Budget Friendly Statistical Software

Harry B. Rowe
www.rowequality.com

December 28, 2012

I frequently see questions on online forums about statistical software. “What is a good statistical software package?” “What do you think about package X?” “Is package Y worth the cost?”

As with many other things, the answer is “That depends.”

The statistical software market presents a number of options. The correct choice *for you* and *for now* depends on several factors:

- What is your available budget? How much money do you have? How valuable is it to you to save time?
- What type of analysis do you want to do?
- For whom are you doing the analysis?
- How frequently will you repeat the analysis?
- What computing platform do you use?
- What is your level of statistical skill?

When quality professionals think about statistical software, or when you do a Google search for “statistical software”, the names that come up first are those of the “top of the line” packages like MiniTab, SAS, JMP, and SPSS. These are sophisticated, capable, and fairly user-friendly options. Unfortunately, the prices are also high – over $1,000 for each unless you are eligible for a special corporate, student, or academic discount.

For some people, these are the logical choice. If you serve colleagues or customers who are used to seeing data in the formats one of these packages produces, this may make a specific package almost mandatory. If your employer has adopted one of these packages as a standard, or simply has a large budget for such expenses, the price may not be a significant obstacle.

But if you have a constrained budget, or if you only occasionally use a statistical package, these high-end packages may be over-kill for you. So let’s look at some more budget friendly options.

Often, the easiest tool to use is one you already have and are familiar with. In most companies, every computer has a spreadsheet package already installed, usually Microsoft Excel. If not, there are open-source office suites available for free download. And you probably already know how to use many of the basic features of the program.

Microsoft Excel, in its most basic installation, has the capability to do basic descriptive statistics and hypothesis testing. Functions are available for most common probability distributions. Linear regression is available also. Graphing support is also quite good.

Although it is not always installed in corporate standard installations, Windows versions of Excel are shipped with an add-in called the "Analysis ToolPak". This add-in provides additional statistical capability for Excel including such things as histograms, covariance, correlation, and analysis of variance.

With these features, Excel is a powerful statistical tool. Although it requires you to look up some formulas, it can easily be used for Statistical Process Control charts. In fact, I have used it several times as the basic tool for a multi-day SPC workshop.

If you use Excel on a Mac, however, the Analysis ToolPak is not available. In its place, Microsoft suggests you download and use StatPlus Mac from AnalystSoft. StatPlus Mac LE (for "Limited Edition" is a free download and provides capabilities similar to the Analysis ToolPak.

If you elect to go with Excel, you may want to consider purchasing a copy of "Six Sigma Statistics with Excel and MiniTab" by Issa Bass. The Kindle version from Amazon is very economical.

If your budget can't support even the modest cost of Excel, or if Linux is your computing platform of choice, then you can download one of the free open-source office suites like LibreOffice.org. These offer a fairly comprehensive set of basic statistics and graphics functions. In this case, you should take a look at "Introduction to Statistics Using LibreOffice.org Calc" by Dana Lee Ling, College of Micronesia at www.comfsm.fm/~dleeling/statistics/text.html#page-011.

Stepping up slightly from Excel, there are a number of third-party statistics add-ins for Excel that automate many standard statistical analysis and graphing tasks. As an example, QI Macros (www.qimacros.com), provides additional menus and wizards to automate tasks like control charting, hypothesis testing, anova, Pareto charts, histograms, etc. QI Macros relieves you of the need to remember or look up formulas, or build templates for frequently used analyses. It costs $200 (in addition to the cost of Excel) and works on both Windows and Mac platforms. While I am personally only familiar with QI Macros, there are a number of other similar products on the market including XLSTAT, Analyse-It, and StatPlus, the paid version of StatPlus LE which was mentioned above.

In the above paragraphs, it may seem that there is a direct correlation between cost and functionality, requiring a higher budget for each increment of capability. There is one final option that turns that model on its head.

R is an extremely powerful and flexible statistical and graphical computing environment. It is also free software. You can find out more about it at www.r-project.org. Versions can be downloaded for Windows, Apple Mac, and Linux/Unix.

R is an open-source version of S, a statistical computing environment developed at Bell Laboratories beginning in the mid-1970's and developed through the 1980's and 1990's. (A commercial version of S, called S-Plus, is available from Tibco Software, Inc. as part of its Spotfire software suite.)

R differs from all of the previously described software in that it is modeled on the paradigm of a programming language and uses a "command line" rather than a "point and click" interface. It also allows the creation of user-written functions that, once entered, can be executed simply by typing the function's name and arguments. This gives rise to the capability to extend the language by writing functions that can be read in as needed. R is incredibly powerful alone, but that power is multiplied by a large number of special-purpose add-on packages that have been contributed to the R library over time.

While R's command-line interface can seem daunting, it is really fairly easy to use after a familiarization period. R provides an extensive "help" facility. Simply use the "help" command at the prompt to open a separate window with help on any command. For example, for help on the t-test:

```
> help(t.test)
```

There are also a number of built-in demonstrations available. Typing "demo()" at the prompt will produce a list of available demonstrations. Typing "demo(graphics)" produces a demonstration of R's extensive graphics capabilities.

A full-blown tutorial on R is too large a topic to include here, but a few examples will show some of the power of the tool.

R indicates it is ready for a command by printing a ">" character. The rest of the line is the command.

Note that variables in R can refer to scalars, vectors, or matrices. If x and y are vectors of equal length, the R command z = x + y results in the creation of a vector z containing the element by element sums of x and y.

Create a vector containing ten numbers drawn from a normal distribution with mean of 1 and standard deviation of 0.1.

```
> x=rnorm(10,mean=1,sd=.1)
```

Display the values of x

```
> x
 [1] 0.9853732 0.9921775 0.9285087 0.8429696 1.2363707 1.2051573
1.0841361
 [8] 1.0362515 0.9743297 0.9436368
```

Calculate the mean of the sample x.

```
> mean(x)
[1] 1.022891
```

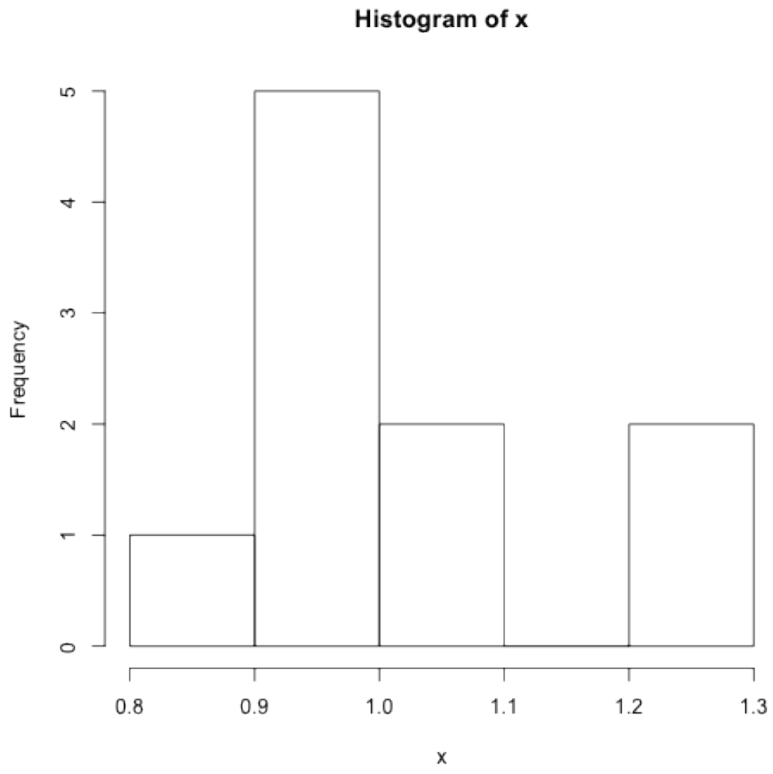
Calculate the standard deviation of the sample.

```
> sd(x)
[1] 0.1224718
```

Plot a histogram of the data.

```
> hist(x)
```

The histogram appears in a separate graphics window, like so.



The plot can be saved to a graphics file for inclusion in a document (like this one) by using another pair of commands. (The text in black is printed by R in response to the commands and may appear different depending on the platform you are using.)

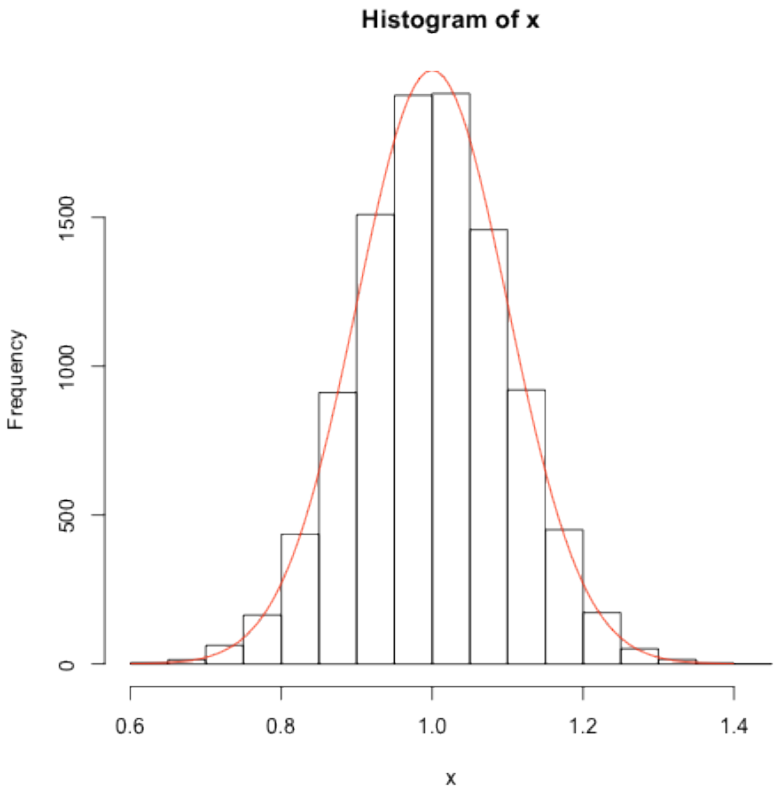
```
> dev.copy(png,"hist_example.png")
```

```
quartz_off_screen
                3
> dev.off()
quartz
     2
```

And to show that R has the capability to do use much larger datasets and produce more sophisticated graphics:

```
> x=rnorm(10000,mean=1,sd=.1)
> hist(x)
> curve(500*dnorm(x,mean=1,sd=.1),from=0.6,to=1.4,col="red",
add=TRUE)
```

These commands generate a vector x consisting of 10,000 values from a normal distribution with mean 1.0 and standard deviation 0.1, plots a histogram of x, and overlays a scaled normal probability density function with those parameters.



And of course we could have specified custom titles, axis labels, scales, etc.

It is also easy to use R with data imported from, or exported to, spreadsheets or any other program that can utilize a comma separated variables file format.

Assume we have an Excel spreadsheet with a table of twenty-five pairs of variables stored as three columns. Column A is the sample number, column b is the x value, and column c has the y value. We export the data from Excel, saving it in a comma separated variables file called “demo.csv”.

In R, we read the data into an object (called a data frame) and name it “data”.

```
> data=read.csv("demo.csv")
```

We can look at the data by entering the object’s name as a command.

```
> data
    X          x         y
1   1 15.5329335 18.478188
2   2 20.8430080 23.990611
3   3  0.2958812  2.918465
4   4 24.1998225 27.091970
5   5  3.2966312  9.095634
6   6 18.4986153 21.180462
7   7 16.0331331 18.890410
8   8 16.7858134 20.372368
9   9  2.5056059  5.082632
10 10 24.8980288 26.560612
11 11  1.0685907  3.508250
12 12  1.4966008  4.122090
13 13 21.3151932 25.202404
14 14  2.4031388  5.570588
15 15 10.7520337 14.926838
16 16  8.5595530 11.096544
17 17 15.0340428 18.099291
18 18  8.0645751 12.169878
19 19 23.2493749 26.448504
20 20 24.3125881 27.499447
21 21  6.3417876 10.528603
22 22  3.1838070  5.852978
23 23  3.8080607  7.372762
24 24 24.7345416 27.524054
25 25  6.4672872  9.106021
```

As you can see, R also displays the column names and a row number.

While we can access the data now by row and column numbers, it would be more convenient if we could refer to them by the column names. We can do this if we issue the following command.

```
> attach(data)
```

We can do some simple descriptive statistics like calculating means and standard deviations.

```
> mean(x)
[1] 12.14723
> sd(x)
[1] 8.860415
> mean(y)
[1] 15.30758
> sd(y)
[1] 8.776824
```

Let's take a closer look at y.

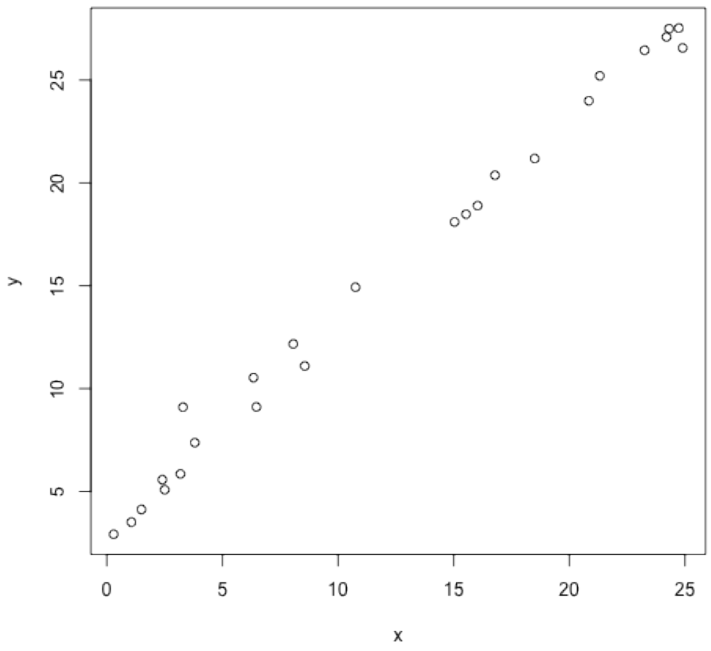
```
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.918   7.373  14.930  15.310  23.990  27.520
```

This shows the values of the minimum, maximum, median, mean, and first and third quartile of y.

We can make a scatter plot of y versus x quite simply.

```
> plot(x,y)
```

This is the plot that appears.



Hmmm. Looks like x and y are correlated. Let's check by calculating the correlation coefficient.

```
> cor(x,y)
[1] 0.995764
```

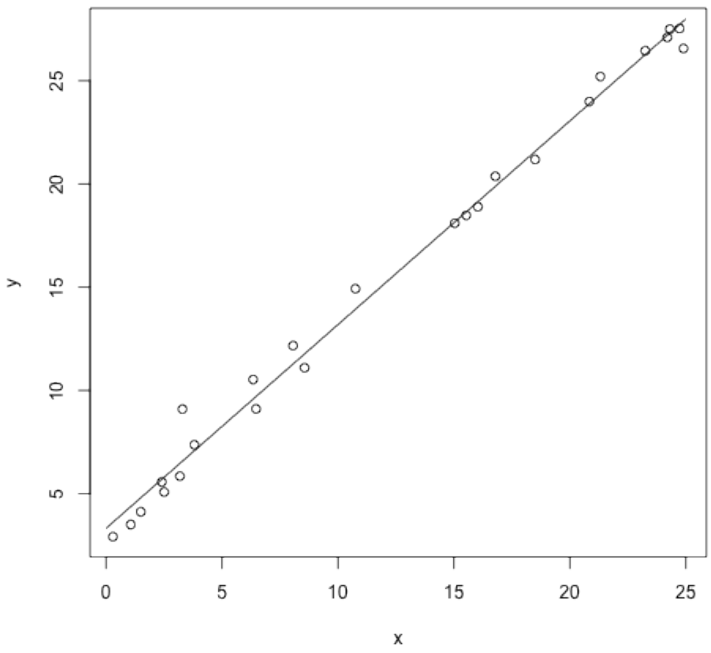
Yes, highly correlated. So let's fit a linear regression model to the data. We use y as the dependent variable and x as the independent variable. (We're barely scratching the surface of R's linear modeling capabilities with this example.) The command says "fit a model to y as a linear function of x".

```
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     3.3259       0.9864
```

This gives us a modeled slope of 0.9864 and intercept of 3.3259. Let's now add this best fit line to the scatter plot.

```
> curve(3.3259+.9864*x,from=0,to=25,add=TRUE)
```

Now we have a plot that looks like this.



It looks like a linear equation with random noise. Let's isolate the noise to look at that. We first subtract the fitted line from the y values and store the resulting deviations in a vector z. In R, we can manipulate the values in a vector with a simple formula.

```
> z=y-3.3259-.9864*x
```

Then we can investigate the characteristics of the noise. We calculate the mean and standard deviation of z as follows.

```
> mean(z)
[1] -0.0003394591
> sd(z)
[1] 0.8069943
```

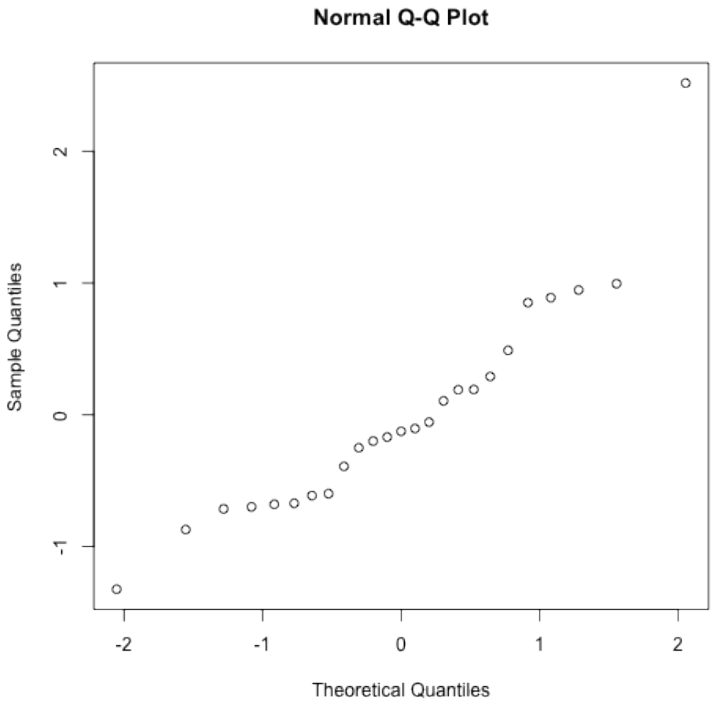
We can also calculate a 90% confidence interval for the mean of the noise. These commands use the inverse normal probability function of R to locate the endpoints of the confidence interval representing 5% and 95% probability.

```
> qnorm(.95,mean=mean(z),sd=sd(z)/sqrt(25))
[1] 0.265138
```

```
> qnorm(.05,mean=mean(z),sd=sd(z)/sqrt(25))
[1] -0.265817
```

We can make a quantile-quantile normal probability plot of x with the qqnorm command.

```
> qqnorm(z)
```



R also provides easy access to many standard statistical tests such as t-tests (the t.test command) and f-tests (the var.test command).

We can apply the Shapiro-Wilk normality test to our noise data with the shapiro.test command.

```
> shapiro.test(z)

    Shapiro-Wilk normality test

data:  z
W = 0.9128, p-value = 0.03513
```

This would have us reject normality at the 95% confidence level.

Inspection of the quantile-quantile plot however suggests this may be because of a single outlier. We can remove the outlier from the data as follows. This command says "Create a variable w containing all of the elements of z whose values are less than 2.5."

```
> w=z[z<2.5]
```

Repeat the Shapiro-Wilk test with the new data set.

```
> shapiro.test(w)

    Shapiro-Wilk normality test

data:  w
W = 0.9555, p-value = 0.3548
```

With the outlier removed, the Shapiro-Wilk test reports a p-value of 0.3548. We cannot reject normality.

This is but a small example of the capabilities of R.

The R-Project web site provides links to a number of R tutorials. There are also a number of books available on R. One that may be particularly useful is "Using R for Introductory Statistics" by John Verzani.

While R does not provide built-in functionality for Statistical Process Control charting, there are a number of (also free) downloadable libraries that provide support for SPC. The one I use is "qcc", authored and maintained by Dr. Luca Scrucca of the Department of Economics, Finance, and Statistics of the University of Perugia in Italy. Many other libraries are available for other specialized types of analysis.

Since leaving my "corporate" job over three years ago, I have used R almost exclusively for my statistical work, as well as for general mathematics, numerical analysis, and modeling. I do descriptive statistics, hypothesis testing, survey analysis, and even some more exotic things like forays into cluster analysis (using K-means). After spending some time climbing the learning curve, I have found R to be a completely acceptable substitute for the expensive software I used in the corporate world.