3

Visualizing Univariate Distributions

Visualizing the distribution of a single continuous variable is a common graphical task for which several specialized methods have evolved. The distribution of a random variable X is defined by the corresponding cumulative distribution function (CDF) $F(x) = P(X \leq x)$. For continuous random variables, or more precisely, random variables with an absolutely continuous CDF, an equivalent representation is the density $f(x) = F'(x)$. One is often also interested in the inverse of F, the quantile function. R provides these functions for many standard distributions; for example, `pnorm()`, `dnorm()`, and `qnorm()` give the distribution, density, and quantile functions, respectively, for the normal distribution. Most of the visualization methods discussed in this chapter involve estimating these functions from data. In particular, density plots and histograms display estimates of the density f, and quantile plots and box-and-whisker plots are based on (partial) estimates of F or its inverse.

Although the mathematical relationships between the theoretical constructs are well-defined, there are no natural relationships between their standard estimates. Furthermore, the task of visualization comes with its own special rules; two plots with exactly the same information can put visual emphasis on entirely different aspects of that information. Thus, the appropriateness of a particular visualization depends to a large extent on the purpose of the analysis. We discuss the merits of different visualizations as we encounter them, but it is helpful to keep this background in mind when reading about them.

3.1 Density Plot

As we have already seen, using the `Chem97` dataset in Chapter 1, the `densityplot()` function produces kernel density plots. In that example, the densities estimated for the six `score` groups were all unimodal (i.e., they had one peak), and differed from each other essentially in their location, variability, and skewness (i.e., the first three moments). This is a common scenario in

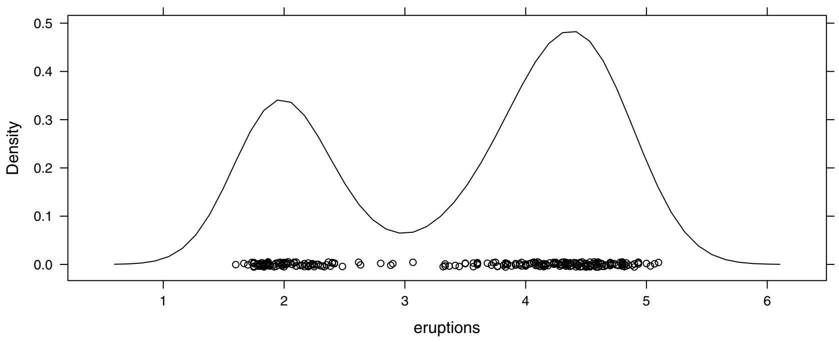

Figure 3.1. A kernel density plot of eruption times of the Old Faithful geyser. All optional arguments retain their defaults; in particular, the Gaussian kernel is used to compute the estimated density, and the raw data values are plotted with slight jittering.

many statistical analyses, but better graphical tools than density plots exist for it, as we show later in this chapter. Density plots are, however, particularly useful for detecting bimodality or multimodality.

Our first example uses the `faithful` dataset (Azzalini and Bowman, 1990; Härdle, 1990), a favorite in density estimation literature. The dataset records the duration of eruptions of the Old Faithful geyser in Yellowstone National Park, and the waiting time to the next eruption, over a period of a few days in 1985. We only look at the distribution of the durations. Figure 3.1 is produced by

```
> densityplot(~ eruptions, data = faithful)
```

By default, along with the estimated density, the points are plotted with some vertical jittering to address overplotting and ties. The `plot.points` argument can be used to change it to a "rug" as in the next example, or omit the points entirely.

There is a variety of approaches to density estimation, of which only the one implemented in the R function `density()` is available through the default panel function `panel.densityplot()`. It is fairly simple to implement other approaches, and we show an example in Figure 13.3. `density()` itself comes with several arguments to control the calculations, and these can be supplied directly to `densityplot()`. The two most important arguments are `kern`, which specifies the "kernel" used, and `bw`, which determines the bandwidth. The default kernel used in Figure 3.1 was the Gaussian. In Figure 3.2, we use the rectangular kernel instead, with a fixed bandwidth rather than a data-dependent one.

```
> densityplot(~ eruptions, data = faithful,
              kernel = "rect", bw = 0.2, plot.points = "rug", n = 200)
```

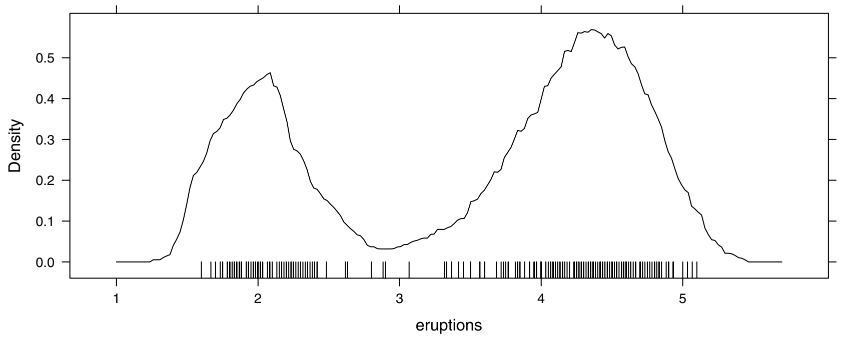

Figure 3.2. Another kernel density plot of the Old Faithful eruption times, this time using the rectangular kernel and a predetermined bandwidth. This is also known as an averaged shifted histogram (Scott, 1985), because it can be obtained as the average of all histograms with a fixed bin width.

Other kernel and bandwidth options are described in the help page for `density()`.

3.2 Large datasets

The datasets we have encountered so far are fairly small. Even the `Chem97` data, the largest we have seen, has only around 30,000 observations. Modern datasets, for example, those that arise from high-throughput biological assays, can easily exceed these sizes by many orders of magnitude. In this paradigm, careful thought is required about the storage of such data, as well as their analysis, including visualization. As a representative example, we use data from a flow cytometry (FCM) experiment. As we are primarily interested in issues related to visualization, we will use, for the most part, a small subset of the data that can be conveniently manipulated in the familiar data frame form. We briefly discuss the important practical issues of storage and efficiency in Chapter 14.

The full dataset (Rizzieri et al., 2007; Brinkman et al., 2007) originated from a collection of weekly peripheral blood samples obtained from several patients following allogeneic blood and marrow transplant. The goal of the study was to identify cellular markers that would predict the development of graft-versus-host disease (GvHD). Samples were taken at various time points before and after transplantation. Our "toy" example, available in the latticeExtra package as the `gvhd10` dataset, represents samples obtained from one patient at seven time points. The blood samples were labeled with four different fluorescent probes to identify targeted biomarkers and a flow cytometer was used to determine fluorescent intensity for individual cells. The number

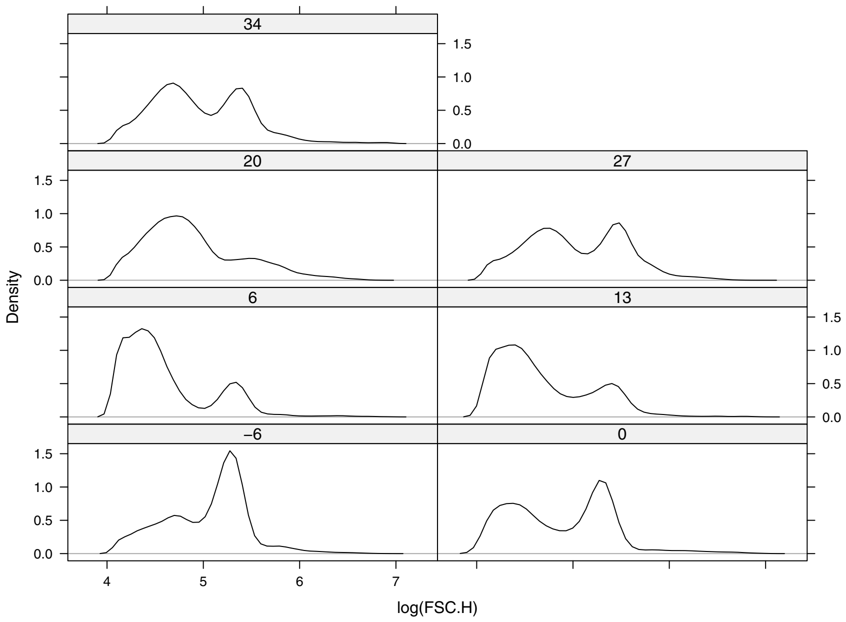

Figure 3.3. Kernel density plots of forward scatter (FSC) measurements of cells taken from a single transplant patient at different time points, on a logarithmic scale. The numbers in the strips represents days past transplant, with negative numbers representing days prior to it. FSC measurements are a surrogate for cell size.

of cells measured varied between approximately 4500 and 25,000 in the seven samples.

Flow cytometry data have their own special set of complexities that make analysis challenging. A full discussion of these complexities is beyond the scope of this book, but one important feature is that observations usually represent a mixture of multiple cell populations, not all of which are of interest. This is usually reflected in the marginal densities of individual marker intensities. In Figure 3.3, we look at densities of forward scatter, a measure of cell size, in blood samples taken at different time points.

```
> library("latticeExtra")
> data(gvhd10)
> densityplot(~log(FSC.H) | Days, data = gvhd10,
              plot.points = FALSE, ref = TRUE, layout = c(2, 4))
```

As the number of points is large, we do not plot them, instead adding a reference line. Other than this, the size of the dataset causes no problems in visualization, although the step of computing the density itself is more intensive.

3.3 Histograms

Histograms are also density estimates, somewhat cruder than kernel density estimates and possessing worse theoretical properties, but invaluable in the days before computers were ubiquitous. Histograms are created by dividing up the range of the data into non-overlapping bins, usually of the same length, and counting the number of observations that fall in them. Each bin is then represented by a rectangle with the bin as its base, where the height of the rectangle is computed to make its area equal the proportion of observations in that bin. This is formally known as the density histogram, because the result is a true probability density whose total area equals one. Other popular variants are the relative frequency histogram, where heights are relative frequencies, and the frequency histogram, where the heights are frequency counts within each bin. As long as all the bins have the same width, the heights in the three cases are multiples of each other (i.e., the corresponding histograms have the same shape but different y-scales). Unequal bin widths are rarely used outside introductory statistics textbooks. In Figure 3.4, we use the `lattice` function `histogram()` to present the same data as Figure 3.3. The `type` argument is used to compute the heights as density rather than the default of relative frequency. The number of bins (intervals) is increased to 50 because the number of observations is fairly large.

```
> histogram(~log2(FSC.H) | Days, gvhd10, xlab = "log Forward Scatter",
            type = "density", nint = 50, layout = c(2, 4))
```

This rendering emphasizes a feature not as obvious in the density plot, namely, that there is a fairly distinct lower bound for the observations, below which the density drops quite abruptly. This is an inherent limitation of kernel density estimates. Some alternative density estimation techniques can address this limitation if the bound is known in advance.

Despite this apparent advantage, it is rather difficult to justify the use of histograms in preference to density plots. For one thing, histograms are rather sensitive to the choice of bin locations; we would prefer estimates that depended more on the data and less on arbitrary parameter choices. Kernel density estimates can be viewed as a natural generalization of histograms that removes some of this arbitrariness, at least when the bins are of equal size. Specifically, consider a histogram with fixed bin width h. The histogram is entirely defined by the location of the left endpoint of any one bin, which is arbitrary. The *averaged shifted histogram* (ASH; Scott, 1985) removes this arbitrariness by defining the estimated density at a point x as the average value at x of all possible density histograms with bin width h. It can be easily shown that the estimate thus obtained is identical to the kernel density estimate computed using the rectangular kernel, as in Figure 3.2. Density plots are also preferable from the visualization perspective as they lend themselves more easily to superposition, as we have seen in Chapter 1. Histograms are nonetheless popular, not least because they are easier to explain to non-statisticians.

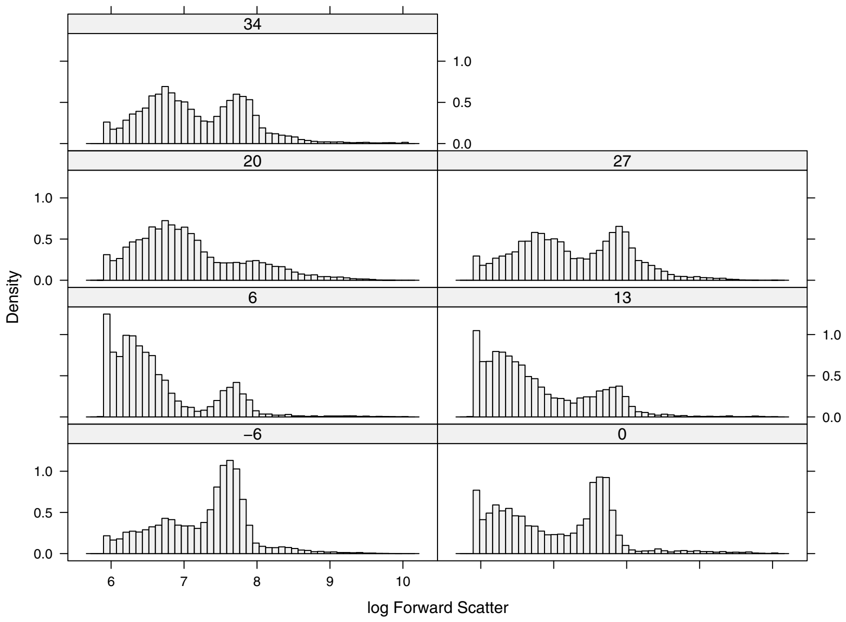

Figure 3.4. Histograms of log-transformed forward scatter measurements for different visits of a patient, with the same layout as Figure 3.3. A distinct lower bound for the measured values stands out much more clearly.

3.4 Normal Q–Q plots

A common task when analyzing continuous univariate data is to compare them to a theoretical distribution. Density estimates emphasize local features such as modes, but are not ideal for judging global features. The most commonly used tool for this job is the theoretical *quantile–quantile* (Q–Q) plot, which graphs quantiles of the observed data against similar quantiles of a probability distribution conjectured to be a reasonable match. For a good fit, a Q–Q plot is roughly linear, with systematic deviations suggesting a lack of fit. This is related to a well-known result from probability theory, that for a continuous random variable X with distribution function F, $F(X)$ has the uniform distribution $\mathcal{U}(0,1)$, which in turn has a linear distribution function. Q–Q plots are particularly effective because the human eye finds it easier to perceive deviations from a straight line than from a curve.

We continue with the `Chem97` example from Chapter 1. The `lattice` function `qqmath()` can be used to create Q–Q plots comparing univariate data to a theoretical distribution. In principle, Q–Q plots can use any theoretical distribution. However, it is most common to use the normal distribution, which is the default choice in `qqmath()`. Figure 3.5 is produced by

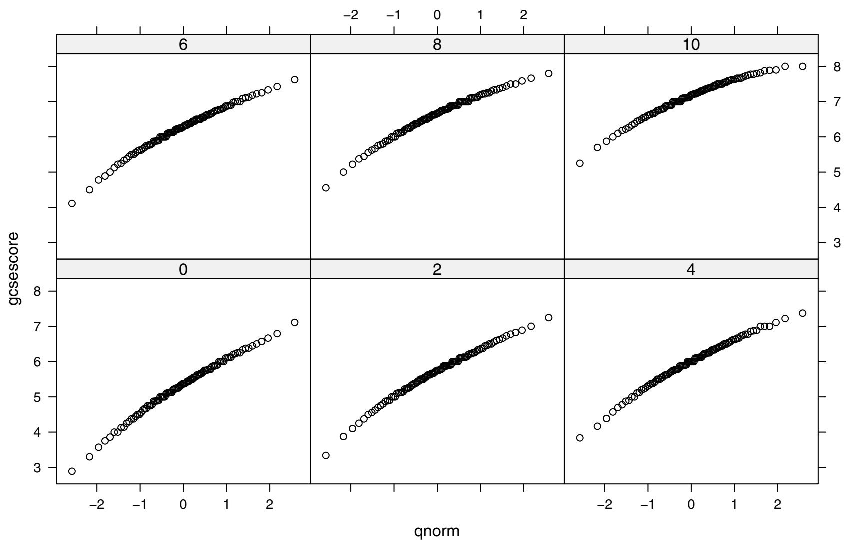

Figure 3.5. Normal Q–Q plots of average GCSE score for different final scores in the A-level chemistry exam. The systematic curvature in the Q–Q plot is indicative of a left-skewed distribution.

```
> qqmath(~ gcsescore | factor(score), data = Chem97,
         f.value = ppoints(100))
```

The formula and the `data` argument used should need no explanation. The other argument, `f.value`, tells `qqmath()` to use only 100 quantiles in each panel, instead of the default of as many quantiles as there are data points (which in this example would give more than 3000 points in each panel).

Figure 3.5 clearly shows systematic convexity, which is consistent with a left-skewed distribution. If we study the plot closely, we can confirm what we observed in Figure 1.3, that higher `score` is associated with higher `gcsescore`, and that the variance of `gcsescore` decreases with `score` (reflected in the decreasing slope of the Q–Q plots). This is clearer if we superpose the Q–Q plots in a single panel as we did with density plots in Chapter 1. Figure 3.6 is produced by

```
> qqmath(~ gcsescore | gender, Chem97, groups = score, aspect = "xy",
         f.value = ppoints(100), auto.key = list(space = "right"),
         xlab = "Standard Normal Quantiles",
         ylab = "Average GCSE Score")
```

We have also added `gender` as a conditioning variable and specified `aspect = "xy"`, which chooses an aspect ratio using the 45° banking rule.

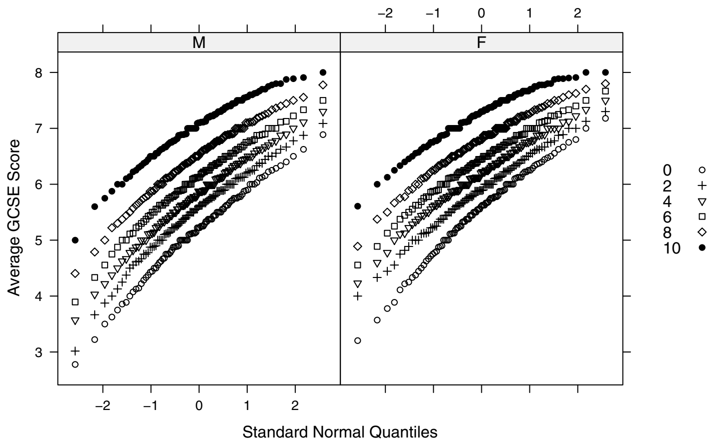

Figure 3.6. Normal Q–Q plots of average GCSE score by gender, grouped by final score. The aspect ratio has been chosen automatically using the 45° banking rule. The systematic curvature is still visible, and superposition now makes it easier to compare slopes, suggesting a systematic change in variance.

3.4.1 Normality and the Box–Cox transformation

The normal distribution plays an important role in many statistical analyses, and nice theoretical results follow if we can assume normality and equal variance, neither of which hold in our example. However, simple power transformations often improve the situation considerably. The Box–Cox transformation (Box and Cox, 1964) is a scale- and location-shifted version of the power transformation, given by

$$f_\lambda(x) = \frac{x^\lambda - 1}{\lambda}$$

for $\lambda \neq 0$, with $f_0(x) = \log x$. This formulation has the advantage of being continuous with respect to the "power" λ at $\lambda = 0$. The "optimal" Box–Cox transformation can be computed by the `boxcox()` function in the MASS package (Venables and Ripley, 2002). A plot of the profile log-likelihood function as a function of λ can be obtained using (result not shown)

```
> library("MASS")
> Chem97.pos <- subset(Chem97, gcsescore > 0)
> with(Chem97.pos,
       boxcox(gcsescore ~ score * gender, lambda = seq(0, 4, 1/10)))
```

One record with a `gcsescore` of 0 has to be omitted from the calculations. In this case, the optimal power is computed as $\lambda = 2.34$. We can visually confirm the success of this transformation using a Q–Q plot of the transformed values, shown in Figure 3.7.

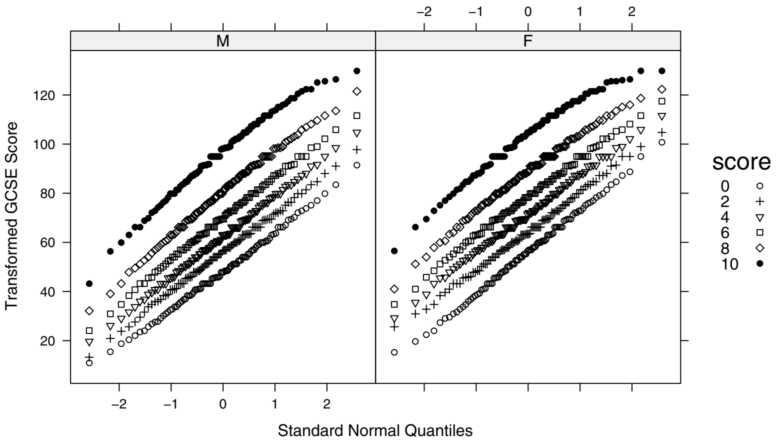

Figure 3.7. Normal Q–Q plots of transformed GCSE score. The transformation appears to have rectified most of the systematic departures from normality and homoscedasticity.

```
> Chem97.mod <- transform(Chem97, gcsescore.trans = gcsescore^2.34)
> qqmath(~ gcsescore.trans | gender, Chem97.mod, groups = score,
         f.value = ppoints(100), aspect = "xy",
         auto.key = list(space = "right", title = "score"),
         xlab = "Standard Normal Quantiles",
         ylab = "Transformed GCSE Score")
```

3.4.2 Other theoretical Q–Q plots

Although less common, distributions other than the normal can also be an appropriate choice as the source of the theoretical quantiles. For example, one standard choice is the uniform distribution, in which case the resulting Q–Q plot is related to the empirical distribution function of the data (see Figure 3.9). If the user is roughly familiar with the shape of common distribution functions, such plots can serve to suggest a good model for the data.

The primary use of quantile plots, however, is as a tool to compare two distributions, and its power comes from the fact that the human eye can better perceive deviations from a straight line than from a curve. To use this fact effectively, the two sets of quantiles compared using a Q–Q plot should arise from the same "expected" distribution. Viewed as a hypothesis test, this means that under the null hypothesis, the distributions compared are effectively the same (up to location and scale); a perceived departure from linearity in the Q–Q plot would lead to a rejection of the null hypothesis. Thus, if the data were expected to come from a certain distribution (not necessarily normal), it would be appropriate to compare against that distribution. One situation where this arises naturally is in simulation studies comparing the empirical and theoretical properties of the sampling distribution of some statistic. Such plots can also serve to demonstrate interesting properties of theoretical distributions; see Figure 10.5 for an example involving the exponential distribution.

3.5 The empirical CDF

A discussion of Q–Q plots would be incomplete without a mention of the empirical cumulative distribution function (ECDF). From a theoretical point of view, the ECDF is the non-parametric maximum likelihood estimate of the cumulative distribution function F. Trellis plots of the the ECDF can be produced by the `ecdfplot()` function in the latticeExtra package. Figure 3.8 is produced by

```
> library("latticeExtra")
> ecdfplot(~ gcsescore | factor(score), data = Chem97,
           groups = gender, auto.key = list(columns = 2),
           subset = gcsescore > 0, xlab = "Average GCSE Score")
```

The `subset` argument is used to remove a single outlier, shrinking the range of the data considerably.

Leaving aside certain technicalities that are largely irrelevant for visualization, the ECDF is closely related to a theoretical Q–Q plot with the uniform distribution as a reference, the difference being that the x- and y-axes are switched. An equivalent Q–Q plot, shown in Figure 3.9, is produced by

```
> qqmath(~ gcsescore | factor(score), data = Chem97, groups = gender,
         auto.key = list(points = FALSE, lines = TRUE, columns = 2),
         subset = gcsescore > 0, type = "l", distribution = qunif,
         prepanel = prepanel.qqmathline, aspect = "xy",
         xlab = "Uniform(0, 1) Quantiles",
         ylab = "Average GCSE Score")
```

It is easy to see that a normal Q–Q plot, or any other theoretical Q–Q plot for that matter, can be obtained by transforming the x-axis of a uniform Q–Q plot by a suitable theoretical quantile function. Similar transformations can be applied to an ECDF plot, but this is less common.

3.6 Two-sample Q–Q plots

Q–Q plots can also be used to directly compare two sets of observations. In theory, these are not much different from Q–Q plots against a theoretical distribution; quantiles from one sample are plotted not against corresponding quantiles from a theoretical distribution, but against those from the other sample. Two-sample Q–Q plots are are created by the `qq()` function. The formula defining such plots may seem somewhat unusual at first, but is natural when the data are stored in a single data frame, and extends naturally to the

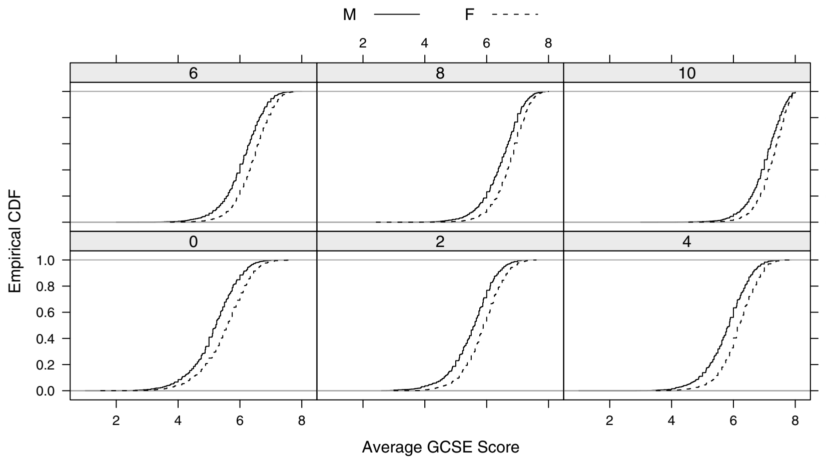

Figure 3.8. Empirical CDF plots of average GCSE scores by final score and gender. The empirical CDF is the non-parametric maximum likelihood estimate of the distribution function F.

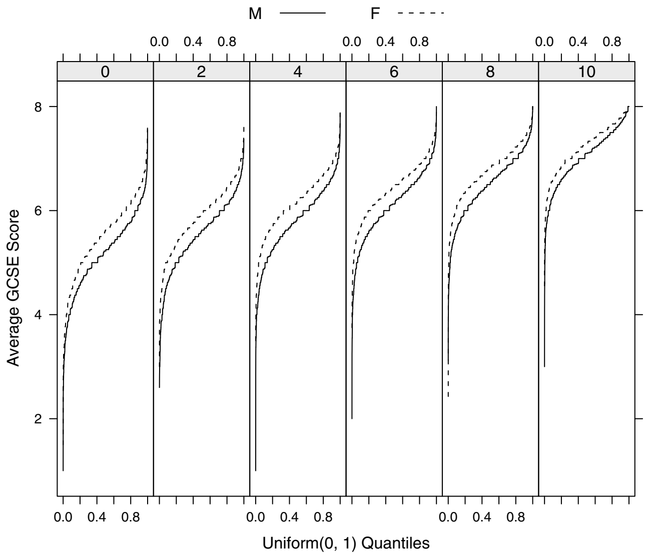

Figure 3.9. Uniform Q–Q plots of average GCSE scores. Modulo certain technicalities, these can be viewed as the inverse of ECDF plots, with the x- and y-axes switched. Neither are particularly useful as diagnostics for lack of fit, but can be used for comparing multiple distributions.

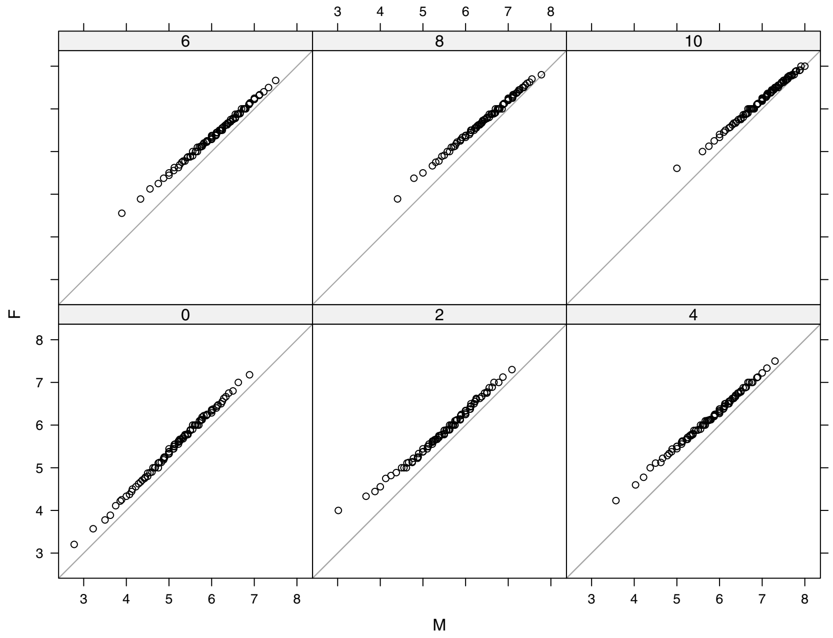

Figure 3.10. Two sample Q–Q plots comparing average GCSE score by gender (which conveniently has two levels), after conditioning on final score. The Q–Q plots are linear, but fall slightly above the diagonal and have slope less than 1 (except for the first panel), suggesting that the distributions of GCSE score are similar, with a higher mean and lower variance for females compared to males. An overall upward shift across panels is also apparent.

`bwplot()` function, which we encounter soon. Specifically, the formula has the form `y ~ x`, where `x` is a numeric vector that consists of both samples, and `y` is a factor of the same length as `x` with exactly two levels defining the two samples. Figure 3.10 shows a Q–Q plot comparing `gcsescore` for males and females after conditioning on `score`.

```
> qq(gender ~ gcsescore | factor(score), Chem97,
     f.value = ppoints(100), aspect = 1)
```

The two axes correspond to quantiles of the two samples. By default, both axes have the same limits, and a diagonal line is added for reference. In this case, the scatter in each panel is linear, but above the diagonal and not quite parallel to it. This suggests that the distributions are similar except for a scale and location change, with `gcsescore` values for females being slightly higher and less variable given a final `score`. A useful variant of this plot can be seen in Figure 11.5.

3.7 Box-and-whisker plots

Two-sample quantile plots can effectively compare two samples at a time, but they do not generalize to more. A matrix of pairwise quantile plots can in principle compare multiple samples, but takes up too much space and can be hard to interpret. A well-known graphical method for comparing multiple samples is the *box-and-whisker plot* (Tukey, 1977). Many variants exist, but essentially each distribution is summarized by five quantiles; three quartiles that define the "box" and two extremes that define the "whiskers". The `bwplot()` function produces box-and-whisker plots with a syntax similar to `qq()`. We illustrate its use by continuing with the `Chem97` example. Figure 3.11 is produced by

```
> bwplot(factor(score) ~ gcsescore | gender, data = Chem97,
         xlab = "Average GCSE Score")
```

Unlike the `qq()` call, the variable defining the samples, `factor(score)` in this case, has more than two levels. For every level, a box-and-whisker plot of the corresponding `gcsescore` values is drawn, allowing us to directly compare the median, indicated by a filled black dot, and the 25*th* and 75*th* quantiles, which determine the range of the box. In some variants, the whiskers extend to the minimum and maximum of the data, but conventionally they are limited to a multiple of the length of the box. This multiple is related to the normal distribution, and points beyond the whiskers, which are plotted explicitly, are thought of as potential outliers; a large number of these indicate tails that are heavier than the normal distribution. These details are controlled by the `coef` and `do.out` arguments of `panel.bwplot()`. In Figure 3.11, the asymmetry in the distribution of `gcsescore` is immediately apparent by looking at the whiskers and the putative outliers, although it is not as clear from the boxes alone.

The next example illustrates the importance of good choices of layout and conditioning. A slightly different version of the previous plot is produced by

```
> bwplot(gcsescore^2.34 ~ gender | factor(score), data = Chem97,
         varwidth = TRUE, layout = c(6, 1),
         ylab = "Transformed GCSE score")
```

The result is shown in Figure 3.12. Although it presents essentially the same data subsets as Figure 3.11 (after applying the optimal Box–Cox transformation to the `gcsescore` values), it orders and orients the boxes differently; in particular, it enables all pairwise comparisons by forcing a common `gcsescore` axis, and emphasizes the differences across `gender` by placing them together. The `varwidth` argument is used to make the widths of the boxes related to sample size, although in this case there is little discernible difference as the sample sizes are all of the same order.

3.7.1 Violin plots

In a sense, the preceding plots summarize all the interesting characteristics of the conditional distribution of `gcsescore`. This usually holds whenever the

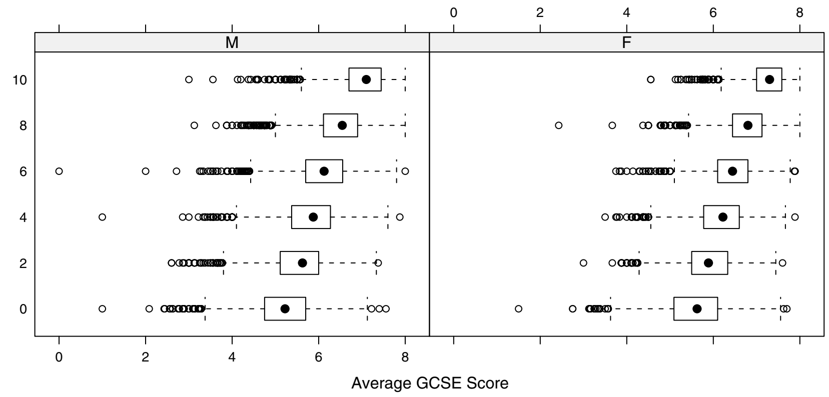

Figure 3.11. Comparative box-and-whisker plots of average GCSE score by final score, conditioned on gender. Systematic skewness and heteroscedasticity, the primary messages of the normal Q–Q plots seen earlier, are readily apparent in this more compact representation as well.

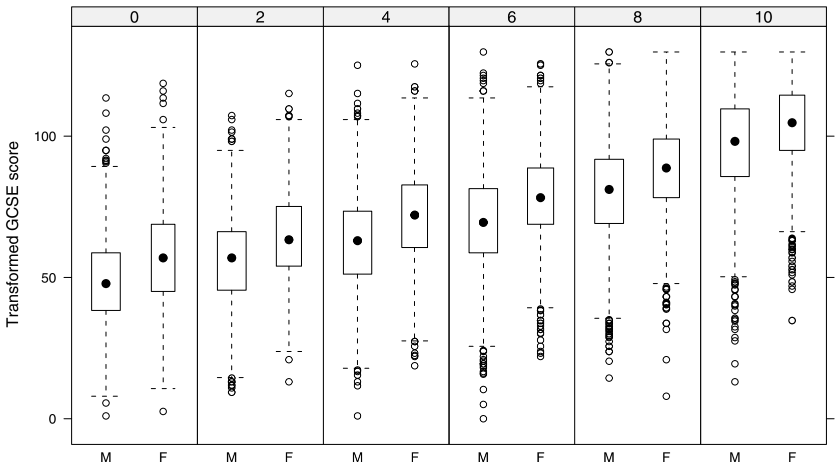

Figure 3.12. Comparative box-and-whisker plots of transformed GCSE scores, representing the same subsets in a slightly different layout. This version highlights a pattern not easily seen in the earlier plots, namely, that boys tend to improve more from bad GCSE scores than girls. This is a good illustration of how layout might affect the information that can be gleaned from a graphic.

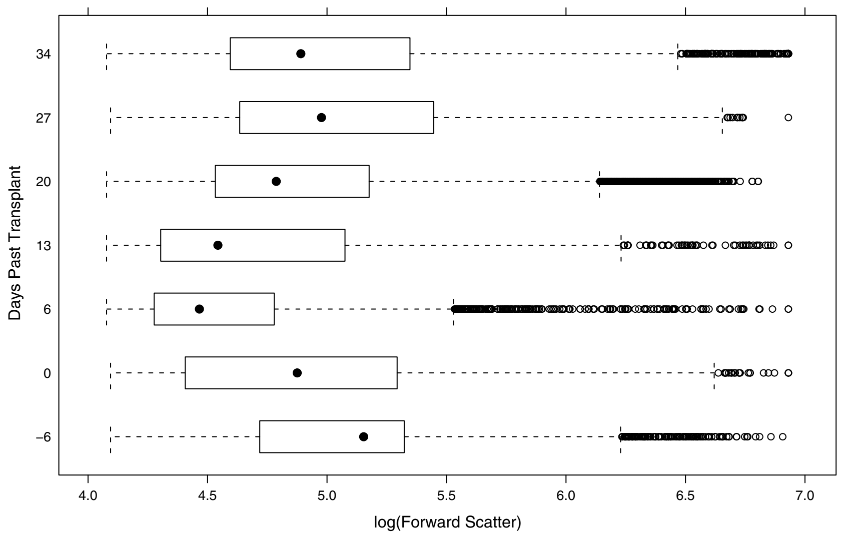

Figure 3.13. Box-and-whisker plots comparing the distribution of log forward scatter values in the `gvhd10` data across time. The multimodality of the distributions, obvious in Figures 3.3 and 3.4, cannot be detected in this encoding.

distribution of interest is unimodal and close to normal. However, box-and-whisker plots can be misleading otherwise. In Figure 3.13, we consider the `gvhd10` data again, this time using a box-and-whisker plot to summarize the distribution of `log(FSC.H)` across `Days`.

```
> bwplot(Days ~ log(FSC.H), data = gvhd10,
         xlab = "log(Forward Scatter)", ylab = "Days Past Transplant")
```

A comparison with Figures 3.3 and 3.4 clearly shows the limitation of this display. A useful alternative that retains the compact structure of a box-and-whisker plot as well as the details of a density plot is the so-called *violin plot* (Hintze and Nelson, 1998). Figure 3.14 is produced by

```
> bwplot(Days ~ log(FSC.H), gvhd10,
         panel = panel.violin, box.ratio = 3,
         xlab = "log(Forward Scatter)",
         ylab = "Days Past Transplant")
```

This uses the predefined panel function `panel.violin()` which can be used as a drop-in replacement for `panel.bwplot()`.

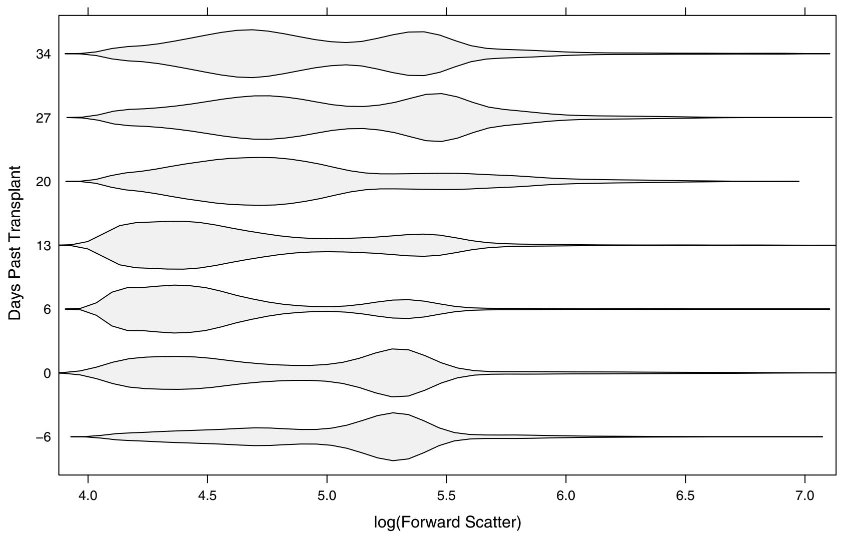

Figure 3.14. A modified version of Figure 3.13, with box-and-whisker plots replaced by violin plots. The bimodal nature of the distributions is readily apparent.

3.8 Strip plots

Box-and-whisker plots summarize the data using a few quantiles, and possibly some outliers. This summarizing can be important when the number of observations is large. When the number of observations per sample is small, it is often sufficient to simply plot the sample values side by side in a common scale. Such plots are known as *strip plots*, also referred to as univariate scatter plots. They are in fact very similar to the bivariate scatter plots we encounter in Chapter 5, except that one of the variables is treated as a categorical variable.

Here, we show a couple of examples using the `quakes` dataset, which records the location (latitude, longitude, and depth) and magnitude of several seismic events near Fiji since 1964. To get a sense of the relationship between magnitude and depth, we might compare the depth values for different magnitudes. Only a few discrete values of magnitude are recorded, and it can be treated as a factor. The call

```
> stripplot(factor(mag) ~ depth, quakes)
```

produces Figure 3.15. There is no particular reason to put the categorical variable on the vertical axis; in fact, the reverse would be the better choice here if we are to have a short wide plot. Figure 3.16 is produced by

```
> stripplot(depth ~ factor(mag), quakes,
            jitter.data = TRUE, alpha = 0.6,
            xlab = "Magnitude (Richter)", ylab = "Depth (km)")
```

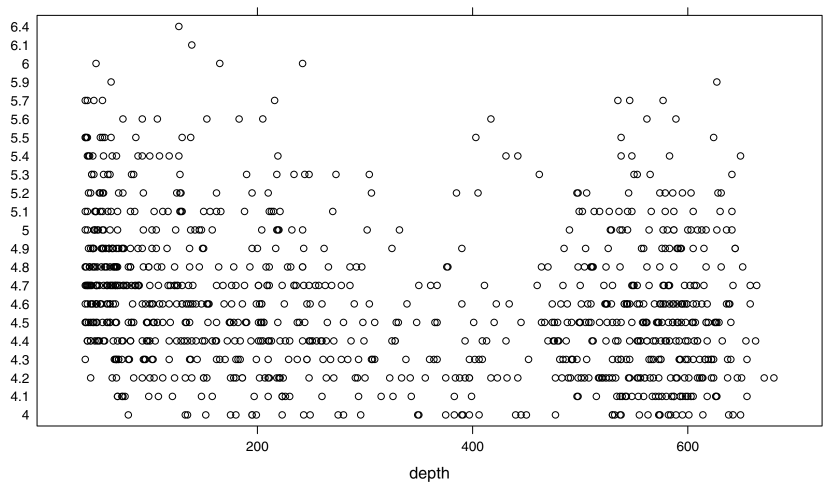

Figure 3.15. Strip plot of depths of the epicenters of seismic events near Fiji, as recorded in the `quakes` dataset. Depths are plotted (on the x-axis) by magnitude of the events on the Richter scale.

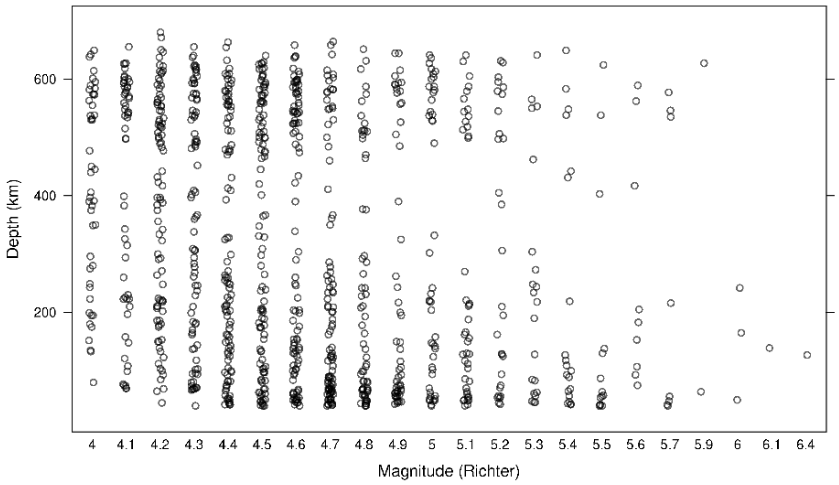

Figure 3.16. Strip plot of epicenter depths by earthquake magnitude, with axes switched. Overplotting is alleviated by jittering the points as well as making them partially transparent.

where additionally we make use of the `alpha` argument to make points semi-transparent[1] and the `jitter.data` argument to randomly displace the points horizontally, both of which help alleviate the effect of overlap.[2] Both plots suggest a weak relationship between depth and magnitude, but the primary visual effect is the clustering of the depth values into two groups, with a gap around 400 km. It is natural to wonder whether this is merely a consequence of some form of spatial clustering of the locations. We follow up on this question in subsequent chapters.

Strip plots can also be used to study residuals from factorial model fits. Figure 3.17, which plots the square roots of the absolute residuals from an additive model fit to the `barley` data, is a variant of the *spread–location* plot (Cleveland, 1993), designed to detect unusual patterns in the variability of residuals.

```
> stripplot(sqrt(abs(residuals(lm(yield~variety+year+site)))) ~ site,
            data = barley, groups = year, jitter.data = TRUE,
            auto.key = list(points = TRUE, lines = TRUE, columns = 2),
            type = c("p", "a"), fun = median,
            ylab = expression(abs("Residual Barley Yield")^{1 / 2}))
```

The call used to produce this plot is somewhat involved, and uses some facts we have not yet encountered. Rather than going into the details now, we wait until we have learned more and analyze the call in Chapter 10.

3.9 Coercion rules

Both the `stripplot()` and `bwplot()` functions expect one of the axes to represent a categorical variable. As with conditioning variables, this can be either a factor or a shingle, and the same coercion rules apply when necessary; that is, a character vector is interpreted as a factor, and a numeric vector as a shingle. The choice of which variable to use as the categorical one is simple when exactly one of the `x` and `y` variables is numeric and the other is a factor or shingle. When the choice is ambiguous, the default is to choose the `y` variable. In all cases, the automatic choice can be overridden by specifying a `horizontal` argument in the high-level call: `TRUE` to have `y` as the categorical variable, `FALSE` to have `x` instead. This choice primarily affects the display produced by the panel function, but also has a subtle effect on axis annotation; by default, for the categorical variable, the axis label is omitted, tick marks are suppressed and the labels do not alternate. These rules also apply to the `dotplot()` and `barchart()` functions discussed in the next chapter.

[1] Note that semi-transparency is not supported on all devices.

[2] Both `alpha` and `jitter.data` are actually passed on to the panel function `panel.stripplot()`.

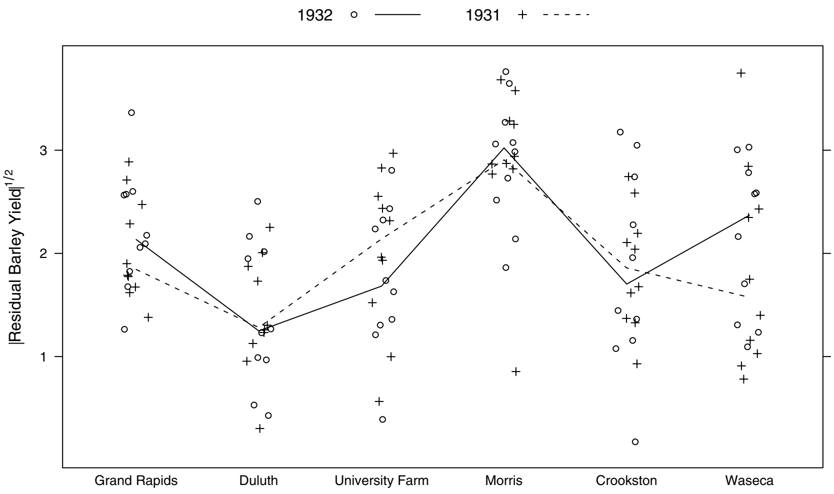

Figure 3.17. A spread–location plot of residuals from a main effects model fit to the `barley` data. The points denote square roots of absolute residuals, jittered horizontally. The lines join the medians of the points within each subgroup, with systematic change in the location indicating a corresponding change in the spread (variance) of the original residuals. It is clear that the model is not entirely appropriate, although the cause is not obvious.

3.10 Discrete distributions

The graphical techniques described in this chapter are designed for continuous random variables. Discrete distributions do not have a density in the conventional sense, but are defined by the analogous probability mass function (p.m.f.). The cumulative distribution function F is still well-defined, as is the quantile function F^{-1} up to certain mathematical caveats. In terms of visualization, this means that density plots and histograms are not really meaningful for discrete data, although Q–Q plots are. The non-parametric maximum likelihood estimator of a p.m.f. is the (relative) frequency table. As with other tables, these can be visualized using bar charts and dot plots (Chapter 4), which serve as a substitute for density plots and histograms; in fact, bar charts are often loosely referred to as histograms, although we prefer not to confuse the two. As with its continuous analogues, bar charts are not very effective tools for judging the goodness of fit of a reference distribution. An innovative visualization for that purpose is Tukey's hanging rootogram (Cleveland, 1988); we do not encounter them in this book, but Trellis rootograms can be created by the `rootogram()` function in the latticeExtra package.

The distinction between continuous and discrete distributions is sometimes unclear. The `gcsescore` variable in the `Chem97` serves as a good case in point;

the 31,022 values, ranging from 0 to 8 with a resolution of three digits after the decimal point, have only 244 unique values. Figure 4.8 in the next chapter, which treats the variable as discrete, reveals an apparent rounding artifact; this is also noticeable in some of the displays we have already seen, but only if we know what we are looking for. Note in particular the ECDF plot in Figure 3.8, which is simply a cumulative version of the bar chart in Figure 4.8, up to differences in conditioning. This emphasizes the important point that two graphical encodings with the same "information" can put visual emphasis on entirely different aspects of that information.

Some distributions are neither continuous nor discrete, but a mixture of the two. These are mostly irrelevant in practical data analysis, with the important exception of *censoring*. Formally, censoring refers to the situation where only the range (most commonly an upper or lower bound) of an observation is known and not its exact value. Often, the fact that an observation is censored is also known, and this can be taken into account during analysis. In other cases, however, censored values may be silently encoded as the bound (which may be the limit of a measuring instrument, for instance), leading to a discrete point mass in an otherwise continuous distribution. Such situations are often hard to identify graphically with density plots or bar charts; the best bet is the Q–Q plot as F and F^{-1} are still well-defined even though the density and p.m.f. are not. Censoring effects can be seen in Figure 10.4.

3.11 A note on the formula interface

We end this chapter with a remark on the formula interface. Although the lattice interface is similar in many ways to the one used in statistical modeling functions in S and R, the interpretation of terms in the formula differs substantially; in fact, the interpretation is not even consistent across lattice functions. A generally helpful rule is the following: given a formula such as `y ~ x`, the `y` variable will be plotted on the y-axis and the `x` variable on the x-axis. The exception to this rule is `qq()`. Similarly, most functions with a formula of the form `~ x` plot `x` on the x-axis, with the exception of `qqmath()` and the yet to be seen `splom()` and `parallel()`. Formulae in the trivariate functions described in Chapter 6 have the form `z ~ x * y`, where again a similar association holds with certain caveats. The upshot is that there is no single rule that governs all uses; the formula interface should be simply viewed as a convenient language that defines the structure of a lattice graphic, and is to be interpreted only in the context of that particular graphic.