

# Package: hextri (via r-universe)

October 24, 2024

**Type** Package

**Title** Hexbin Plots with Triangles

**Version** 0.9.17

**Author** Thomas Lumley (Jie Fu Yu made a prototype)

**Maintainer** Thomas Lumley <t.lumley@auckland.ac.nz>

**Description** Display hexagonally binned scatterplots for multi-class data, using coloured triangles to show class proportions.

**License** MIT + file LICENSE

**Imports** hexbin, graphics, FNN, grid, grDevices

**Suggests** lattice, knitr, datasets

**VignetteBuilder** knitr

**Repository** <https://tslumley.r-universe.dev>

**RemoteUrl** <https://github.com/tslumley/hextri>

**RemoteRef** HEAD

**RemoteSha** 3e90299a816287cc08a13404ab99f1e375661822

## Contents

hexclass . . . . .	2
panel.hextri . . . . .	4
sainte_lague . . . . .	5
<b>Index</b>	<b>7</b>

hexclass

*Hexagonal binning with classes***Description**

Displays a hexagonally-binned scatterplot with coloured subtriangles to indicate number of observations in each class. With a single class, gives a base-graphics version of the ordinary hexbin plot.

**Usage**

```
hextri(x,...)
## Default S3 method:
hextri(x, y, class, colours, nbins = 10, border =
TRUE, diffuse=FALSE, style=c("alpha","size"), weights=NULL,
sorted=!diffuse,minfrac=0, ...)
## S3 method for class 'formula'
hextri(x, data=parent.frame(), class,colours,nbins=10,
border=TRUE, diffuse=FALSE, style=c("alpha","size"),weights=NULL, sorted=!diffuse,
xlab=NULL, ylab=NULL,minfrac=0,...)
```

**Arguments**

x	Coordinates of points, or for the formula method, a model formula
y	coordinates
data	The class and weights arguments are looked up here, in addition to the formula itself
class	Factor giving class membership for points
colours	Vector of colors (number, name, or hashtag format) giving the colours for each class
nbins	Number of bins in the x-axis direction
border	Include a narrow transparent border around each triangle and hexagon
diffuse	Pass on rounding error to nearest not-yet-drawn hexes so that rare classes get represented
style	Represent data by hexagon size or by alpha-blending
weights	If not NULL, sampling weights or frequency weights. The hexbin plot will be based on the sum of weights in the hex
sorted	Should the triangles be sorted into a consistent order within each hex? More attractive but can cause 3-d artifacts and may be less accurate.
xlab, ylab	Specify to override the formula-based defaults
minfrac	Cells with radius smaller than this fraction of a whole cell will be drawn as points. Most useful with style="size"
...	Passed to the plot call that sets up the plot region

**Details**

Uses the Sainte-Lague method to apportion counts to triangles, ensuring the counts round to exactly six triangles per hexagon.

The binning/aspect ratio are adjusted to give regular hexagons on the figure region set by `plot(x,y)`. If you reshape the window you will need to redraw the plot.

**Value**

A list with components `x`, `y`, and `col` suitable as input to `polygon()`

**Author(s)**

Thomas Lumley

**See Also**

[sainte\\_lague](#)

**Examples**

```
xx<-rnorm(1000)
yy<-rnorm(1000)
cc<-cut(xx*yy,c(-Inf,-.4,0,.4,Inf))

plot(xx,yy,col=(1:4)[cc])
hextri(xx,yy,cc,1:4,nbins=20,border=TRUE,style="size")

##formula method
data(NHANES, package="hexbin")
hextri(Weight~Age, class=Smoke, col=c("red","orange","green","grey40"),
  data=NHANES, style="size",nbins=20, main="Smoking")
legend("topright",fill=c("red","orange","green","grey40"),
  legend=c("Current","Past","Never","unknown"),bty="n")

## minimum sizes

data(NHANES, package="hexbin")
hextri(Weight~Age, class=Smoke, col=c("red","orange","green","grey40"),
  data=NHANES, style="size",nbins=20, main="Smoking",minfrac=.2)
legend("topright",fill=c("red","orange","green","grey40"),
  legend=c("Current","Past","Never","unknown"),bty="n")

## using the return value of hextri
rval<-hextri(xx,yy,cc,1:4,nbins=20,border=TRUE,style="alpha")
plot(y~x,data=rval,type="n")
with(rval, polygon(x,y,col=col,border=NA))

## diffusion
xx<-runif(10000)
yy<-runif(10000)
```

```
cc<-rep(1:3,c(4750,4750,500))
hextri(xx,yy,cc,2:4,border=TRUE,diffuse=FALSE,style="size")
hextri(xx,yy,cc,2:4,border=TRUE,diffuse=TRUE,sorted=TRUE, style="size")

hextri(xx,yy,cc,2:4,border=TRUE,diffuse=TRUE,style="size",weights=cc)
```

---

panel.hextri

*A lattice panel function for hexagonal binning with classes*


---

### Description

Displays a hexagonally-binned scatterplot with coloured subtriangles to indicate number of observations in each class. Uses the Sainte-Lague method to apportion counts to triangles, ensuring the counts round to exactly six triangles per hexagon.

### Usage

```
panel.hextri(x, y, groups, subscripts, colours, nbins = 10, border = TRUE,
  diffuse = FALSE, style = c("alpha", "size"), weights = NULL,
  sorted=!diffuse, shape = 1, ...)
```

### Arguments

x, y	point coordinates for the subset being plotted
groups	A factor giving the class identity for all points (will be filled in automatically by xyplot)
subscripts	Vector selecting the elements of groups that are in the panel (will be filled in automatically by xyplot)
colours	Vector of colors (number, name, or hashtag format) giving the colours for each class
nbins	Number of bins along the x axis
border	If TRUE, leave a transparent border around each element drawn
diffuse	If TRUE pass on rounding error to nearest not-yet-drawn hexes so that rare classes get represented
style	Represent data by hexagon size or by alpha-blending
weights	If not NULL, sampling weights or frequency weights. The hexbin plot will be based on the sum of weights in the hex
sorted	Sort the triangles into a consistent order within each hex?
shape	Aspect ratio for each hex.
...	because you have to.

### Value

A panel

**Author(s)**

Thomas Lumley

**See Also**[hextri](#) is the base-graphics plot**Examples**

```

library(lattice)
xx<-rnorm(1000)
yy<-rnorm(1000)
cc<-cut(xx*yy,c(-Inf,-.4,0,.4,Inf))
zz<-cut(xx+yy,c(-Inf,-1,0,1,Inf))

plot(xx,yy,col=(1:4)[cc])
xyplot(yy~xx|zz, panel=panel.hextri,groups=cc,colours=1:4,style="size",nbins=10)

data(airquality)
airquality$o3group<-with(airquality, cut(Ozone, c(0,18,60,Inf)))

xyplot(Temp~Solar.R|equal.count(Wind,4), groups=o3group, panel=panel.hextri,
  data=na.omit(airquality),colours=c("royalblue","grey60","goldenrod"),
  strip=strip.custom(var.name="Wind Speed"),xlab="Solar Radiation (langley)",
  ylab="Temperature (F)")

```

---

sainte\_lague

*Proportional representation by Sainte-Lague method*


---

**Description**

Originally an algorithm for proportional allocation of seats to parties in elections, used here to assign the six triangles in each hex to classes.

**Usage**

```
sainte_lague(votes, nseats)
```

**Arguments**

`votes`            Vector of 'votes' for each party, non-negative numeric values  
`nseats`            Single integer giving the number of seats to be allocated (6, here)

**Value**

Numeric vector of length `nseats` giving the class membership for each seat, with an error attribute giving the rounding errors for each party.

**References**

[https://en.wikipedia.org/wiki/Sainte-Lagu%C3%AB\\_method](https://en.wikipedia.org/wiki/Sainte-Lagu%C3%AB_method)

**See Also**

[hextri](#)

**Examples**

```
sainte_lague(c(100,200,300,50),6)
```

# Index

## \* **hplot**

hexclass, [2](#)

panel.hextri, [4](#)

hexclass, [2](#)

hextri, [5](#), [6](#)

hextri (hexclass), [2](#)

panel.hextri, [4](#)

sainte\_lague, [3](#), [5](#)