R news and tutorials contributed by (573) R bloggers

Welcome!

Here you will find daily **news and tutorials about R**, contributed by over 573 bloggers. There are many ways to **follow us -**

By e-mail:

On Facebook:

If you are an R blogger yourself you are invited to add your own R content feed to this site (**Non-English** R bloggers should add themselves- here)

Jobs for R-

users

- Economic Analyst
- Research Analyst
- Research and Analytics Associate
- Monitoring and Data Analytics Officer
- Worldwide CoC Team –Data Scientist and Managing Strategy Consultant

Search & Hit Enter

Popular Searches

- web scraping
- heatmap
- twitteR
- maps
- time series
- animation
- shiny
- boxplot
- hadoop
- Ggplot2
- sql
- trading
- finance
- how to import image file to R
- eclipse
- LaTeX
- ggplot
- googlevis
- excel
- quantmod
- knitr
- PCA
- rattle
- market research
- RStudio
- regression
- Tutorial
- coplot
- Rcmdr
- map

Recent Posts

- Data Science Competitions 101: Anatomy and Approach
- New chapters for 50 shades of grey….
- Reproducible art with R
- Notes from the 4th R in Insurance Conference
- Voting clusters in the U.N.
- 2016-09 An Introduction to the 'DOM' Package
- Stock prices analysis part 1 exercises
- R and openMP: boosting compiled code on multi-core cpu-s
- Update on the R Shapefile Contest
- Use quick formula functions in purrr::map (+ base vs tidtyverse idiom comparisons/examples)
- Ooms Magical Polyglot World
- DeployR Enterprise Security Model
- The reproducibility crisis in science and prospects for R
- Calculating Pi using Buffon's Needle
- rotl paper published

er sites

SAS blogs
Jobs for R-users

Ways to Do 2D Histograms in

September 1, 2014
By Myles Harrison

Introduction

Lately I was trying to put together some 2D histograms in R and found that there are many ways to do it, with directions on how to do so scattered across the internet in blogs, forums and of course, Stackoverflow.

As such I thought I'd give each a go and also put all of them together here for easy reference while also highlighting their difference.

For those not "in the know" a 2D histogram is an extensions of the regular old histogram, showing the distribution of values in a data set across the range of two quantitative variables. It can be considered a special case of the heat map, where the intensity values are just the count of observations in the data set within a particular area of the 2D space (bucket or bin).

So, quickly, here are 5 ways to make 2D histograms in R, plus one additional figure which is pretty neat.

First and foremost I get the palette looking all pretty using RColorBrewer, and then chuck some normally distributed data into a data frame (because I'm lazy). Also one scatterplot to justify the use of histograms.

```
# Color housekeeping
library(RColorBrewer)
rf <- colorRampPalette(rev(brewer.pal(11,'Spectral')))
r <- rf(32)

# Create normally distributed data for plotting
x <- rnorm(mean=1.5, 5000)
y <- rnorm(mean=1.6, 5000)
df <- data.frame(x,y)

# Plot
plot(df, pch=16, col='black', cex=0.5)
```

Option 1: hexbin

The *hexbin* package slices the space into 2D hexagons and then counts the number of points in each hexagon. The nice thing about *hexbin* is that it provides a legend for you, which adding manually in R is s a pain. The default invocation provides a pretty sparse looking chrome figure. Adding the colramp parameter with a suitable produced from colorRampPalette makes things nicer. The l placement is a bit strange – I adjusted it after the fact though st as well do so in the R code.

```
 OPTION 1: hexbin from package 'hexbin' #######
y(hexbin)
te hexbin object and plot
exbin(df)
```

```
plot(h)
plot(h, colramp=rf)
```

Using the *hexbinplot* function provides greater flexibility, allowing specification of endpoints for the bin counting, and also allowing the provision of a transformation functions. Here I did log scaling. Also it appears to handle the legend placement better; no adjustment was required for these figures.

```
# hexbinplot function allows greater flexibility
hexbinplot(y~x, data=df, colramp=rf)
# Setting max and mins
hexbinplot(y~x, data=df, colramp=rf, mincnt=2, maxcnt=60)


# Scaling of legend - must provide both trans and inv functions
hexbinplot(y~x, data=df, colramp=rf, trans=log, inv=exp)
```

Option 2: hist2d

Another simple way to get a quick 2D histogram is to use the *hist2d* function from the *gplots* package. Again, the default invocation leaves a lot to be desired:

```
##### OPTION 2: hist2d from package 'gplots' #######
library(gplots)

# Default call
h2 <- hist2d(df)
```

Setting the colors and adjusting the bin sizing coarser yields a more desirable result. We can also scale so that the intensity is logarithmic as before.

```
# Coarser binsizing and add colouring
h2 <- hist2d(df, nbins=25, col=r)

# Scaling with log as before
h2 <- hist2d(df, nbins=25, col=r, FUN=function(x) log(length(x)))
```

Option 3: stat_2dbin from ggplot

And of course, where would a good R article be without reference to the ggplot way to do things? Here we can use the *stat_bin2d* function, other added to a ggplot object or as a type of geometry in the call to qplot.

```
##### OPTION 3: stat_bin2d from package 'ggplot' #######
library(ggplot2)

ult call (as object)
gplot(df, aes(x,y))
p + stat_bin2d()


ult call (using qplot)
x,y,data=df, geom='bin2d')
```

, we probably want to adjust the bin sizes to a desired number, so ensure that ggplot uses our colours that we created before. tter is done by adding the *scale_fill_gradientn* function with our

colour vector as the colours argument. Log scaling is also easy to add using the trans parameter.

```
# Add colouring and change bins
h3 <- p + stat_bin2d(bins=25) + scale_fill_gradientn(colours=r)
h3

# Log scaling
h3 <- p + stat_bin2d(bins=25) + scale_fill_gradientn(colours=r, trans="log")
h3
```

Option 4: kde2d

Option #4 is to do kernel density estimation using *kde2d* from the MASS library. Here we are actually starting to stray from discrete bucketing of histograms to true density estimation, as this function does interpolation.
The default invocation uses n = 25 which is actually what we've been going with in this case. You can then plot the output using *image()*.

Setting n higher does interpolation and we are into the realm of kernel density estimation, as you can set your "bin size" lower than how your data actually appear. Hadley Wickham notes that in R there are over 20 packages [PDF] with which to do density estimation so we'll keep that to a separate discussion.

```
##### OPTION 4: kde2d from package 'MASS' #######
# Not a true heatmap as interpolated (kernel density estimation)
library(MASS)

# Default call
k <- kde2d(df$x, df$y)
image(k, col=r)

# Adjust binning (interpolate - can be computationally intensive for large datasets)
k <- kde2d(df$x, df$y, n=200)
image(k, col=r)
```

Option 5: The Hard Way

Lastly, an intrepid R user was nice enough to show on Stackoverflow how do it "the hard way" using base packages.

```
##### OPTION 5: The Hard Way (DIY) #######
# http://stackoverflow.com/questions/18089752/r-generate-2d-histogram-from-raw-data
nbins <- 25
x.bin <- seq(floor(min(df[,1])), ceiling(max(df[,1])), length=nbins)
y.bin <- seq(floor(min(df[,2])), ceiling(max(df[,2])), length=nbins)

freq <-  as.data.frame(table(findInterval(df[,1], x.bin),findInterval(df[,2], y.bin)))
freq[ 1] <- as.numeric(freq[,1])
 2] <- as.numeric(freq[,2])

 <- diag(nbins)*0
[cbind(freq[,1], freq[,2])] <- freq[,3]

al
x.bin, y.bin, freq2D, col=r)


x.bin, y.bin, log(freq2D), col=r)
```

Not the way I would do it, given all the other options available, however if you want things "just so" maybe it's for you.

Bonus Figure

Lastly I thought I would include this one very cool figure from Computational Actuarial Science with R which is not often seen, which includes both a 2D histogram with regular 1D histograms bordering it showing the density across each dimension.

```
##### Addendum: 2D Histogram + 1D on sides (from Computational ActSci w R) #######
#http://books.google.ca/books?id=YWcLBAAAQBAJ&pg=PA60&lpg=PA60&dq=kde2d+log&source=bl&ots=7AB-RAoMqY&sig=gFaHSoQCoGMXrR9BTaLOdCs198U&hl=en&sa=X&ei=8mQDVPqtMsi4ggSRnILQDw&redir_esc=y#v=onepage&q=kde2d%20log&f=false

h1 <- hist(df$x, breaks=25, plot=F)
h2 <- hist(df$y, breaks=25, plot=F)
top <- max(h1$counts, h2$counts)
k <- kde2d(df$x, df$y, n=25)

# margins
oldpar <- par()
par(mar=c(3,3,1,1))
layout(matrix(c(2,0,1,3),2,2,byrow=T),c(3,1), c(1,3))
image(k, col=r) #plot the image
par(mar=c(0,2,1,0))
barplot(h1$counts, axes=F, ylim=c(0, top), space=0, col='red')
par(mar=c(2,0,0.5,1))
barplot(h2$counts, axes=F, xlim=c(0, top), space=0, col='red', horiz=T)
```

Conclusion

So there you have it! 5 ways to create 2D histograms in R, plus some additional code to create a really snappy looking figure which incorporates the regular variety. I leave it to you to write (or find) some good code for creating legends for those functions which do not include them. Hopefully other R users will find this a helpful reference.

References

code on github
https://github.com/mylesmharrison/5_ways_2D_histograms
R generate 2D histogram from raw data (Stackoverflow)
http://stackoverflow.com/questions/18089752/r-generate-2d-histogram-from-raw-data
Computational Actuarial Science with R (Google Books)
http://books.google.ca/books?id=YWcLBAAAQBAJ&pg=PA60&lpg=PA60&dq=kde2d+log&source=bl&ots=7AB-RAoMqY&sig=gFaHSoQCoGMXrR9BTaLOdCs198U&hl=en&sa=X&ei=8mQDVPqtMsi4ggSRnILQDw&redir_esc=y#v=onepage&q=kde2d%20log&f=false
Wickham, Hadley. Density Estimation in R [PDF]
vita.had.co.nz/papers/density-estimation.pdf

Jobs for R users