Probability Plot Examples

Dave Lorenz

July 7, 2015

These examples demonstrate variations of types of probability plots that can be generated by functions in the **smwrGraphs** package. All of the examples use randomly generated sets of data. **NOTE:** to use any of the high-level plotting functions, you must first call a function to set up the graphics environment like `setPage` or `setPDF`, but these are not included here to use the graphics tools in `Sweave`.

```
> # Load the smwrGraphs package
> library(smwrGraphs)
> # Generate the random data
> set.seed(2736)
> Xnorm <- rnorm(32)
> Xlogn <- rlnorm(32)
> Xmix <- exp(c(rnorm(15), rnorm(15, 0.5))) + .5
> Xbig <- rnorm(100)
```

1 Empirical Distribution Function

The empirical distribution function describes the cumulative distribution given an empirical measure of the observations. The empirical measure is the step function that increases by 1/n for each of the sorted observations. This graph is often used to explore or describe the observations without describing them in the context of any distribution, like normal.

```
> # setSweave is a specialized function that sets up the graphics page for
> # Sweave scripts. It should be replaced by a call to setPage or setPDF
> # in a regular script.
> setSweave("probplot01", 6 ,6)
> # Create the graph. Note that by default, the x-axis is log-transformed and
> #  requires strictly positive data. Setting xaxis.log to
> #  FALSE relaxes that requirement.
> ecdfPlot(Xmix)
> # Required call to close PDF output graphics
> graphics.off()
```

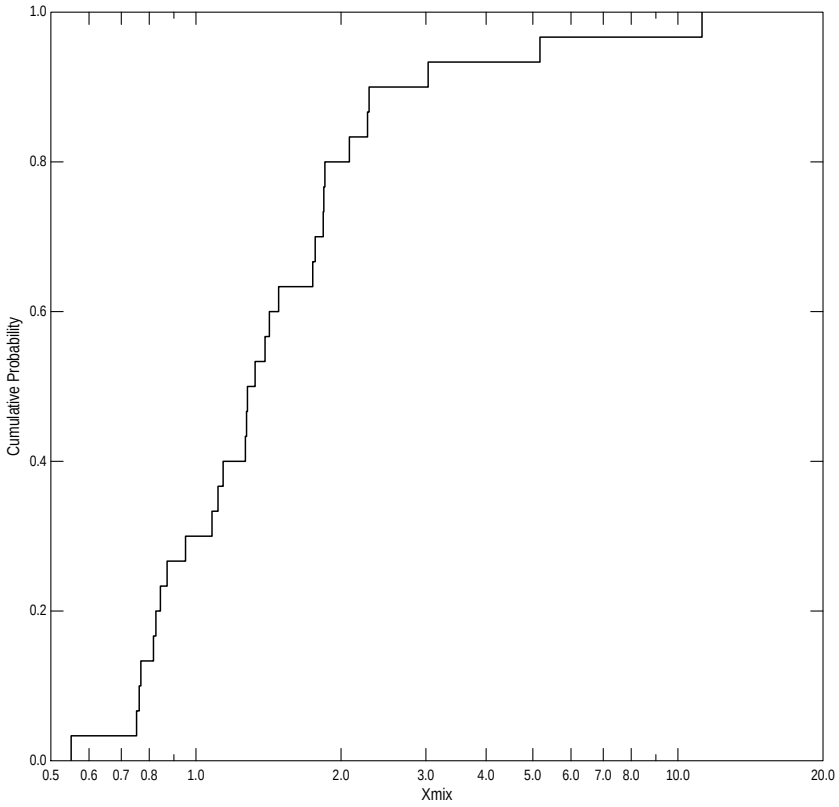

Figure 1. The ECDF plot.

2 Probability Plots

Probability plots describe the observed values in the context of a known distribution. The most common, and default distribution for the `probPlot` function, is the normal distribution. Many other distributions are possible. Log-normally distributed data can be portrayed in either of two ways–either use set `yaxis.log` to `TRUE` and optionally provide the mean and standard deviations of the natural logarithms of the data or `yaxis.log` to `FALSE`, `distribution` to "lognormal" and provide the mean and standard deviations of the natural logarithms of the data. The first option is shown in figure 2..

```
> setSweave("probplot02", 6 ,6)
> # For the normal distribution, the mean and sd arguments are optional, if
> #  supplied, then a line for the fitted distribution is drawn. The default
> #  setting for yaxis.log is TRUE, so the logs of the data are required for the
> #  mean and standard deviation.
> probPlot(Xlogn, mean=mean(log(Xlogn)), sd=sd(log(Xlogn)))
> graphics.off()
```

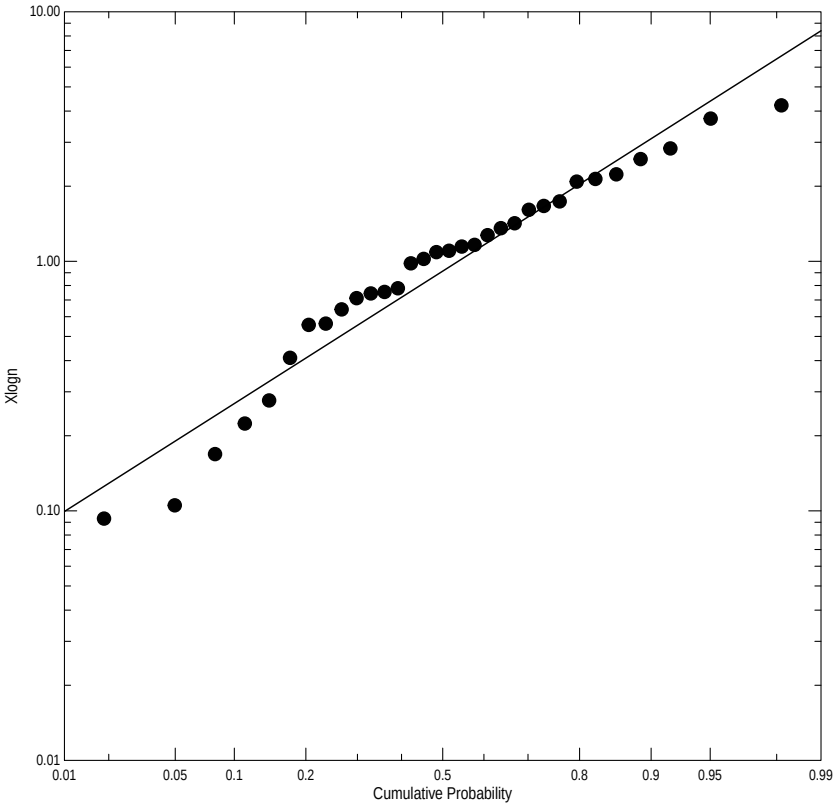

Figure 2. Probability plot for lognormal data.

3 The Q-normal Plot

The Q-normal plot is related to the probability plot for normal distributions, but uses quantiles of the normal distribution instead of probabilities. It is typically used for diagnostic plots to quickly indicate deviations from an assumption of normality.

```
> setSweave("probplot03", 6 ,6)
> # Accept all of the defaults, the line is based on the mean and the standard
> #  deviation of the data.
> qqPlot(Xnorm)
> graphics.off()
```

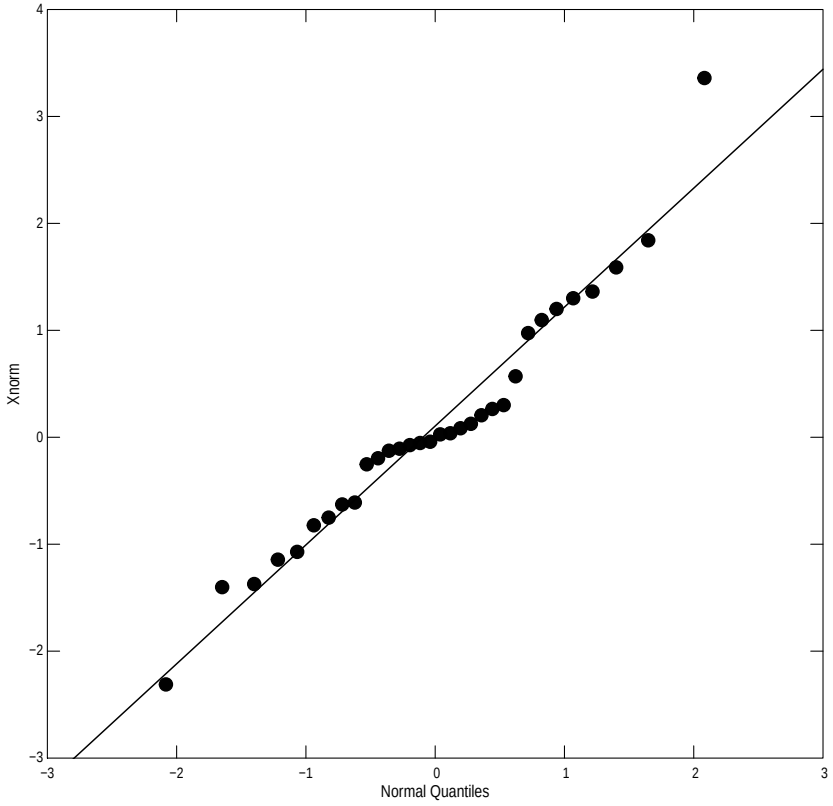

Figure 3. The Q-normal plot.

4 The Q-Q Plot

The Q-Q plot is designed to compare the distribution of two samples. The larger sample is subsetted to match the length of the smaller sample, if necessary and matching quantiles are plotted against each other. For this plot, two lines are shown—the best fit line (black) and the 1:1 line (gray). If the distributions are similar the lines will be close to each other. No assumption is made about the distribution other than similarity.

```
> setSweave("probplot04", 6 ,6)
> # Accept all of the defaults.
> qqPlot(Xnorm, Xbig)
> graphics.off()
```

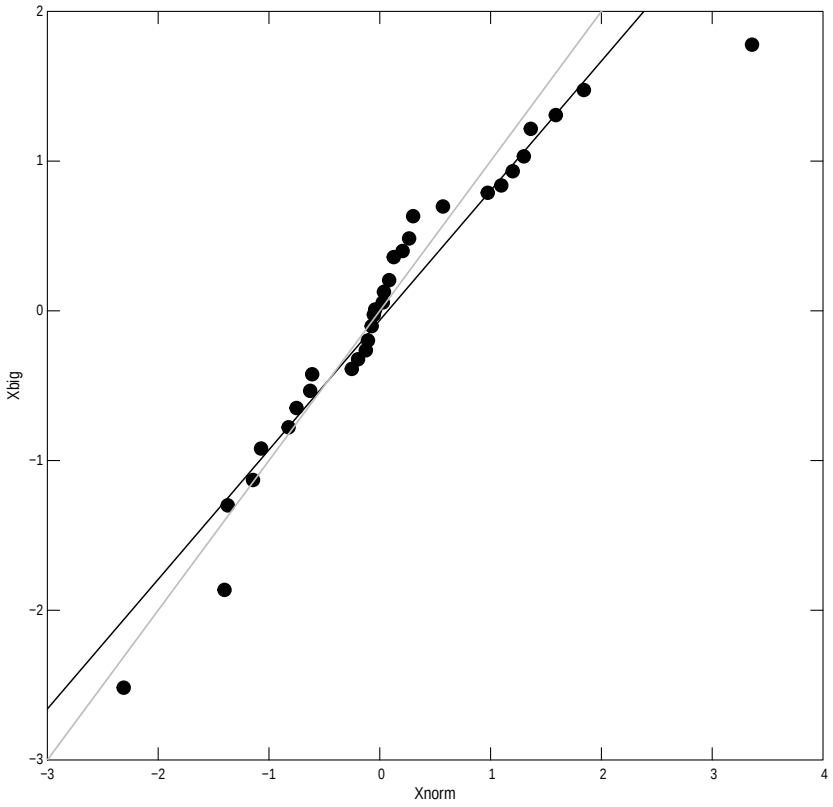

Figure 4. The Q-Q plot.

5 Histogram

The histogram describes the probability distribution given counts or the proportion of the data in bins along the range of the observations. The density of the data are portrayed as rectangles with fixed width and height proportional to the number of observations within each bin. The histogram is most useful for relatively large datasets because the appearance is sensitive the number in each bin, which can vary widely for small data sets. No assumption is made about the distribution.

```
> setSweave("probplot05", 6 ,6)
> histGram(Xbig)
> # Required call to close PDF output graphics
> graphics.off()
```

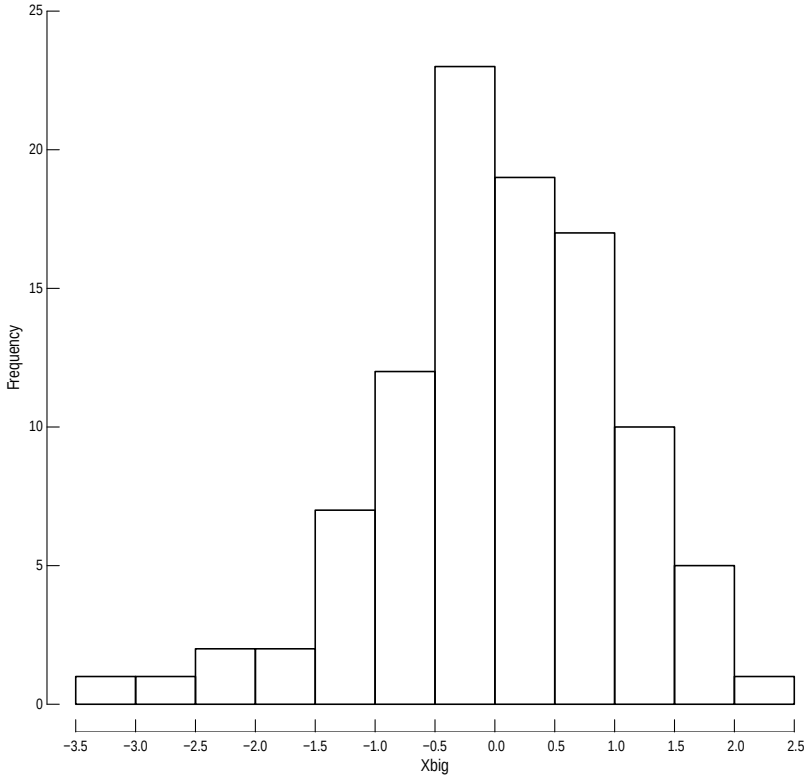

Figure 5. The histogram.

A frequent variation on the histogram is plotting the density curve on top of the histogram. That type of histogram requires the `type` component of the `Hist` argument be set to "density." Because density estimation generally extends the x-axis beyond the usual limits of histogram estimation, the first step should be to compute the density estimate and set the x-axis limits to match those from the density estimate.

```
> # Compute the density
> Xbig.den <- density(Xbig)
> range(Xbig.den$x)

[1] -4.019955  3.310997

> setSweave("probplot06", 6, 4)
> # Set type to density and xaxis range to -4, 3.5 by setting breaks
> histGram(Xbig, breaks=seq(-4, 3.5, by=.5), Hist=list(type="density"))
> # Add the density line, the defaults all work so current arg not needed
> with(Xbig.den, addXY(x, y))
> # Required call to close PDF output graphics
> graphics.off()
```

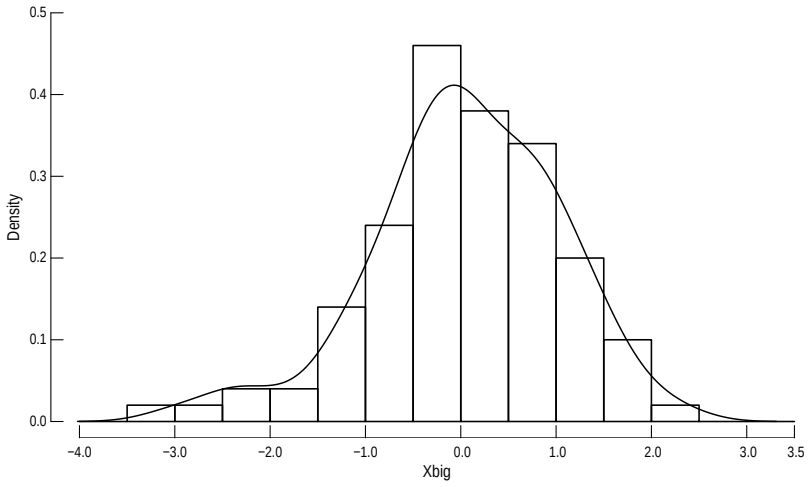

Figure 6. The density histogram.

6 Adding Plots to Probability Graphs

In general adding plots to any of the probability plots requires additional manipulations to sort the data and construct the correct coordinates for plotting. The examples in this section demonstrate how to add plots to the ECDF plot, the probability plot, and the Q-normal plot.

The `ecdfPlot` function has the `group` argument that can be used to construct multiple ECDF plots in the same graph. If the data to plot come from different sources and are not stacked, then the `addXY` function can be used to add the additional ECDFs. The required steps are sort the data and repilcate the first (smallest) value. The steps are described for the production of figure 7A shown in the following code.

Adding plots to the graphs created by the `probPlot` function requires sorting the data and creating plotting points using the `ppoints` function with the `a` argument set to the value for the `alpha` argument in `probPlot`; the default value is 0.4. The `probPlot` function is set up to use plotting positions regardless of the distribution that is used, but statistical parameters cannot be supplied for that distribution. The steps for adding to a probability plot are described for the production of figure 7B shown in the following code.

Adding plots to a Q-normal graph created with the `qqPlot` function is very similar to adding plots to the probability plot, except that the x-axis coordiante must be converted to the normal quaintiles. The steps for adding to a Q-normal plot are described for the production of figure 7C shown in the following code.

```
> # Set up the output
> setSweave("probplot07", 6, 8)
> # Allocate 3 graphs
> AA.lo <- setLayout(num.rows=3)
> # Figure 7A
> setGraph(1, AA.lo)
> AA.pl <- ecdfPlot(Xmix)
> # Add another plot, simply Xmix + 0.5
> Xadd <- Xmix + 0.5
> # Replicate the smallest values and sort the data
> Xadd <- c(min(Xadd), sort(Xadd))
> # The y-axis coordinates are the sequence from 0 to 1 with legnth matching the data
> addXY(Xadd, seq(0, 1, length.out=length(Xadd)),
+   Plot=list(what="stairstep", color="blue"), current=AA.pl)
> addTitle(Heading="A")
> # Figure 7B
> setGraph(2, AA.lo)
> AA.pl <- probPlot(Xlogn)
> # Add another plot, simply Xlogn + 0.5
> Xadd <- Xlogn + 0.5
> # Sort the data
> Xadd <- sort(Xadd)
> # The x-axis coordinate are the plotting positions
> addXY(ppoints(Xadd, a=0.4), Xadd,
+   Plot=list(what="points", color="blue"), current=AA.pl)
> addTitle(Heading="B")
> # Figure 7C
> setGraph(3, AA.lo)
```

```
> AA.pl <- qqPlot(Xnorm)
> # Add another plot, simply subset Xbig
> Xadd <- sample(Xbig, 20)
> # Sort the data
> Xadd <- sort(Xadd)
> # The x-axis coordinate are the plotting positions
> addXY(qnorm(ppoints(Xadd, a=0.4)), Xadd,
+   Plot=list(what="points", color="blue"), current=AA.pl)
> # And add the line representing the mean and sd
> refLine(coefficients=c(mean(Xadd), sd(Xadd)),
+   Plot=list(color="blue"), current=AA.pl)
> addTitle(Heading="C")
> # Required call to close PDF output graphics
> graphics.off()
```

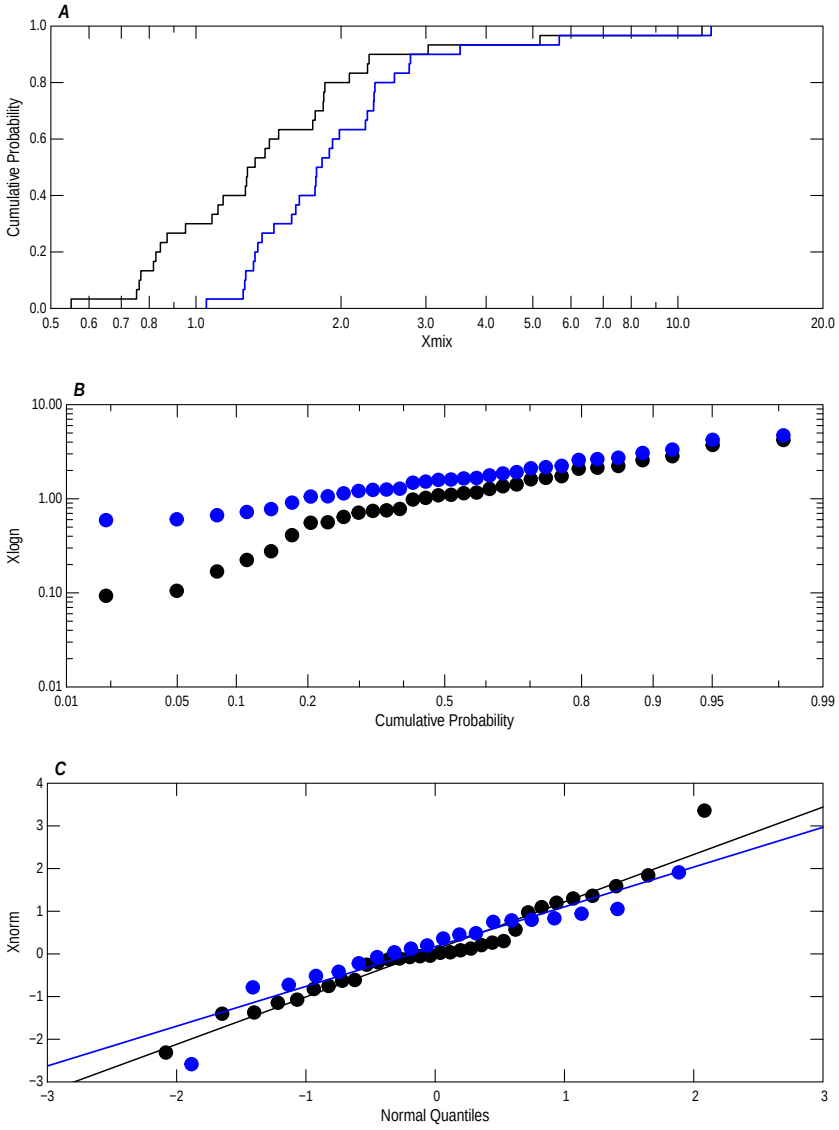

Figure 7. Example probability plots with added plots.