Plotting: An Iterative Process

Plotting is an iterative process. First we find a way to represent the data that focusses on the important aspects of the data. What is considered an important aspect may have to do with the purpose of the plot. Next we add information that helps the reader easily interpret the plot. Both of these steps are an iteratvie process.

We consider an example from the spam case study where the goal is to choose a threshhold for the Naive Bayes method of spam filtering. We have computed the Type I (when a ham message is classified as spam) and Type II (when a spam message is classified as ham) error rates for various values of the log odds-ratio, and we need to choose the threshhold for the log odds-ratio, where electronic mail messages with a log odds-ratio above the threshhold will be classified as spam, and below will be classified as ham.

As we need to compare the Type I and II errors, a plot that includes both of these would be more useful than producing separate plots for the two types of errors. The **matplot** function does just that.

```
matplot(logOdds, cbind(TypeI, TypeII))
```

Notice that the region of greatest interest, where the Type I and II errors are both small is too compressed. We need to zoom in on the region around (0, 0). To do this we can specify the limits of the x and y axes. We iterate a few times, until we settle on a region that gives us the needed resolution. At the same time, we start to improve the visual presentation and specify that the points should be connected by lines.

```
matplot(logOdds, cbind(TypeI, TypeII), ylim = c(0, 0.2),
        xlim = c(-100, 100), type="l")
matplot(logOdds, cbind(TypeI, TypeII), ylim = c(0, 0.1),
        xlim = c(-60, 30), type="l")
```

Next we want to highlight controlling the Type I error. To do this, we place markers, horizontal and vertical lines, that help us easily read the error rates and log odds. For example, if we wanted to bound the Type I error rate at 0.5% and ascertain what the threshhold and Type II error rate would be for a Type I error of 0.5% then we could place on the plot a marker at 0.005, a vertical marker where this horizontal line crosses the Type I curve, and finally a second horizontal marker where the vertical line that indicates the size of the Type II error rate for this threshhold, i.e. the threshhold that gives us a Type I error rate at most 0.5%. The **abline** helps here. We may want to wrap the commands up into a function to make it easier for us to try a few different values for the threshhold.

```
markers =
 function(lo = logOdds, t1 = TypeI, t2 = TypeII, p = 0.005)
{
    v1 = min(lo[t1 < p])
    h2 = max(t2[lo <= v1])
    abline(v = v1)
    segments(min(lo), p, v1, p)
    segments(min(lo), h2, v1, h2)
}
```

We have completed the first step, having determined how to present the basic information. Now we turn to the second step of adding context to the plot to make it easier for for the reader to read the plot. We add a title, labels on the axes, labels on the markers, color to distinguish the lines, and a legend for the colors.

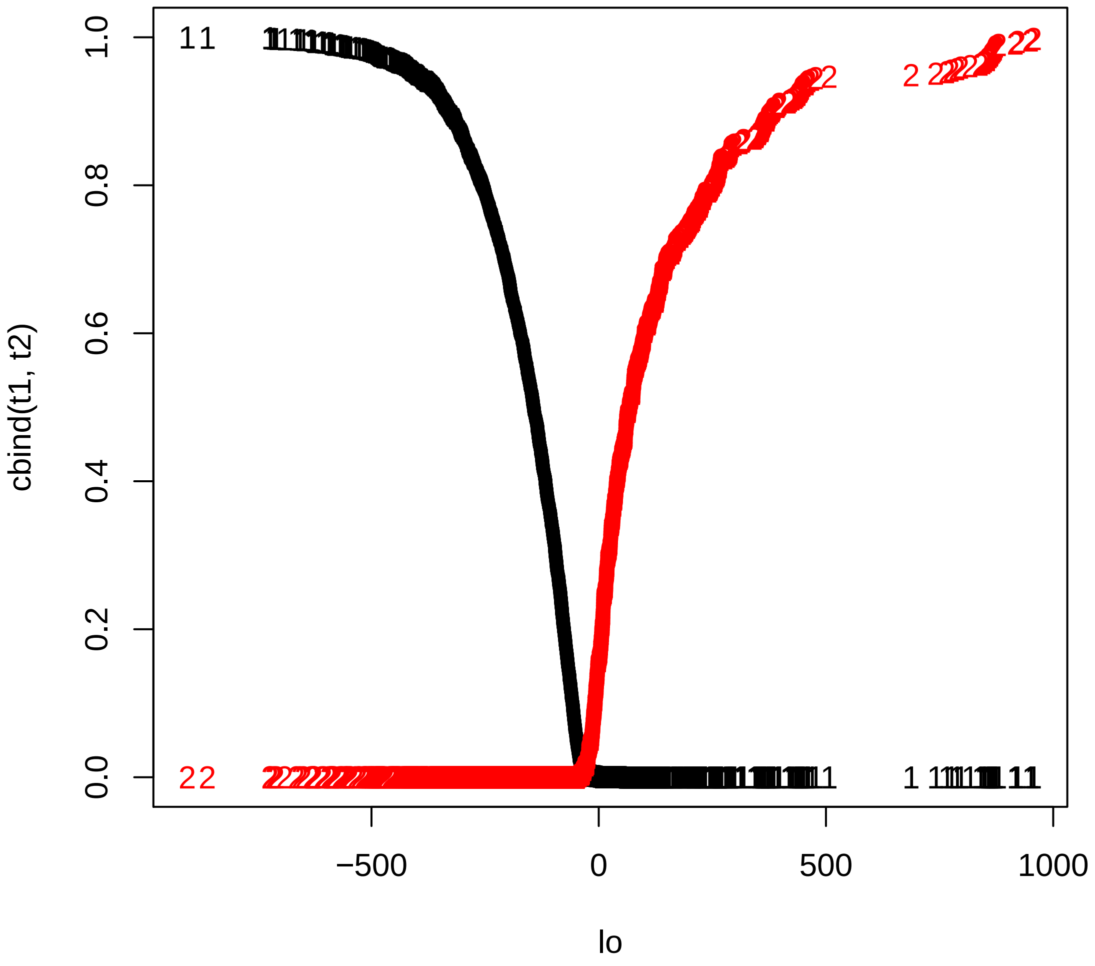
```
addContext =
function(lo = logOdds, t1 = TypeI, t2 = TypeII, p = 0.005, xlim = c(-60, 30),
     ylim = c(0., 0.05), markers = TRUE) {
```

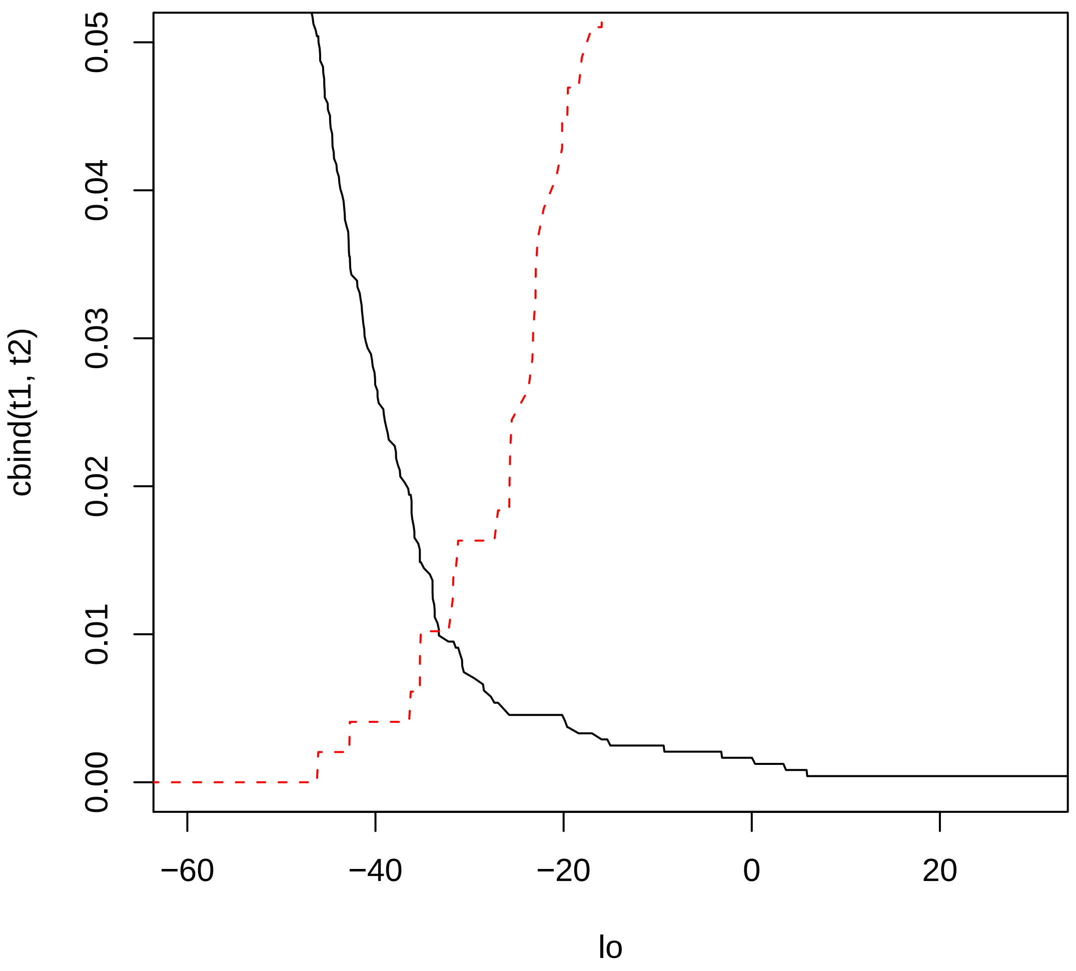
```
    matplot(lo, cbind(t1,t2), col = c("red","blue"), ylim = ylim,
        xlim = xlim, type = "l", lty = 1, xlab = "Threshhold",
ylab="Proportion of Errors", lwd = 4, cex = 1)
    if (markers) {
      title(main = "Cross-validation Estimates of Type I and Type II Error")
      v1 = min(lo[t1 < p])
      h2 = max(t2[lo <= v1])
      abline(v= v1, lty=2, lwd=2, col="green")
      segments(xlim[1], p, v1, p, lty=2, lwd=2, col="green")
      segments(xlim[1], h2, v1, h2, lty=2, lwd=2, col="green")
      text(5 + c(xlim[1], xlim[1]), c(p, h2), as.character(round(c(p, h2),3)),
           pos = 3, offset = 0.5, cex=1, col = c("red", "blue"))
      legend(10, 0.04, legend = c("Type I", "Type II"),
             fill=c("red","blue"),yjust=0,cex=1)
     }
}
```

As a final addition to put the problem in context, we add a second plot as an inlay to the first in order to see that the full range of values for Type I and II errors. The **layout** function helps us here.

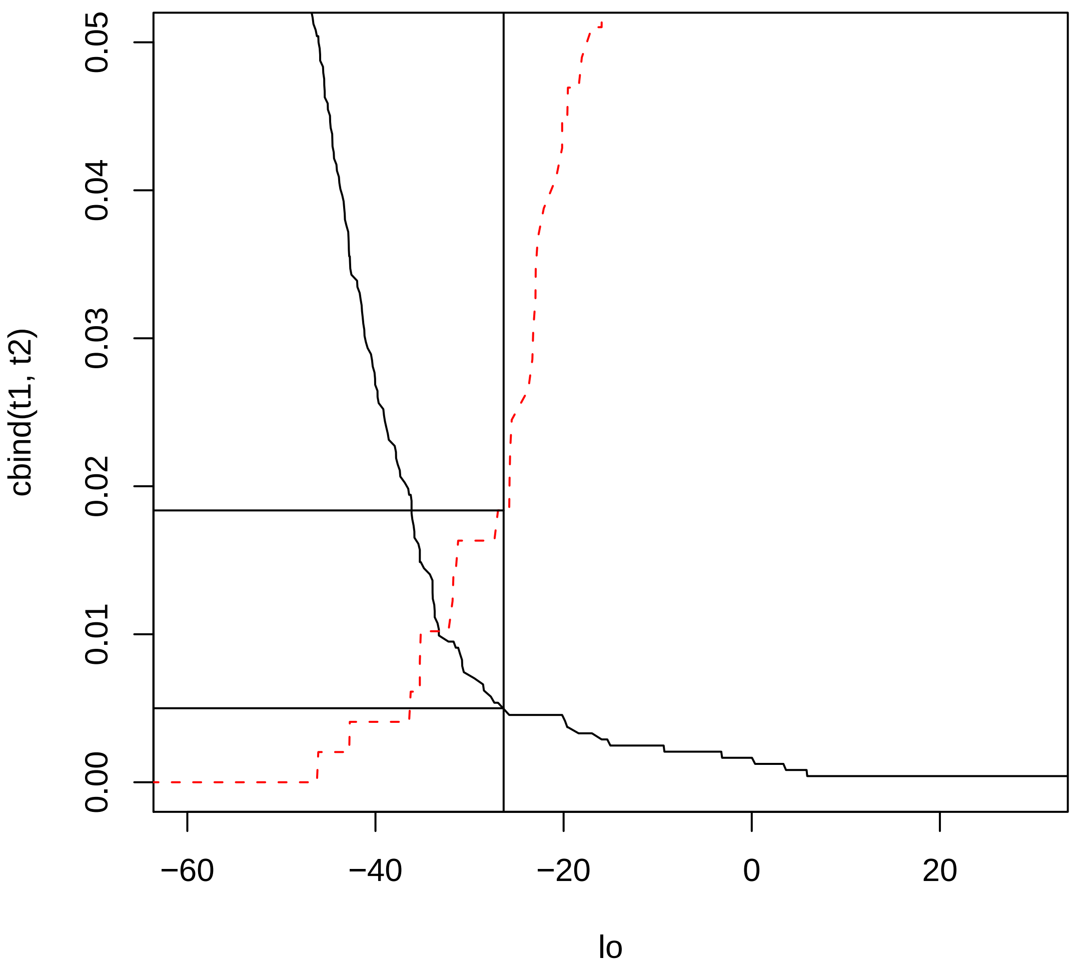
```
inLay =
function(widths = c(3, 3, 3, 3, 1), heights = c(1, 3, 3, 1, 1) )
{
  layout(matrix(c(1, 0, 0, 0, 1,
                  0, 0, 2, 2, 0,
  0, 0, 2, 2, 0,
  0, 0, 0, 0, 0,
  1, 0, 0, 0, 1,), 5, 5, byrow = TRUE),
widths = widths,
heights = heights)
```

```
  addContext()
  addContext(xlim = c(-100,100), ylim = c(0,0.8), markers = FALSE)
}
```

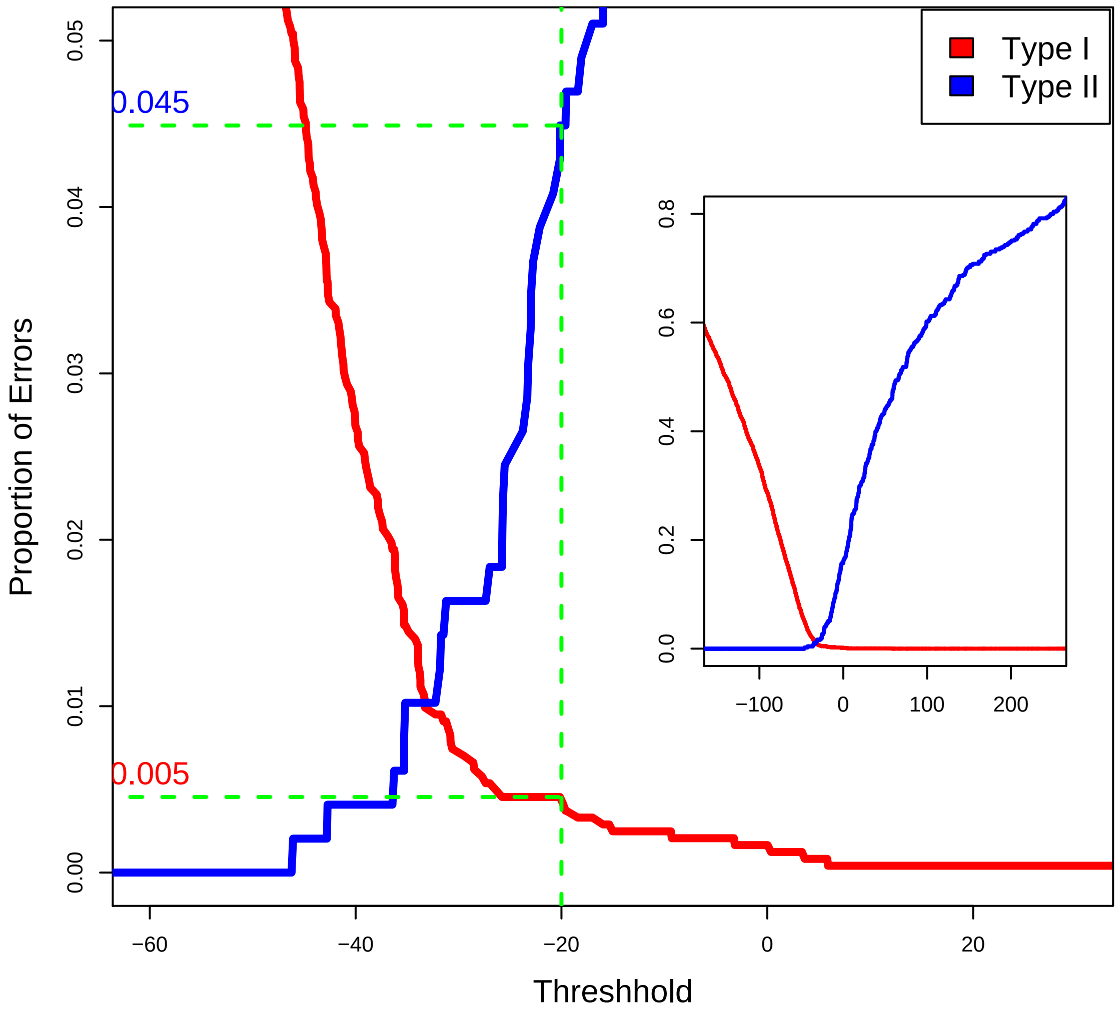

cbind(t1, t2)
lo
1.0
0.8
0.6
0.4
0.2
0.0
−500
0
500
1000



cbind(t1, t2)
0.00
0.01
0.02
0.03
0.04
0.05
−60
−40
−20
0
20
lo



cbind(t1, t2)
0.00
0.01
0.02
0.03
0.04
0.05
−60
−40
−20
0
20
lo


Proportion of Errors
Threshhold
0.045
0.005
Type I
Type II
0.00
0.01
0.02
0.03
0.04
0.05
−60
−40
−20
0
20
0.0
0.2
0.4
0.6
0.8
−100
0
100
200