4.6: Bar Graphs

Another form of graph that you often want to plot is the ***bar graph***. The main function that you can use in R to draw them is the `barplot()` function.[102] And to illustrate the use of the function, I'll use the `finalists` variable that I introduced in Section 5.1.7. What I want to do is draw a bar graph that displays the number of finals that each team has played in over the time spanned by the `afl` data set. So, let's start by creating a vector that contains this information. I'll use the `tabulate()` function to do this (which will be discussed properly in Section **??**, since it creates a simple numeric vector:

```
> freq <- tabulate( afl.finalists )
> print( freq )
 [1] 26 25 26 28 32  0  6 39 27 28 28 17  6 24 26 39 24
```

This isn't exactly the prettiest of frequency tables, of course. I'm only doing it this way so that you can see the `barplot()` function in it's "purest" form: when the input is just an ordinary numeric vector. That being said, I'm obviously going to need the team names to create some labels, so let's create a variable with those. I'll do this using the `levels()` function, which outputs the names of all the levels of a factor (see Section 4.7:

```
> teams <- levels( afl.finalists )
> print( teams )
 [1] "Adelaide"         "Brisbane"         "Carlton"          "Collingwood"
 [5] "Essendon"         "Fitzroy"          "Fremantle"        "Geelong"
 [9] "Hawthorn"         "Melbourne"        "North Melbourne"  "Port Adelaide"
[13] "Richmond"         "St Kilda"         "Sydney"           "West Coast"
[17] "Western Bulldogs"
```

Okay, so now that we have the information we need, let's draw our bar graph. The main argument that you need to specify for a bar graph is the `height` of the bars, which in our case correspond to the values stored in the `freq` variable:

```
> barplot( height = freq )  # specifying the argument name (panel a)
> barplot( freq )   # the lazier version (panel a)
```

Either of these two commands will produce the simple bar graph shown in Figure 6.21.

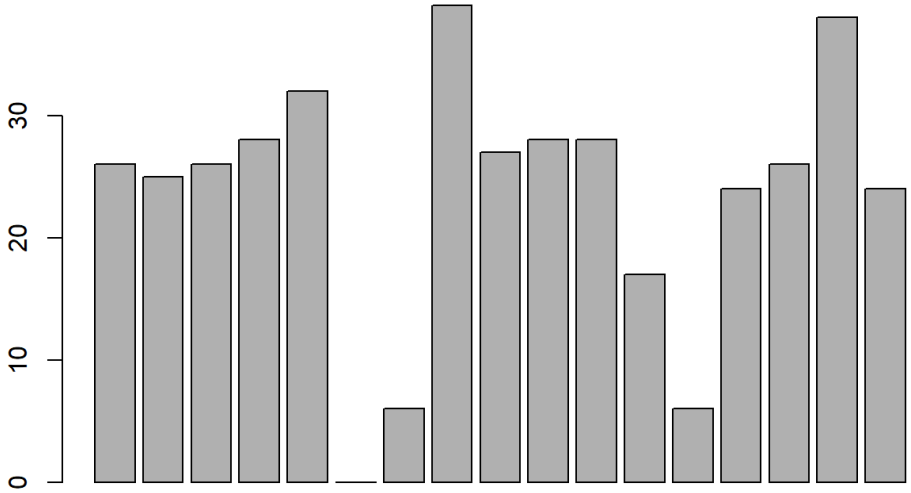

Figure 6.21: the simplest version of a bargraph, containing the data but no labels

As you can see, R has drawn a pretty minimal plot. It doesn't have any labels, obviously, because we didn't actually tell the `barplot()` function what the labels are! To do this, we need to specify the `names.arg` argument. The `names.arg` argument needs to be a vector of character strings containing the text that needs to be used as the label for each of the items. In this case, the `teams` vector is exactly what we need, so the command we're looking for is:

```
barplot( height = freq, names.arg = teams )
```

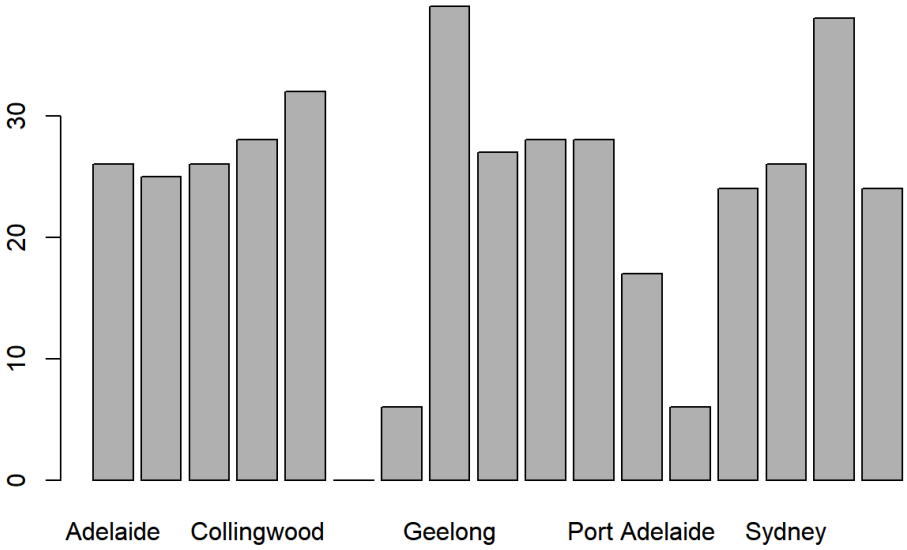

Figure 6.22: we've added the labels, but because the text runs horizontally R only includes a few of them

This is an improvement, but not much of an improvement. R has only included a few of the labels, because it can't fit them in the plot. This is the same behaviour we saw earlier with the multiple-boxplot graph in Figure 6.16. However, in Figure 6.16 it wasn't an issue: it's pretty obvious from inspection that the two unlabelled plots in between 1987 and 1990 must correspond to the data from 1988 and 1989. However, the fact that `barplot()` has omitted the names of every team in between Adelaide and Fitzroy is a lot more problematic.

The simplest way to fix this is to rotate the labels, so that the text runs vertically not horizontally. To do this, we need to alter set the `las` parameter, which I discussed briefly in Section **??**. What I'll do is tell R to rotate the text so that it's always perpendicular to the axes (i.e., I'll set `las = 2`). When I do that, as per the following command…

```
barplot(height = freq,  # the frequencies
        names.arg = teams,  # the label
        las = 2)            # rotate the labels
```

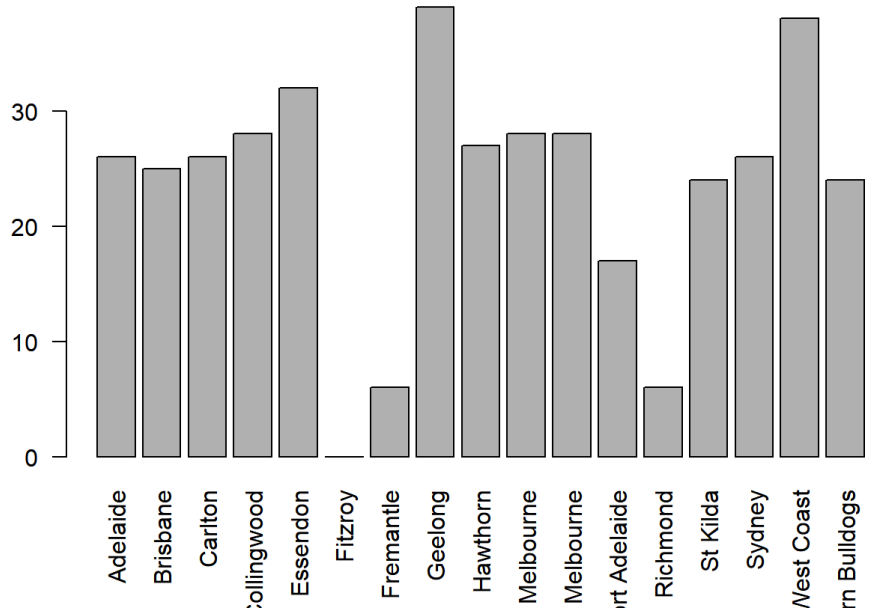

Figure 6.23: we've rotated the labels, but now the text is too long to fit

… the result is the bar graph shown in Figure 6.23. We've fixed the problem, but we've created a new one: the axis labels don't quite fit anymore. To fix this, we have to be a bit cleverer again. A simple fix would be to use shorter names rather than the full name of all teams, and in many situations that's probably the right thing to do. However, at other times you really do need to create a bit more space to add your labels, so I'll show you how to do that.

4.6.1 Changing global settings using `par()`

Altering the margins to the plot is actually a somewhat more complicated exercise than you might think. In principle it's a very simple thing to do: the size of the margins is governed by a graphical parameter called `mar`, so all we need to do is alter this parameter. First, let's look at what the `mar` argument specifies. The `mar` argument is a vector containing four numbers: specifying the amount of space at the bottom, the left, the top and then the right. The units are "number of `lines'". The default value for` mar `is` c(5.1, 4.1, 4.1, 2.1)`, meaning that R leaves 5.1"lines" empty at the bottom, 4.1 lines on the left and the bottom, and only 2.1 lines on the right. In order to make more room at the bottom, what I need to do is change the first of these numbers. A value of 10.1 should do the trick.

So far this doesn't seem any different to the other graphical parameters that we've talked about. However, because of the way that the traditional graphics system in R works, you need to specify what the margins will be *before* calling your high-level plotting function. Unlike the other cases we've see, you can't treat `mar` as if it were just another argument in your plotting function. Instead, you have to use the `par()` function to change the graphical parameters beforehand, and only then try to draw your figure. In other words, the first thing I would do is this:

```
> par( mar = c( 10.1, 4.1, 4.1, 2.1) )
```

There's no visible output here, but behind the scenes R has changed the graphical parameters associated with the current device (remember, in R terminology all graphics are drawn onto a "device"). Now that this is done, we could use the exact same command as before, but this time you'd see that the labels all fit, because R now leaves twice as much room for the labels at the bottom. However, since I've now figured out how to get the labels to display properly, I might as well play around with some of the other options, all of which are things you've seen before:

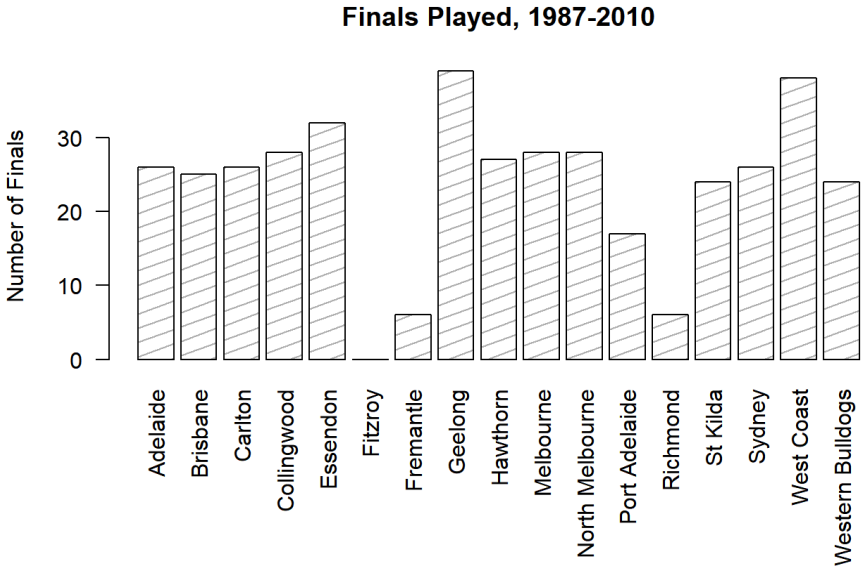

Figure 6.24: we fix this by expanding the margin at the bottom, and add several other customisations to make the chart a bit nicer

```
barplot( height = freq,
        names.arg = teams,
        las=2,
        ylab = "Number of Finals",
        main = "Finals Played, 1987-2010",
        density = 10,
        angle = 20)
```

However, one thing to remember about the `par()` function is that it doesn't just change the graphical parameters for the current *plot*. Rather, the changes pertain to any subsequent plot that you draw onto the same *device*. This might be exactly what you want, in which case there's no problem. But if not, you need to reset the graphical parameters to their original settings. To do this, you can either close the device (e.g., close the window, or click the "Clear All" button in the Plots panel in Rstudio) or you can reset the graphical parameters to their original values, using a command like this:

```
> par( mar = c(5.1, 4.1, 4.1, 2.1) )
```
