quiz.density.estimation

Stat 5301

We've discussed using statistics like $\bar{X}$ to estimate parameters like μ. We also use statistics to estimate more complicated things, like functions. For example, we might want to estimate a probability density function (pdf).

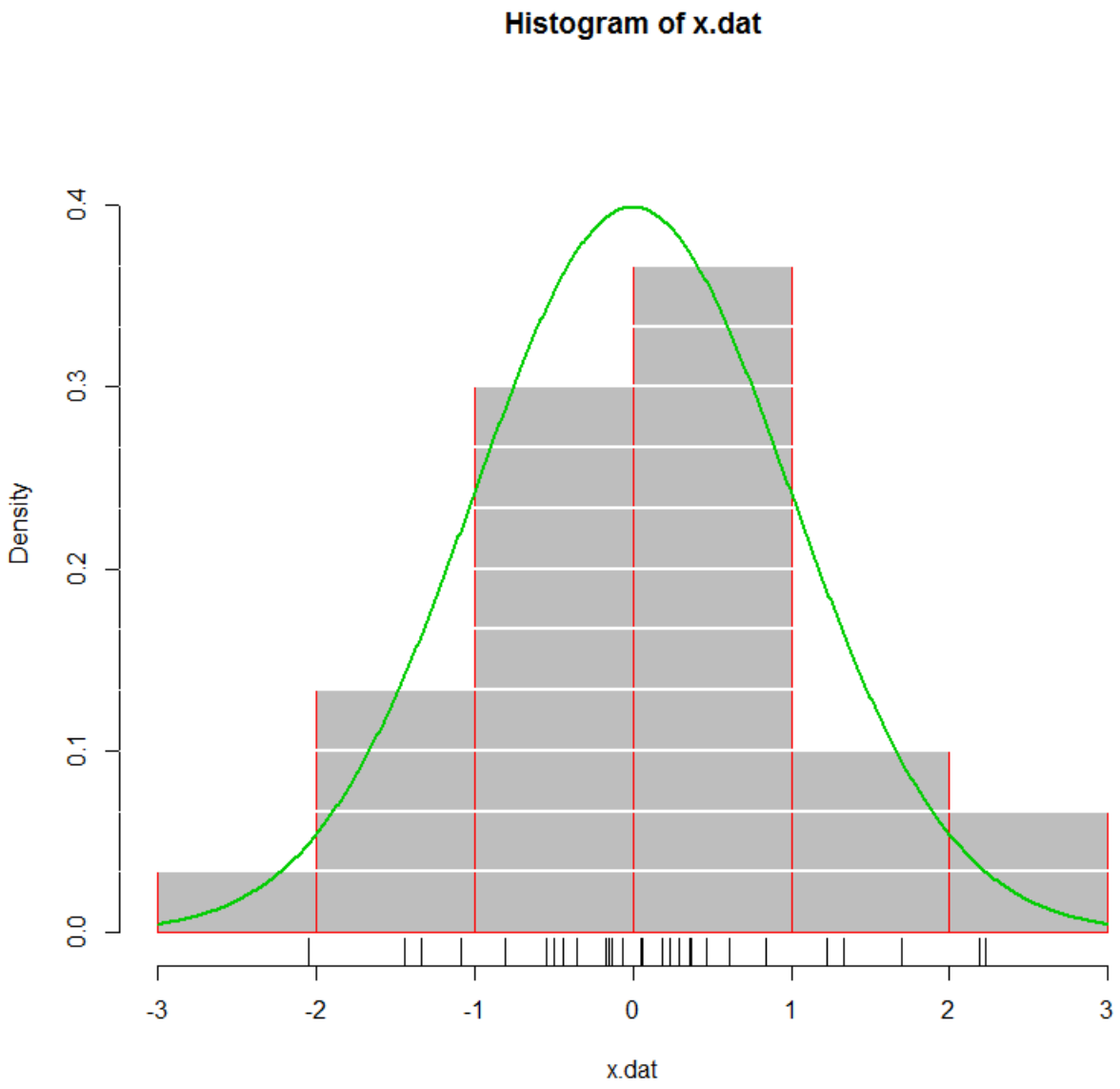
Consider a realization $x_1, \ldots, x_n$ of an iid sample $X_1, \ldots, X_n$. A histogram based on the x_i's (scaled by density, not frequency) is an estimate of the probability density function which governs the X_i's. Code below is for the case where the X_i's are $N(0, 1)$ and $n = 30$.

```
# sample n=30 observations from N(0,1)
# sort so smallest data point is x.dat[1], largest is x.dat[30]
x.dat <- sort(rnorm(30))

# build histogram for x.dat; let R choose number of bins; add rug and horiz grid
hist(x.dat, prob=T, col='grey', border=2, ylim = c(0, .45))
rug(x.dat)
abline(h=(1/30)*(1:30), col='white', lwd=2)

# overlay N(0,1) pdf
xvec <- seq(-3, 3, by= .01)
lines(xvec, dnorm(xvec), col=3, lwd=2)
```

The plot below shows that a histogram consists of stacks of boxes in bins. Area of each box is $1/n$. Here, the width of each box happens to be 1, so the height is also $1/n$. In each bin, number of boxes = number of data points, and total area of histogram = 1.



We can also think of the histogram as the sum of n "box functions" evaluated at each data point. For this particular histogram, the box function could be like the one below. It takes value 1/30 in the bin which contains the data point x and value 0 elsewhere (the .001 is there to avoid annoying edge effects).

```
boxfunction <- function(x, xvec){
        # x is a data.point from the sample
        # xvec is a grid of points on the x-axis
        # returns outvec = vector of y values
        # outvec = 1/30 on the bin which contains x; outvec = 0 elsewhere
        outvec <- 1/30 * dunif(xvec, floor(x)+.001, ceiling(x))
        return(outvec)
}
```

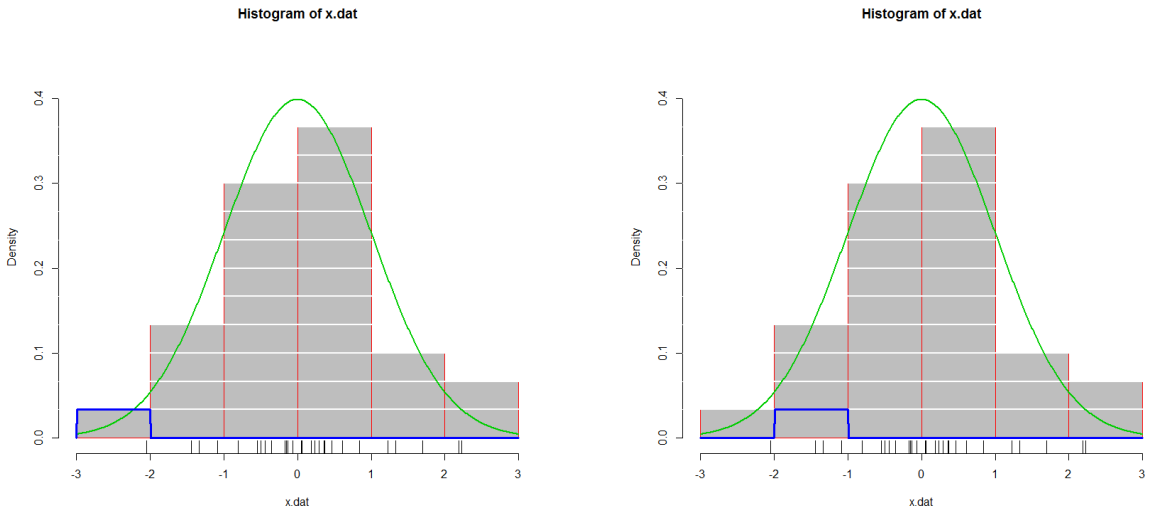
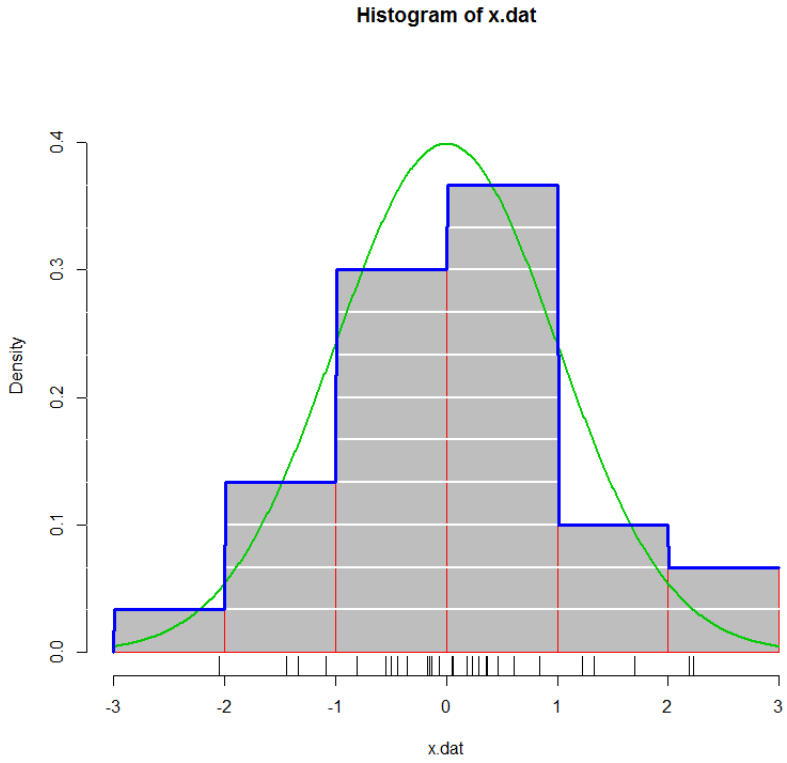


Figure 1: Box functions for `x.dat[1]` and `x.dat[2]`

Figure 1 shows the box function for the smallest data point `x.dat[1]` and the next data point `x.dat[2]`. The box function for `x.dat[3]` would be the same as the one for `x.dat[2]` because they are in the same bin. The code below stacks the box functions for all 30 data points to reproduce the histogram:




```
hist.curve <- rep(0, length(xvec))
```

```
for(k in 1:length(x.dat)){
        boxk <- boxfunction(x.dat[k], xvec)
        hist.curve <- hist.curve + boxk
}
lines(xvec, hist.curve, col=4, lwd=3)
```

Notice that the boxes are centered on the bin centers. It doesn't matter exactly where the data points fall within the bin. In this quiz we will produce an alternative to the histogram which doesn't use bins. It will stack box functions, but each box will be centered on a data point. The resulting plot will have more corners than a histogram and should better approximate the underlying pdf.

1) Generate a data vector like `x.dat <- sort(rnorm(30))`. You don't have to use `rnorm()` and you don't have to use $n = 30$.

2) Write a new box function `mybox()` with inputs x = a data point from the sample, $xvec$ = a grid of points on the x-axis, and bw = the width of the boxes (we didn't need the bandwidth parameter above because the bins were chosen by R). Use `dunif()` again, but the `floor()` and `ceiling()` parts will have to change. In `mybox()` the box should be **centered** on the sample data point x with width given by bw.

3) Write a function `myhist()` with arguments $xdat$ and bw. Build a sensible $xvec$ inside your function. Then write a loop similar to the one above. For each element of the data vector $xdat$, the function should evaluate `mybox(xdat[i], xvec, bw)` and add the result to the stack.

After the loop runs, plot the stack value versus $xvec$ and add a rug plot. Then overlay a plot of the true pdf for your data. Choose good colors. The function `myhist()` can return the stack value, but it doesn't have to. The plot is the important thing. How does it compare to a histogram of the same data?

4) (90 points) Run `myhist(x.dat, bw)` for several values of bw. Submit at least three plots in your code: one **oversmoothed**, one **undersmoothed**, and one that feels **just right**. **(Larger bw will produce wider, smoother plots. Make bw large enough, and the plot will be a single box; every data point will be covered by every box. Smaller bw will produce spikier plots with more gaps. Make bw small enough, and the plot will consist of n non-overlapping boxes; this is making a rug plot the hard way. The meaning of 'large' and 'small' depends on the variance of your data vector. So does the meaning of 'just right'. So you will have to experiment.)**

5) (20 points) Instead of stacking boxes, we can stack any shape with area $= 1/n$. Write a new function `mybump()` to replace `mybox()`. The new function should use `dnorm()` instead of `dunif()`, and use the bw input to specify the standard deviation for `dnorm()`. Modify `myhist()` to call `mybump()` instead of `mybox()` and repeat step 4 above.