# qcc: An R package for quality control charting and statistical process control

*by Luca Scrucca*

## Introduction

The **qcc** package for the R statistical environment allows to:

- plot Shewhart quality control charts for continuous, attribute and count data;
- plot Cusum and EWMA charts for continuous data;
- draw operating characteristic curves;
- perform process capability analyses;
- draw Pareto charts and cause-and-effect diagrams.

I started writing the package to provide the students in the class I teach a tool for learning the basic concepts of statistical quality control, at the introductory level of a book such as Montgomery (2000). Due to the intended final users, a free statistical environment was important and R was a natural candidate.

The initial design, albeit it was written from scratch, reflected the set of functions available in S-Plus. Users of this last environment will find some similarities but also differences as well. In particular, during the development I added more tools and re-designed most of the library to meet the requirements I thought were important to satisfy.

## Creating a `qcc` object

A `qcc` object is the basic building block to start with. This can be simply created invoking the `qcc` function, which in its simplest form requires two arguments: a data frame, a matrix or a vector containing the observed data, and a string value specifying the control chart to be computed. Standard Shewhart control charts are available. These are often classified according to the type of quality characteristic that they are supposed to monitor (see Table 1).

Control charts for continuous variables are usually based on several samples with observations collected at different time point. Each sample or "rationale group" must be provided as a row of a data frame or a matrix. The function `qcc.groups` can be used to easily group a vector of data values based on a sample indicator. Unequal sample sizes are allowed.

Suppose we have extracted samples for a characteristic of interest from an ongoing production process:

```
> data(pistonrings)
> attach(pistonrings)
> dim(pistonrings)
[1] 200   3
> pistonrings
    diameter sample trial
1     74.030      1  TRUE
2     74.002      1  TRUE
3     74.019      1  TRUE
4     73.992      1  TRUE
5     74.008      1  TRUE
6     73.995      2  TRUE
7     73.992      2  TRUE
...
199   74.000     40 FALSE
200   74.020     40 FALSE
> diameter <- qcc.groups(diameter, sample)
> dim(diameter)
[1] 40  5
> diameter
     [,1]   [,2]   [,3]   [,4]   [,5]
1  74.030 74.002 74.019 73.992 74.008
2  73.995 73.992 74.001 74.011 74.004
...
40 74.010 74.005 74.029 74.000 74.020
```

Using the first 25 samples as training data, an $\overline{X}$ chart can be obtained as follows:

```
> obj <- qcc(diameter[1:25,], type="xbar")
```

By default a Shewhart chart is drawn (see Figure 1) and an object of class 'qcc' is returned. Summary statistics can be retrieved using the `summary` method (or simply by printing the object):

```
> summary(obj)

Call:
qcc(data = diameter[1:25, ], type = "xbar")

xbar chart for diameter[1:25, ]

Summary of group statistics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  73.99   74.00   74.00   74.00   74.00   74.01

Group sample size:  5
Number of groups:  25
Center of group statistics:  74.00118
Standard deviation:  0.009887547

Control limits:
      LCL      UCL
 73.98791 74.01444
```

| type | Control charts for variables | |
|---|---|---|
| "xbar" | $\overline{X}$ chart | Sample means are plotted to control the mean level of a continuous process variable. |
| "xbar.one" | $\overline{X}$ chart | Sample values from a one–at–time data process to control the mean level of a continuous process variable. |
| "R" | $R$ chart | Sample ranges are plotted to control the variability of a continuous process variable. |
| "S" | $S$ chart | Sample standard deviations are plotted to control the variability of a continuous process variable. |
| type | Control charts for attributes | |
| "p" | $p$ chart | The proportion of nonconforming units is plotted. Control limits are based on the binomial distribution. |
| "np" | $np$ chart | The number of nonconforming units is plotted. Control limits are based on the binomial distribution. |
| "c" | $c$ chart | The number of defectives per unit are plotted. This chart assumes that defects of the quality attribute are rare, and the control limits are computed based on the Poisson distribution. |
| "u" | $u$ chart | The average number of defectives per unit is plotted. The Poisson distribution is used to compute control limits, but, unlike the $c$ chart, this chart does not require a constant number of units. |

Table 1: Shewhart control charts available in the **qcc** package.

The number of groups and their sizes are reported in this case, whereas a table is provided in the case of unequal sample sizes. Moreover, the center of group statistics (the overall mean for an $\overline{X}$ chart) and the within-group standard deviation of the process are returned.

The main goal of a control chart is to monitor a process. If special causes of variation are present the process is said to be "out of control" and actions are to be taken to find, and possibly eliminate, such causes. A process is declared to be "in control" if all points charted lie randomly within the control limits. These are usually computed at $\pm 3\sigma$s from the center. This default can be changed using the argument `nsigmas` (so called "american practice") or specifying the confidence level (so called "british practice") through the `confidence.level` argument.

Assuming a Gaussian distribution is appropriate for the statistic charted, the two practices are equivalent, since limits at $\pm 3\sigma$s correspond to a two-tails probability of 0.0027 under a standard normal curve.

Instead of using standard calculations to compute the group statistics and the corresponding control limits (see Montgomery (2000), Wetherill and Brown (1991)), the center, the within-group standard deviation and the control limits may be specified directly by the user through the arguments `center`, `std.dev` and `limits`, respectively.

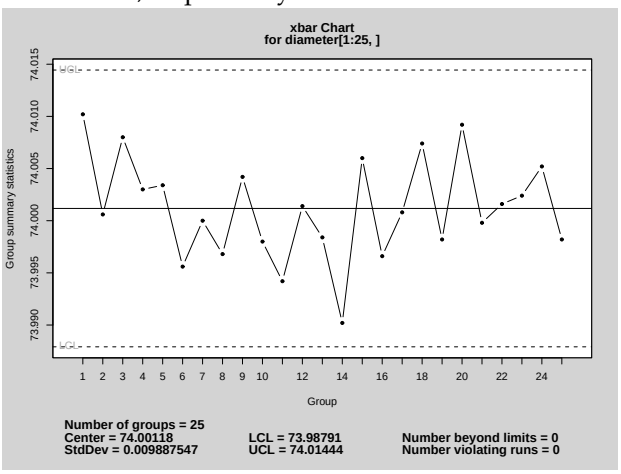


Figure 1: A $\overline{X}$ chart for samples data.

## Shewhart control charts

A Shewhart chart is automatically plotted when an object of class 'qcc' is created, unless the `qcc` function is called with the argument `plot=FALSE`. The method responsible for plotting a control chart can however be invoked directly as follows:

```
> plot(obj)
```

giving the plot in Figure 1. This control chart has the center line (horizontal solid line), the upper and lower control limits (dashed lines), and the sample group statistics are drawn as points connected with lines. Unless we provide the argument `add.stats=FALSE`, at the bottom of the plot some summary statistics are shown, together with the number of points beyond control limits and the number of violating runs (a run has length 5 by default).

Once a process is considered to be "in-control", we may use the computed limits for monitoring new data sampled from the same ongoing process. For example,

```
> obj <- qcc(diameter[1:25,], type="xbar",
+            newdata=diameter[26:40,])
```

plot the $\overline{X}$ chart for both calibration data and new data (see Figure 2), but all the statistics and the control limits are solely based on the first 25 samples.

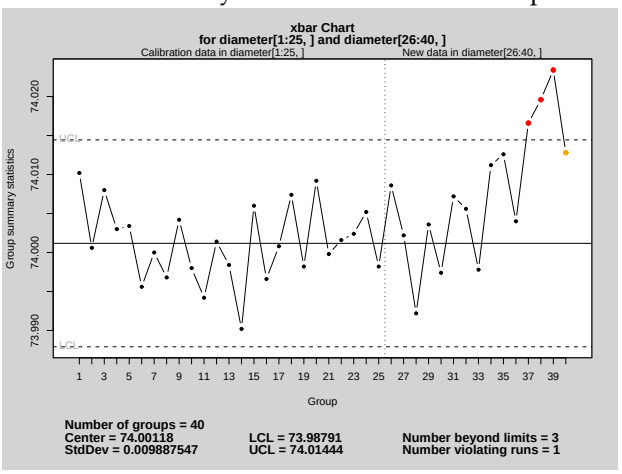


Figure 2: A $\overline{X}$ chart with calibration data and new data samples.

If only the new data need to be plotted we may provide the argument `chart.all=FALSE` in the call to the `qcc` function, or through the plotting method:

```
> plot(obj, chart.all=FALSE)
```

Many other arguments may be provided in the call to the `qcc` function and to the corresponding plotting method, and we refer the reader to the on–line documentation for details (`help(qcc)`).

As reported on Table 1, an $\overline{X}$ chart for one–at–time data may be obtained specifying `type="xbar.one"`. In this case the data charted are simply the sample values, and the within-group standard deviation is estimated by moving ranges of $k$ (by default 2) points.

Control charts for attributes mainly differ from the previous examples in that we need to provide sample sizes through the `size` argument (except for the $c$ chart). For example, a $p$ chart can be obtained as follows:

```
> data(orangejuice)
> attach(orangejuice)
```

```
> orangejuice
   sample  D size trial
1       1 12   50  TRUE
2       2 15   50  TRUE
3       3  8   50  TRUE
...
53     53  3   50 FALSE
54     54  5   50 FALSE
> obj2 <- qcc(D[trial], sizes=size[trial], type="p")
```

where we use the logical indicator variable `trial` to select the first 30 samples as calibration data.

## Operating characteristic function

An operating characteristic (OC) curve provides information about the probability of not detecting a shift in the process. This is usually referred to as the type II error, that is, the probability of erroneously accepting a process as being "in control".

The OC curves can be easily obtained from an object of class 'qcc':

```
> par(mfrow=c(1,2))
> oc.curves(obj)
> oc.curves(obj2)
```

The function `oc.curves` invisibly returns a matrix or a vector of probabilities values for the type II error. More arguments are available (see `help(oc.curves)`), in particular `identify=TRUE` allows to interactively identify values on th e plot.

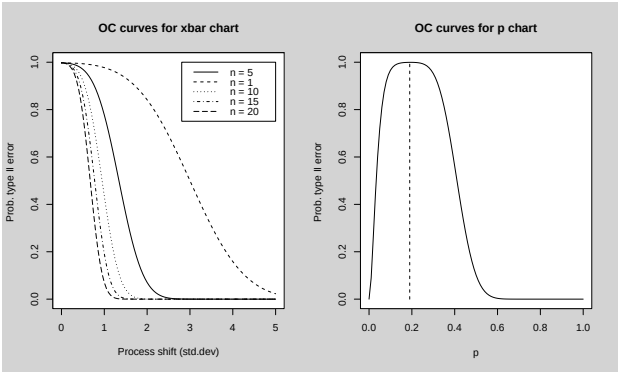


Figure 3: Operating characteristics curves.

## Cusum charts

Cusum charts display how the group summary statistics deviate above or below the process center or target value, relative to the standard error of the summary statistics. They are useful to detect small and permanent variation on the mean of the process.

The basic cusum chart implemented in the **qcc** package is only available for continuous variables at the moment. Assuming a 'qcc' object has been created, we can simply input the following code:

```
> cusum(obj)
```

and the resulting cusum chart is shown in Figure 4. This displays on a single plot both the positive deviations and the negative deviations from the target, which by default is set to the center of group statistics. If a different target for the process is required, this value may be provided through the argument `target` when creating the object of class 'qcc'. Two further aspects may need to be controlled. The argument `decision.int` can be used to specifies the number of standard errors of the summary statistics at which the cumulative sum displays an out of control; by default it is set at 5. The amount of shift in the process to be detected by the cusum chart, measured in standard errors of the summary statistics, is controlled by the argument `se.shift`, by default set at 1.

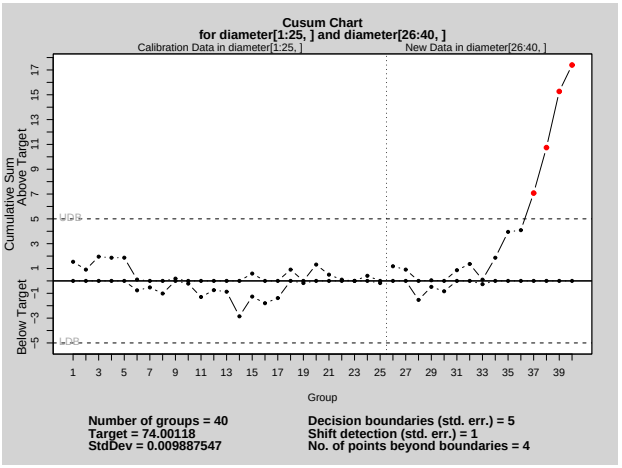


Figure 4: Cusum chart.

## EWMA charts

Exponentially weighted moving average (EWMA) charts smooth a series of data based on a moving average with weights which decay exponentially. As for the cusum chart, also this plot can be used to detect small and permanent variation on the mean of the process.

The EWMA chart depends on the smoothing parameter $\lambda$, which controls the weighting scheme applied. By default it is set at 0.2, but it can be modified using the argument `lambda`. Assuming again that a 'qcc' object has been created, an EWMA chart can be obtained as follows:

```
> ewma(obj)
```

and the resulting graph is shown in Figure 5.

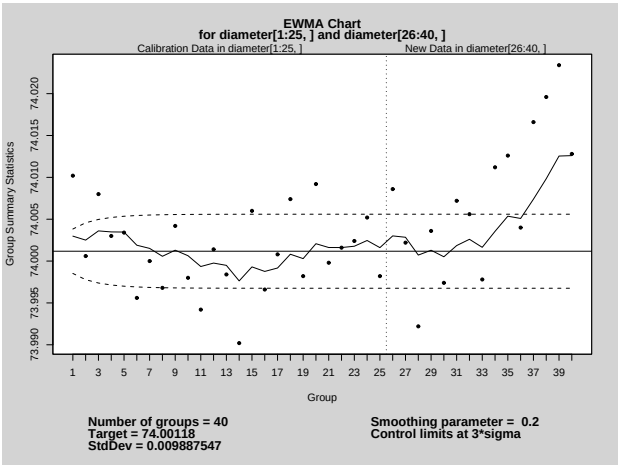


Figure 5: EWMA chart.

## Process capability analysis

Process capability indices for a characteristic of interest from a continuous process can be obtained through the function `process.capability`. This takes as mandatory arguments an object of class 'qcc' for a `"xbar"` or `"xbar.one"` type, and `spec.limits`, a vector specifying the lower (LSL) and upper (USL) specification limits. For example, using the previously created 'qcc' object, we may simply use:

```
> process.capability(obj,
+                    spec.limits=c(73.95,74.05))

Process Capability Analysis

Call:
process.capability(object = obj,
                   spec.limits = c(73.95, 74.05))

Number of obs = 125          Target = 74
       Center = 74.00118        LSL = 73.95
       StdDev = 0.009887547     USL = 74.05

Capability indices:

      Value   2.5%  97.5%
Cp    1.686  1.476  1.895
Cp_l  1.725  1.539  1.912
Cp_u  1.646  1.467  1.825
Cp_k  1.646  1.433  1.859
Cpm   1.674  1.465  1.882

Exp<LSL 0%   Obs<LSL 0%
Exp>USL 0%   Obs>USL 0%
```

The graph shown in Figure 6 is an histogram of data values, with vertical dotted lines for the upper and the lower specification limits. The target is shown as a vertical dashed line and its value can be provided using the `target` argument in the call; if missing, the value from the 'qcc' object is used if available, otherwise the target is set at the middle value between the specification limits.

The dashed curve is a normal density for a Gaussian distribution matching the observed mean and standard deviation. The latter is estimated using the within-group standard deviation, unless a value is specified in the call using the `std.dev` argument.

The capability indices reported on the graph are also printed at console, together with confidence intervals computed at the level specified by the `confidence.level` argument (by default set at 0.95).

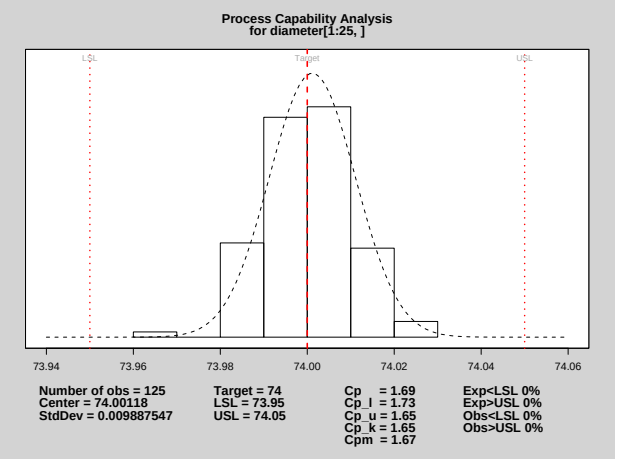


Figure 6: Process capability analysis

A further display, called "process capability sixpack plots", is also available. This is a graphical summary formed by plotting on the same graphical device:

- a $\overline{X}$ chart
- a $R$ or $S$ chart (if sample sizes $> 10$)
- a run chart
- a histogram with specification limits
- a normal Q–Q plot
- a capability plot

As an example we may input the following code:

```
> process.capability.sixpack(obj,
+                    spec.limits=c(73.95,74.05),
+                    target= 74.01)
```

## Pareto charts and cause-and-effect diagrams

A Pareto chart is a barplot where the categories are ordered using frequencies in non increasing order, with a line added to show their cumulative sum. Pareto charts are useful to identify those factors that have the greatest cumulative effect on the system,

and thus screen out the less significant factors in an analysis.

A simple Pareto chart may be drawn using the function `pareto.chart`, which in its simpler form requires a vector of values. For example:

```
> defect <- c(80, 27, 66, 94, 33)
> names(defect) <-
+   c("price code", "schedule date",
+     "supplier code", "contact num.",
+     "part num.")
> pareto.chart(defect)
```

The function returns a table of descriptive statistics and the Pareto chart shown in Figure 7. More arguments can be provided to control axes labels and limits, the main title and bar colors (see `help(pareto.chart)`).

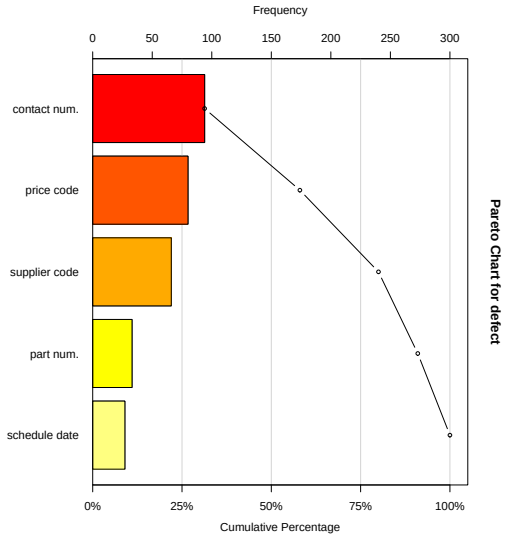


Figure 7: Pareto chart.

A cause-and-effect diagram, also called an Ishikawa diagram (or fish bone diagram), is used to associate multiple possible causes with a single effect. Thus, given a particular effect, the diagram is constructed to identify and organize possible causes for it.

A very basic implementation of this type of diagram is available in the **qcc** package. The function `cause.and.effect` requires at least two arguments: `cause`, a list of causes and branches providing descriptive labels, and `effect`, a string describing the effect. Other arguments are available, such as `cex` and `font` which control, respectively, the character expansions and the fonts used for labeling branches, causes and the effect. The following example creates the diagram shown in Figure 8.

```
> cause.and.effect(
+   cause=list(Measurements=c("Microscopes",
+                             "Inspectors")
+              Materials=c("Alloys",
+                          "Suppliers"),
+              Personnel=c("Supervisors",
+                          "Operators"),
+              Environment=c("Condensation",
+                            "Moisture"),
+              Methods=c("Brake", "Engager",
+                        "Angle"),
+              Machines=c("Speed", "Bits",
+                         "Sockets")),
+   effect="Surface Flaws")
```

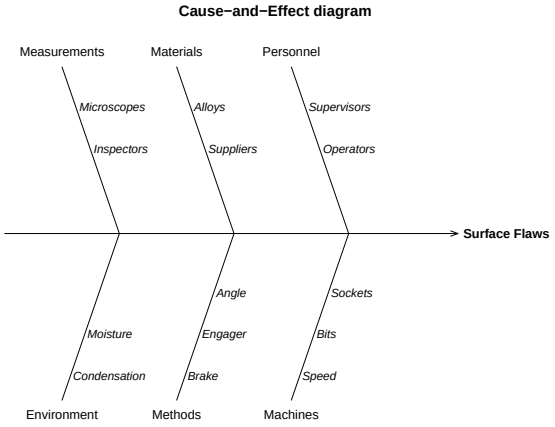


Figure 8: Cause-and-effect diagram.

## Tuning and extending the package

Options to control some aspects of the **qcc** package can be set using the function `qcc.options`. If this is called with no arguments it retrieves the list of available options and their current values, otherwise it sets the required option. From a user point of view, the most interesting options allow to set the plotting character and the color used to highlight points beyond control limits or violating runs, the run length, i.e. the maximum value of a run before to signal a point as out of control, the background color used to draw the figure and the margin color, the character expansion used to draw the labels, the title and the statistics at the bottom of any plot. For details see `help(qcc.options)`.

Another interesting feature is the possibility to easily extend the package defining new control charts. Suppose we want to plot a $p$ chart based on samples with different sizes. The resulting control chart will have non–constant control limits and, thus, it may result difficult to read by some operators. One possible solution is to draw a standardized control chart. This can be accomplished defining three new functions: one which computes and returns the group statistics to be charted (i.e. the $z$ scores) and their center (zero in this case), another one which returns the within-group standard deviation (one in this case), and finally a function which returns the control limits. These functions need to be

called `stats.`*type*, `sd.`*type* and `limits.`*type*, respectively, for a new chart of type "*type*". The following code may be used to define a standardized $p$ chart:

```
stats.p.std <- function(data, sizes)
{
  data <- as.vector(data)
  sizes <- as.vector(sizes)
  pbar <- sum(data)/sum(sizes)
  z <- (data/sizes - pbar)/sqrt(pbar*(1-pbar)/sizes)
  list(statistics = z, center = 0)
}

sd.p.std <- function(data, sizes) return(1)

limits.p.std <- function(center, std.dev, sizes, conf)
{
  if (conf >= 1) { lcl <- -conf
                   ucl <- +conf }
  else
     { if (conf > 0 & conf < 1)
          { nsigmas <- qnorm(1 - (1 - conf)/2)
            lcl <- -nsigmas
            ucl <- +nsigmas }
       else stop("invalid 'conf' argument.") }
  limits <- matrix(c(lcl, ucl), ncol = 2)
  rownames(limits) <- rep("", length = nrow(limits))
  colnames(limits) <- c("LCL", "UCL")
  return(limits)
}
```

Then, we may source the above code and obtain the control charts in Figure 9 as follows:

```
# set unequal sample sizes
> n <- c(rep(50,5), rep(100,5), rep(25, 5))
# generate randomly the number of successes
> x <- rbinom(length(n), n, 0.2)
> par(mfrow=c(1,2))
# plot the control chart with variable limits
> qcc(x, type="p", size=n)
# plot the standardized control chart
> qcc(x, type="p.std", size=n)
```

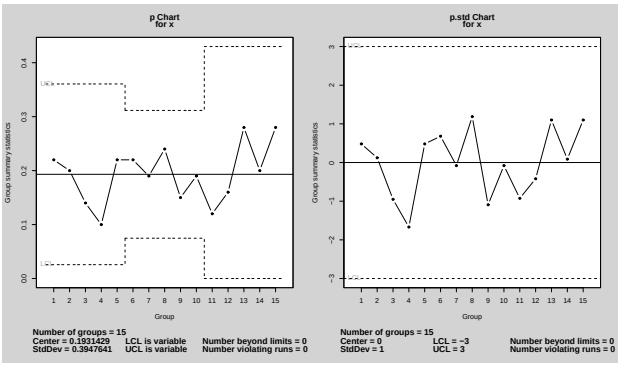


Figure 9: A $p$ chart with non–constant control limits (left panel) and the corresponding standardized control chart (right panel).

## Summary

In this paper we briefly describe the **qcc** package. This provides functions to draw basic Shewhart quality control charts for continuous, attribute and count data; corresponding operating characteristic curves are also implemented. Other statistical quality tools available are Cusum and EWMA charts for continuous data, process capability analyses, Pareto charts and cause-and-effect diagram.

## References

Montgomery, D.C. (2000) *Introduction to Statistical Quality Control*, 4th ed. New York: John Wiley & Sons.

Wetherill, G.B. and Brown, D.W. (1991) *Statistical Process Control*. New York: Chapman & Hall.

*Luca Scrucca*
*Dipartimento di Scienze Statistiche, Università degli Studi di Perugia, Italy*
luca@stat.unipg.it

# Least Squares Calculations in R

**Timing different approaches**

*by Douglas Bates*

## Introduction

The S language encourages users to become programmers as they express new ideas and techniques of data analysis in S. Frequently the calculations in these techniques are expressed in terms of matrix operations. The subject of numerical linear algebra - how calculations with matrices can be carried out accurately and efficiently - is quite different from what many of us learn in linear algebra courses or in courses on linear statistical methods.

Numerical linear algebra is primarily based on decompositions of matrices, such as the LU decomposition, the QR decomposition, and the Cholesky decomposition, that are rarely discussed in linear algebra or statistics courses.

In this article we discuss one of the most common operations in statistical computing, calculating least squares estimates. We can write the problem mathe-