Grouped one dimensional data comparison (violinpoint version 0.3.0)

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 **violinpoint** 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.

beanplot 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.

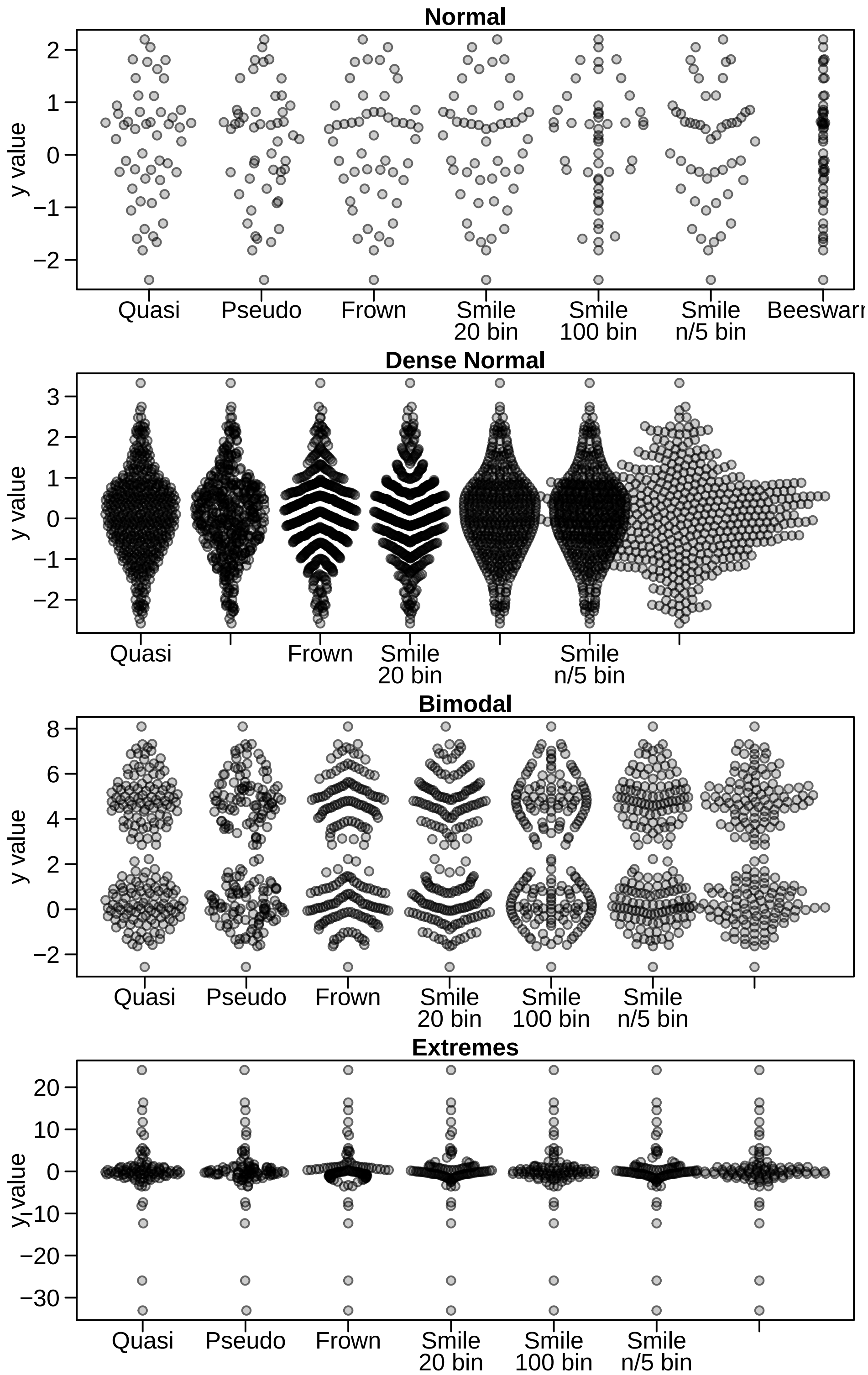
2. Simulated data

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

```
> library(beeswarm)
> library(violinpoint)
> 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")
```

Here, we plot the data using quasirandom, pseudorandom, alternating 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]]
+     offsets <- list(
+       'Quasi'=offsetX(y),  # Default
+       'Pseudo'=offsetX(y, method='pseudorandom',nbins=100),
+       'Frown'=offsetX(y, method='frowney',nbins=20),
+       'Smile\n20 bin'=offsetX(y, method='smiley',nbins=20),
+       'Smile\n100 bin'=offsetX(y, method='smiley',nbins=100),
+       'Smile\nn/5 bin'=offsetX(y, method='smiley',nbins=round(length(y)/5)),
+       'Beeswarm'=swarmx(rep(0,length(y)),y)$x
+     )
+     ids <- rep(1:length(offsets), each=length(y))
+     plot(unlist(offsets) + ids, rep(y, length(offsets)),
+         xlab='', xaxt='n', pch=21,las=1,main=label, ylab='y value',
+         col='#00000099',bg='#00000033')
+   par(lheight=.8)
+   axis(1, 1:length(offsets), names(offsets),padj=1,mgp=c(0,-.3,0),tcl=-.5)
+ })
```


Normal
y value
Quasi
Pseudo
Frown
Smile 20 bin
Smile 100 bin
Smile n/5 bin
Beeswarm
Dense Normal
y value
Quasi
Frown
Smile 20 bin
Smile n/5 bin
Bimodal
y value
Quasi
Pseudo
Frown
Smile 20 bin
Smile 100 bin
Smile n/5 bin
Extremes
y value
Quasi
Pseudo
Frown
Smile 20 bin
Smile 100 bin
Smile n/5 bin

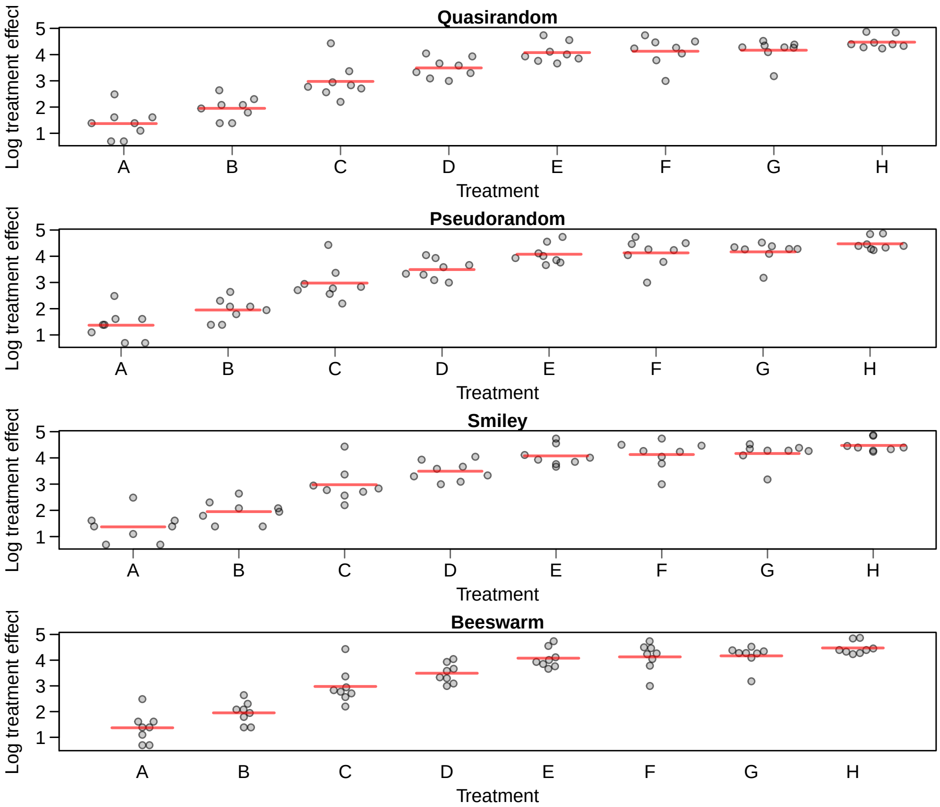
3. Real data

3.1. 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(4,1), 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)
> #simple function to avoid repeating code
> plotFunc<-function(x,y,offsetXArgs){
+   vpPlot(
+    x,y,
+    las=1, ylab='Log treatment effect',
+    pch=21, col='#00000099',bg='#00000033',
+    offsetXArgs=offsetXArgs
+   )
+   title(xlab='Treatment')
+   addMeanLines(x,y)
+ }
> addMeanLines<-function(x,y){
+   means<-tapply(y,x,mean)
+   segments(
+    1:length(means)-.3,means,1:length(means)+.3,means,
+    col='#FF000099',lwd=2
+   )
+ }
> #quasirandom
> plotFunc(OrchardSprays$treatment,log(OrchardSprays$decrease),NULL)
> title(main='Quasirandom')
> #pseudorandom
> plotFunc(
+   OrchardSprays$treatment,log(OrchardSprays$decrease),
+   list(method='pseudo')
+ )
> title(main='Pseudorandom')
> #smiley
> plotFunc(
+   OrchardSprays$treatment,log(OrchardSprays$decrease),
+   list(method='smiley',nbins=5)
+ )
> title(main='Smiley')
> #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)
+    )
```

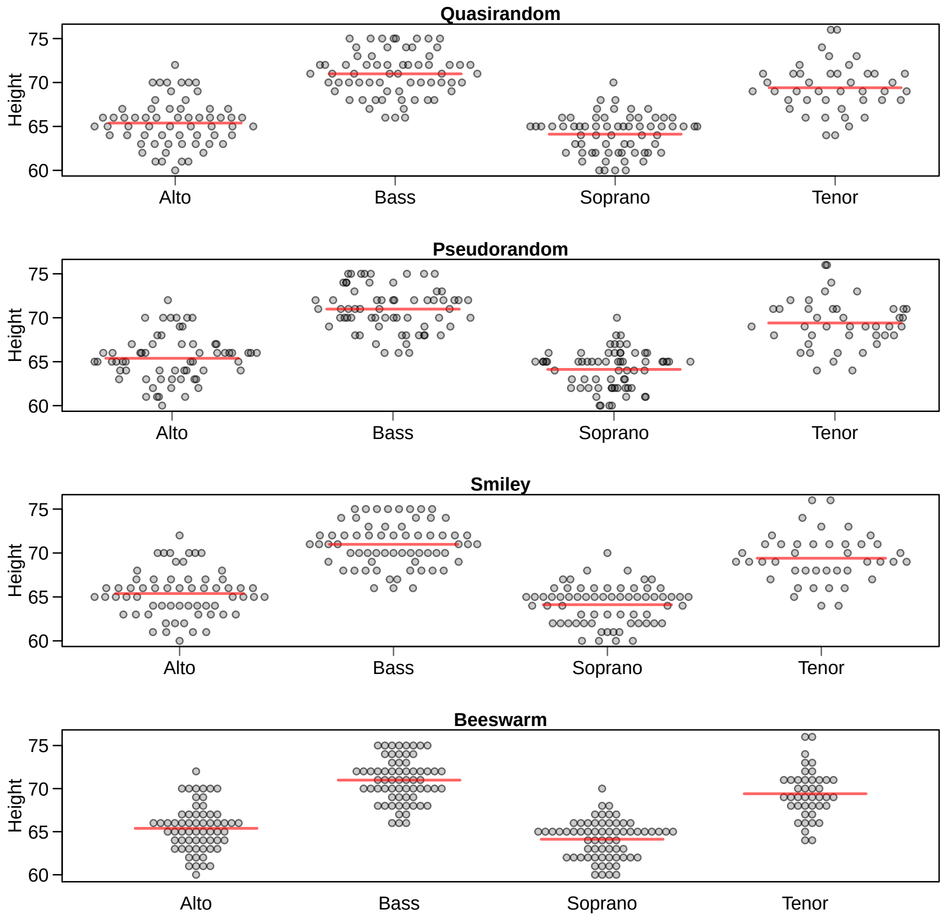


3.2. Discrete data

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(4,1), 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)
>    #simple function to avoid repeating code
>    plotFunc<-function(x,y,offsetXArgs){
```

```
+     vpPlot(
+      x,y,
+      las=1, ylab='Height',
+      pch=21, col='#00000099',bg='#00000033',
+      offsetXArgs=offsetXArgs
+     )
+     addMeanLines(x,y)
+   }
>   #quasirandom
>   plotFunc(parts,singer$height,NULL)
>   title(main='Quasirandom')
>   #pseudorandom
>   plotFunc(
+     parts,singer$height,
+     list(method='pseudo')
+   )
>   title(main='Pseudorandom')
>   #smiley
>   plotFunc(
+     parts,singer$height,
+     list(method='smiley',nbins=10)
+   )
>   title(main='Smiley')
>   #beeswarm
>   beeInput<-split(singer$height, parts)
>   beeswarm(
+     beeInput,las=1,
+     ylab='Height',main='Beeswarm',
+     pch=21, col='#00000099',bg='#00000033',
+   )
>   addMeanLines(parts,singer$height)
```

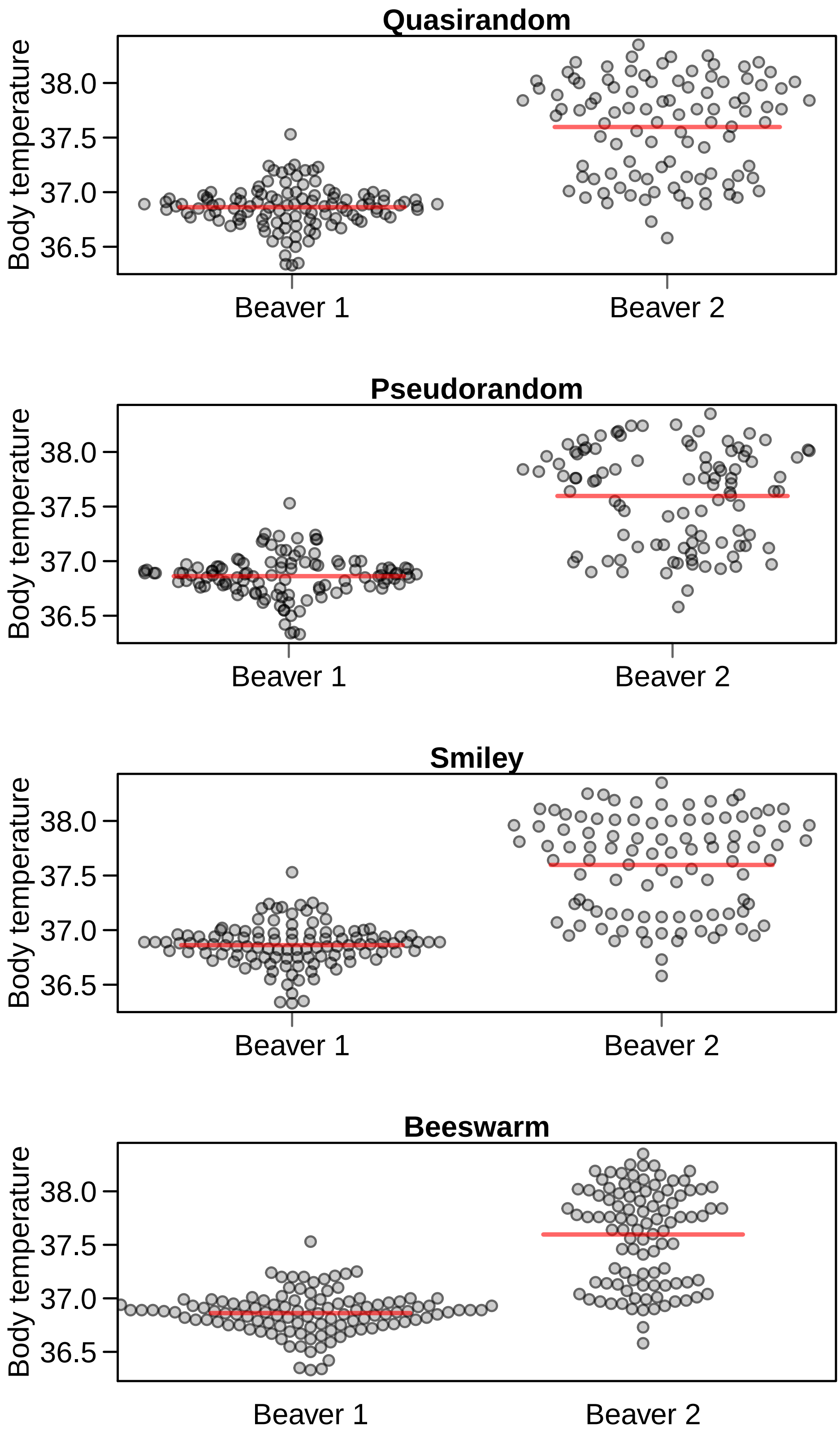


3.3. Moderately sized data

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(4,1), mar=c(3.5,4, 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,offsetXArgs){
+     vpPlot(
+      x,y,
+      las=1, ylab='Body temperature',
+      pch=21, col='#00000099',bg='#00000033',
+      offsetXArgs=offsetXArgs
+     )
```

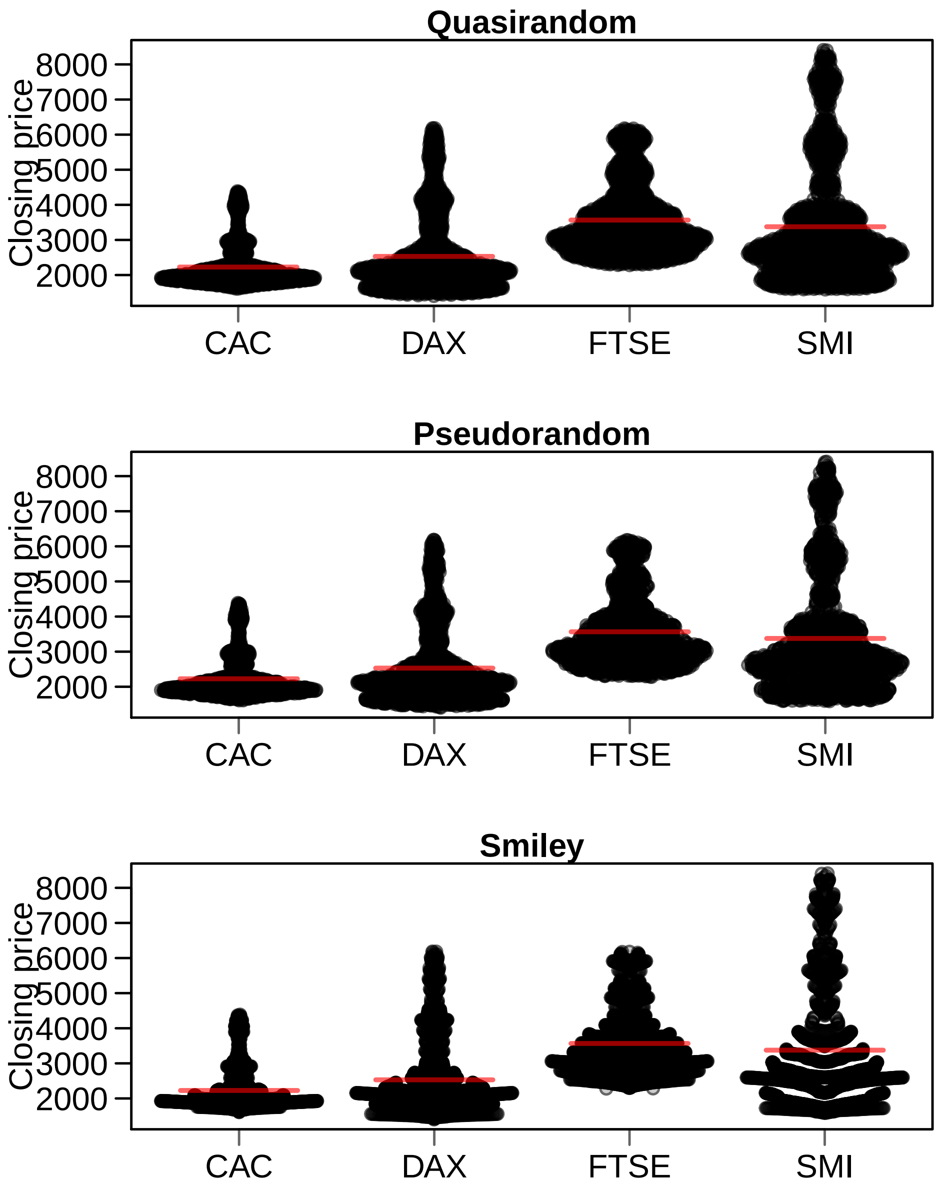
```
+     addMeanLines(x,y)
+   }
>   #quasirandom
>   plotFunc(x,y,NULL)
>   title(main='Quasirandom')
>   #pseudorandom
>   plotFunc(
+     x,y,
+     list(method='pseudo')
+   )
>   title(main='Pseudorandom')
>   #smiley
>   plotFunc(
+     x,y,
+     list(method='smiley',nbins=20)
+   )
>   title(main='Smiley')
>   #beeswarm
>   beeInput<-split(y,x)
>   beeswarm(
+     beeInput,las=1,
+     ylab='Body temperature',main='Beeswarm',
+     pch=21, col='#00000099',bg='#00000033',
+   )
>   addMeanLines(x,y)
```


Quasirandom
Body temperature
38.0
37.5
37.0
36.5
Beaver 1
Beaver 2
Pseudorandom
Body temperature
38.0
37.5
37.0
36.5
Beaver 1
Beaver 2
Smiley
Body temperature
38.0
37.5
37.0
36.5
Beaver 1
Beaver 2
Beeswarm
Body temperature
38.0
37.5
37.0
36.5
Beaver 1
Beaver 2

3.4. Larger data

An example with using the `EuStockMarkets` data from the **datasets** package:

```
> y<-as.vector(EuStockMarkets)
> x<-rep(colnames(EuStockMarkets), each=nrow(EuStockMarkets))
> par(mfrow=c(4,1), mar=c(3.5,4, 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,offsetXArgs){
+   vpPlot(
+    x,y,
+    las=1, ylab='Closing price',
+    pch=21, col='#00000099',bg='#00000033',
+    offsetXArgs=offsetXArgs
+   )
+   addMeanLines(x,y)
+ }
> #quasirandom
> plotFunc(x,y,NULL)
> title(main='Quasirandom')
> #pseudorandom
> plotFunc(
+   x,y,
+   list(method='pseudo')
+ )
> title(main='Pseudorandom')
> #smiley
> plotFunc(
+   x,y,
+   list(method='smiley',nbins=20)
+ )
> title(main='Smiley')
> #beeswarm
> beeInput<-split(y,x)
> #beeswarm(
>   #beeInput,las=1,
>   #ylab='Closing price',main='Beeswarm',
>   #pch=21, col='#00000099',bg='#00000033',
> #)
> #addMeanLines(x,y)
```


Quasirandom
Pseudorandom
Smiley
Closing price
2000
3000
4000
5000
6000
7000
8000
CAC
DAX
FTSE
SMI

Affiliation:

Github: http://github.com/sherrillmix/violinpoint