Grouped one-dimensional data method comparison (vipor version 0.4.3)

Scott Sherrill-Mix, Erik Clarke

Abstract

This is a comparison of various methods for visualizing groups of 1-dimensional data with an emphasis on the **vipor** package.

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

1. Methods

There are several ways to plot grouped one-dimensional data combining points and density estimation:

pseudorandom The kernel density is estimated then points are distributed uniform randomly within the density estimate for a given bin. Selection of an appropriate number of bins does not greatly affect appearance but coincidental clumpiness is common.

alternating within bins The kernel density is estimated then points are distributed within the density estimate for a given bin evenly spaced with extreme values alternating from right to left e.g. max, 3rd max, ..., 4th max, 2nd max. If maximums are placed on the outside then these plots often form consecutive "smiley" patterns. If minimums are placed on the outside then "frowny" patterns are generated. Selection of the number of bins can have large effects on appearance important.

tukey An algorithm described by Tukey and Tukey in "Strips displaying empirical distributions: I. textured dot strips" using constrained permutations of offsets to distrbute the data.

beeswarm The package **beeswarm** provides methods for generating a "beeswarm" plot where points are distibuted so that no points overlap. Kernel density is not calculated although the resulting plot does provide an approximate density estimate. Selection of an appropriate number of bins affects appearance and plot and point sizes must be known in advance.

quasirandom The kernel density is estimated then points are distributed quasirandomly using the von der Corput sequence within the density estimate for a given bin. Selection of an appropriate number of bins does not greatly affect appearance and position does not depend on plotting parameters.

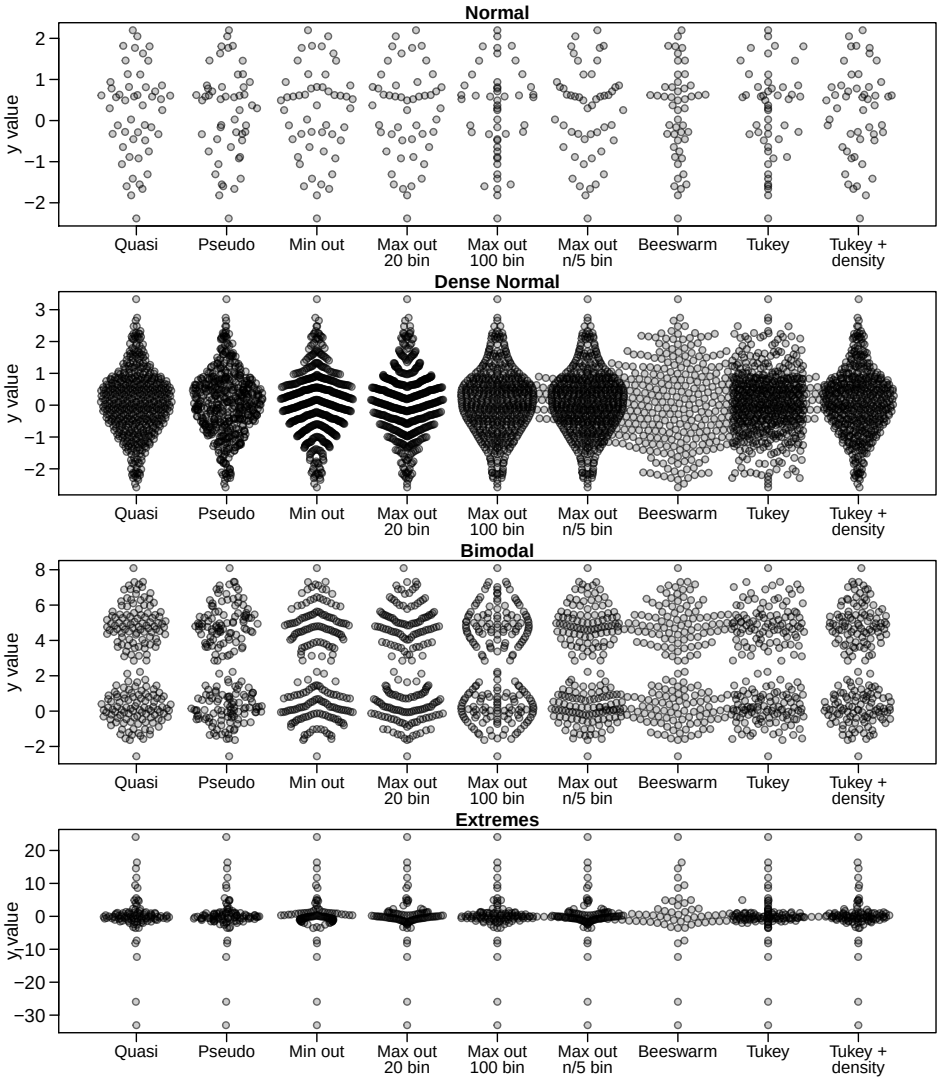
2. Simulated data

To compare between methods we'll generate some simulated data from normal, bimodal (two normal) and Cauchy distributions:

```
>   library(vipor)
>   library(beeswarm)
>   library(beanplot)
>   library(vioplot)
>   set.seed(12345)
>   dat <- list(rnorm(50), rnorm(500), c(rnorm(100),
+     rnorm(100,5)), rcauchy(100))
>   names(dat) <- c("Normal", "Dense Normal", "Bimodal", "Extremes")
```

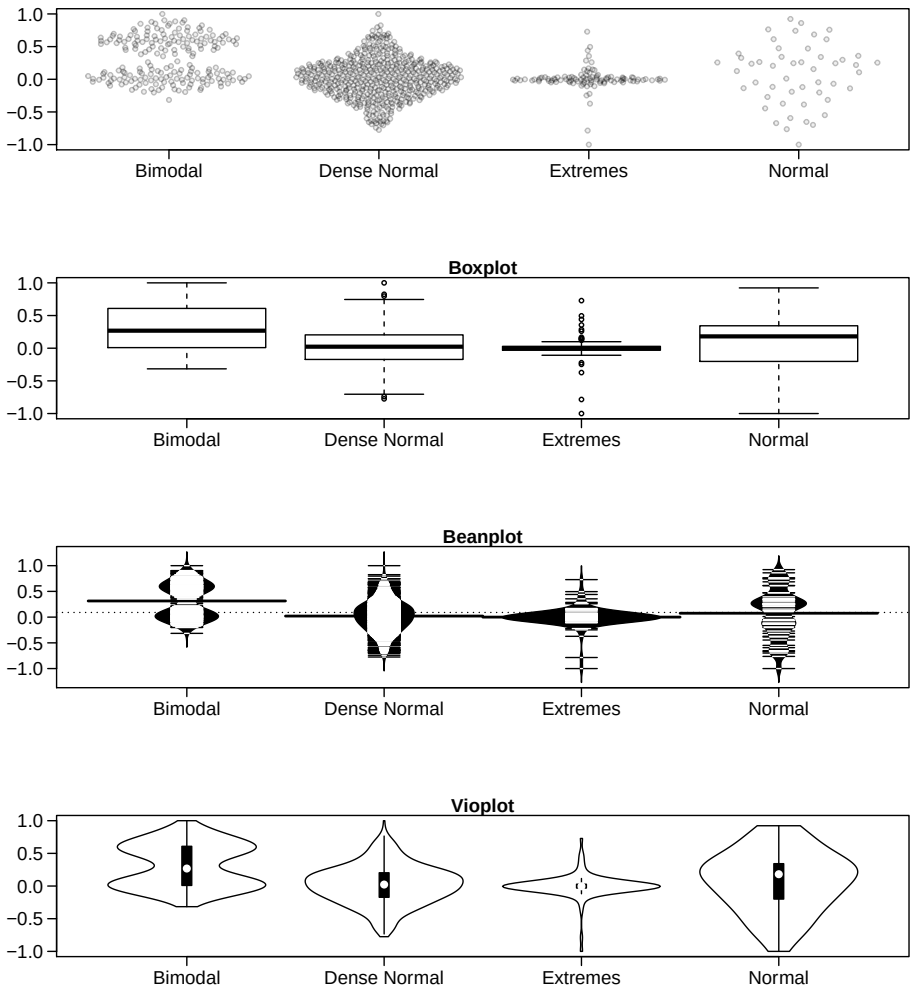
And plot the data using quasirandom, pseudorandom, alternating, Tukey texture and beeswarm methods:

```
>   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]]
+     # need to plot first so beeswarm can figure out pars
+     # xlim is a magic number due to needing plot for beeswarm
+     plot(1,1,type='n',xlab='',xaxt='n',ylab='y value',las=1,main=label,
+       xlim=c(0.5,9.5),ylim=range(y))
+     offsets <- list(
+       'Quasi'=offsetX(y),  # Default
+       'Pseudo'=offsetX(y, method='pseudorandom',nbins=100),
+       'Min out'=offsetX(y, method='minout',nbins=20),
+       'Max out\n20 bin'=offsetX(y, method='maxout',nbins=20),
+       'Max out\n100 bin'=offsetX(y, method='maxout',nbins=100),
+       'Max out\nn/5 bin'=offsetX(y, method='maxout',nbins=round(length(y)/5)),
+       'Beeswarm'=swarmx(rep(0,length(y)),y)$x,
+       'Tukey'=offsetX(y,method='tukey'),
+       'Tukey +\ndensity'=offsetX(y,method='tukeyDense')
+     )
+     ids <- rep(1:length(offsets), each=length(y))
+     points(unlist(offsets) + ids, rep(y, length(offsets)),
+            pch=21,col='#00000099',bg='#00000033')
+   par(lheight=.8)
+   axis(1, 1:length(offsets), names(offsets),padj=1,
+     mgp=c(0,-.1,0),tcl=-.5,cex.axis=1.1)
+ })
```



And also plot using `boxplot`, `beanplot` and `vioplot` methods:

```
>   x<-rep(names(dat),sapply(dat,length))
>   y<-unlist(lapply(dat,function(x)x/max(abs(x))))
>   par(mfrow=c(4,1), mar=c(6,4.5, 1.2, 0.5),mgp=c(3.3,.75,0),
+     cex.axis=1.2,cex.lab=1.2,cex.main=1.2,las=1)
>   vpPlot(x,y, ylab='',cex=.7, pch=21,
+     col='#00000044',bg='#00000011')
>   boxplot(y~x,main='Boxplot',ylab='')
>   beanplot(y~x,main='Beanplot',ylab='')
>   vioInput<-split(y,x)
>   labs<-names(vioInput)
>   names(vioInput)[1]<-'x'
>   do.call(vioplot,c(vioInput,list(names=labs,col='white')))
>   title(main='Vioplot')
```



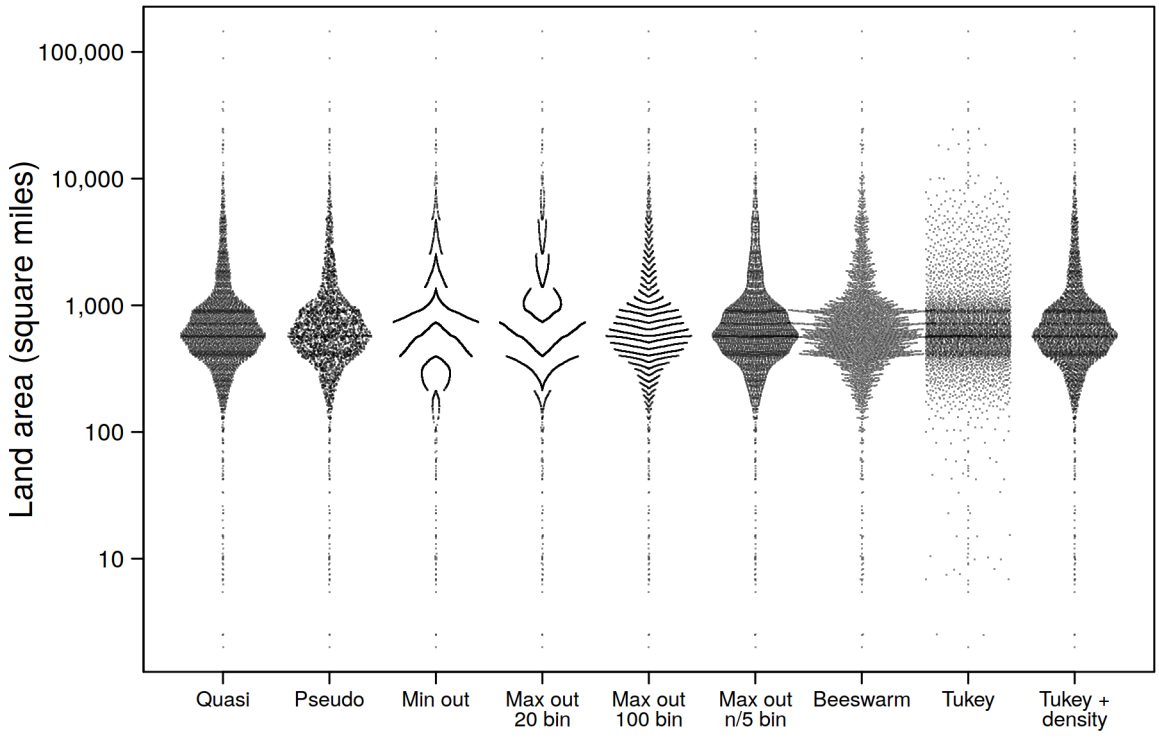
3. Real data

3.1. County data

An example using USA county land area (similar to figure 7 of Tukey and Tukey's "Strips displaying empirical distributions: I. textured dot strips"):

```
>   y<-log10(counties$landArea)
>   offsets <- list(
+     'Quasi'=offsetX(y),  # Default
+     'Pseudo'=offsetX(y, method='pseudorandom',nbins=100),
+     'Min out'=offsetX(y, method='minout',nbins=20),
+     'Max out\n20 bin'=offsetX(y, method='maxout',nbins=20),
+     'Max out\n100 bin'=offsetX(y, method='maxout',nbins=100),
```

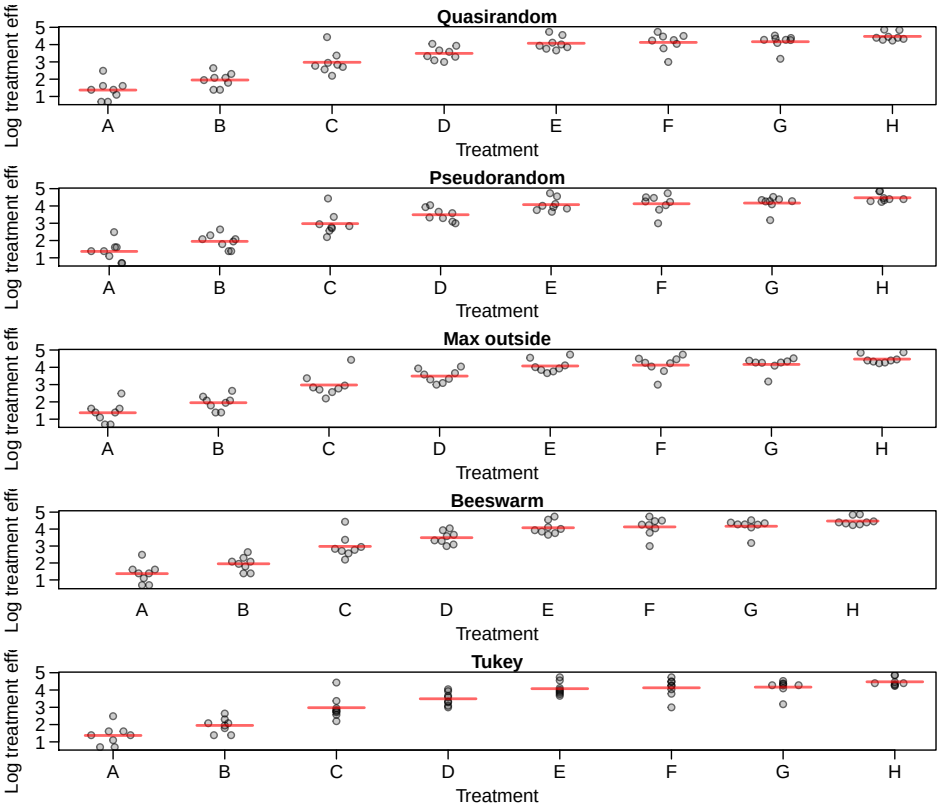
```
+     'Max out\nn/5 bin'=offsetX(y, method='maxout',nbins=round(length(y)/5)),
+     'Beeswarm'=swarmx(rep(0,length(y)),y)$x,
+     'Tukey'=offsetX(y,method='tukey'),
+     'Tukey +\ndensity'=offsetX(y,method='tukeyDense')
+   )
>   ids <- rep(1:length(offsets), each=length(y))
>   #reduce file size by rendering to raster
>   tmpPng<-tempfile(fileext='.png')
>   png(tmpPng,height=1200,width=1800,res=300)
>   par(mar=c(2.5,3.5,.2,0.2))
>   plot(
+     unlist(offsets) + ids, rep(y, length(offsets)),
+     xlab='', xaxt='n', yaxt='n',pch='.',
+     ylab='Land area (square miles)',mgp=c(2.7,1,0),
+     col='#00000077'
+   )
>   par(lheight=.8)
>   axis(1, 1:length(offsets), names(offsets),padj=1,
+     mgp=c(0,-.3,0),tcl=-.3,cex.axis=.65)
>   axis(2, pretty(y), format(10^pretty(y),scientific=FALSE,big.mark=','),
+     mgp=c(0,.5,0),tcl=-.3,las=1,cex.axis=.75)
>   dev.off()
```



3.2. Few data points

An example with few data points (maybe a bit too few for optimal use of this package) using the `OrchardSprays` data from the **datasets** package:

```
>   par(mfrow=c(5,1), mar=c(3.5,3.1, 1.2, 0.5),mgp=c(2.1,.75,0),
+     cex.axis=1.2,cex.lab=1.2,cex.main=1.2,las=1)
>   #simple function to avoid repeating code
>   plotFunc<-function(x,y,offsetXArgs){
+     vpPlot(x,y, ylab='Log treatment effect', pch=21,
+       col='#00000099',bg='#00000033', offsetXArgs=offsetXArgs)
+     title(xlab='Treatment')
+     addMeanLines(x,y)
+   }
>   addMeanLines<-function(x,y,col='#FF000099'){
+     means<-tapply(y,x,mean)
+     segments(
+      1:length(means)-.25,means,1:length(means)+.25,means,
+      col=col,lwd=2
+     )
+   }
>   #quasirandom
>   plotFunc(OrchardSprays$treatment,log(OrchardSprays$decrease),
+     list(width=.2))
>   title(main='Quasirandom')
>   #pseudorandom
>   plotFunc(OrchardSprays$treatment,log(OrchardSprays$decrease),
+     list(method='pseudo',width=.2))
>   title(main='Pseudorandom')
>   #smiley
>   plotFunc(OrchardSprays$treatment,log(OrchardSprays$decrease),
+     list(method='maxout',width=.2))
>   title(main='Max outside')
>   #beeswarm
>   beeInput<-split(log(OrchardSprays$decrease), OrchardSprays$treatment)
>   beeswarm(beeInput,las=1,ylab='Log treatment effect',xlab='Treatment',
+     pch=21, col='#00000099',bg='#00000033', main='Beeswarm')
>   addMeanLines(OrchardSprays$treatment,log(OrchardSprays$decrease))
>   plotFunc(OrchardSprays$treatment,log(OrchardSprays$decrease),
+     list(method='tukey',width=.2))
>   title(main='Tukey')
```

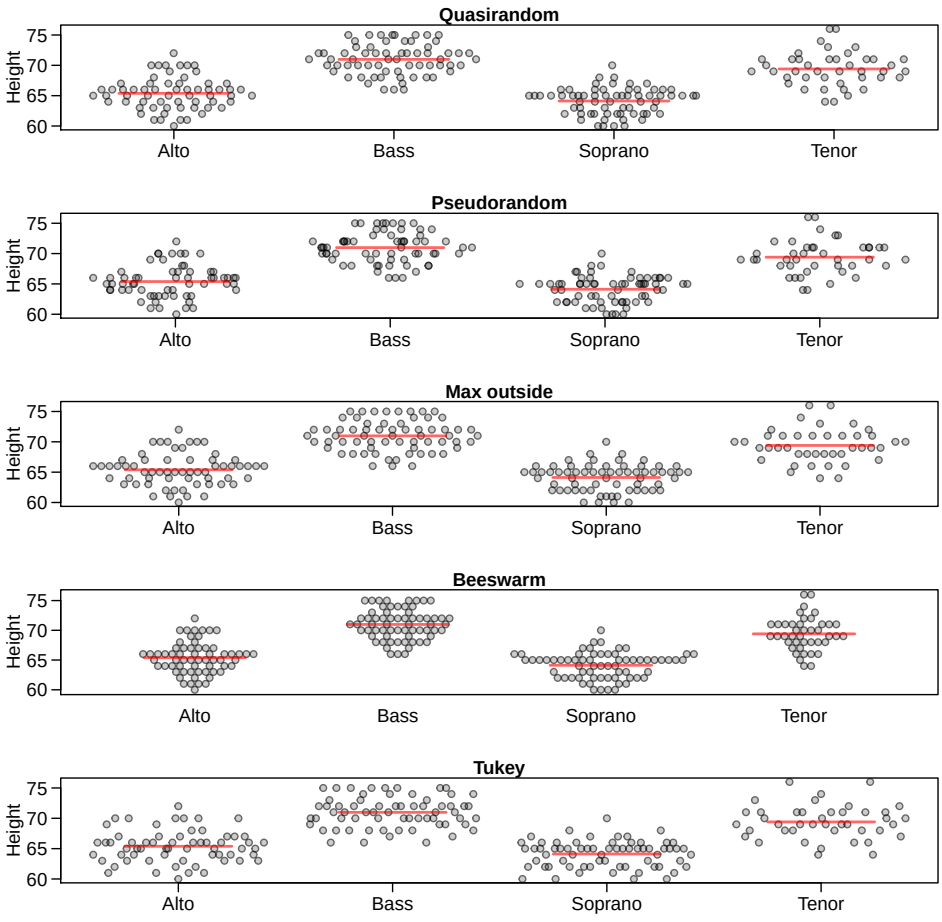


3.3. Discrete data

Data with discrete bins are plotted adequately although other display choices (e.g. multiple barplots) might be better for final publication. For example the `singer` data from the **lattice** package has its data rounded to the nearest inch:

```
>   data('singer',package='lattice')
>   parts<-sub(' [0-9]+$','',singer$voice)
>   par(mfrow=c(5,1), mar=c(3.5,3.1, 1.2, 0.5),mgp=c(2.1,.75,0),
+     cex.axis=1.2,cex.lab=1.2,cex.main=1.2,las=1)
>   #simple function to avoid repeating code
>   plotFunc<-function(x,y,...){
+     vpPlot(x,y, ylab='Height',pch=21,col='#00000099',bg='#00000033',...)
+     addMeanLines(x,y)
+   }
>   #quasirandom
>   plotFunc(parts,singer$height,
+     main='Quasirandom')
>   #pseudorandom
>   plotFunc(parts,singer$height,offsetXArgs=list(method='pseudo'),
+     main='Pseudorandom')
>   #smiley
>   plotFunc(parts,singer$height,offsetXArgs=list(method='maxout'),
```

```
+     main='Max outside')
>   #beeswarm
>   beeInput<-split(singer$height, parts)
>   beeswarm(beeInput,ylab='Height',main='Beeswarm',
+     pch=21, col='#00000099',bg='#00000033')
>   addMeanLines(parts,singer$height)
>   #tukey
>   plotFunc(parts,singer$height,offsetXArgs=list(method='tukey'),
+     main='Tukey')
```

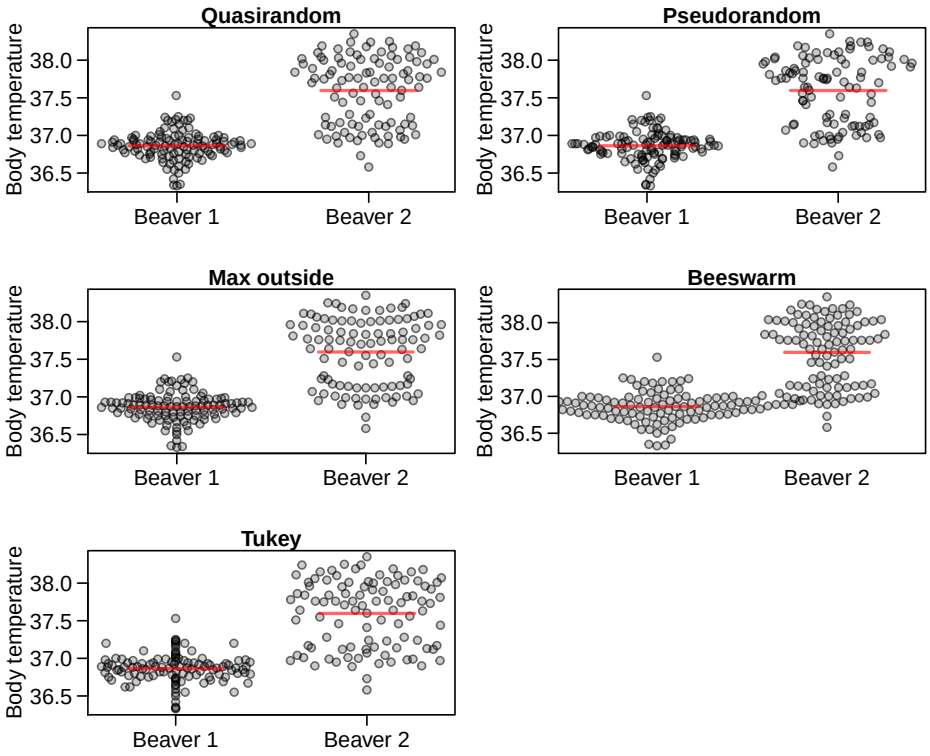


3.4. Moderately sized data

An example with 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)))
>   par(mfrow=c(3,2), mar=c(3.5,4.5, 1.2, 0.5),mgp=c(3,.75,0),
+     cex.axis=1.2,cex.lab=1.2,cex.main=1.2)
>   #simple function to avoid repeating code
```

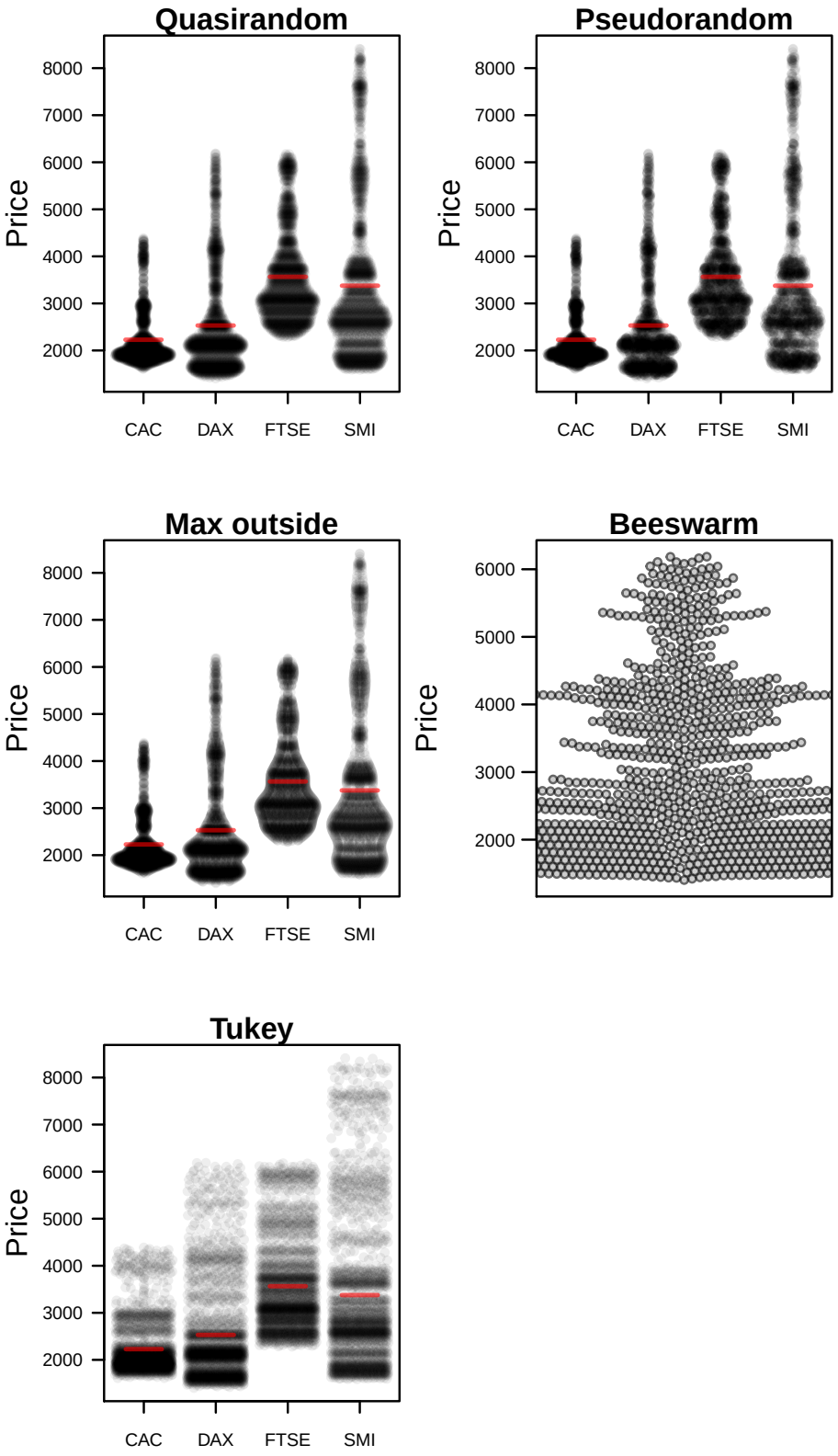
```
>   plotFunc<-function(x,y,...){
+     vpPlot(x,y, las=1, ylab='Body temperature',pch=21,
+       col='#00000099',bg='#00000033',...)
+     addMeanLines(x,y)
+   }
>   #quasirandom
>   plotFunc(x,y,main='Quasirandom')
>   #pseudorandom
>   plotFunc(x,y,offsetXArgs=list(method='pseudo'),main='Pseudorandom')
>   #smiley
>   plotFunc(x,y,offsetXArgs=list(method='maxout'),main='Max outside')
>   #beeswarm
>   beeInput<-split(y,x)
>   beeswarm(beeInput,las=1,ylab='Body temperature',main='Beeswarm',
+     pch=21, col='#00000099',bg='#00000033')
>   addMeanLines(x,y)
>   #tukey
>   plotFunc(x,y,offsetXArgs=list(method='tukey'),main='Tukey')
```



3.5. Larger data

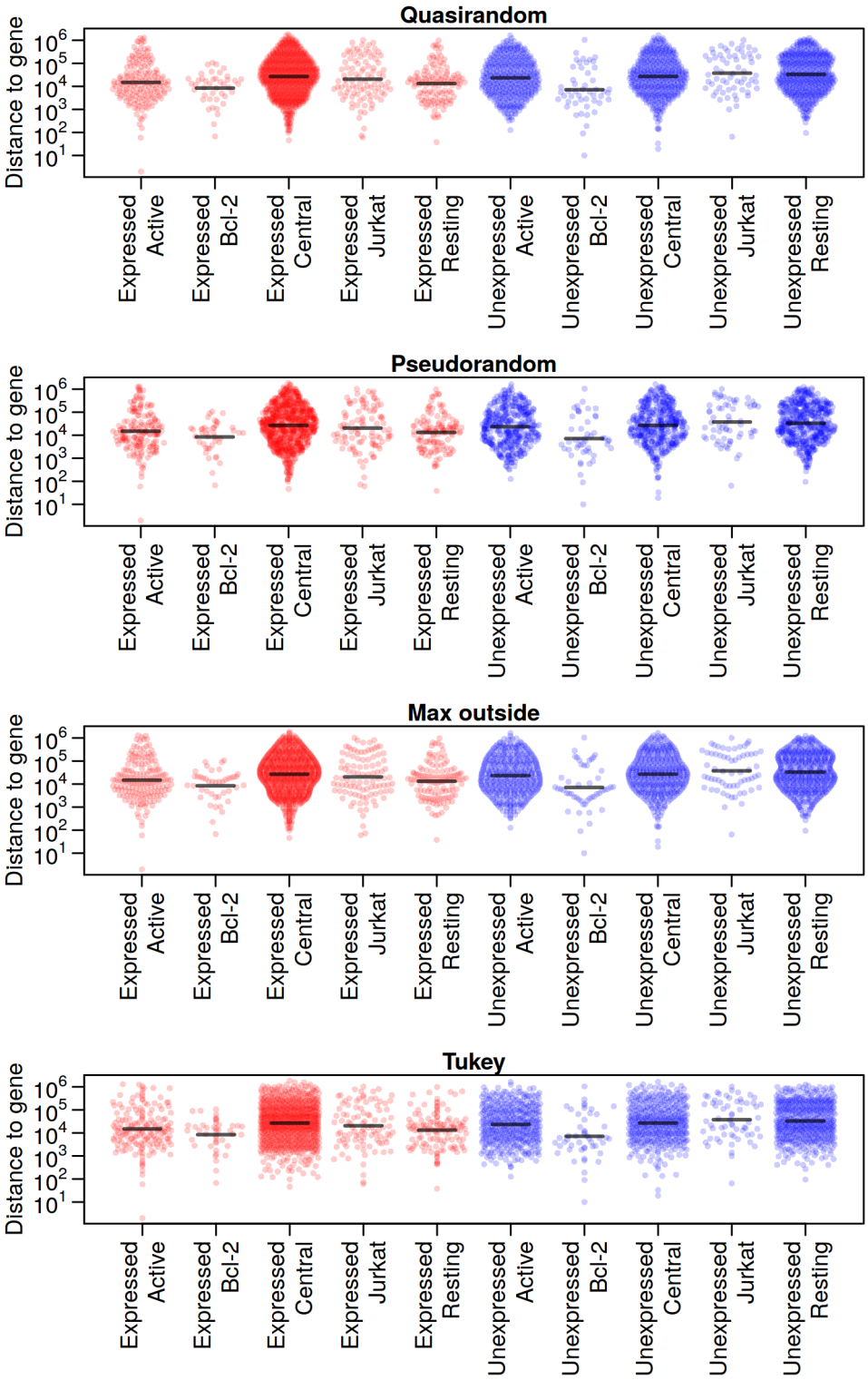
An example using the `EuStockMarkets` data from the **datasets** package. Here `beeswarm` takes too long to run and generates overlap between entries and so only a single group is displayed:

```
>   y<-as.vector(EuStockMarkets)
>   x<-rep(colnames(EuStockMarkets), each=nrow(EuStockMarkets))
>   par(mfrow=c(3,2), mar=c(4,4.3, 1.2, 0.5),mgp=c(3.3,.75,0),
+     cex.axis=1.2,cex.lab=1.2,cex.main=1.2,las=1)
>   #simple function to avoid repeating code
>   plotFunc<-function(x,y,...){
+     vpPlot(x,y, ylab='Price',cex=.7,cex.axis=.7,
+       mgp=c(2.5,.75,0),tcl=-.4, pch=21,
+       col='#00000011',bg='#00000011',...)
+     addMeanLines(x,y)
+   }
>   #quasirandom
>   plotFunc(x,y,main='Quasirandom')
>   #pseudorandom
>   plotFunc(x,y,offsetXArgs=list(method='pseudo'),main='Pseudorandom')
>   #smiley
>   plotFunc(x,y,offsetXArgs=list(method='maxout'),main='Max outside')
>   #beeswarm
>   #beeInput<-split(y,x)
>   beeswarm(EuStockMarkets[,'DAX',drop=FALSE],cex=.7, ylab='Price',
+     main='Beeswarm',pch=21, col='#00000099',bg='#00000033',cex.axis=.7)
>   #tukey
>   plotFunc(x,y,offsetXArgs=list(method='tukey'),main='Tukey')
```


Quasirandom
Pseudorandom
Max outside
Beeswarm
Tukey
Price
8000
7000
6000
5000
4000
3000
2000
CAC
DAX
FTSE
SMI

Another example using the HIV `integrations` data from this package. Here `beeswarm` takes too long to run and is omitted:

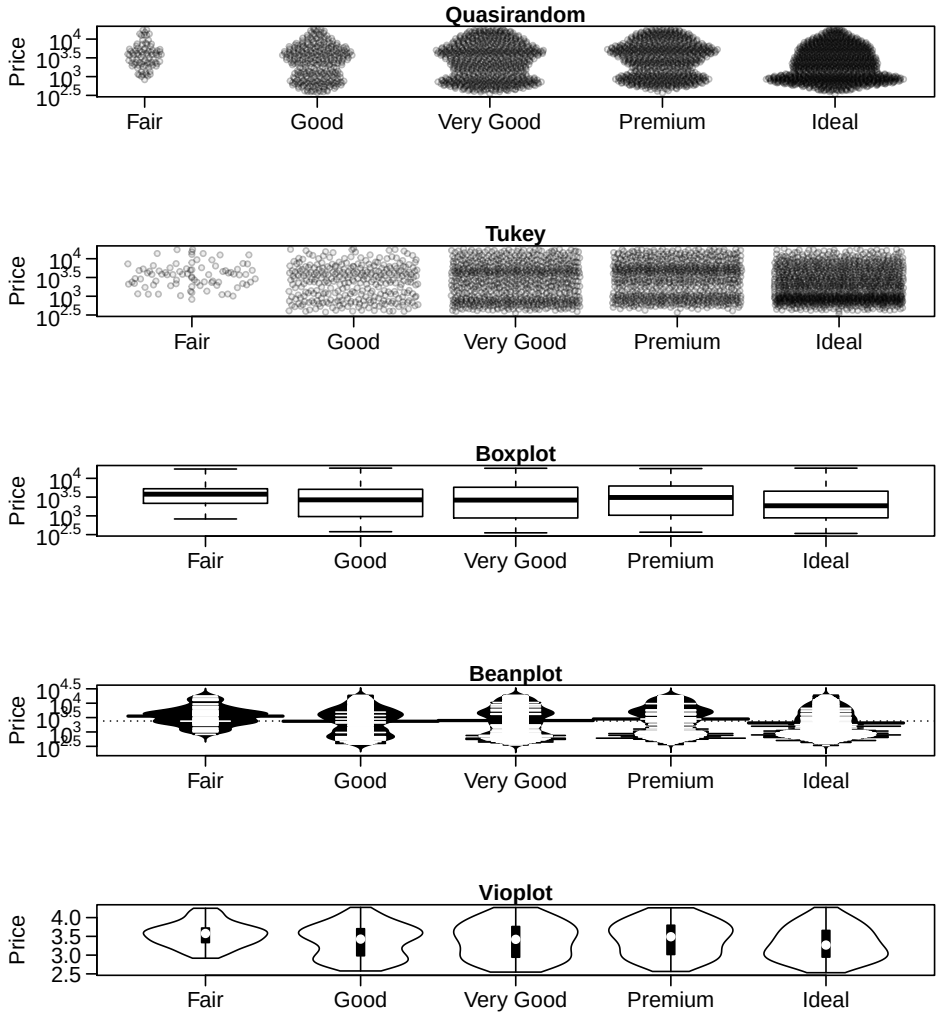
```
>   ints<-integrations[integrations$nearestGene>0,]
>   y<-log10(ints$nearestGene)
>   x<-paste(ints$latent,ints$study,sep='\n')
>   #reduce file size by rendering to raster
>   tmpPng<-tempfile(fileext='.png')
>   png(tmpPng,height=2400,width=1500,res=300)
>   par(mfrow=c(4,1), mar=c(7.5,3.5, 1.2, 0.5),mgp=c(2.5,.75,0),
+     cex.axis=1.2,cex.lab=1.2,cex.main=1.2)
>   #simple function to avoid repeating code
>   plotFunc<-function(x,y,...){
+     cols<-ifelse(grepl('Expressed',x),'#FF000033','#0000FF33')
+     vpPlot(x,y,las=2, ylab='Distance to gene',cex=.7,yaxt='n',
+       pch=21, col=NA,bg=cols,lheight=.4,...)
+     prettyY<-pretty(y)
+     yLabs<-sapply(prettyY,function(x)as.expression(bquote(10^.(x))))
+     axis(2,prettyY,yLabs,las=1)
+     addMeanLines(x,y,col='#000000AA')
+   }
>   #quasirandom
>   plotFunc(x,y,main='Quasirandom')
>   #pseudorandom
>   plotFunc(x,y,offsetXArgs=list(method='pseudo'),main='Pseudorandom')
>   #smiley
>   plotFunc(x,y,offsetXArgs=list(method='maxout'),main='Max outside')
>   #tukey
>   plotFunc(x,y,offsetXArgs=list(method='tukey'),main='Tukey')
>   #beeswarm
>   #beeInput<-split(y,x)
>   #beeswarm(beeInput,las=1,cex=.7, ylab='Log distance to gene',
>     #main='Beeswarm',pch=21, col='#00000099',bg='#00000033')
>   #addMeanLines(x,y)
>   dev.off()
```



Another example using 3000 entries of the `diamonds` data from the **ggplot2** package. Here `beeswarm` takes too long to run and is omitted:

```
>   select<-sample(1:nrow(ggplot2::diamonds),3000)
```

```
> y<-unlist(log10(ggplot2::diamonds[select,'price']))
> x<-unlist(ggplot2::diamonds[select,'cut'])
> par(mfrow=c(5,1), mar=c(6,4.5, 1.2, 0.5),mgp=c(3.3,.75,0),
+   cex.axis=1.2,cex.lab=1.2,cex.main=1.2,las=1)
> #simple function to avoid repeating code
> prettyYAxis<-function(y){
+   prettyY<-pretty(y)
+   yLabs<-sapply(prettyY,function(x)as.expression(bquote(10^.(x))))
+   axis(2,prettyY,yLabs)
+ }
> #quasirandom
> vpPlot(x,y,offsetXArgs=list(varwidth=TRUE),
+   ylab='Price',cex=.7,pch=21, col='#00000044',
+   bg='#00000011',yaxt='n',main='Quasirandom')
> prettyYAxis(y)
> #tukey
> vpPlot(x,y,offsetXArgs=list(method='tukey'),
+   ylab='Price',cex=.7,pch=21, col='#00000044',
+   bg='#00000011',yaxt='n',main='Tukey')
> prettyYAxis(y)
> #boxplot
> boxplot(y~x,main='Boxplot',ylab='Price',yaxt='n')
> prettyYAxis(y)
> #beanplot
> beanplot(y~x,main='Beanplot',ylab='Price',yaxt='n')
> prettyYAxis(y)
> vioInput<-split(y,x)
> labs<-names(vioInput)
> names(vioInput)[1]<-'x'
> #vioplot
> do.call(vioplot,c(vioInput,list(names=labs,col='white')))
> title(ylab='Price', main='Vioplot')
```



Affiliation:

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