vipor package usage example (version 0.4.2)

Scott Sherrill-Mix, Erik Clarke

Abstract

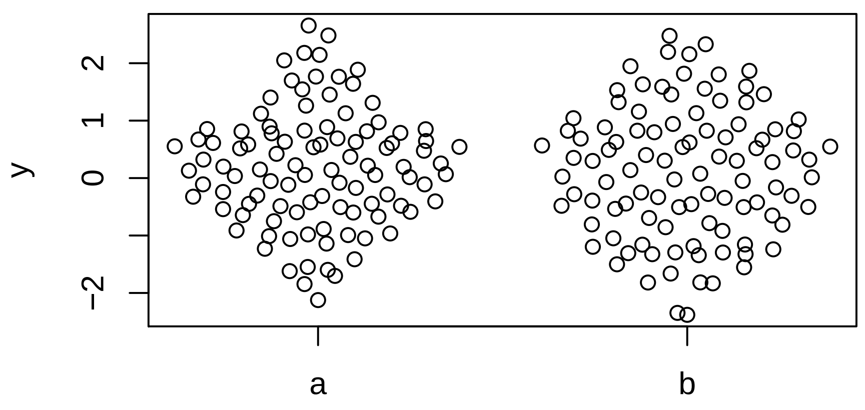
This is a collection of examples of usage for the **vipor** package.

Keywords: visualization, display, one dimensional, grouped, groups, violin, scatter, points, quasirandom, beeswarm, van der Corput.

1. The basics

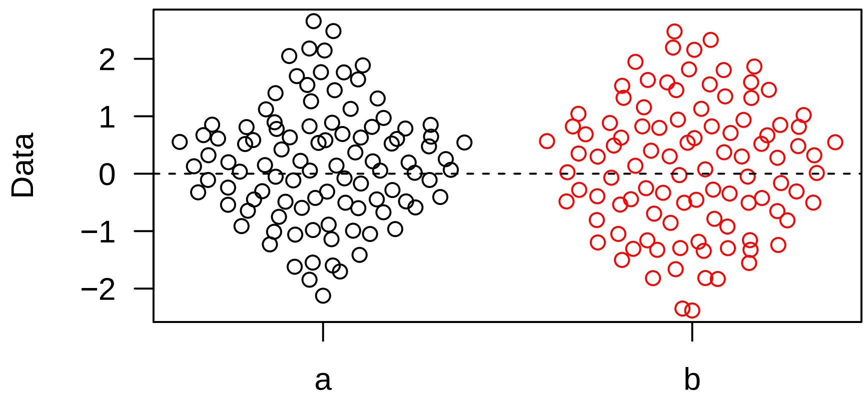
This is the simplest example of using the `vpPlot` function to generate violin scatter plots:

```
>   library(vipor)
>   set.seed(12345)
>   n<-100
>   dat<-rnorm(n*2)
>   labs<-rep(c('a','b'),n)
>   vpPlot(labs,dat)
```



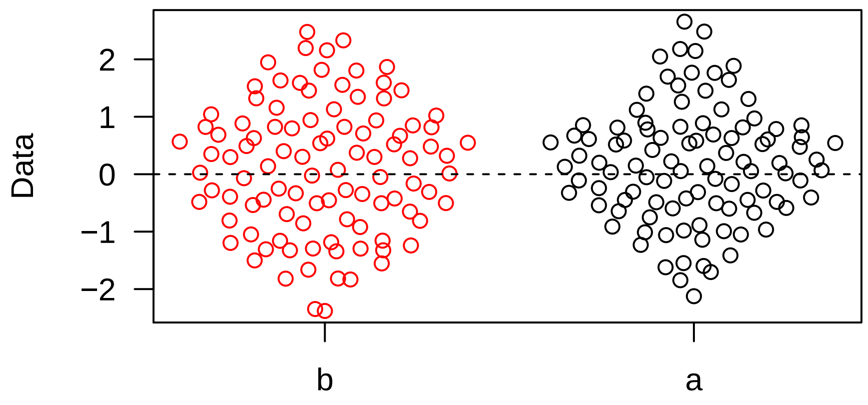
`vpPlot` is just a wrapper around `plot` so standard graphical options can be used and the plot can be annotated with R plotting functions:

```
>   vpPlot(labs,dat,las=1,ylab='Data',col=rep(1:2,n))
>   abline(h=0,lty=2)
```



Factors can be used to generate custom group orderings:

```
>   labs2<-factor(labs,levels=c('b','a'))
>   vpPlot(labs2,dat,las=1,ylab='Data',col=rep(1:2,n))
>   abline(h=0,lty=2)
```



For custom plotting, the offsets for a group of points can be calculated using the `offsetX` function. The adjusted x position of the points is also returned invisibly from `vpPlot`:

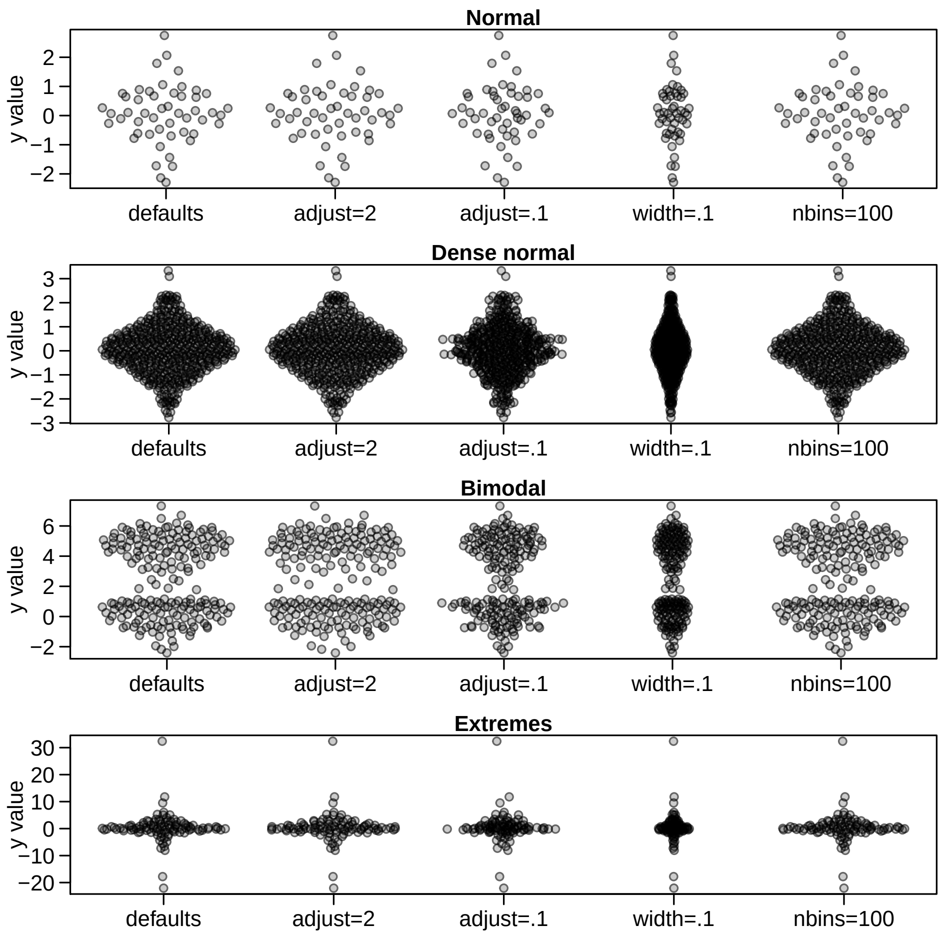
```
>   offsets<-offsetX(dat,labs)
>   head(offsets,4)

[1] -0.18939738  0.10387013  0.28854590  0.01104955

>   xPos<-vpPlot(labs,dat)
>   head(xPos,4)

[1] 0.8106026 2.1038701 1.2885459 2.0110496

>   xPos2<-rep(1:2,n)+offsets
>   head(xPos2,4)

[1] 0.8106026 2.1038701 1.2885459 2.0110496

>   all(xPos==xPos2)

[1] TRUE
```

Note that `offsetX` returns offsets centered around 0 which will need to be added to the original x positions.

2. Options

`offsetX` calls `stats::density` to compute kernel density estimates. The tightness of the fit can be adjusted with the `adjust` option and the width of the offset with `width`. `nbins` to adjust the number of bins used in the kernel density is also provided but this can usually be left at its default when using quasirandom offsets:

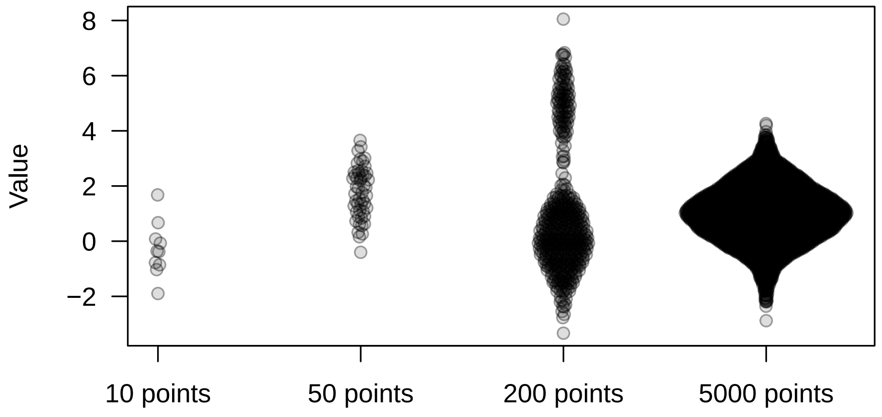
```
>   dat <- list(
+     'Normal'=rnorm(50),
+     'Dense normal'= rnorm(500),
+     'Bimodal'=c(rnorm(100), rnorm(100,5)),
+     'Extremes'=rcauchy(100)
+   )
>   par(mfrow=c(4,1), mar=c(2.5,3.1, 1.2, 0.5),mgp=c(2.1,.75,0),
+   cex.axis=1.2,cex.lab=1.2,cex.main=1.2)
>   dummy<-sapply(names(dat),function(label) {
+     y<-dat[[label]]
+     offsets <- list(
+       'defaults'=offsetX(y),  # Default
+       'adjust=2'=offsetX(y, adjust=2),    # More smoothing
+       'adjust=.1'=offsetX(y, adjust=0.1),  # Tighter fit
+       'width=.1'=offsetX(y, width=0.1),     # Less wide
+       'nbins=100'=offsetX(y, nbins=100)     # Less bins
+     )
+     ids <- rep(1:length(offsets), each=length(y))
+     plot(unlist(offsets) + ids, rep(y, length(offsets)), ylab='y value',
+       xlab='', xaxt='n', pch=21,
+       col='#00000099',bg='#00000033',las=1,main=label)
+     axis(1, 1:length(offsets), names(offsets))
+   })
```



The `varwidth` argument scales the width of a group by the square root of the number of observations in that group (as in the function `boxplot`). Arguments to `offsetX` can be passed into `vpPlot` as a list through the `offsetXArgs` argument.

```
>   dat <- list(
+     '10 points'=rnorm(10),
+     '50 points'=rnorm(50,2),
+     '200 points'=c(rnorm(400), rnorm(100,5)),
+     '5000 points'= rnorm(5000,1)
+   )
```

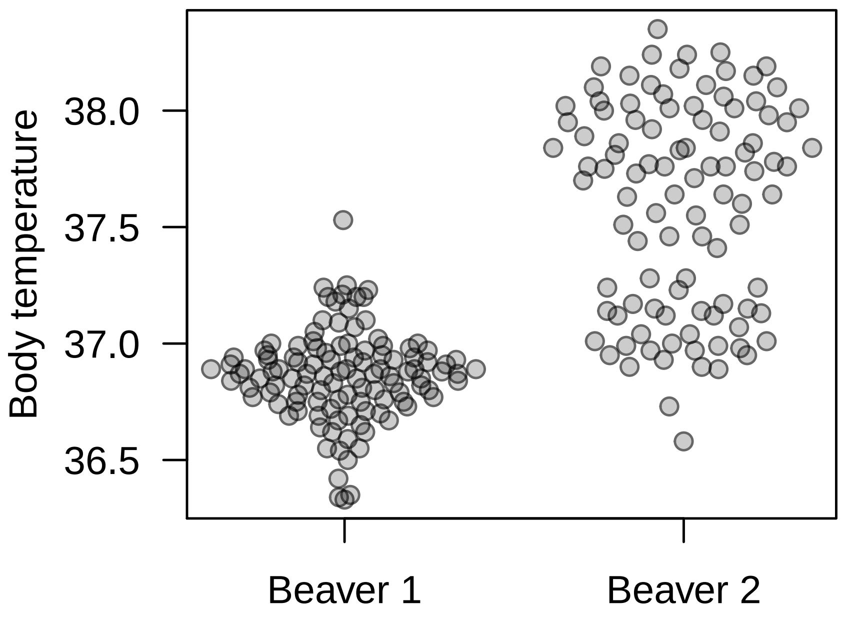
```
>   labs<-rep(names(dat),sapply(dat,length))
>   labs<-factor(labs,levels=unique(labs))
>   vpPlot( labs,unlist(dat),offsetXArgs=list(varwidth=TRUE),
+     las=1,ylab='Value',col='#00000066',bg='#00000022',pch=21)
```



3. Real data

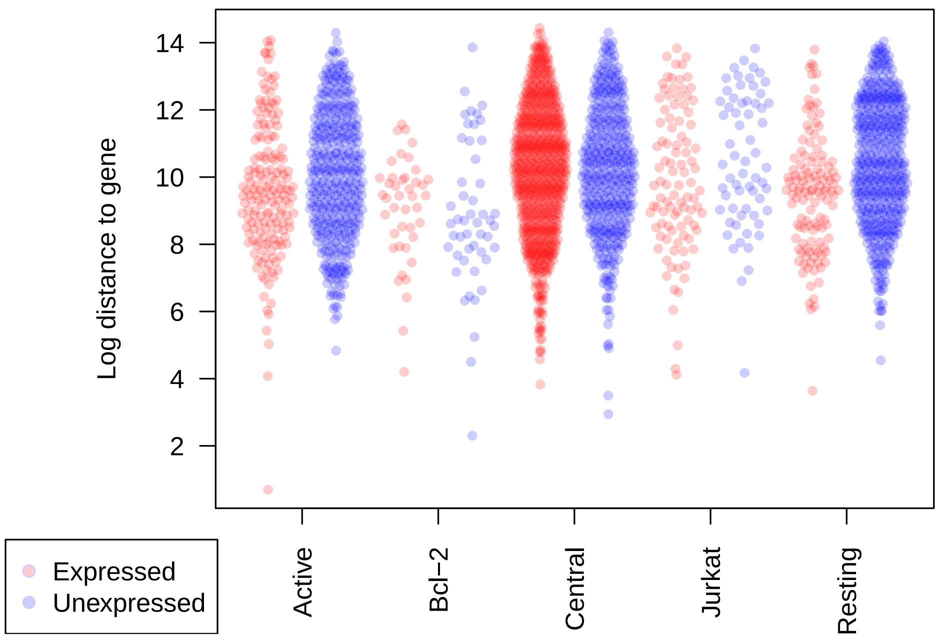
An example using the `beaver1` and `beaver2` data from the **datasets** package:

```
>   y<-c(beaver1$temp,beaver2$temp)
>   x<-rep(
+     c('Beaver 1','Beaver 2'),
+     c(nrow(beaver1),nrow(beaver2))
+   )
>   vpPlot(x,y,las=1, ylab='Body temperature',
+     pch=21, col='#00000099',bg='#00000033')
```



An example using the `integrations` data from this package:

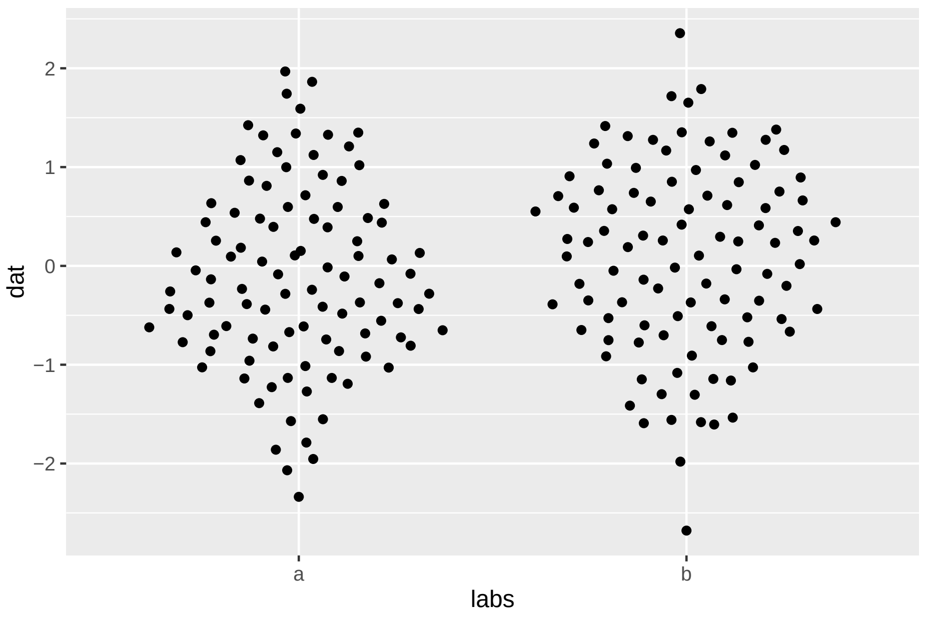
```
>   ints<-integrations[integrations$nearestGene>0,]
>   y<-log(ints$nearestGene)
>   x<-as.factor(paste(ints$study,ints$latent))
>   activeCols<-c('Expressed'='#FF000033','Unexpressed'='#0000FF33')
>   cols<-activeCols[ints$latent]
>   par(mar=c(4,7,.1,.1))
>   vpPlot(x,y,las=2, ylab='Log distance to gene',xaxt='n',
+     pch=21, col=cols,bg=cols,cex=.7)
>   uniqX<-levels(x)
>   prettyX<-tapply(1:length(uniqX),sub('(Une|E)xpressed$','',uniqX),mean)
>   axis(1,prettyX,names(prettyX),las=2)
>   legend(grconvertX(0.01,from='ndc'),grconvertY(0.15,from='ndc'),
+     names(activeCols),pch=21,col=cols,pt.bg=activeCols,xpd=NA)
```



4. ggbeeswarm package

This package is also wrapped by the **ggbeeswarm** package so if you prefer `ggplot` then you can do something like:

```
>   library(ggbeeswarm)
>   n<-100
>   dat<-rnorm(n*2)
>   labs<-rep(c('a','b'),n)
>   ggplot(mapping=aes(labs,dat))+geom_quasirandom()
```



Affiliation:

Github: http://github.com/sherrillmix/vipor
Cran: https://cran.r-project.org/package=vipor