Hexagon Binning: an Overview

Nicholas Lewin-Koh*

November 25, 2003

1 Overview

Hexagon binning is a form of bivariate histogram useful for visualizing the structure in datasets with large n. The underlying concept of hexagon binning is extremely simple;

1. the xy plane over the set (range(x), range(y)) is tessellated by a regular grid of hexagons.
2. the counts of points falling in each hexagon are counted and stored in a data structure
3. the hexagons with count > 0 are plotted using a color ramp or varying the radius of the hexagon in proportion to the counts.

The algorithm is extremely fast and effective for displaying the structure of datasets with $n \geq 10^6$. If the size of the grid and the cuts in the color ramp are chosen in a clever fashion than the structure inherent in the data should emerge in the binned plots. The same caveats apply to hexagon binning as apply to histograms and care should be exercised in choosing the binning parameters.

The hexbin package is a set of function for creating and plotting hexagon bins. The package extends the basic hexagon binning ideas with several functions for doing bivariate smoothing, finding an approximate bivariate median, and looking at the difference between two sets of bins on the same scale. The basic functions can be incorporated into many types of plots. This package is based on the original package for splus by Dan Carr at George Mason University and is mostly the fruit of his graphical genius and intuition.

2 Basic Hexagon Binning Functions

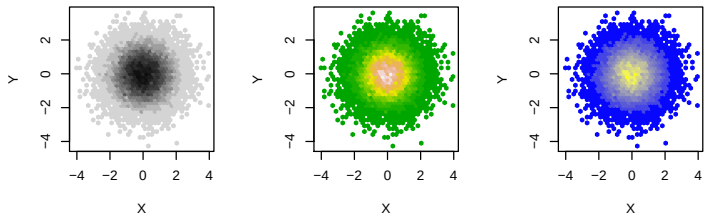
Using the basic hexagon binning functions are not much more involved than using the basic plotting functions. The following little example shows the basic features of the basic plot and binning functions. We start by loading the package and generating a toy example data set.

*with minor assistance by Martin Mächler

```
> library("hexbin")
> x <- rnorm(20000)
> y <- rnorm(20000)
> hbin <- hexbin(x, y, xbins = 40)
> plot(hbin)
```

Note that the default color scheme for the hexplot is a gray scale, but that there is an argument to plot.hexbin, colramp, that allows the use of any function that excepts an argument `n` and returns n colors. Several functions are supplied that provide alternative color-ramps to R's built in color ramp functions. Figure 1 shows the effect of using different color ramps to generate hexagon plots.

```
[1] "done 'colorscale'"

[1] "done 'colorscale'"

[1] "done 'colorscale'"
```



Three examples of using hexagons in a plot for large n with different color schemes. a) Is the default gray scale b) is the R base terrain.colors() and c) is the BTY() , Blue to Yellow, color ramp supplied with hexbin.

The code for Figure 1 is

```
> def.par <- par(no.readonly = TRUE)
> nf <- layout(matrix(1:3, nr = 1), widths = c(1, 1, 1))
> x <- rnorm(20000)
> y <- rnorm(20000)
> hbin <- hexbin(x, y, xbins = 40)
> rx <- range(x)
> ry <- range(y)
> par(pty = "s")
> plot(rx, ry, type = "n", xlab = "X", ylab = "Y")
> hexagons(hbin)

[1] "done 'colorscale'"

> plot(rx, ry, type = "n", xlab = "X", ylab = "Y")
> hexagons(hbin, colramp = terrain.colors)

[1] "done 'colorscale'"

> plot(rx, ry, type = "n", xlab = "X", ylab = "Y")
> hexagons(hbin, colramp = BTY)

[1] "done 'colorscale'"

> par(def.par)
```

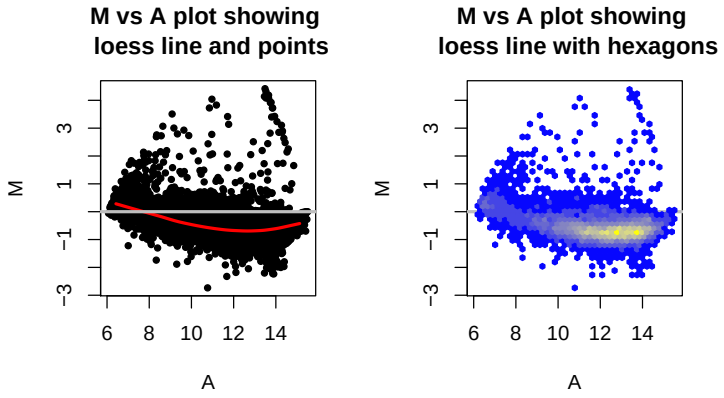
The hexbin package supplies a plotting method for the hexbin data structure. The plotting method `plot.hexbin` accepts all the parameters for the hexagon function and supplies a legend as well, for easy interpretation of the plot. Figure 2 shows a hex binned plot with a legend. A function `hex.legend` is supplied for creating user specified hexagon legends.

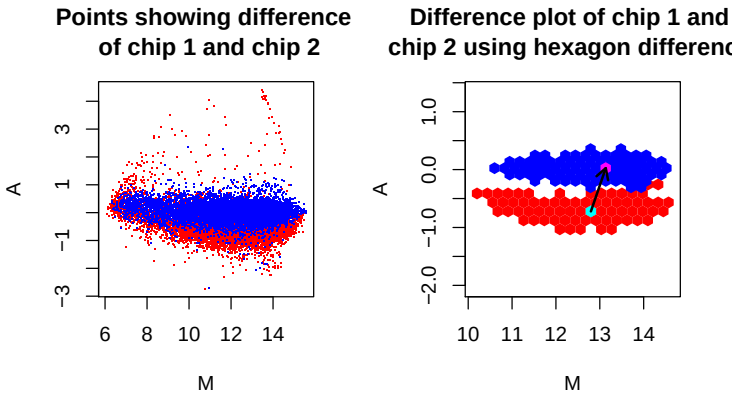
3 cDNA Chip Normalization

This example is taken from the marray suite of packages, which supplies methods and classes for the normalization and diagnostic plots of cDNA microarrays. In this example the goal is not to make any comments about the normalization methodology, but rather to show how the diagnostic plots can be enhanced using hexagon binning due to the large number of points ($n = 8,448$ cDNA probes per chip).

First we look at the diagnostic plot M vs A, where M is the log–ratio, $M = \log < -2\frac{R}{G}$ and A is the overall intensity, $A = \log < -2\sqrt{RG}$. Figure 3 shows the plot using points and on the right hexagons. The hexagon binned plot shows that most of the pairs are well below zero, and that the overall shape is more like a comet with most of the mass at the bottom of the curve, rather than a thick bar of points curving below the line.

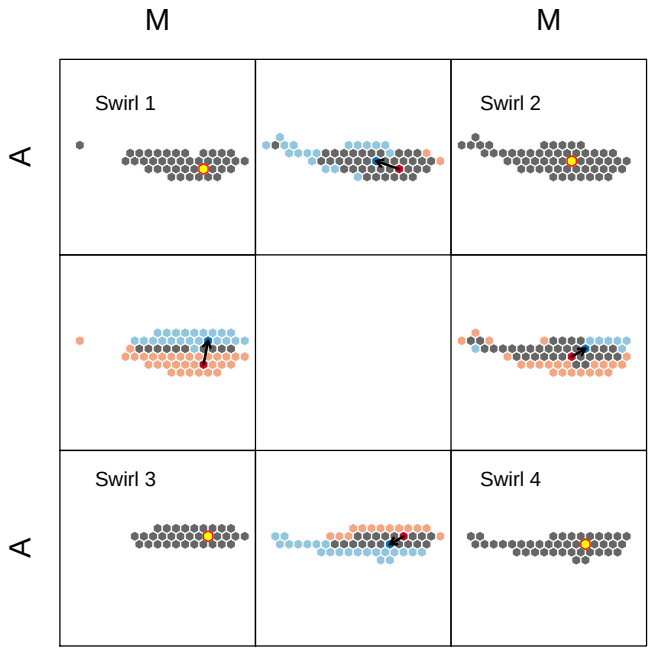
```
Loading required package: marrayClasses
Welcome to Bioconductor
         Vignettes contain introductory material.  To view,
         simply type: openVignette()
         For details on reading vignettes, see
         the openVignette help page.
[1] TRUE

Loading required package: marrayPlots
[1] TRUE

[1] "done 'colorscale'"
```





As a final example with the swirl data we use the hmatplot to demonstrate the need for normalization of all four arrays to a common scale. Figure 4 shows a matrix of figures. The four corner arrays are bivariate box plots and the four plots between are the differences of the two distributions. The arrows highlight the distance between the two bivariate medians.

```
[1]  5.96687 15.17425
[1] -3.786204  3.067059
```



To generate Figure 4 takes a bit of explanation. Before we generate any of the hexbin objects we need to make sure that all the hexagon grids have the same range, otherwise we can get distortion in the hexagons, especially in hdiffplots where two unequally scaled sets of hexagons will be plotted.

```
> require("marrayClasses")
```

```
[1] TRUE
```

```
> require("marrayPlots")
```

```
[1] TRUE
```

```
> hxrange <- range(c(maA(swirl[, 1]), maA(swirl[, 2]), maA(swirl[,
+     3]), maA(swirl[, 4])))
> hyrange <- range(c(maM(swirl[, 1]), maM(swirl[, 2]), maM(swirl[,
+     3]), maM(swirl[, 4])))
```

After we obtain the appropriate range of all the hexagons we need to generate the hexagon bin objects and their erosion components. Note that we use the extended ranges through the `xbnds` and `ybnds` arguments. Also note the use of the cdf argument which gives the cut–off for bins that should be included in the erosion component. Also we need to generate a matrix of the eroded hbin names and row and column labels.

```
> hb1 <- hexbin(maA(swirl[, 1]), maM(swirl[, 1]), xbins = 22, xbnds = hxrange,
+     ybnds = hyrange)
> hb1.e <- erode.hexbin(hb1, cdf = 0.25)
> hb2 <- hexbin(maA(swirl[, 2]), maM(swirl[, 2]), xbins = 22, xbnds = hxrange,
+     ybnds = hyrange)
> hb2.e <- erode.hexbin(hb2, cdf = 0.25)
> hb3 <- hexbin(maA(swirl[, 3]), maM(swirl[, 3]), xbins = 22, xbnds = hxrange,
+     ybnds = hyrange)
> hb3.e <- erode.hexbin(hb3, cdf = 0.25)
> hb4 <- hexbin(maA(swirl[, 4]), maM(swirl[, 4]), xbins = 22, xbnds = hxrange,
+     ybnds = hyrange)
> hb4.e <- erode.hexbin(hb4, cdf = 0.25)
> nam <- matrix(c("hb1.e", "hb2.e", "hb3.e", "hb4.e"), ncol = 2)
> row.lab <- c("A", "A")
> col.lab <- c("M", "M")
> tpo <- list(x = c(0.055, 0.77, 0.055, 0.77), y = c(0.98, 0.98,
+     0.18, 0.18))
> text(tpo, c("Swirl 1", "Swirl 3", "Swirl 2", "Swirl 4"))
```

Now we a ready to plot. Note we need to specify two lists of colors, one for the borders and one for the hexagon fill. These lists need to be dived for the hbox and hdiff component.

```
> hplt.par <- hmatplot(nam, row.lab, col.lab, border = list(hbox = c("red",
+     "white"), hdiff = rep("white", 6)), pen = list(hbox = c("yellow",
+     gray(0.4)), hdiff = c("#F4A582", gray(0.4), "#92C5DE", "#CA0020",
+     "#0571B0", "black")))

[1]  5.96687 15.17425
[1] -3.786204  3.067059
```

Finally we add annotation using global parameters since the original device parameters are restored on exit.

```
> tpo <- list(x = c(0.055, 0.77, 0.055, 0.77), y = c(0.98, 0.98,
+     0.18, 0.18))
> text(tpo, c("Swirl 1", "Swirl 3", "Swirl 2", "Swirl 4"))
```

There are many applications of hexagon binning, and this package in later versions will give a set of methods and tools for extending hexagon binning to many situations.