4

Displaying Multiway Tables

An important subset of statistical data comes in the form of tables. Tables usually record the frequency or proportion of observations that fall into a particular category or combination of categories. They could also encode some other summary measure such as a rate (of binary events) or mean (of a continuous variable). In R, tables are usually represented by arrays of one (vectors), two (matrices), or more dimensions. To distinguish them from other vectors and arrays, they often have class *"table"*. The R functions `table()` and `xtabs()` can be used to create tables from raw data.

Graphs of tables do not always convey information more easily than the tables themselves, but they often do. The `barchart()` and `dotplot()` functions in lattice are designed to display tabulated data. As with other high-level functions, the primary formula interface requires the data to be available as a data frame. The `as.data.frame.table()` function can be used for converting tables to suitable data frames. In addition, there are methods in lattice that work directly on tables. We focus on the latter in this chapter; examples using the formula interface can be found in Chapter 2.

4.1 Cleveland dot plot

Dot plots (Cleveland, 1985) provide simple and effective graphical summaries of tables that are perhaps less often used than they should be. For illustration, we use the `VADeaths` data, which is a cross-classification of death rates in the U.S. state of Virginia in 1940 by age and population groups (Molyneaux et al., 1947).

```
> VADeaths
      Rural Male Rural Female Urban Male Urban Female
50-54       11.7          8.7       15.4          8.4
55-59       18.1         11.7       24.3         13.6
60-64       26.9         20.3       37.0         19.3
```

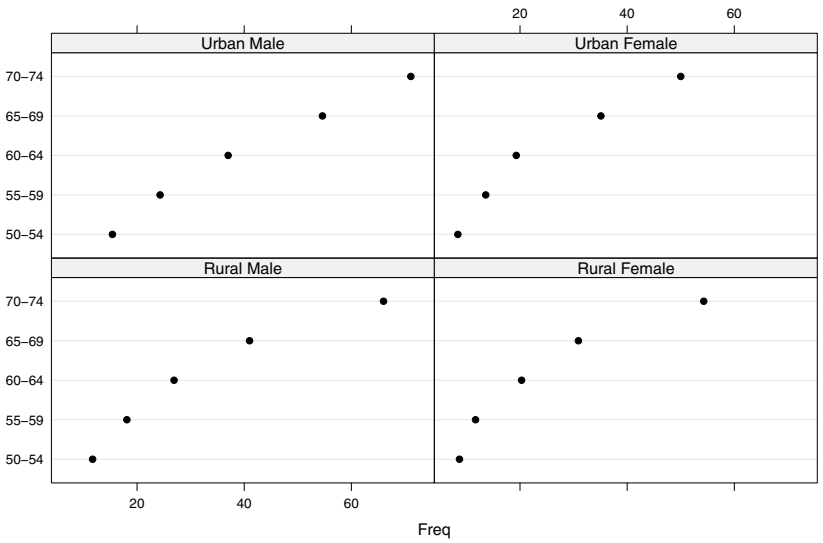

Figure 4.1. Dot plots of death rates (per 1000) in Virginia in 1940, cross-tabulated by age and demographic groups.

```
65-69       41.0          30.9          54.6          35.1
70-74       66.0          54.3          71.1          50.0
```

The `VADeaths` object is of class *"matrix"*.

```
> class(VADeaths)
[1] "matrix"
```

We can check what methods are available for the `dotplot()` function using

```
> methods("dotplot")
[1] dotplot.array*   dotplot.default* dotplot.formula*
[4] dotplot.matrix*  dotplot.numeric* dotplot.table*

   Non-visible functions are asterisked
```

As we can see, there is a method for *"matrix"* objects, which we can use directly in this case. The corresponding help page can be viewed by typing `help(dotplot.matrix)`. Figure 4.1 is produced by

```
> dotplot(VADeaths, groups = FALSE)
```

which uses one additional argument to disable grouping. As is almost inevitable with a first attempt, there is much scope for improvement. The default label on the horizontal axis says `Freq`, even though the table values are not frequencies. More importantly, this display does not easily allow us to compare the rates for males and females, as they are displayed in different

columns. One way to rectify this is to force the display to have one column. To prevent the panels from getting too flattened, we add an explicit aspect ratio. Because rates have a well-defined origin (0), it may also be interesting to make a judgment about their relative magnitude, and not just their differences. To this end, we ask for the points to be joined to a baseline using the `type` argument. Figure 4.2 is produced by

```
> dotplot(VADeaths, groups = FALSE,
          layout = c(1, 4), aspect = 0.7,
          origin = 0, type = c("p", "h"),
          main = "Death Rates in Virginia - 1940",
          xlab = "Rate (per 1000)")
```

Even more direct comparison can be achieved using superposition, which is in fact the default in this `dotplot()` method. By omitting the `groups = FALSE` argument, we can plot rates for all the population groups in a single panel, but with different graphical parameters. The following call produces Figure 4.3.

```
> dotplot(VADeaths, type = "o",
          auto.key = list(lines = TRUE, space = "right"),
          main = "Death Rates in Virginia - 1940",
          xlab = "Rate (per 1000)")
```

4.2 Bar chart

Bar charts (along with pie charts[1]) are among the most popular graphical representations of tables. However, they are less useful than dot plots in most situations. A bar chart analogous to the dot plot in Figure 4.1 is produced by

```
> barchart(VADeaths, groups = FALSE,
           layout = c(1, 4), aspect = 0.7, reference = FALSE,
           main = "Death Rates in Virginia - 1940",
           xlab = "Rate (per 100)")
```

The resulting plot, shown in Figure 4.4, conveys exactly the same information with some additional and redundant graphical structure. In fact, bar charts can actually mislead when the "origin" is arbitrary, as they convey the incorrect impression that the quantity encoded by the length (or area) of the bar has some meaning. Another popular but questionable practice is to add confidence intervals to bar charts; dot plots with confidence intervals are almost invariably easier to interpret.

One variant of the bar chart does encode more information than a dot plot could. A grouping variable can be incorporated in a bar chart display either by plotting the bars for the various groups side by side or by stacking

[1] lattice does not contain a function that produces pie charts. This is entirely by choice, as pie charts are a highly undesirable form of graphical representation (see Cleveland (1985) for a discussion), and their use is strongly discouraged.

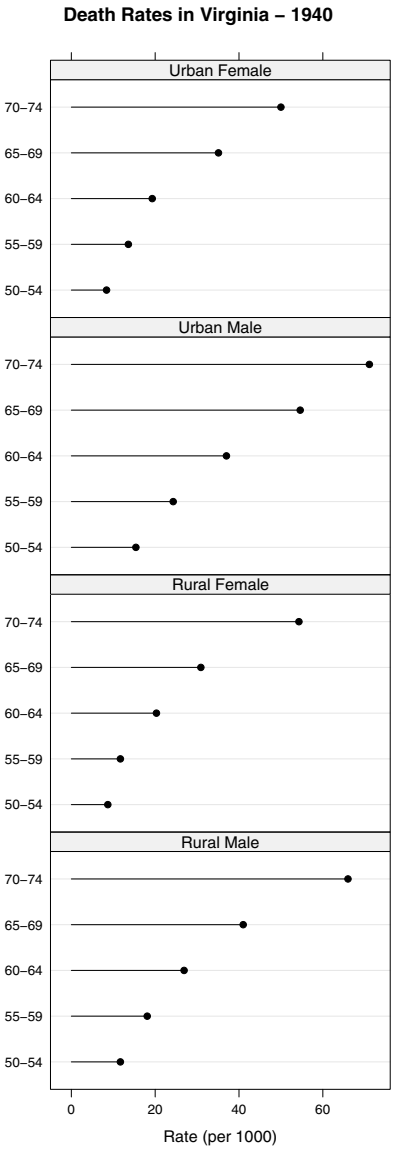

Figure 4.2. Dot plot of death rates in Virginia in 1940, arranged in a single column layout with more informative labels. The origin is included in the plot, and points are joined to it to enable comparison of absolute rates.

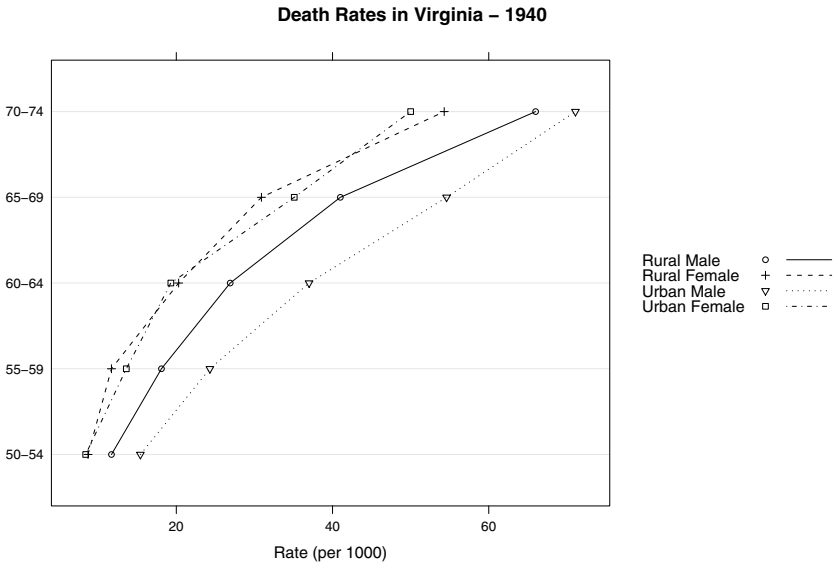

Figure 4.3. Death rates in Virginia in 1940, with population groups superposed within a single panel. Points within a group are joined to emphasize group membership. This plot suggests that the rates are virtually identical in the rural female and urban female subgroups, with a systematic increase among rural males and a further increase for urban males. This pattern is hard to see in the multipanel versions.

them on top of each other. The first case is similar to a grouped dot plot and contains no extra information. In the second case, a stacked bar chart, the total length of each bar encodes the marginal totals, in addition to the lengths of the component bars, which breaks up this total according to the grouping variable. We have seen stacked bar charts previously in Chapter 2 (e.g., Figure 2.9). For another example, consider the data in Table 4.2, based on a survey of doctorate degree recipients in the United States who went on to pursue a postdoctoral position. The data are available in the latticeExtra package.

```
> data(postdoc, package = "latticeExtra")
```

Stacked bar charts are generally produced by adding a `stack = TRUE` argument to `barchart()`, but this is unnecessary for the *"table"* method as it is the default. A stacked bar chart of the `postdoc` data, shown in Figure 4.5, is produced by

```
> barchart(prop.table(postdoc, margin = 1), xlab = "Proportion",
           auto.key = list(adj = 1))
```

The data plotted are proportions, computed by `prop.table()`, as these are the quantities of interest; the counts could have been plotted as well, but that would not have told us much except that the "Biological Sciences" field

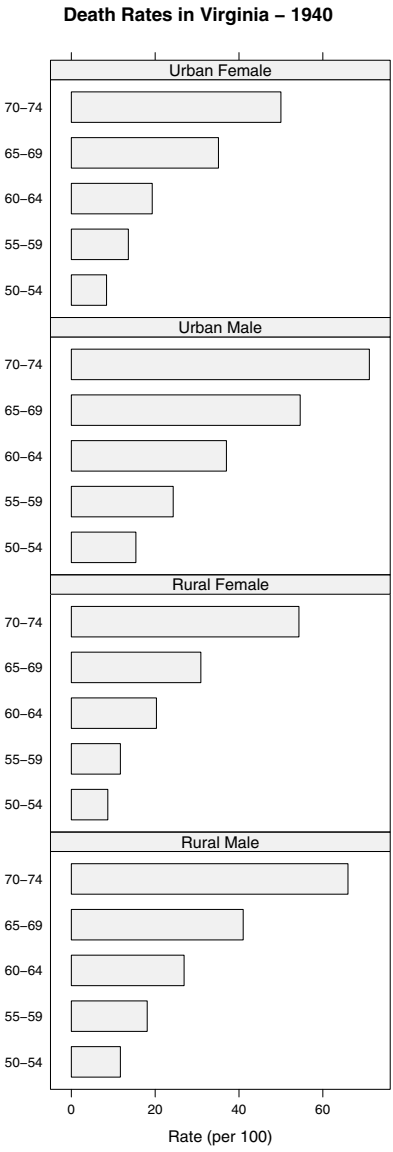

Figure 4.4. Bar charts of Virginia death rates by population group, in a layout similar to the dot plots in Figure 4.1. This encoding contains more graphical structure, but no more information.

	Expected or Additional Training	Work with Specific Person	Training Outside PhD Field	Other Employment Not Available	Other
Biological Sciences	6404	2427	1950	1779	602
Chemistry	865	308	292	551	168
Earth, Atm., & Ocean Sciences	343	75	75	238	80
Engineering	586	464	288	517	401
Medical Sciences	205	137	82	68	74
Physics & Astronomy	1010	347	175	399	162
Social & Behavioral Sciences	1368	564	412	514	305
All Postdoctorates	11197	4687	3403	4406	1914

Table 4.1. Reasons for choosing a postdoctoral position after graduating from U.S. universities, by different fields of study.

contributes the majority of postdocs. We also make the levels of the grouping variable right-justified in the legend using the `auto.key` argument.

A multipanel dot plot encoding the same information can be produced by

```
> dotplot(prop.table(postdoc, margin = 1), groups = FALSE,
          xlab = "Proportion",
          par.strip.text = list(abbreviate = TRUE, minlength = 10))
```

creating Figure 4.6. Even though the stacked bar chart is more concise, it is not necessarily better if one is primarily interested in comparing the proportions of reasons across fields. The bar chart encodes this quantity using length, whereas the dot plot does so using relative position which is more easily judged by the human eye.

4.2.1 Manipulating order

A point worth making in the context of this example is the importance of visual order. In many situations, the levels of a categorical variable have no natural order; this is true for both margins of the `postdoc` table. Often, choosing the order in which these levels are displayed based on the data can significantly increase the impact of the display. An effective order is usually obtained by sorting levels by the value of a corresponding continuous response, or perhaps a summary measure when multiple observations or multiple responses are involved. Facilities available in lattice that aid such reordering are discussed, along with examples, in Chapter 10. Unfortunately, our example is slightly complicated by the fact that the responses are proportions that add up to one for each field of study, making it difficult to find a common order that is appropriate for all panels.

One solution is to use a different order for each panel. This is not particularly difficult to achieve, but involves several concepts we have not yet

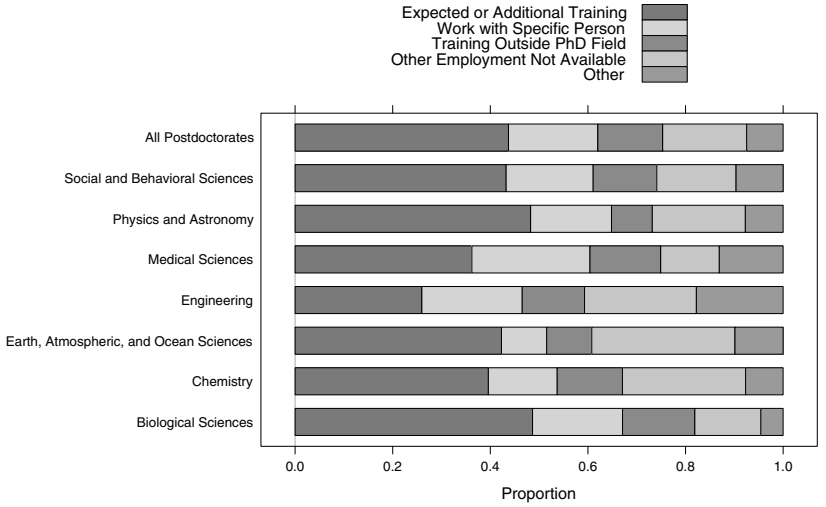

Figure 4.5. A stacked bar chart showing the proportion of reasons for choosing a postdoc by field of study. Because the bars encode proportions, their lengths add up to one within each field. Comparison is done through lengths (except for the first and last group), which is less effective than comparison through position. Notice the long labels on the vertical axis, for which enough space has been allocated automatically.

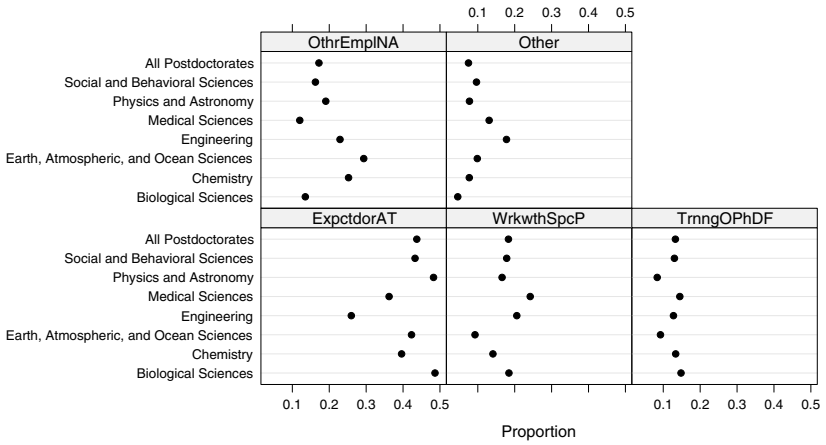

Figure 4.6. Reasons for choosing a postdoc position; an alternative visualization using multipanel dot plots. Although the display is less compact, it makes comparison within fields easier. Long axis labels are not uncommon in situations such as these, therefore the labels are all shown on one side by default to save space. The strip labels have been abbreviated as they would not have fit in the available area.

encountered. We give the solution here for the sake of completeness, and refer the reader to later chapters for details. Figure 4.7 is produced by

```
> dotplot(prop.table(postdoc, margin = 1), groups = FALSE,
          index.cond = function(x, y) median(x),
          xlab = "Proportion", layout = c(1, 5), aspect = 0.6,
          scales = list(y = list(relation = "free", rot = 0)),
          prepanel = function(x, y) {
              list(ylim = levels(reorder(y, x)))
          },
          panel = function(x, y, ...) {
              panel.dotplot(x, reorder(y, x), ...)
          })
```

The critical additions in this call are the use of the `index.cond` argument and the `reorder()` function, both of which are discussed in Chapter 10. The order of the fields is changed inside the panel function, and a corresponding change is required in the `prepanel` function to ensure that the axis labels match. We also need to specify an appropriate `scales` argument to allow panels to have independent axis annotation. Details about these arguments can be found in Chapter 8.

4.2.2 Bar charts and discrete distributions

As mentioned in Chapter 3, bar charts can be viewed as analogues of density plots or histograms for discrete distributions. In the examples we have seen so far, the data come in the form of a table. When only raw data are available, frequency tables can be easily constructed with the `xtabs()` function. Consider the following two-way table of `gcsescore` by `gender` derived from the `Chem97` data.

```
> gcsescore.tab <- xtabs(~gcsescore + gender, Chem97)
```

We might attempt to produce a bar chart directly from this table, but this will not give us the result we want; because `gcsescore` is now interpreted as a categorical variable, its levels will be plotted as equispaced integers[2] and not as the original numeric values. Additionally, `barchart()` will print the labels for every level of `gcsescore`, causing substantial overlap. In this case, it is easier to first manipulate the data, after converting the table into a data frame, and then use the `xyplot()` function, which does not require either of the variables to be categorical. The next chapter discusses `xyplot()` in detail; here we use the convenient `type` argument to create Figure 4.8.

```
> gcsescore.df <- as.data.frame(gcsescore.tab)
> gcsescore.df$gcsescore <-
      as.numeric(as.character(gcsescore.df$gcsescore))
> xyplot(Freq ~ gcsescore | gender, data = gcsescore.df,
         type = "h", layout = c(1, 2), xlab = "Average GCSE Score")
```

[2] To be precise, the numeric codes in the underlying representation of a *"factor"*.

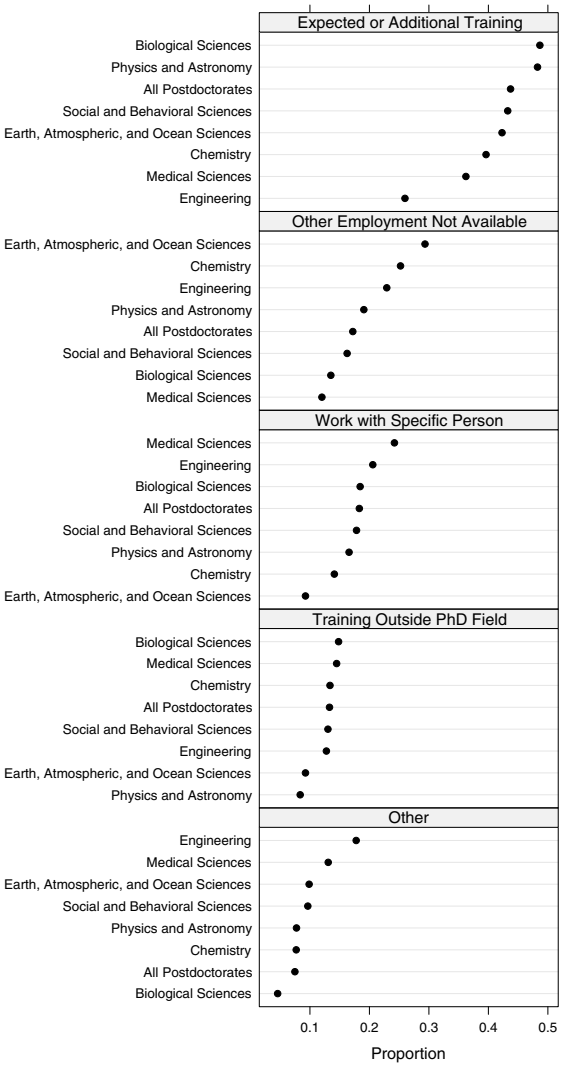

Figure 4.7. Yet another visualization of reasons for choosing a postdoc. Both margins have been ordered by the response (proportions within field): the panels (reasons) are ordered by the median proportion over all fields, and fields are ordered by proportion within each panel. Reordering often makes it easier to see patterns in the data when there is no intrinsic order.

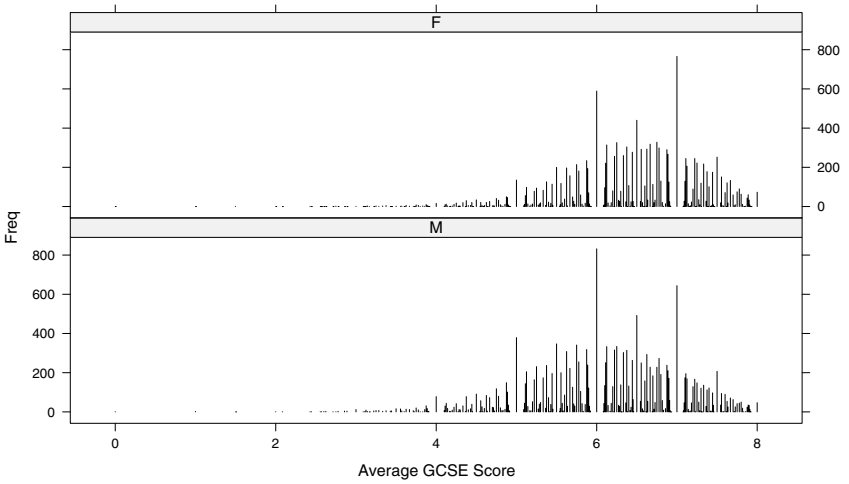

Figure 4.8. A bar chart of sorts, visualizing the frequency table of average GCSE score by gender. The main differences from a conventional bar chart are that the x-axis is continuous and the "bars" are actually zero-width lines. Apart from showing that girls tend to do better than boys on the GCSE, the most interesting feature of the display is the spikes of high frequencies for certain values, most noticeable for whole numbers. At first glance, this might appear to be some sort of rounding error. In fact, the artifact is due to averaging; most of the GCSE score values are the average of 8, 9, or 10 scores, where the total scores are integers.

Note that the use of `barchart()` is perfectly reasonable when the number of levels is small; for example, Figure 4.9 is produced by

```
> score.tab <- xtabs(~score + gender, Chem97)
> score.df <- as.data.frame(score.tab)
> barchart(Freq ~ score | gender, score.df, origin = 0)
```

4.3 Visualizing categorical data

Tables are examples of the more general class of categorical data. Specialized visualization methods for such data exist, but are less well known compared to methods for continuous data. Support for visualizing categorical data in lattice is limited to dot plots and bar charts, and those interested in such data are strongly encouraged to look at the vcd package, which implements many techniques described by Friendly (2000).

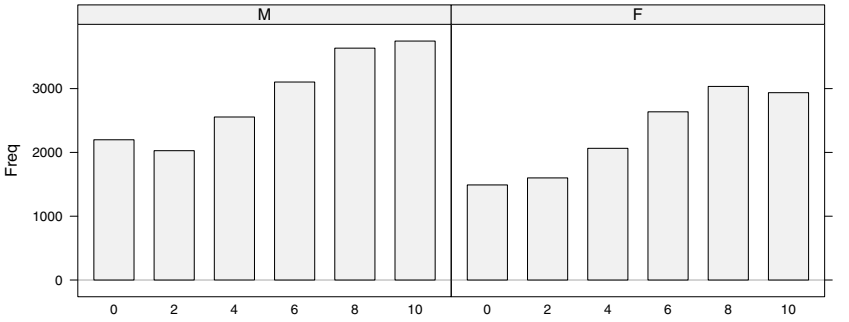

Figure 4.9. Bar chart displaying the frequency distribution of final score in the A-level chemistry examination, by gender.