It injects small bias but generally greatly reduces the MSE, i.e., gives more reliable estimates of β.

Our first task in this section is to understand the reasons behind collinearity more deeply by recalling (1.9.8) by inserting the SVD of X in place of X as

$$y = \mathrm{X}\beta + \varepsilon = U\Lambda^{1/2}G'\beta + \varepsilon = X^*\gamma + \varepsilon, \tag{1.10.1}$$

where we are using a parametrization (change of notation) with $X^* = U\Lambda^{1/2}$ and $\gamma = G'\beta$. If c denotes the $p \times 1$ vector of elements of γ, applying OLS to (1.10.1) gives

$$c = (X^{*\prime}X^*)^{-1}X^{*\prime}y = \Lambda^{-1/2}U'y, \tag{1.10.2}$$

where we have used the analogy between b and c to get the OLS estimates of γ. The last equality uses $(X^{*\prime}X^*) = \Lambda^{1/2}\ U'U\Lambda^{1/2} = \Lambda$, since $U'U = I$. We can carry out the analogy further and write the covariance matrix of c from (1.9.11) as

$$V(c) = \sigma^2(X^{*\prime}X^*)^{-1} = \sigma^2(\Lambda^{1/2}U'U\Lambda^{1/2})^{-1} = \sigma^2\Lambda^{-1}, \quad (1.10.3)$$

$$V(c) = \sigma^2 \sum_{i=1}^{p} (\lambda_i)^{-1}, \quad V(c_i) = \sigma^2/\lambda_i. \quad (1.10.4)$$

The important insight is that $V(c)$ is diagonal, implying that the estimates c of γ are uncorrelated with each other. We refer to c as "uncorrelated components of b", ready for shrinkage as we shall see later.

Since $\gamma = G'\beta$, another way of estimating γ is given by $c = \mathrm{G}'b$. The pre-multiplication of the OLS vector by G' gives the c vector, implying that each element of c is a linear combination of all elements of b. We can think of c as a decomposition of b into uncorrelated components. So if we shrink (down-weight) any individual c_i, it will have no effect on the sampling variance of remaining c_j, $j \neq i$. By contrast, the OLS coefficients b are known to be generally interdependent or correlated with each other. Equation (1.10.4) states that sampling variance $V(c_i)$ is proportional to the reciprocal of the ith eigenvalue of $X'X$. Since $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$, the variance is large when $i = p$ and smallest for c_1. Thus, c_p is the least reliably estimated component and deserves to be shrunk (down-weighted), if not deleted altogether (shrunk to zero). The point is that using the uncorrelated c components, shrinkage can be clean without hurting anything else.

When X data are near-collinear, the smallest eigenvalue λ_p is close to zero, its reciprocal is very large, so the sum of reciprocals of all eigenvalues of $X'X$, including the last one, is also very large. Hence, $V(c) \to \infty$, as does $V(b) \to \infty$. Recall from Remark 1 that collinearity increases MSE(b), leading b to be "far away" in Euclidean distance from the true vector β, or essentially useless estimates. We have shown from (1.10.4) that a solution to the collinearity problem lies in shrinkage, and the offending linear combination of regressors is given by the weights in the last column of matrix G, or g_p. Hence, the weights defining the linear combination are proportional to the eigenvector associated with λ_p. For our example in the snippet #R1.9.1, its elements are: (0.9987, 0.0499, −0.0125).

1.10.1. *Ridge regression*

Ridge estimator represents a family of estimators parameterized by the biasing parameter $k > 0$. The estimator and the corresponding variance–covariance matrix are, respectively,

$$\begin{aligned} b_k &= (X'X + kI)^{-1}X'y, \\ V(b_k) &= \sigma^2(X'X + kI)^{-1}X'X(X'X + kI)^{-1}. \end{aligned} \tag{1.10.5}$$

A large number of choices of $k \in [0, \infty)$ are possible. Since each choice gives a new ridge estimator, the researcher needs some guidance on choosing k. Hence, we denote the ridge regression (RR) estimator by b with subscript k. If $k = 0$, we have OLS; if $k = \infty$, you are dividing by infinity, making shrinkage equal to zero. Since it amounts to deleting all coefficients, $k = \infty$ is not a serious choice. In fact, Hoerl and Kennard, who invented RR, suggested k in the 0 to 1 range. The motivation behind shrinkage, as discussed above, is that it helps to reduce the severity of the consequences of collinearity. Vinod (J2020s) describes how injecting k has turned out to be a "Big Idea". vital for modern data science.

Equation (1.10.5) shows that we are adding a constant k to the diagonal of $X'X$ before inverting it. This amounts to adding $k > 0$ to all eigenvalues of $X'X$, and obviously reduces ill-conditioning of the matrix. Verify that the condition number $K^{\#}$=sqrt(max(λ_i)/ min(λ_i)) becomes $K^{\text{ridg}\#}$=sqrt (max($\lambda_i + k$)/ min($\lambda_i + k$)), and often dramatically reduces $K^{\#}$. For example, if $p = 3$, max(λ_i) $= 9$ and min(λ_i)/ $= 0.01$, $K^{\#} =$ sqrt(9/0.01) $= 30$. Since $30 = 10p$, collinearity is present. Now choosing a rather small biasing parameter $k = 0.1$, the $K^{\text{ridg}\#} =$ sqrt(9.1/(0.01 + 0.1)) $=$ 9.095453 has been reduced, removing the ill-conditioning. Numerical mathematicians assure us that well-conditioned matrices can be reliably inverted and are insensitive to perturbations.

What does ridge regression really do to b, the OLS regression coefficients? To see this, let us recall the decomposition: $X'X = G\Lambda G'$, where G is a matrix of eigenvectors (G is orthogonal matrix $G'G = I = GG'$, that is, its inverse equals its transpose) and Λ is a diagonal matrix of ordered eigenvalues diag(λ_i). Also, recall the SVD: $X = U\Lambda^{1/2}G'$. Now substituting these decompositions in (1.10.5), we have

$$b_k = (G\Lambda G' + kGG')^{-1}[U\Lambda^{1/2}G']'y. \tag{1.10.6}$$

Our next task is to simplify (1.10.6) using matrix algebra till we write it as a shrinkage estimator $b_k = G\Delta c = G\Delta G'b$, where Δ is a $p \times p$ matrix of shrinkage factors $\delta_i < 1$.

1.10.1.1. *Derivation of shrinkage factors* Δ

Since $G'G = I = GG'$ holds, we can replace kI with kGG' in (1.10.6). Now, the transpose $[U\Lambda^{1/2}G']'$ is evaluated using the rule $(ABC)' = C'B'A'$ and noting that the transpose of a diagonal matrix $\Lambda^{1/2}$ is itself. Substituting in (1.10.6), we have

$$\begin{aligned} b_k &= (G\Lambda G' + kGG')^{-1}G\Lambda^{1/2}U'y \\ &= G(\Lambda + kI)^{-1}G'G\Lambda^{1/2}U'y, \end{aligned} \tag{1.10.7}$$

where we peeled off G matrix outside the inverse. Now using $G'G = I$, and since $0.5 = [1 - 0.5]$, we write $\Lambda^{1/2} = \Lambda\Lambda^{-1/2}$. Then

$$b_k = G(\Lambda + kI)^{-1}\Lambda\Lambda^{-1/2}U'y. \tag{1.10.8}$$

If $k = 0$, we have the special case of OLS and (1.10.8) becomes

$$b = G\Lambda^{-1}\Lambda\Lambda^{-1/2}U'y = G\Lambda^{-1/2}U'y. \tag{1.10.9}$$

Note that we want to understand exactly how b_k shrinks b. It involves the diagonal matrix of shrinkage factors $\Delta = \text{diag}(\delta_i)$, obtained from the matrix multiplication:

$$\begin{aligned} \Delta = \text{diag}(\delta_i) &= (\Lambda + kI)^{-1}\Lambda, \\ &\text{where } 1 > \delta_1 \geq \delta_2 \geq \cdots \geq \delta_p \geq 0, \end{aligned} \tag{1.10.10}$$

showing that ith shrinkage factor $\delta_i = \lambda_i/[\lambda_i + k] < 1$ for all i since $k > 0$. Thus, δs are all fractional shrinkage factors. Since $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$, the shrinkage factors satisfy the declining deltas property stated in (1.10.10). In (1.10.8), substitute $\Delta = (\Lambda + kI)^{-1}\Lambda$ to give

$$b_k = G\Delta\Lambda^{-1/2}U'y = G\Delta c = G\Delta G'b \tag{1.10.11}$$

showing that ridge regression shrinks uncorrelated components c, and where the last equality uses the definitions $\gamma = G'\beta$ implying $c = G'b$. The weights δ_i are declining. The smallest weight is on the last c_p, and the highest weight is on the first uncorrelated component c_1, which is eminently sensible. The least reliable component with the highest variance is given the smallest δ weight.

In practice, ridge regression not only removes collinearity problems but also removes wrong signs and makes the coefficients not so sensitive to extremely minor perturbations in data. See a detailed monograph by Gruber (1998), which also covers the James–Stein shrinkage estimator.

1.10.1.2. *Declining deltas and justification for shrinking to zero prior*

In ridge regression, the weights δ are progressively smaller (declining) according to (1.10.10), since λs are declining and the biasing parameter $k > 0$. The shrinkage to zero can be viewed as shrinkage to a prior in a Bayesian framework, explained in Vinod and Ullah's (B1981) monograph focused on ridge regression. Why zero? The usual null hypothesis assumes that, in the absence of evidence to the contrary, coefficients are zero (Null means zero). The usual testing of significance is actually a test of whether a coefficient is significantly different from zero. In the absence of knowledge to the contrary, a conservative choice is zero. Hence, shrinkage methods shrink coefficients toward zero.

1.10.1.3. *Choice of k in ridge regression*

The ridge regression solution is not known till we choose a suitable k value for a particular problem at hand. A graphical method called Ridge Trace has some intuitive advantages since it reveals the behavior of b_k as k slowly increases from the zero value associated with OLS. Figure 1.13 has an example. The "MASS" package of R provides enough information to plot the Ridge Trace. It also provides k values for three choices: modified Hoerl–Kennard–Baldwin (HKB), modified Lawless–Wang (LW), and GCV (generalized cross-validation) with a simple command "select". A researcher can easily pick a suitable solution k from among these three in conjunction with the ridge trace. In the Ridge Trace, one looks for the neighborhood of k and makes sure that the coefficient values (vertical axis) are stable there. If the values are stable at all three choices, one chooses the smallest k, since as a biasing parameter, the smaller the bias, the better.

We discuss an example using Bell System data in the next section. In Fig. 1.10, k values larger than 0.004 are such that coefficients do not change very much near there. The HKB solution $k = 4.467141\text{e} - 05$ and GCV solution $k = 0.0025$ are slightly unstable, whereas the LW solution $k = 0.0074$ is a bit too stable. Vinod (J1976a) suggests that k should be chosen from intuitively plausible rank deficiency m. He proves that there is a direct relationship between k and m (see Section 1.9.1.1). Upon rounding, the eigenvalues are: (551851.2, 314.56090, 0.82867, 0.38353, 0.00631, 0.00004), where the last two eigenvalues are less than 0.007. Hence, rank deficiency appears to be at least $m = 1$ and perhaps $m = 2$. The monotonic relation between m and k shown in Fig. 1.9 plotting m against k suggests that the choice $m \in [1, 2]$ would imply a desired range of $k \in [0.0006, 0.040]$. Since both k values are rather small, we expect that the injected bias will not be large for this application of ridge regression. The GCV solution $k = 0.0025$ lying in the desired range seems worthy of further study.

1.10.2. *Principal components regression*

Recall from (1.10.11) the definition $\gamma = G'\beta$, which transforms the parameter vector β. Recall also that a popular method of dealing with collinearity in Economics is to omit some regressors, which amounts to shrinking elements of β to zero. This is not the wisest choice for several reasons discussed here. Since the OLS estimate of β is b, the OLS estimate of γ is $c = G'b$, where the vector c has as many rows (components) as b. In Principal Components Regression (PCR), some components of c are simply deleted (weight = 0). Thus, the key decisions in using PCR are to decide how many components to delete and which one(s) to delete. The second question has an easy answer. The above discussion clarifies that one should not delete principal components (eigenvectors) willy-nilly, but focus on deleting the relatively "unreliably estimated" components having high variances. These are precisely the components associated with small eigenvalues or in order of preference set by: c_p, c_{p-1}, $c_{p-2}, \ldots$.

How many components should one delete? Our answer is m. Recall from the discussion following (1.9.4) that we use m to denote the rank deficiency of the $X'X$ matrix. It is easy to verify from SVD that m is also the rank deficiency of X of dimension $T \times p$, where there are p regressors after including the column of ones for the intercept. The matrix X is of full column rank if the rank of X is p; otherwise, it is deficient. Knowing m, we can define c_{pc} as the $p \times 1$ column vector of elements $(c_1,\, c_2, \ldots, c_{p-m},\, 0, \ldots, 0)$, where we append exactly m zeros, where m must be an integer. Now, the PCR is given by

$$b_{pc} = Gc_{pc}. \tag{1.10.12}$$

1.11. Bell System Production Function in Anti-Trust Trial

Bell System was the name given to the telephone monopoly in the United States controlled by AT&T till 1982. For a numerical example, let us use the data and model in Vinod (J1976a). Several expert witnesses (including professors Christensen, Jorgenson, and myself) have cited this paper in the multi-billion dollar anti-trust trial of Bell System. It is one of the few examples where econometric research was so crucial for a real-world issue. The phrase "anti-trust" in the United States law refers to preventing business conglomerates from artificially monopolizing the markets.

It is well known in the economic theory of regulation that certain public utilities (e.g., electricity providers) need to be monopolies to avoid wasteful duplication of facilities (several electricity lines criss-crossing a city for each provider). These are called natural monopolies in regulatory economics. Empirical evidence for natural monopoly depends on strong economies of scale. Vinod (J1972a, J1976a) shows that Bell System was a "natural monopoly", since it enjoyed strong economies of scale.

The empirical evidence was based on fitting a three-input trans-log production function to Bell System data on output, capital, labor, and technology. The technology was carefully measured from data going back to 1925 using x_6 as a Poisson-weighted model described in Vinod (J1976a), using the trans-log production. In our version, we use the log of the Poisson technology variable denoted by (Lx_6), in place of x_6 used in Vinod (J1976a). Our production function is defined as

$$\begin{aligned} Ly_t = a_0 + a_1 Lx_{1t} + a_2 Lx_{2t} + a_3 Lx_{1t} Lx_{2t} + a_4 [Lx_{1t}]^2 \\ + a_5 [Lx_{2t}]^2 + a_6 Lx_{6t} + \varepsilon_t, \end{aligned} \tag{1.11.1}$$

where L is an abbreviation for log, y_t is output at time t (years 1947–1976), x_{1t} and x_{2t} are capital and labor inputs, respectively, and x_{6t} is the third input representing Poisson-weighted technology index. The three-input Cobb–Douglas production function is a special case of (1.11.1) when true unknown coefficients (all interaction terms) satisfy $a_3 = a_4 = a_5 = 0$. The R package called "car" has a simple way of testing interaction terms denoted by a colon. The F statistic p-value is smaller than 0.05, suggesting rejection of the null. Statistical journals have recently complained about the abuse of p-values. R package "relevance" has formal tools for assessing economic relevance in addition to purely statistical significance using p-values. It defines and computes a relevance measure $Rl = \mu/\zeta$, where μ is the effect, and ζ denotes a relevant scale factor.

Vinod's (J1972a) MNH production function of (1.8.1) is a special case when $a_4 = a_5 = 0$. Snippet #R1.11.1 shows that we reject this null since the F statistic p-value is smaller than 0.05. The R package called "micEcon" contains several useful tools, including the computation of the Hessian matrix for the trans-log function. Chapters 9 and 10 discuss how Hessians are relevant for computing standard errors.

```
#R1.11.1 Bell System Production Function Estimation, testing
rm(list=ls()) #rm means remove ls() means every object to
#clean slate
mylink=("https://faculty.fordham.edu/vinod/belldata.csv")
bell=read.table(file=mylink, header=T,sep=",")
#csv are comma-separated files created from excel
#worksheet, need "sep"
summary(bell); attach(bell)
Ly=log(y); Lx1=log(k); Lx2=log(lab); Lx6=log(poiss6)
Lx12=(Lx1*Lx2);Lx11=(Lx1^2); Lx22=(Lx2^2)
regnh=lm(Ly~Lx1+Lx2+Lx12)
library(car)#does hypothesis testing
linear.hypothesis(regnh, "1*Lx12 = 0") #p-val=8.585e-11 Reject
regb=lm(Ly~Lx1+Lx2+Lx12 +Lx11 +Lx22 +Lx6)
summary(regb)
regbi=lm(Ly~Lx1+Lx2+Lx1:Lx2 +Lx6);regbi
bcross=regbi$co[5] #coefficient of all interaction terms
## test null that all interaction coefficients  0.4729
equal to 0
coefs <- names(coef(regbi))
```

Table 1.3. Bell system data trans-log production function OLS estimation (using Poisson-weighted index of technological change).

	Estimate	Std. error	t value	$\Pr(>\lvert t\rvert)$
(Intercept)	−42.3670	61.8755	−0.68	0.5004
Lx_1	−7.0485	6.2506	−1.13	0.2711
Lx_2	23.3892	26.9507	0.87	0.3944
Lx_{12}	0.9221	1.4766	0.62	0.5385
Lx_{11}	0.0779	0.2170	0.36	0.7230
Lx_{22}	−2.3611	2.9886	−0.79	0.4376
Lx_6	0.2289	0.0624	3.67	0.0013

```
linear.hypothesis(regbi, coefs[grep(":", coefs)],
verbose=TRUE)
# the p-value for above test =0.0005175
#hence, we reject the null that all interactions =0
```

The OLS coefficients in Table 1.3. suggest that the coefficient a_1 in the notation of Eq. (1.11.1) is large and negative (−7.0485), suggesting a wrong sign. By contrast, coefficient a_2 is even larger and positive (23.3892). At the same time, the standard errors are also large. This gives a prima facie indication of near collinearity. Yet, on the key issue of scale elasticity, the OLS estimate SCE $= 1.341316$ remains plausible. Vinod and Ullah (B1981) explain how certain linear combinations (so-called estimable) can be well estimated under collinearity even though the signs and relative magnitudes of individual coefficients are implausible.

The snippet #R1.11.2 computes various elasticities for these data. For (1.11.1) the marginal elasticities with respect to x_1 and x_2 are: $\mathrm{ME}_1 = a_1 + a_3 Lx_2 + 2a_4 Lx_1$, and $\mathrm{ME}_2 = a_2 + a_3 Lx_1 + 2a_5 Lx_2$. Since Lx_1 and Lx_2 vary for each observation, MEs are variable. However, they can be estimated at the mean of these values by using the formula $\mathrm{ME}_1 = a_1 + a_3 \text{ mean}(Lx_2) + 2a_4 \text{ mean}(Lx_1)$, and similarly for ME_2. The scale elasticity SCE $= \mathrm{ME}_1 + \mathrm{ME}_2$, the sum of the two. In the snippet #R1.11.2, SCE $= 1.341316$. Since this is larger than unity (SCE > 1), the data support strong economies of scale, and hence a natural monopoly in telecommunications.

```
#R1.11.2 Bell System elasticity computation
#Make sure that #R1.11.1 is run before this
a=regb$coe[2:7]
ME1=a[1]+a[3]* mean(Lx2) +2*a[4]* mean(Lx1) #marginal
#elasticity
ME2=a[2]+a[3]* mean(Lx1) +2*a[5]* mean(Lx2)#ME for x2
SCE=ME1+ME2;SCE #scale elasticity=1.341316 >>1, natural
#monopoly
EOS=SCE/(SCE+2*bcross);EOS #elasticity of substitution
#or EOS evaluated at the means=0.739
```

1.11.1. *Collinearity diagnostics for Bell data trans-log*

According to snippet #R1.11.3, the condition number for the six regressors of the model is (=115562.6). This $K^{\#}$ is enormous compared to 10 times the number of regressors (=70); we have collinearity in (1.11.1). We can see in Table 1.1 that the signs and relative magnitudes of individual coefficients are unreliable (MSE is large). Consistent with the consequences of near collinearity discussed in Section 1.9.2, the estimated OLS coefficient vector b is *too long* compared to the true $\beta'\beta$, and we need to shrink its length.

```
#R1.11.3 Collinearity Diagnostics for Bell System Trans-log
#Make sure that #R1.11.1&2 are all run before this
X=cbind(Lx1,Lx2,Lx12, Lx11,Lx22,Lx6)
# COMPUTE SVD and condition number
svdx=svd(X);cond.no=svdx$d[1]/svdx$d[ncol(X)]
cond.no # 115562.6 lot larger than 10p=70, hence collinear
```

1.11.2. *Shrinkage solution and ridge regression for Bell data*

Omitting the second-order terms in Eq. (1.11.1) could remove the collinearity. However, it would make the production function homogeneous and imply several unrealistic hidden restrictions (discussed in Section 1.6.0.2) on the nature of Bell System production technology, without any justification. Thus, here is an example where ridge regression-type shrinkage (declining deltas discussed in Section 1.10.1.2) is obviously appropriate.

Before we compute the ridge regression coefficients of the model, we need some understanding of the appropriate range of choices of the biasing parameter k. The choice of k always involves some trial and error. According to Vinod (J1976a), the rank deficiency m can be directly known from the number of very "small" eigenvalues of $X'X$, and then the known m can assist in the choice of k. Accordingly, we propose a new R function to relate k to the rank deficiency in snippet #R1.11.4, which will also provide a graph relating k to m in Fig. 1.9. The rank deficiency between about 1.5 and 2 seems appropriate, implying the biasing $k = 0.025$ from Fig. 1.9.

```
#R1.11.4 New R Function for deeper understanding of rank
#deficiency
# Function gets m (rank deficiency) from k (ridge biasing
#parameter).  ## BEGIN new function on next line
getmfromk=function(ei, maxk=2, showplot=T, n100=100){
#Input       ei= eigenvalues
#Input      maxk= largest k allowed
p=length(ei)
k=seq(0,maxk, maxk/n100)
m=rep(0,length(k))
del=rep(0,length(ei))
for (i in 1:(n100+1)){
for (j in 1: length(ei)){
del[j]=ei[j]/(ei[j]+k[i])}
sumd=sum(del)
m[i]=p-sumd
i=i+1}
if (showplot){
plot(k,m, type="l", main="Ridge biasing parameter versus rank
deficiency")}
list(m=m,k=k)
}  #End of the function called getkfromm (get k from m)
#### Important Note, run following only after #R1.11.1to5
getmfromk(svdx$d^2,maxk=0.1)#squared singular values input
```

1.11.3. *Ridge regression from existing R packages*

Before we finalize our choice of shrinkage by ridge regression, it helps to consider what the existing R packages have to offer. Snippet #R1.11.5 is for that purpose. The MASS package helps select k by

Table 1.4. Ridge results for Bell System data estimation of Eq. (1.11.1).

	lm.ridge MASS coefficients	Unstandardized coefficients	Standardized coefficients	Standard error	t-Statistic
Intercept	2.7484	−0.7483			
Lx_1	−0.0602	0.2118	0.6415	0.0604	10.6186
Lx_2	−0.2189	0.2892	0.2335	0.0537	4.3483
$Lx_1\ Lx_2$	0.0278	0.0205	0.5889	0.0286	20.6209
$[Lx_1]^2$	0.0231	0.0119	0.7358	0.0502	14.6702
$[Lx_2]^2$	0.0448	0.0248	0.2742	0.0547	5.0085
Lx_6	0.4059	0.3663	1.2025	0.103	11.6722

giving three choices discussed in Section 1.10.1.3. Recall that for our model, the $k(\text{HKB}) = 4.467141\text{e} - 05$, $k(\text{LW}) = 0.007425944$, and $k(\text{GCV}) = 0.0025$. We find that these and some other choices of k within the MASS package scaling do not work well for our purposes. The MASS package results are reported along with our proposed results in the next section (Table 1.4). Note that some negative coefficients remain.

Unfortunately, implicit scaling of second-order terms has failed to shrink the size of marginal elasticities. For example, the marginal elasticity of capital exceeds 6 and ME of labor exceeds 2 for one choice of k, and the latter is negative for another choice of k. The snippet contains actual results and hints on producing the Ridge Trace by using the "matplot" command of R. A major disadvantage of the ridge implementation in the MASS package is that it fails to provide standard errors for estimated coefficients, which are generally required in econometric applications. One must compute the diagonals of square roots of $V(b_k)$ stated in (1.10.5), replacing σ^2 by [(residual sum of squares)/(degrees of freedom)].

Obenchain's "RXshrink" R package has tools for finding MSE reducing choices for generalized ridge regression (GRR) with two parameters (k and Q),

$$b_{kQ} = (X'X + k\,(X'X)^Q)^{-1}X'y.$$

Certain demonstration functions fully work out various data examples, using the commands `demo(RXshrink::longley2)`, `demo(mpg), demo(haldport), demo(tycobb)`. RXshrink implements sophisticated GRR solving the collinearity problem in terms of $0 \le m \le p$ the multicollinearity allowance, or rank deficiency replacing the infinite range ridge parameter $0 \le k < \infty$, using the transformation in Fig. 1.10. Snippet #R1.11.5 includes hints on implementing Obenchain's package and how it needs "`as.data.frame`" function of R to make it work.

```
#R1.11.5 Ridge regression from package MASS & RXshrink of R
li=("https://faculty.fordham.edu/vinod/belldata.csv")
bell=read.table(file=li, header=T,sep=",")
attach(bell)
Ly=log(y); Lx1=log(k); Lx2=log(lab); Lx6=log(poiss6)
Lx12=(Lx1*Lx2);Lx11=(Lx1^2); Lx22=(Lx2^2)
formu=Ly~Lx1+Lx2+Lx12 +Lx11 +Lx22 +Lx6
library(MASS)#lm.ridge
select(lm.ridge(Ly~Lx1+Lx2+Lx12 +Lx11 +Lx22 +Lx6, lambda =
seq(0,0.01,0.0025)))
HKB=lm.ridge(formu,lambda=4.467141e-05)
LW=lm.ridge(formu,lambda=0.007425944)
GCV=lm.ridge(formu,lambda=0.0025)
library(RXshrink)# for two-parameter ridge k and Q
mydf=data.frame(cbind(Ly,Lx1,Lx2,Lx12 ,Lx11 ,Lx22 ,Lx6))
#the above command is needed to get the package to work
#qmr=qm.ridge(formu, mydf);qmr;plot(qmr)
#commented out for brevity
mc=MLcalc(formu,data=mydf,rscale=2)
Ob=mc$beta[2,]#relevant coefficients along 2nd row
ME1=rep(NA,4); ME2=ME1# storing elasticities
for (j in 1:4){
if(j==1) a=coef(HKB)[2:7]
if(j==2) a=coef(LW)[2:7]
if(j==3) a=coef(GCV)[2:7]
if(j==4) a=Ob
ME1[j]=a[1]+a[3]* mean(Lx2) +2*a[4]* mean(Lx1)
ME2[j]=a[2]+a[3]* mean(Lx1) +2*a[5]* mean(Lx2)
}#end of j loop
mtx=cbind(ME1,ME2)#preparing to print results
rownames(mtx)=c("HKB","LW","GCV","RXshrink")
print(mtx)
```

The following output of the above code shows the various ridge regression methods along the first four rows yield comparable results. OLS result along the last row for ME1 is a bit larger while that of ME2 is a bit smaller than others. All show strong economies of scale since ME1+ME2 exceed unity. Thus, we find that despite negative coefficients in Table 1.3, properly computed marginal elasticities allowing for nonlinear form do give reasonable estimates.

```
               ME1       ME2
HKB      0.8330580 0.5153294
LW       0.6506425 0.6433128
GCV      0.7153687 0.5999542
RXshrink 0.8286777 0.5219063
OLS      0.8557328 0.4855832
```

1.12. Comments on Wrong Signs, Collinearity, and Ridge Scaling

Collinearity is ubiquitous in Econometrics, especially with time series data and practitioners are all too familiar with the "wrong signs" problems in the empirical work, even though it is a taboo subject in published journal articles. Recall Table 1.1, which illustrates the wrong negative sign for the coefficient of log of capital (Lx_1) input. In many natural science and engineering applications, researchers use ridge regression quite regularly. The "lm.ridge" function of the MASS package is generally used by scientists and engineers. Its code uses advanced tools in R without comments and does not explain what its computing steps do.

This section describes a generally applicable scaling from Vinod (J1976a). We do not wish to imply that all problems in economics will face a similar difficulty with the lm.ridge scaling. The scaling described in this section can be potentially useful in natural sciences. We also provide standard errors based on the covariance matrix formula, $V(b_k) = \sigma^2(X'X+kI)^{-1}X'X(X'X+kI)^{-1}$ as well as Student's t values not available in the MASS package.

Applied researchers need convenient software, which will guide a user toward a good choice of k, which is done by the "select" function of the MASS package. Obenchain's two-parameter generalized ridge regression, where b_{kQ} is as defined above, is worthy of study by economists. It prints a nice table to help choose k and Q. A description is available at http://members.iquest.net/~softrx/equation.htm. The option `rscale=2` of the function `MLcalc` reports rescaled (unstandardized) results.

Economists are often focused on regression coefficients as partial derivatives. One solution to collinearity is to drop regressors. However, that solution has other problems in economic applications. Recall from Section 1.6.0.2 that the Cobb–Douglas functional form forces the economist to accept hidden restrictions on the underlying production technology: (parallel isoquants, constant elasticity of substitution or marginal rates of technical substitution, scale elasticity SCE depends only on the output level). There is no economic reason to impose such restrictions on the data, and estimated coefficients subject to unrealistic restrictions are obviously suspect. But, including the second-order terms involving squares and cross products inevitably introduces collinearity. What do we do? Vinod (J1976a) and Vinod and Ullah (B1981) propose ridge regression as the solution with standardization of regressor matrix, making $X'X$ a correlation matrix.

Our standardization means ith regressor x_i (excluding the dummy regressor column of ones for the intercept) is subjected to a change of origin and a change of scale as in (1.12.1). Similar to the usual standardization, we also use deviations from the mean. However, instead of dividing by the standard deviation (s_i), we divide by a much larger number: $\sqrt{}(T-1)$ times s_i, and define

$$x_{i,\text{std}} = (x_i - \bar{x})/[\sqrt{}(T-1)s_i], \tag{1.12.1}$$

where subscript std suggests standardization. Note that the dependent variable is generally not divided by its standard deviation for the technical reason that the dependent variable is a random variable, similar to ε, and that it is not involved in checking the ill-conditioning of X. Now, the model $y = X\beta + \varepsilon$ is standardized and becomes one without the intercept as

$$y - \bar{y} = X_{\text{std}}\beta_{\text{std}} + \varepsilon. \tag{1.12.2}$$

Note that $(X'_{\text{std}}X_{\text{std}})$ is a correlation matrix. The correlation matrix is guaranteed to be free from the measurement units, since it does not change after any linear transformation of data. After all, correlation coefficients are pure numbers. The ridge estimates for (1.12.2) are

$$b_{k,\text{std}} = (X'_{\text{std}}X_{\text{std}} + kI_T)^{-1}X'_{\text{std}}(y - \bar{y}). \tag{1.12.3}$$

```
#R1.12.1 Function to standardize so that X'X is a correlation
#matrix. Following commands implement equation (1.12.1)
# in a general setting.  #  BEGIN new function in R
stdze = function(oldx){
# standardize oldx=Original Matrix
bigt=nrow(oldx)
sqb=sqrt(bigt-1)
m=ncol(oldx)
cmn=apply(oldx,2,mean)#column means c=column
csd=apply(oldx,2,sd)#column standard deviations
newx=oldx#---Initialize----#
for (j in 1:m){
newx[,j]=(oldx[,j]-cmn[j])/(sqb*csd[j])
} #end loop for j
return(newx)
}
#example is commented out since it need not always be run
#set.seed(231)
#x=runif(20,min=2,max=10)
#oldx=matrix(x,10,2)
#cor(oldx) #correlation matrix of data
#newx=stdze(oldx)
#t(newx) %*% newx
```

The motivation behind standardization is to make sure that adding k times the identity matrix affects all regressor variables in a symmetric fashion, free from units of measurement. Of course, our real interest is in the regression slopes β as partial derivatives, not in standardized ones. Hence, we unstandardize the ith ridge regression slope coefficient estimated from the standardized model $(b_{k,\text{std}})_i$ by using

$$b_i = (b_{k,\text{std}})_i/[\sqrt{(T-1)}s_i]. \tag{1.12.4}$$

Now, the intercept is estimated indirectly from the sample mean of y and sample means of all regressor variables included in the original data matrix X denoted by $(\bar{x})_i$ multiplied by the estimated (unstandardized) slopes as

$$\text{intercept} = \bar{y} - \sum_i b_i \bar{x}_i. \tag{1.12.5}$$

```
#R1.12.2 Function to unstandardize the data and coefficients
# It implements equations (1.12.4) and (1.12.5)
#source("c:/r-functions/stdze.txt") #good idea to store
#functions in a subdirectory called, say, r-functions
#The source command brings the code into current R memory
#Alternatively, simply run #R1.12.1 snippet before the
#following
unstdze = function(y, oldx, b) {
#  oldx=Original Matrix of y and regressors without column of
#ones, INPUT: y= dependent variable data and oldx is X matrix
#INPUT: b=regr coefficients based on standardized data
bigt=nrow(oldx)
sqb=sqrt(bigt-1)
m=ncol(oldx)
if (m != length(b) ) print("Error in unstdze function ncol
#not=length(b)")
cmn=apply(oldx,2,mean)#column means c=column
csd=apply(oldx,2,sd)#column standard deviations
unstdb=b #---Initialize----#
for (j in 1:m){
unstdb[j]=(b[j])/(sqb*csd[j])
} #end loop for j
for (j in 1:m){
unstdb[j]=(b[j])/(sqb*csd[j])
} #end loop for j
intercept=mean(y)-sum(cmn*unstdb)
list(intercept=intercept, unstdb=unstdb) }
#example
#unstdze(y, oldx, b)
```

The snippets #R1.12.1 and #R1.12.2 describe how to standardize the data and unstandardize the estimated regression coefficients to restore original interpretation as partial derivatives.

```
#R1.12.3 R function to do Ridge regression with
#standardization
# after suitably standardizing and un-standardizing the data
```

```
ridge.std=function(y,x,k=c(0,.05,0.01), bestk=0, plot=T)
{  #estimate the ridge regression model regressing y on X
#version standardizes the data
ys=y-mean(y)
xs=stdze(x)
xx= t(xs) %*%xs
mink=min(k)
p=ncol(xs)
if (length(k)>1) {
mymtx=matrix(NA,length(k),p)
for (i in 1:length(k)){
ki=k[i]*diag(ncol(xs))
xy= t(xs) %*%ys
sinv=solve(xx+ki)
bk=sinv %*%xy
mymtx[i,]=bk
if (k[i]==0) {lsfit=xs%*%bk; lscoef=bk}
resid=ys-xs%*%bk
s2=sum(resid^2)/(length(ys)-ncol(xs)-1)
if(mink==0){
if(k[i]==0){
HKB <- (p - 2) * s2/sum(lscoef^2); print(c("HoerlKennard
#Baldwin
k=",HKB),q=F)
LW <- (p - 2) * s2 * length(ys)/sum(lsfit^2);print(c("Lawless
Wang k=",
LW),q=F)}}
}# end k loop
if(plot){
mynum=as.character(1:p)
nam=1:p
matplot(k,mymtx,main="Ridge Trace: Look for stable region",
type="b",pch=mynum,
xlab="Ridge biasing parameter k", ylab="Ridge regression
coefficients", lty=1:6)
} }#end if length(k)>1 end if for plot=T
print(c("value of ridge biasing parameter chosen is =",bestk),
q=F)
ki=bestk*diag(ncol(xs))
sinv=solve(xx+ki)
xy= t(xs) %*%ys
bk=sinv %*%xy
resid=ys-xs%*%bk
s2=sum(resid^2)/(length(ys)-ncol(xs)-1)
varbk=s2*(sinv %*%xx %*% sinv)
se.bk=sqrt(diag(varbk))
coef=as.numeric(bk)
unst1=unstdze(y,x,coef)
unst=as.numeric(unst1$unstdb)
unst.int=as.numeric(unst1$intercept)
tstat=coef/se.bk
out=cbind(unst,coef,se.bk,tstat)
print(out)
list(bk=bk,varbk=varbk,se.bk=se.bk,unst=unst,
intercept=unst.int)}
#example
#ridge.std(y,X,k=0,bestk=0.1)
```

The snippet #R1.12.3 provides a useful function to do entire ridge regression on any set of y and X data with suitable standardizations and unstandardization. The snippet #R1.12.4 illustrates the use of above R functions using Bell System data.

```
#R1.12.4 Standardized Ridge regression for Bell Data
#Make sure that #R1.11.1 #R1.11.2,#R1.12.1 and #R1.12.2
#are run before this snippet
X=cbind(Lx1,Lx2,Lx12 ,Lx11 ,Lx22 ,Lx6)
## Ridge regression after calling 3 programs from above
#snippets
# or from storage in your c drive and using the source command.
rr=ridge.std(Ly,X,k=seq(0,.1,.01),bestk=.025)
# See Table 1.12.1 for results
a=rr$unst  #use unstandardized regression coefficients
ME1=a[1]+a[3]* mean(Lx2) +2*a[4]* mean(Lx1);ME1
#ME1=0.5966401 is now sensible
ME2=a[2]+a[3]* mean(Lx1) +2*a[5]* mean(Lx2);ME2
# ME2=0.8386455 is now sensible
print(c("Marginal elasticities for Bell Data at mean",ME1,ME2),
q=F)
SCE=ME1+ME2
print(c("Scale elasticity at mean=SCE=",SCE),q=F)
# SCE= 1.43529 >1 suggests economies of scale
```

The ridge results of Table 1.4 should be compared to the OLS results of Table 1.3. The ridge output of the MASS package is in the first column. Note that OLS coefficients and MASS results have some negative signs. The OLS magnitudes are large. Our ridge coefficients are in the column entitled "Unstandardized coefficients". The t-statistics are available in the last column.

1.12.1. *Comments on the 1982 Bell System breakup*

The bottom line of this research is that Bell System did have strong economies of scale and that the SCE > 1 is a robust result. For example, for OLS, the SCE $= 1.34$, and if we use ridge regression with $k = 0.025$ within the standardized model of this section, the SCE $= 1.4353$.

The breakup of the Bell System in 1982 was done for non-economic reasons, notably because the Justice Department lawyers were able to prove that the top management of AT&T did intend to "crush" the competition by the MCI Corporation. As an expert witness, I had numerous discussions with the lawyers working on the case. To the best of my knowledge, the proof for "intent" was found in handwritten lecture notes jotted down by one or more of the Presidents of subsidiaries of AT&T. The notes seem to suggest that the Former President, Mr. Charlie Brown of AT&T, said at a meeting of all top brass that "We will crush MCI". Such talk was considered predatory and illegal under the Sherman Anti-Trust Act, coming from the president of a near-monopoly, which controlled over 90% of the market. In light of this knowledge, and the huge cost of litigation, the then management of AT&T agreed to settle the anti-trust lawsuit and agreed to be broken up.

After the 1982 breakup, the companies are merging again since the 1990s, and the process is accelerating since 2000. This shows that economic forces based on strong economies of scale cannot be ignored by competitive ideology. There is such a thing as a natural monopoly. The consumer is better off if a natural monopoly is smartly regulated rather than broken up.

Bell System data for production function estimation consisting of 30 annual observations from 1947 to 1976 (as rows) contains five columns of comma-separated values (csv). The column headings with self-explanatory names for production function estimation are yr=year, y=output, k=capital input, lab=labor input, and poiss6 for a specially constructed Poisson-weighted index measuring the contribution of technology to Bell System output using data starting with

R&D expenses from 1925 onward (Vinod, J1976a). The file named "belldata.csv" can be directly imported into your R by using the following snippet.

```
#R1.13.1
ur=https://faculty.fordham.edu/vinod/belldata.csv
bell=read.table(file=ur, header=T, sep=",")
head(bell,2)
#      yr       y        k        lab       poiss6
#    1947    1866     9473    707.7      19.63616
#    1948    1995    10996    740.6      18.6777
tail(bell,3)
      yr       y        k        lab       poiss6
#    1975   15756    61889      1181     99.03202
#    1976   17108    63854    1175.7   105.0231
```

1.13. Comparing Productive Efficiency of Firms

The neoclassical theory of production assumes that markets are always efficient (prices equal marginal costs), and firms always maximize profits and reside at the surface of the production possibility frontier, paying input prices equal to their marginal productivity. This paradigm was questioned by Leibenstein (1966), which is considered an early example of behavioral economics. Literature dealing with frontier production functions assumes that one can rank the firms by their efficiency.

Stochastic Frontier Analysis (SFA) begins with the usual error ϵ in estimating production functions similar to (1.1.5). A simple version of SFA splits errors into two parts, $\epsilon = \nu - u$, where $\nu \sim N(0, \sigma_\nu^2)$ and $u \sim |N(\mu, \sigma_u^2)|$ represent inefficiency with positive mean and variance. The absolute values of the standard Normal ensure that $u \geq 0$. The first step uses the usual regression and estimates a vector of residuals with elements $\hat{\epsilon}_i, i = 1, 2, \ldots, T$. The second step adjusts the intercept by adding $M_\epsilon = max(\hat{\epsilon})$. Now technical inefficiency for a particular firm is $M_\epsilon - \hat{\epsilon}_i$. The large literature on this topic solves various statistical estimation and inference problems with the naive formulation described above. Rovigatti (2017) provides a helpful R package called "prodest" for panel data estimates of the productivity of individual firms. Sickles *et al.* (2020) extend SFA, Data Envelopment Analysis (DEA), and Free Disposable Hull (FDH) estimators with R software at https://sites.google.com/site/productivityinr. The link also has an example using OECD data with columns for "Country", "Year", "GDP", "co2", "K", "L", and "E" for energy input. Daraio *et al.* (2019) provide a survey of various software options listing some 15 packages.

1.14. Regression Model Selection and Inequality Constraints

Exploratory regression model selection is often challenging. Aravind Hebbali's package "olsrr" has many OLS regression tools for model choice using Akaike Information Criterion (AIC), Mallows' Cp criterion, forward stepwise regression, etc. For small problems, it can produce all possible regressions. It also provides graphics and various tests (Bartlett, Bonferroni outlier, score). The package has many examples. A good model should have its right-hand side regressors "approximately cause" the left-hand side variable and have a good fit out-of-sample.

Economic theory can impose inequality constraints. For example, expenses cannot exceed some multiple of the budget, or any price cannot be negative. Ulrike Gromping's R package "ic.infer" uses "quadratic programming" to estimate regression coefficients with linear inequality constraints (e.g., all positive or monotonically increasing). One can also test the null hypothesis that restrictions hold vs. the alternative that they are violated by data.