ggbeeswarm package usage example (version 0.5.1)

Erik Clarke [aut, cre], Scott Sherrill-Mix [aut]

Abstract

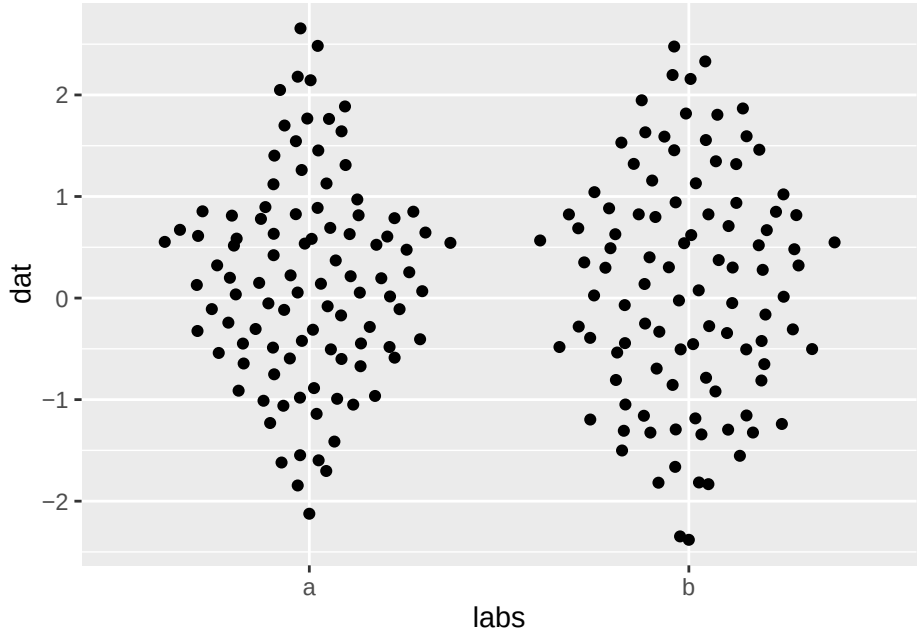
This is a collection of examples of usage for the **ggbeeswarm** package.

Keywords: visualization, display, one dimensional, grouped, groups, violin, scatter, points, quasirandom, beeswarm, van der Corput, beeswarm, ggplot, ggplot2.

1. The basics

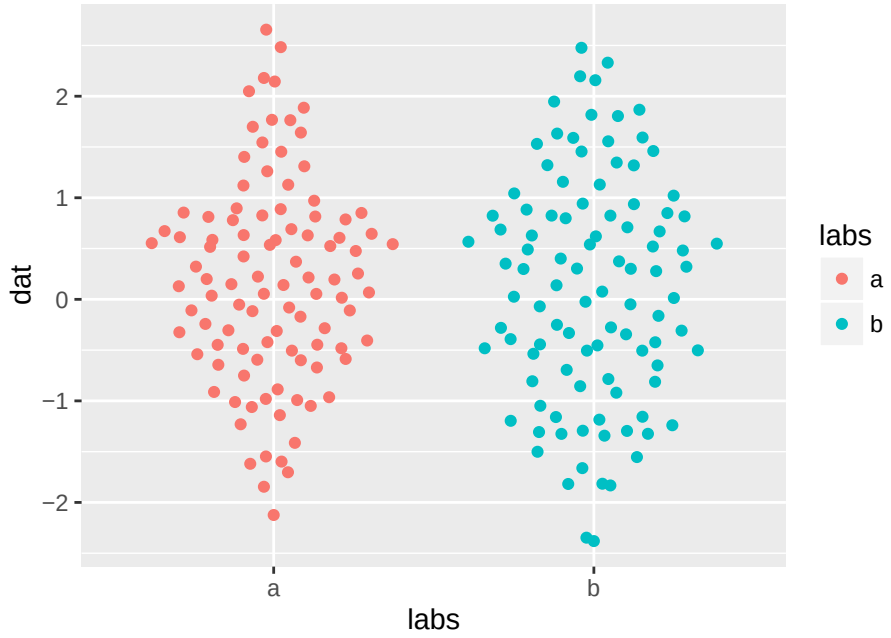
This is the simplest example of using `geom_quasirandom` to generate violin scatter plots:

```
>   library(ggbeeswarm)
>   set.seed(12345)
>   n<-100
>   dat<-rnorm(n*2)
>   labs<-rep(c('a','b'),n)
>   ggplot(mapping=aes(labs, dat)) + geom_quasirandom()
```



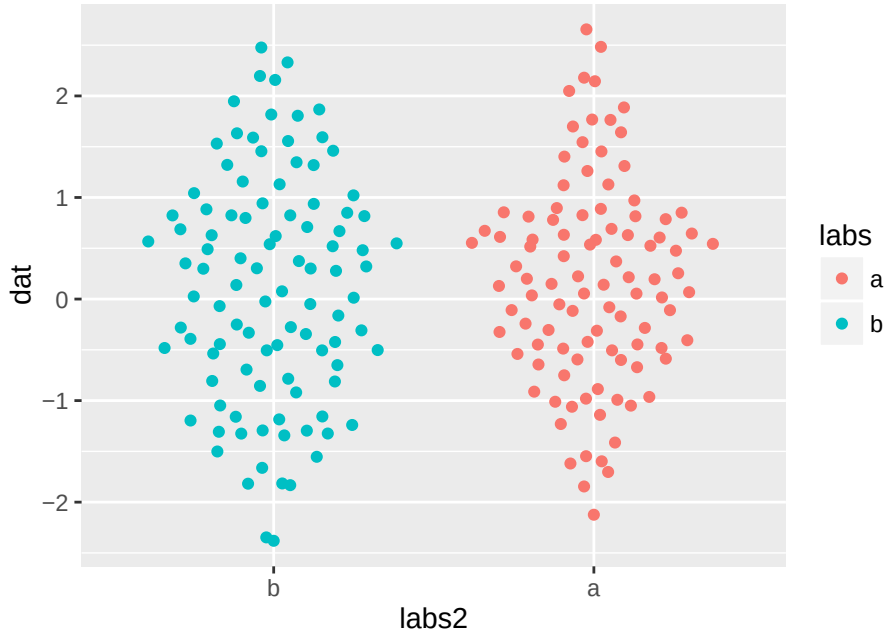
Normal `ggplot` options can be used:

```
> ggplot(mapping=aes(labs, dat)) + geom_quasirandom(aes(color=labs))
```



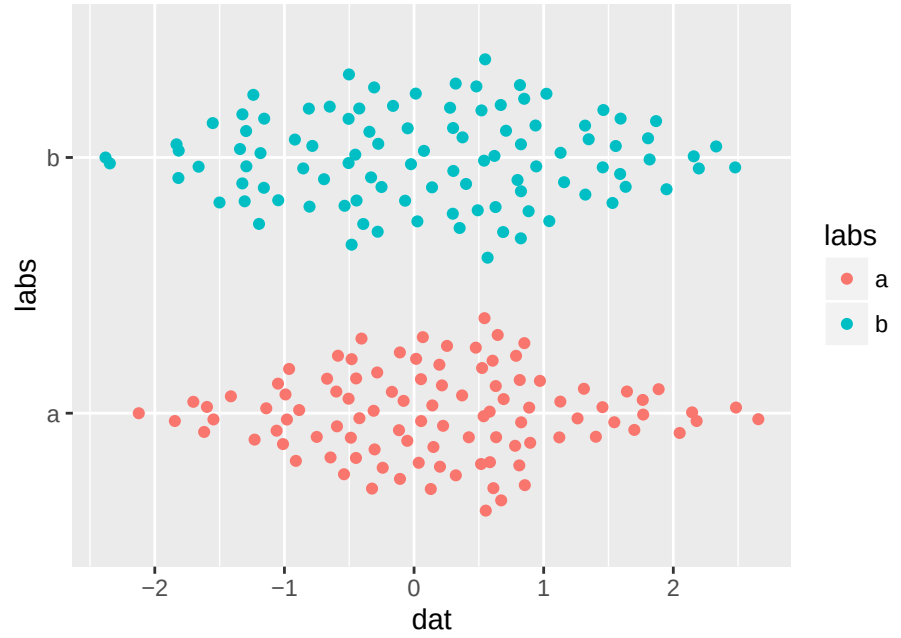
Factors can be used to generate custom group orderings:

```
> labs2<-factor(labs,levels=c('b','a'))
> ggplot(mapping=aes(labs2, dat)) + geom_quasirandom(aes(color=labs))
```



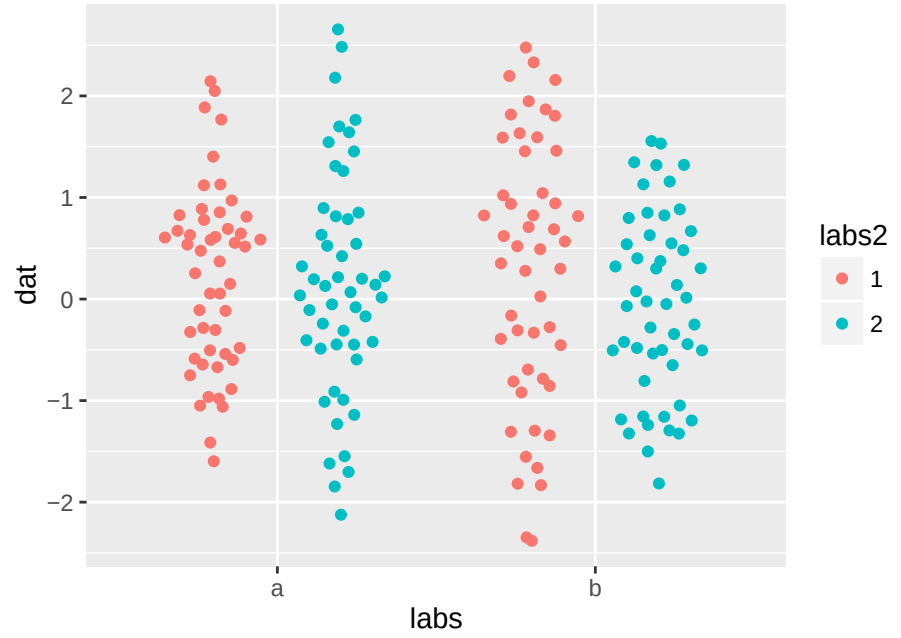
The axes can also be switched with a categorical y-axis:

```
>   ggplot(mapping=aes(dat,labs)) + geom_quasirandom(aes(color=labs))
```



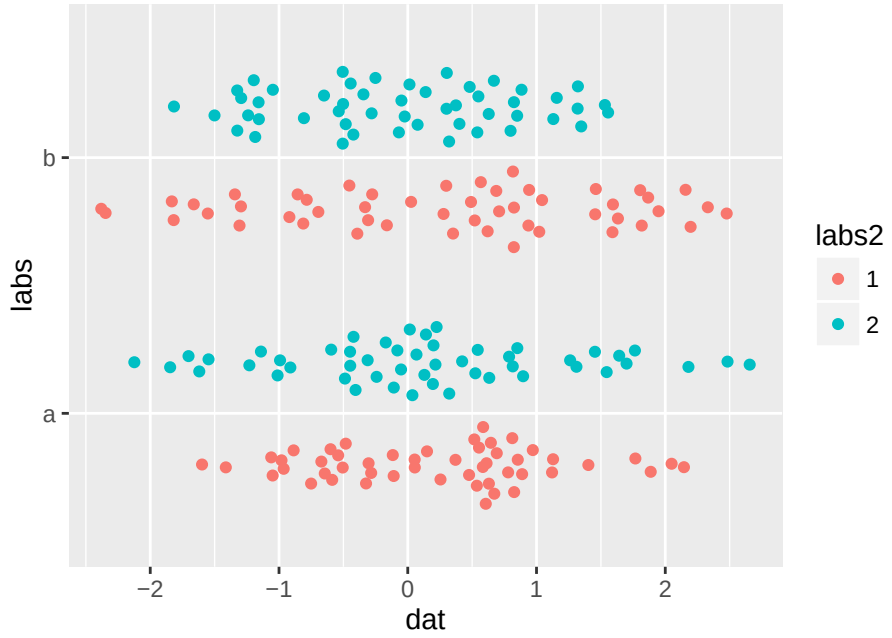
And dodging can be used to compare within groups:

```
>   labs2<-factor(rep(1:2,each=n))
>   ggplot(mapping=aes(labs,dat,color=labs2)) + geom_quasirandom(dodge.width=.8)
```



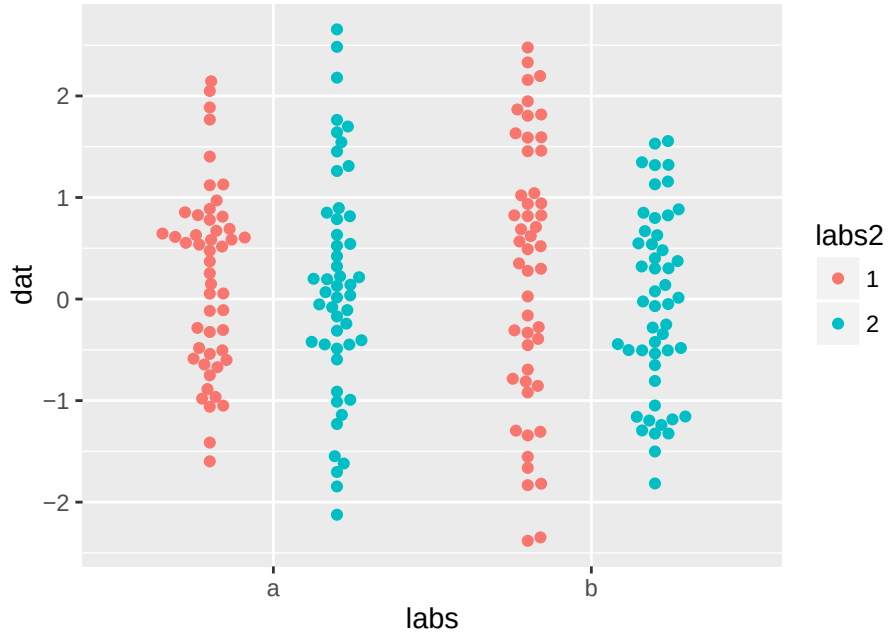
Or on the y-axis:

```
> labs2<-factor(rep(1:2,each=n))
> ggplot(mapping=aes(dat,labs,color=labs2)) + geom_quasirandom(dodge.width=.8)
```

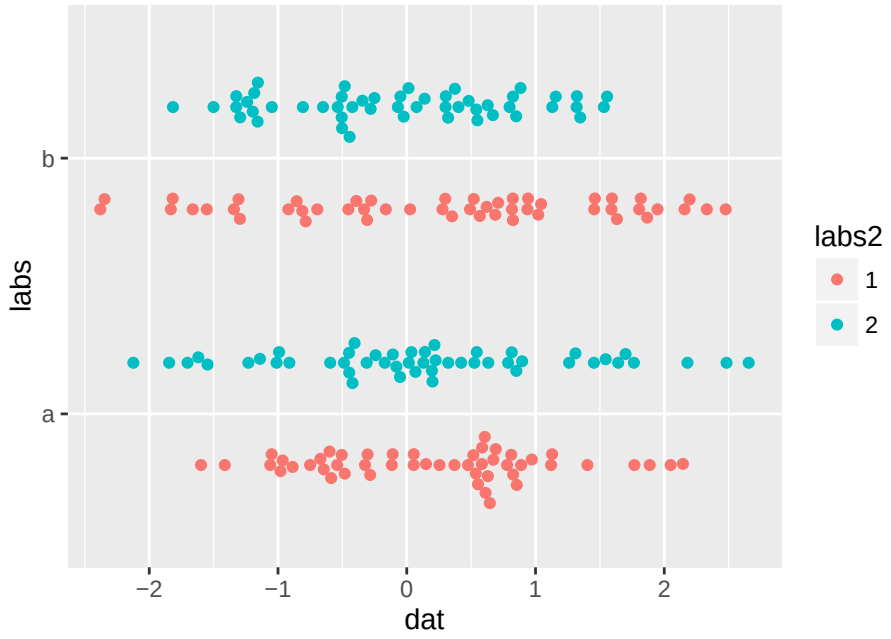


And with `geom_beeswarm`:

```
> ggplot(mapping=aes(labs,dat,color=labs2)) + geom_beeswarm(dodge.width=.8)
```




```
> ggplot(mapping=aes(dat,labs,color=labs2)) + geom_beeswarm(dodge.width=.8)
```



2. Options

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

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.

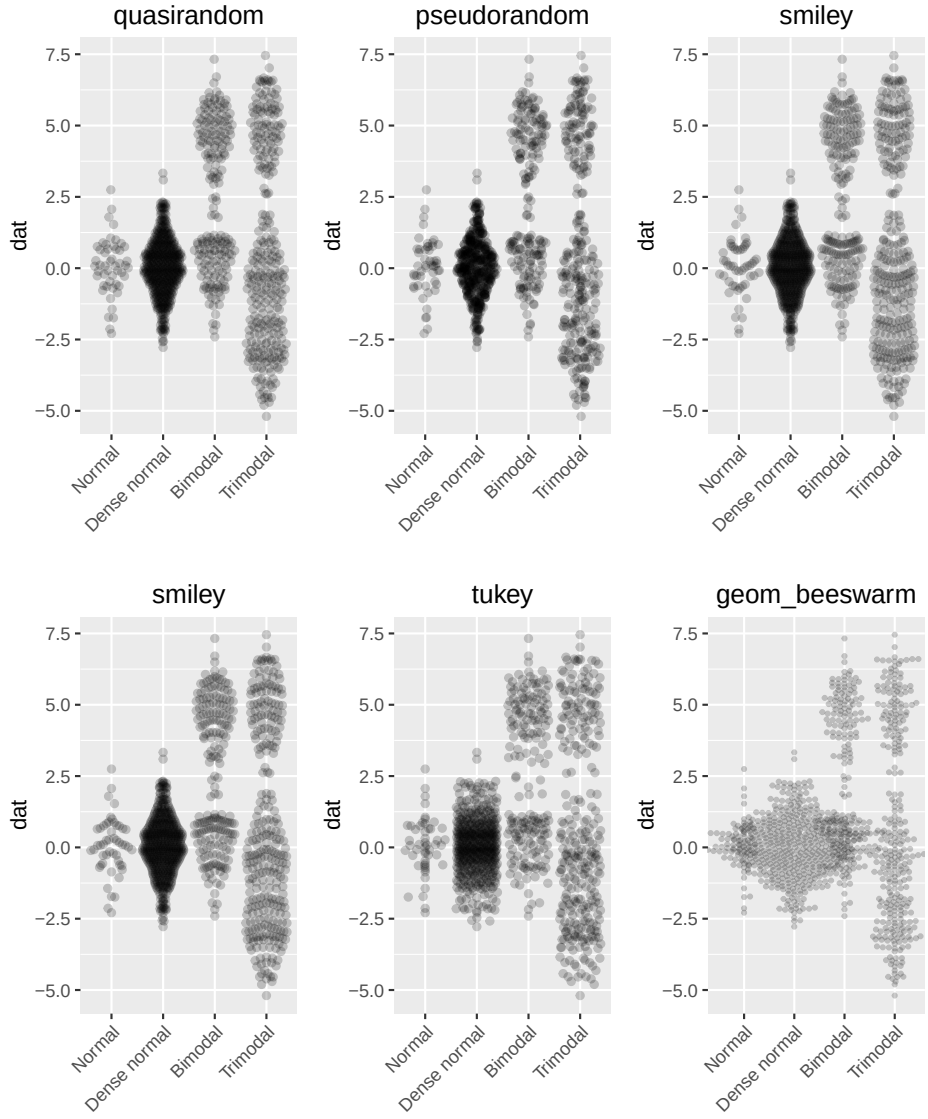
quasirandom The kernel density is estimated then points are distributed quasirandomly using the von der Corput sequence within the density estimate for a given bin. Selection of an appropriate number of bins does not greatly affect appearance and position does not depend on plotting parameters.

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.

The first four options are included within `geom_quasirandom` using the `method=` argument and beeswarm plots are generated with `geom_beeswarm`:

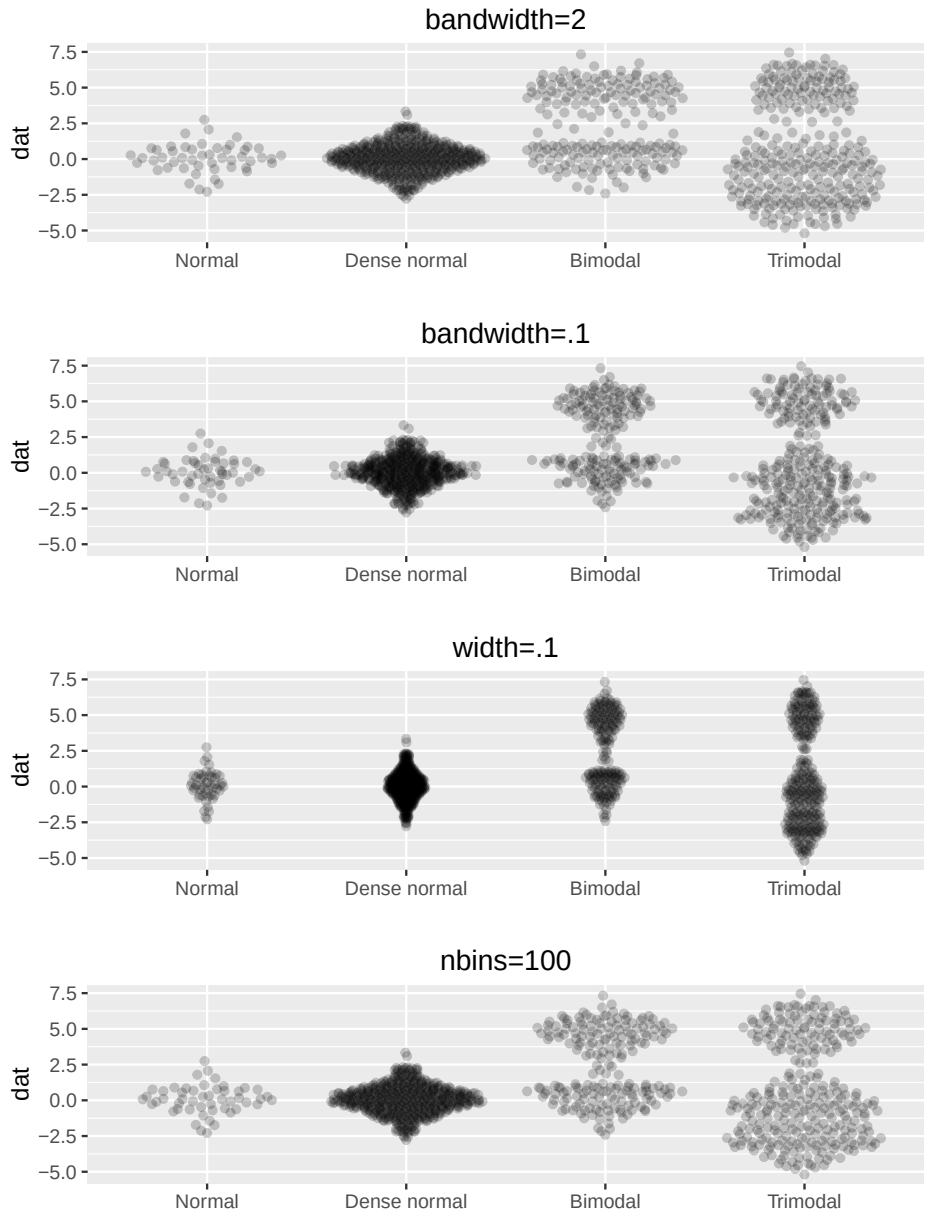
```
>   library(gridExtra)
>   dat <- list(
+     'Normal'=rnorm(50),
+     'Dense normal'= rnorm(500),
+     'Bimodal'=c(rnorm(100), rnorm(100,5)),
+     'Trimodal'=c(rnorm(100), rnorm(100,5),rnorm(100,-3))
+   )
>   labs<-rep(names(dat),sapply(dat,length))
>   labs<-factor(labs,levels=unique(labs))
>   dat<-unlist(dat)
>   p1<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(alpha=.2) +
+     ggtitle('quasirandom') + labs(x='') +
+     theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
>   p2<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='pseudorandom',alpha=.2) +
+     ggtitle('pseudorandom') + labs(x='') +
+     theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
>   p3<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='smiley',alpha=.2) +
+     ggtitle('smiley') + labs(x='') +
+     theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
>   p4<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='frowney',alpha=.2) +
+     ggtitle('smiley') + labs(x='') +
+     theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
>   p5<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='tukey',alpha=.2) +
+     ggtitle('tukey') + labs(x='') +
+     theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
>   p6<-ggplot(mapping=aes(labs, dat)) +
+     geom_beeswarm(alpha=.2,cex=8,size=.75) +
+     ggtitle('geom_beeswarm') + labs(x='') +
+     theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
>   grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)
```



`quasirandom` calls `vipor::offsetX` which calls `stats::density` to compute kernel density estimates. The tightness of the fit can be adjusted with the `bandwidth` 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 but is useful for non-quasirandom methods:

```
>   library(gridExtra)
>   p1<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(bandwidth=2,alpha=.2) +
+     ggtitle('bandwidth=2') + labs(x='')
>   p2<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(bandwidth=.1,alpha=.2) +
+     ggtitle('bandwidth=.1') + labs(x='')
>   p3<-ggplot(mapping=aes(labs, dat)) +
```

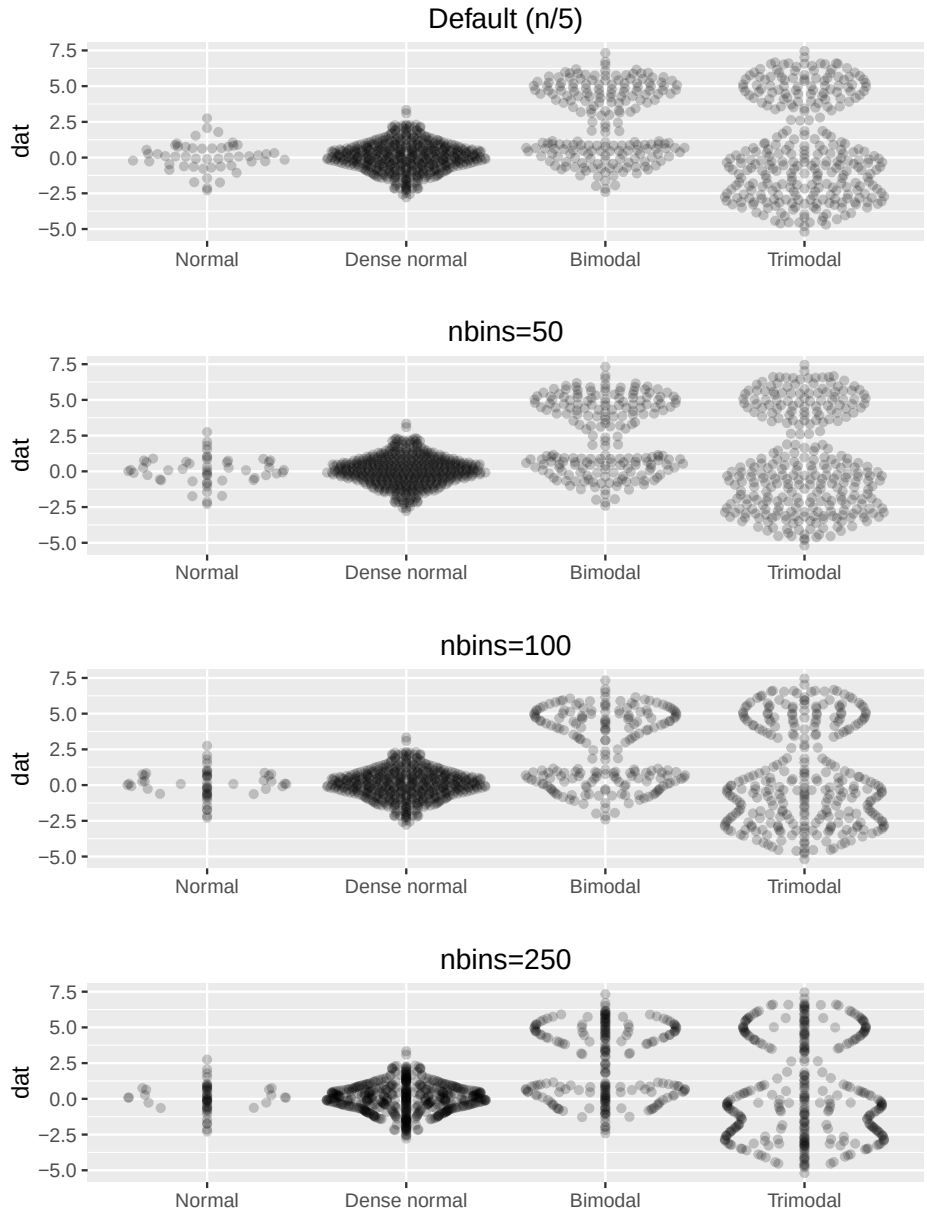
```
+     geom_quasirandom(width=.1,alpha=.2) +
+     ggtitle('width=.1') + labs(x='')
>   p4<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(nbins=100,alpha=.2) +
+     ggtitle('nbins=100') + labs(x='')
>   grid.arrange(p1, p2, p3, p4, ncol=1)
```



The `frowney` or `smiley` methods are sensitive to the number of bins so the argument `nbins` is more useful/necessary with them:

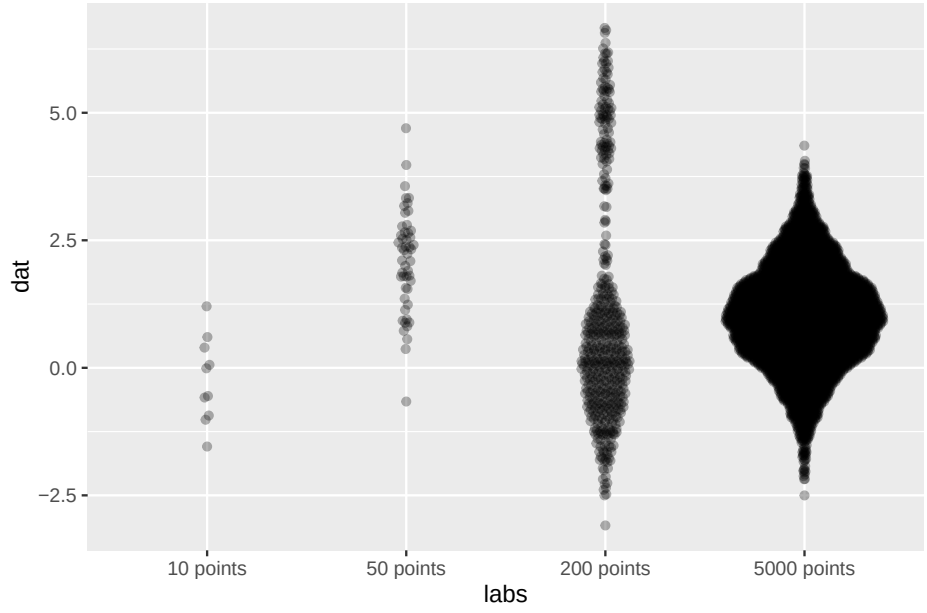
```
>   p1<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='smiley',alpha=.2) +
```

```
+     ggtitle('Default (n/5)') + labs(x='')
>   p2<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='smiley',nbins=50,alpha=.2) +
+     ggtitle('nbins=50') + labs(x='')
>   p3<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='smiley',nbins=100,alpha=.2) +
+     ggtitle('nbins=100') + labs(x='')
>   p4<-ggplot(mapping=aes(labs, dat)) +
+     geom_quasirandom(method='smiley',nbins=250,alpha=.2) +
+     ggtitle('nbins=250') + labs(x='')
>   grid.arrange(p1, p2, p3, p4, ncol=1)
```



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`):

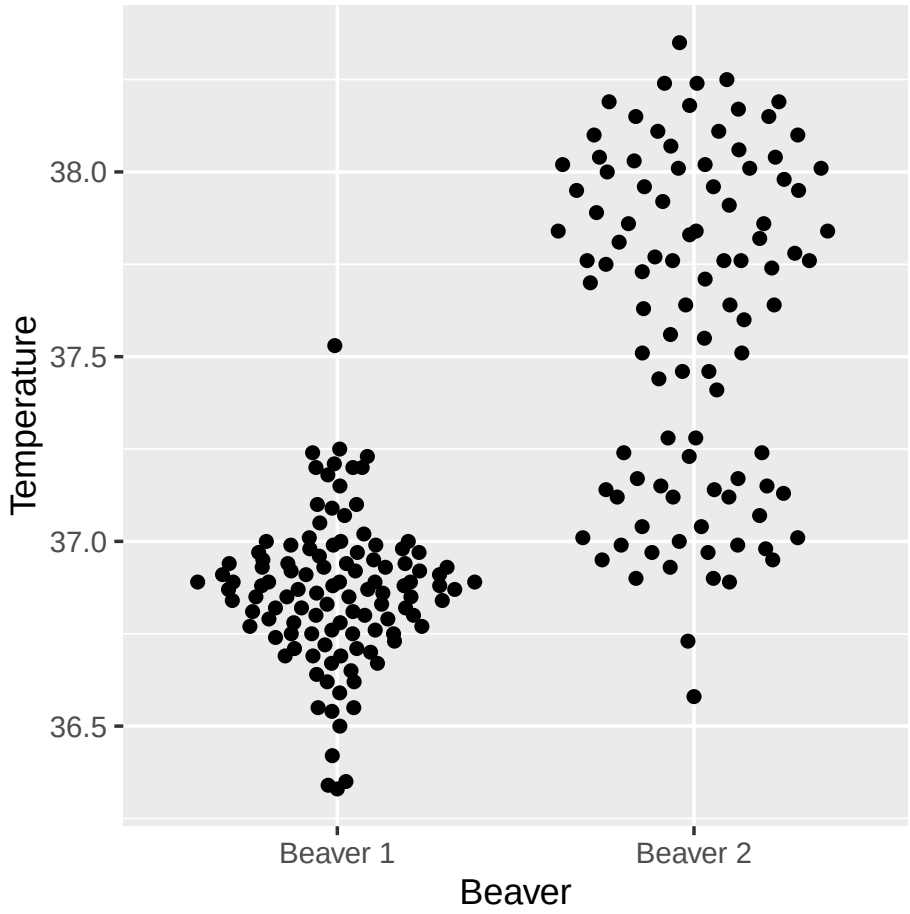
```
>   dat <- list(
+     '10 points'=rnorm(10),
+     '50 points'=rnorm(50,2),
+     '200 points'=c(rnorm(400), rnorm(100,5)),
+     '5000 points'= rnorm(5000,1)
+   )
>   labs<-rep(names(dat),sapply(dat,length))
>   labs<-factor(labs,levels=unique(labs))
>   dat<-unlist(dat)
>   ggplot(mapping=aes(labs, dat)) + geom_quasirandom(alpha=.3,varwidth=TRUE)
```



3. Real data

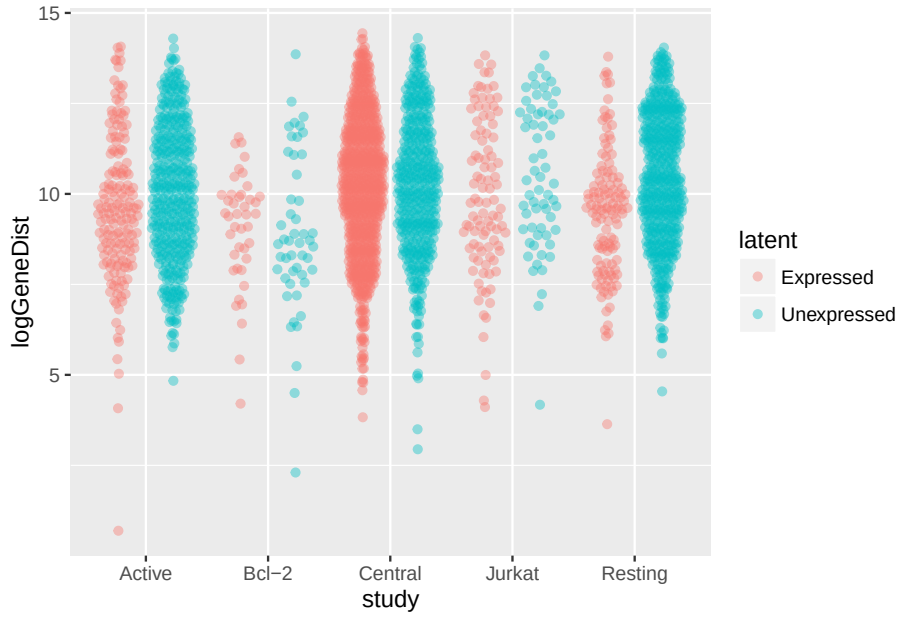
An example using the `beaver1` and `beaver2` data from the **datasets** package:

```
>   beaver<-data.frame(
+     'Temperature'=c(beaver1$temp,beaver2$temp),
+     'Beaver'=rep(
+       c('Beaver 1','Beaver 2'),
+       c(nrow(beaver1),nrow(beaver2))
+     )
+   )
>   ggplot(beaver,mapping=aes(Beaver, Temperature)) + geom_quasirandom()
```



An example using the `integrations` data from the **vipor** package and the argument `dodge.width`:

```
>   library(vipor)
>   ints<-integrations[integrations$nearestGene>0,]
>   ints$logGeneDist<-log(ints$nearestGene)
>   ggplot(ints,mapping=aes(study, logGeneDist,color=latent)) +
+   geom_quasirandom(dodge.width=.9,alpha=.4)
```



Affiliation:

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