

# Package ‘latticedl’

February 14, 2012

**Maintainer** Toby Dylan Hocking <toby.hocking@inria.fr>

**Author** Toby Dylan Hocking

**Version** 1.2

**License** GPL-3

**Title** Lattice direct labels

**Description** OBSOLETE, still on CRAN for historical purposes. Please use the more modern directlabels package instead.

**Depends** plyr, lattice, grid

**Suggests** nlme, mlmRev, ggplot2

**Repository** CRAN

**Repository/R-Forge/Project** directlabels

**Repository/R-Forge/Revision** 429

**Date/Publication** 2011-05-23 15:59:04

## R topics documented:

latticedl-package . . . . .	2
bottom.points . . . . .	3
compare.methods . . . . .	3
direct.label . . . . .	4
dl . . . . .	6
dl.indep . . . . .	8
dl.text . . . . .	8
dl.trans . . . . .	9
empty.grid . . . . .	10
empty.grid.2 . . . . .	11
first.points . . . . .	11
get.means . . . . .	12

high.points . . . . .	13
label.positions . . . . .	13
last.points . . . . .	14
left.points . . . . .	15
low.points . . . . .	15
maxvar.points . . . . .	16
need.trans . . . . .	16
panel.superpose.dl . . . . .	17
perpendicular.lines . . . . .	19
positioning.functions . . . . .	20
right.points . . . . .	21
rug.mean . . . . .	21
top.points . . . . .	22
trans.densityplot . . . . .	22
trans.qqmath . . . . .	23

<b>Index</b>	<b>24</b>
--------------	-----------

---

latticedl-package	<i>Lattice direct labels</i>
-------------------	------------------------------

---

## Description

Direct labeling functions that use the lattice package.

## Details

Package:	latticedl
Maintainer:	Toby Dylan Hocking <toby.hocking@inria.fr>
Author:	Toby Dylan Hocking
Version:	1.1
License:	GPL-3
Title:	Lattice direct labels
Description:	Direct labeling functions that use the lattice package.
Depends:	plyr, lattice, grid
Suggests:	nlme, mlmRev, ggplot2

## Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

---

bottom.points	<i>bottom points</i>
---------------	----------------------

---

**Description**

Positioning Function for the bottom of a group of points.

**Usage**

```
bottom.points(d, ...)
```

**Arguments**

d  
...

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

compare.methods	<i>compare methods</i>
-----------------	------------------------

---

**Description**

Plot several label placement methods on the same page.

**Usage**

```
compare.methods(m, ..., horiz = FALSE)
```

**Arguments**

m	Vector of label placement function names.
...	Args to pass to dl
horiz	Arrange plots horizontally or vertically?

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

direct.label	<i>direct label</i>
--------------	---------------------

---

### Description

Add direct labels to a grouped lattice plot. This works by parsing the trellis object returned by the high level plot function, and returning it with a new panel function that will plot direct labels using the specified method.

### Usage

```
direct.label(p, method = NULL, debug = FALSE)
```

### Arguments

p	The lattice plot (result of a call to a high-level lattice function).
method	Method for direct labeling as described in <code>?label.positions</code> .
debug	Show debug output?

### Value

The lattice plot.

### Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

### Examples

```
library(latticed1)
library(proto)
library(ggplot2)
data(mpg)
m <- lm(cty~displ,data=mpg)
mpgf <- fortify(m,mpg)
mpg.scatter <- xyplot(.resid~.fitted,mpgf,groups=factor(cyl))
plot(direct.label(mpg.scatter))
plot(direct.label(mpg.scatter,debug=TRUE))
plot(direct.label(
  xyplot(.resid~.fitted,mpgf,groups=factor(cyl),
    panel=function(...){panel.abline(1);panel.xyplot(...)},
    main="foobar2")
  ,method=perpendicular.lines))
## Should plot but show direct label placement error in each panel:
## default method includes perpendicular line calculation, which makes
## no sense for only 1 group per panel
trellised <- xyplot(.resid~.fitted|cyl,mpgf,groups=factor(cyl))
plot(direct.label(trellised))
## Should work, but not very informative:
```

```

plot(direct.label(trellised,method=empty.grid))
mpgf$cyl10 <- sapply(mpgf$cyl,function(i)paste(rep(i,l=10),collapse=""))
plot(direct.label(
  xyplot(.resid~.fitted|cyl,mpgf,groups=factor(cyl10))
  ,method=empty.grid))
## Some label placements fail, some dont:
plot(direct.label(
  xyplot(.resid~.fitted|manufacturer,mpgf,groups=factor(cyl))
  ,method=empty.grid.2))

data(BodyWeight,package="nlme")
print(direct.label(
  xyplot(weight~Time|Diet,BodyWeight,groups=Rat,type='l',layout=c(3,1))
  ))
## Say we want to use a simple linear model to explain rat body weight:
fit <- lm(weight~Time+Diet+Rat,BodyWeight)
bw <- fortify(fit,BodyWeight)
## And we want to use this panel function to display the model fits:
panel.model <- function(x,subscripts,col.line,...){
  panel.xyplot(x=x,subscripts=subscripts,col.line=col.line,...)
  llines(x,bw[subscripts,".fitted"],col=col.line,lty=2)
}
## Just specify the custom panel function as usual:
print(direct.label(
  xyplot(weight~Time|Diet,bw,groups=Rat,type='l',layout=c(3,1),
  panel=panel.superpose,panel.groups=panel.model)
  ,method=last.points))

## Fails: default method for scatterplot doesn't make sense here
##print(direct.label(xyplot,BodyWeight,weight~Time|Diet,Rat))
loci <- data.frame(ppp=c(rbeta(800,10,10),rbeta(100,0.15,1),rbeta(100,1,0.15)),
  type=factor(c(rep("NEU",800),rep("POS",100),rep("BAL",100))))
plot(direct.label(
  densityplot(~ppp,loci,groups=type,n=500)
  ))
## Not very informative but it should work:
plot(direct.label(
  densityplot(~ppp|type,loci,groups=type,n=500)
  ))

data(Chem97,package="mlmRev")
qqm <- qqmath(~gcscscore,Chem97,groups=gender,
  type=c("p","g"),f.value=ppoints(25),auto.key=TRUE)
plot(direct.label(qqm))
static.labels <- data.frame(x=c(-2,0),y=c(6,4),groups=c("F","M"))
plot(direct.label(qqm,method=function(d,...)static.labels,debug=TRUE))
plot(direct.label(qqm,method=static.labels,debug=TRUE))
## Should work: static.labels overwrites values from
## last.points. Applying last.points should change the hjust as well:
plot(direct.label(qqm,method=c("last.points",static.labels),debug=TRUE))
plot(direct.label(qqm,method=c(static.labels,"last.points"),debug=TRUE))
plot(direct.label(qqmath(~gcscscore|gender,Chem97,groups=factor(score),

```

```

type=c('l', 'g'), f.value=ppoints(100)))

plot(direct.label(densityplot(~gcsescore, Chem97, groups=factor(score))))
## This would be more effective superimposed:
plot(direct.label(densityplot(~gcsescore|gender, Chem97, groups=factor(score), layout=c(1,2))))

plot(direct.label(dotplot(VADeaths, type="o"), method=list("last.points", rot=30)))

```

---

dl

*Quick lattice direct label plot*


---

### Description

Shortcut for a lattice plot with direct labels. This is a convenience function so you do not have to explicitly type `groups=`. This simply constructs the lattice plot and then calls `direct.label` on it.

### Usage

```
dl(lattice.fun, data, x, groups, method = NULL, debug = FALSE, ...)
```

### Arguments

<code>lattice.fun</code>	High-level lattice plot function to use.
<code>data</code>	Data to use.
<code>x</code>	Lattice model formula.
<code>groups</code>	To be passed to <code>lattice</code> as <code>groups=</code> argument.
<code>method</code>	Method for direct labeling as described in <code>?label.positions</code> .
<code>debug</code>	Show debug output?
<code>...</code>	Other arguments to be passed to <code>lattice.fun</code> .

### Value

The lattice plot.

### Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

### Examples

```

library(latticedl)
library(proto)
library(ggplot2)
data(mpg)
m <- lm(cty~displ, data=mpg)
mpgf <- fortify(m, mpg)
plot(dl(xyplot, mpgf, .resid~.fitted, factor(cyl), method=get.means, par.settings=list()))

```

```

plot(dl(xyplot,mpgf,.resid~.fitted,factor(cyl),
      panel=function(...){panel.abline(1);panel.xyplot(...)},
      main="foobar2",
      method=perpendicular.lines))
plot(dl(xyplot,mpgf,.resid~.fitted,factor(cyl),debug=TRUE))
## Should fail: (default method includes perpendicular line
## calculation, which makes no sense for only 1 group per panel
plot(dl(xyplot,mpgf,.resid~.fitted|cyl,factor(cyl)))
## Should work, but not very informative:
plot(dl(xyplot,mpgf,.resid~.fitted|cyl,factor(cyl),method=empty.grid))
mpgf$cyl10 <- sapply(mpgf$cyl,function(i)paste(rep(i,l=10),collapse=""))
plot(dl(xyplot,mpgf,.resid~.fitted|cyl,factor(cyl10),method=empty.grid))
plot(dl(xyplot,mpgf,.resid~.fitted|manufacturer,factor(cyl),method=empty.grid.2))

data(BodyWeight,package="nlme")
print(dl(xyplot,BodyWeight,weight~Time|Diet,Rat,
      type='l',layout=c(3,1)))
## Say we want to use a simple linear model to explain rat body weight:
fit <- lm(weight~Time+Diet+Rat,BodyWeight)
bw <- fortify(fit,BodyWeight)
## And we want to use this panel.groups function to display the model
## fits:
panel.model <- function(x,subscripts,col.line,...){
  panel.xyplot(x=x,subscripts=subscripts,col.line=col.line,...)
  llines(x,bw[subscripts, ".fitted"],col=col.line,lty=2)
}
## Custom panel.groups functions:
print(dl(xyplot,bw,weight~Time|Diet,Rat,type="l",layout=c(3,1),
      panel=panel.superpose,
      panel.groups=panel.model,method=first.points))
## Custom panel function which highlights min and max values:
panel.range <- function(y,...){
  panel.abline(h=range(y))
  panel.superpose(y=y,...)
}
print(dl(xyplot,bw,weight~Time|Diet,Rat,type="l",layout=c(3,1),
      panel=panel.range))
## Custom panel and panel.groups functions:
print(dl(xyplot,bw,weight~Time|Diet,Rat,type="l",layout=c(3,1),
      panel=panel.range,panel.groups=panel.model,method=first.points))

## Fails: default method for scatterplot doesn't make sense here
##print(dl(xyplot,BodyWeight,weight~Time|Diet,Rat))

## dl with densityplots:
loci <- data.frame(ppp=c(rbeta(800,10,10),rbeta(100,0.15,1),rbeta(100,1,0.15)),
  type=factor(c(rep("NEU",800),rep("POS",100),rep("BAL",100))))
print(dl(densityplot,loci,~ppp,type,n=500))
## Not very informative but it should work:
print(dl(densityplot,loci,~ppp|type,type,n=500))

```

---

dl.indep	<i>Direct label groups independently</i>
----------	--

---

**Description**

Makes a function you can use to specify the location of each group independently.

**Usage**

```
dl.indep(expr)
```

**Arguments**

expr	Expression that takes a subset of the d data frame, with data from only a single group, and returns the direct label position.
------	--

**Value**

A Positioning Function.

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

**Examples**

```
library(latticedl)
complicated <- list(dl.trans(x=x+10),
                   dl.indep(d[-2,]),
                   rot=c(30,180))
direct.label(
  dotplot(VADeaths,type="o")
  ,method=complicated)
```

---

dl.text	<i>dl text</i>
---------	----------------

---

**Description**

To be used as panel.groups= argument in panel.superpose. Analyzes arguments to determine correct text color for this group, and then draws the direct label text.

**Usage**

```
dl.text(labs, group.number, col.line = NULL, col.points = NULL, col = NULL, col.symbol = NULL, type = NULL)
```

**Arguments**

labs	table of labels and positions constructed by label.positions
group.number	which group we are currently plotting, according to levels(labs\$groups)
col.line	line color
col.points	point color
col	general color
col.symbol	symbol color
type	plot type
...	ignored

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

dl.trans                      *Direct label data transform*

---

**Description**

Make a function that transforms the data. This is for conveniently making a function that calls transform on the data frame, with the arguments provided. See examples.

**Usage**

```
dl.trans(...)
```

**Arguments**

...                      Arguments to pass to transform.

**Value**

A Positioning Function.

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

**Examples**

```
library(latticedl)
complicated <- list(dl.trans(x=x+10),
                   dl.indep(d[-2,]),
                   rot=c(30,180))
direct.label(
  dotplot(VADeaths, type="o")
  ,method=complicated)
```

---

 empty.grid

*empty.grid*


---

## Description

Label placement method for scatterplots that ensures labels are placed in different places. A grid is drawn over the whole plot. Each cluster is considered in sequence and assigned to the point on this grid which is closest to the point given by `loc.fun()`.

## Usage

```
empty.grid(d, debug = FALSE, loc.fun = get.means, ...)
```

## Arguments

<code>d</code>	Data frame of points on the scatterplot with columns groups x y.
<code>debug</code>	Show debugging info on the plot? This is passed to <code>loc.fun</code> .
<code>loc.fun</code>	Function that takes <code>d</code> and returns a data frame with 1 column for each group, giving the point we will use to look for a close point on the grid, to put the group label.
<code>...</code>	ignored.

## Value

Data frame with columns groups x y, 1 line for each group, giving the positions on the grid closest to each cluster.

## Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

## Examples

```
library(latticed1)
library(proto)
library(ggplot2)
data(mpg)
m <- lm(cty~displ,data=mpg)
mpgf <- fortify(m,mpg)
plot(dl(xyplot,mpgf,.resid~.fitted,factor(cyl)))

plot(dl(xyplot,mpgf,.resid~.fitted,factor(cyl),
       panel=function(...){panel.abline(1);panel.xyplot(...)},
       main="foobar2",
       method=perpendicular.lines))
plot(dl(xyplot,mpgf,.resid~.fitted,factor(cyl),debug=TRUE))
## Should fail: (default method includes perpendicular line calculation, which makes no sense for only 1 group per p
plot(dl(xyplot,mpgf,.resid~.fitted|cyl,factor(cyl)))
```

```
## Should work, but not very informative:
plot(dl(xyplot,mpgf,.resid~.fitted|cyl,factor(cyl),method=empty.grid))
mpgf$cyl10 <- sapply(mpgf$cyl,function(i)paste(rep(i,l=10),collapse=""))
plot(dl(xyplot,mpgf,.resid~.fitted|cyl,factor(cyl10),method=empty.grid))
plot(dl(xyplot,mpgf,.resid~.fitted|manufacturer,factor(cyl),method=empty.grid.2))
```

---

empty.grid.2

*empty grid 2*

---

### Description

Use the perpendicular lines method in combination with the empty grid method.

### Usage

```
empty.grid.2(d, debug, ...)
```

### Arguments

d	Data frame with columns groups x y.
debug	Show debugging graphics on the plot?
...	ignored.

### Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

---

first.points

*first points*

---

### Description

Positioning Function for the first of a group of points.

### Usage

```
first.points(d, ...)
```

### Arguments

d
...

### Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

## Examples

```
library(latticed1)
library(ggplot2)
data(BodyWeight,package="nlme")
print(dl(xyplot,BodyWeight,weight~Time|Diet,Rat,
        type='l',layout=c(3,1)))
## Say we want to use a simple linear model to explain rat body weight:
fit <- lm(weight~Time+Diet+Rat,BodyWeight)
bw <- fortify(fit,BodyWeight)
## And we want to use this panel function to display the model fits:
panel.model <- function(x,subscripts,col.line,...){
  panel.xyplot(x=x,subscripts=subscripts,col.line=col.line,...)
  llines(x,bw[subscripts,".fitted"],col=col.line,lty=2)
}
## Just specify the custom panel function as usual:
print(dl(xyplot,bw,weight~Time|Diet,Rat,
        type='l',layout=c(3,1),panel=panel.model))

## Fails: default method for scatterplot doesn't make sense here
##print(dl(xyplot,BodyWeight,weight~Time|Diet,Rat))
```

---

get.means

*get means*

---

## Description

Positioning Function for the mean of each cluster of points.

## Usage

```
get.means(d, ...)
```

## Arguments

d  
...

## Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

---

high.points	<i>high points</i>
-------------	--------------------

---

**Description**

```
dl.indep(data.frame(d[which.max(d$y),],hjust=0.5,vjust=0))
```

**Usage**

```
high.points(d, ...)
```

**Arguments**

d  
...

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

label.positions	<i>label positions</i>
-----------------	------------------------

---

**Description**

Calculates table of positions of each label. It does not draw anything, but is called for its return value. Normally you don't have to call label.positions explicitly. Instead, it is called for you by direct.label.

**Usage**

```
label.positions(x, y, subscripts, groups, debug = FALSE, method, ...)
```

**Arguments**

x	x values of points to draw.
y	y values of points to draw.
subscripts	Subscripts of groups to consider.
groups	Vector of groups.
debug	Show debug output? If TRUE, the resulting table of label positions will be printed.

method            Method for direct labeling, specified in one of the following ways: (1) a Positioning Function, (2) the name of a Positioning Function as a character string, or (3) a list containing any number of (1), (2), or additionally named values. Starting from the data frame of points to plot for the panel, the elements of the list are applied in sequence, and each row of the resulting data frame is used to draw a direct label. See examples in `?direct.label` and `?positioning.functions`. NULL indicates to choose a Positioning Function based on the high-level plot function chosen (this is done in `panel.superpose.dl`).

...                Passed to positioning method(s).

**Value**

Data frame of direct label positions. Each row describes the position of 1 label to be drawn later.

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

`last.points`

*last.points*

---

**Description**

Positioning Function for the last of a group of points.

**Usage**

```
last.points(d, ...)
```

**Arguments**

d  
...

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

left.points	<i>left points</i>
-------------	--------------------

---

**Description**

```
dl.indep(data.frame(d[which.min(d$x)],hjust=1,vjust=0.5))
```

**Usage**

```
left.points(d, ...)
```

**Arguments**

```
d  
...
```

**Author(s)**

```
Toby Dylan Hocking <toby.hocking@inria.fr>
```

---

low.points	<i>low points</i>
------------	-------------------

---

**Description**

```
dl.indep(data.frame(d[which.min(d$y)],hjust=0.5,vjust=1))
```

**Usage**

```
low.points(d, ...)
```

**Arguments**

```
d  
...
```

**Author(s)**

```
Toby Dylan Hocking <toby.hocking@inria.fr>
```

maxvar.points            *maxvar points*

---

**Description**

Do first or last, whichever has points most spread out.

**Usage**

```
maxvar.points(d, ...)
```

**Arguments**

d  
...

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

need.trans            *need trans*

---

**Description**

Functions which need translation before applying positioning function.

**Usage**

```
need.trans
```

**Format**

The format is: chr [1:2] "qqmath" "densityplot"

---

panel.superpose.dl      *panel superpose dl*

---

## Description

Call panel.superpose for the data points and then for the direct labels. This is a proper lattice panel function that behaves much like panel.superpose.

## Usage

```
panel.superpose.dl(x, y = NULL, subscripts, groups, panel.groups, method = NULL, .panel.superpose = pan
```

## Arguments

x	Vector of x values.
y	Vector of y values.
subscripts	Subscripts of x,y,groups.
groups	Vector of group ids.
panel.groups	To be parsed for default labeling method, and passed to panel.superpose.
method	Method for direct labeling as described in ?label.positions.
.panel.superpose	The panel function to use for drawing data points.
type	Plot type, used for default method dispatch.
...	Additional arguments to panel.superpose.

## Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

## Examples

```
loci <- data.frame(ppp=c(rbeta(800,10,10),rbeta(100,0.15,1),rbeta(100,1,0.15)),
                    type=factor(c(rep("NEU",800),rep("POS",100),rep("BAL",100))))
library(latticedl)
## 4 ways to make the same plot:
print(dl(densityplot,loci,~ppp,type,n=500))
print(direct.label(
  densityplot(~ppp,loci,groups=type,n=500)
))
print(direct.label(
  densityplot(~ppp,loci,groups=type,n=500,
             panel=panel.superpose,
             panel.groups="panel.densityplot")
))
print(densityplot(~ppp,loci,groups=type,n=500,
                 panel=panel.superpose.dl,panel.groups="panel.densityplot"))
```

```

### Exploring custom panel and panel.groups functions
library(ggplot2)
data(BodyWeight,package="nlme")
## Say we want to use a simple linear model to explain rat body weight:
fit <- lm(weight~Time+Diet+Rat,BodyWeight)
bw <- fortify(fit,BodyWeight)
## lots of examples to come, all with these arguments:
ratxy <- function(...){
  xyplot(weight~Time|Diet,bw,groups=Rat,type="l",layout=c(3,1),...)
}
## No custom panel functions:
regular <- ratxy()
print(regular) ## normal lattice plot
print(direct.label(regular)) ## with direct labels

## The direct label panel function panel.superpose.dl can be used to
## display direct labels as well:
print(ratxy(panel=panel.superpose.dl,panel.groups="panel.xyplot"))
print(ratxy(panel=function(...){
  panel.superpose.dl(panel.groups="panel.xyplot",...)
}))

## Not very user-friendly, since default label placement is
## impossible, but these should work:
print(ratxy(panel=panel.superpose.dl,panel.groups=panel.xyplot,
  method=first.points))
print(ratxy(panel=function(...){
  panel.superpose.dl(panel.groups=panel.xyplot,...),
  method=first.points))
}))

### Custom panel.groups functions:
## This panel.groups function will display the model fits:
panel.model <- function(x,subscripts,col.line,...){
  panel.xyplot(x=x,subscripts=subscripts,col.line=col.line,...)
  llines(x,bw[subscripts,".fitted"],col=col.line,lty=2)
}
pg <- ratxy(panel=panel.superpose,panel.groups=panel.model)
print(pg)
## If you use panel.superpose.dl with a custom panel.groups function,
## you need to manually specify the Positioning Function, since the
## name of panel.groups is used to infer a default:
print(direct.label(pg,method=first.points))
print(ratxy(panel=panel.superpose.dl,panel.groups="panel.model",
  method=first.points))

## Custom panel function that draws a box around values:
panel.line1 <- function(ps=panel.superpose){
  function(y,...){
    panel.abline(h=range(y))
    ps(y=y,...)
  }
}
custom <- ratxy(panel=panel.line1())

```

```
print(custom)
print(direct.label(custom))
## Alternate method, producing the same results, but using
## panel.superpose.dl in the panel function. This is useful for direct
## label plots where you use several datasets.
print(ratxy(panel=panel.line1(panel.superpose.dl),panel.groups="panel.xyplot"))

## Lattice plot with custom panel and panel.groups functions:
both <- ratxy(panel=panel.line1(),panel.groups="panel.model")
print(both)
print(direct.label(both,method=first.points))
print(ratxy(panel=panel.line1(panel.superpose.dl),
            panel.groups=panel.model,method=first.points))
```

---

perpendicular.lines    *perpendicular lines*

---

## Description

Draw a line between the centers of each cluster, then draw a perpendicular line for each cluster that goes through its center. For each cluster, return the point the lies furthest out along this line.

## Usage

```
perpendicular.lines(d, debug = FALSE, ...)
```

## Arguments

d	Data frame with groups x y.
debug	If TRUE will draw points at the center of each cluster and some lines that show how the points returned were chosen.
...	ignored.

## Value

Data frame with groups x y, giving the point for each cluster which is the furthest out along the line drawn through its center.

## Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

---

positioning.functions *Built-in Positioning Functions for direct label placement*

---

### Description

When adding direct labels to a grouped plot, label placement can be specified using a Positioning Function (or a list of them), of the form `function(d,...)`, where `d` is a data frame of the points to plot, with columns `x` `y` groups. The job of the Positioning Function(s) is to return the position of each direct label you want to plot as a data frame, with 1 row for each label. Thus normally a Positioning Function will return 1 row for each group. Several built-in Positioning Functions are discussed below, but you can also create your own, either from scratch or by using `dl.indep` and `dl.trans`.

### Usage

```
## Longitudinal data:
## first.points
## left.points    ## same as first.points
## last.points
## right.points  ## same as last.points
## bottom.points
## low.points    ## same as bottom.points

## Also good for density plots:
## top.points
## high.points   ## same as top.points

## Scatter plots:
## get.means
## perpendicular.lines
## empty.grid
## empty.grid.2
```

### Arguments

```
d          Data frame of points to plot, with columns x y groups.
...        Ignored.
```

### Value

Data frame of label positions.

### Author(s)

Toby Dylan Hocking <toby.hocking@inria.fr>

---

`right.points`                      *right points*

---

**Description**

`dl.indep(data.frame(d[which.max(d$x),],hjust=0,vjust=0.5))`

**Usage**

`right.points(d, ...)`

**Arguments**

`d`  
`...`

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

`rug.mean`                              *rug mean*

---

**Description**

Place points on top of the mean value of the rug.

**Usage**

`rug.mean(d, ..., end)`

**Arguments**

`d`  
`...`  
`end`

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

top.points                    *top points*

---

**Description**

Positioning Function for the top of a group of points.

**Usage**

```
top.points(d, ...)
```

**Arguments**

d  
...

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

**Examples**

```
loci <- data.frame(ppp=c(rbeta(800,10,10),rbeta(100,0.15,1),rbeta(100,1,0.15)),
                      type=factor(c(rep("NEU",800),rep("POS",100),rep("BAL",100))))
library(latticed1)
print(dl(densityplot,loci,~ppp,type,n=500))
## Not very informative but it should work:
print(dl(densityplot,loci,~ppp|type,type,n=500))
```

---

trans.densityplot            *trans densityplot*

---

**Description**

Transformation function for 1d densityplots.

**Usage**

```
trans.densityplot(d, ...)
```

**Arguments**

d  
...

**Author(s)**

Toby Dylan Hocking <toby.hocking@inria.fr>

---

`trans.qqmath`

*trans qqmath*

---

**Description**

Transformation function for 1d qqmath plots.

**Usage**

`trans.qqmath(d, ...)`

**Arguments**

`d`

`...`

**Author(s)**

Toby Dylan Hocking <[toby.hocking@inria.fr](mailto:toby.hocking@inria.fr)>

# Index

- \*Topic **package**
  - [latticedl-package, 2](#)
  
- [bottom.points, 3](#)
  
- [compare.methods, 3](#)
  
- [direct.label, 4](#)
- [dl, 6](#)
- [dl.indep, 8](#)
- [dl.text, 8](#)
- [dl.trans, 9](#)
  
- [empty.grid, 10](#)
- [empty.grid.2, 11](#)
  
- [first.points, 11](#)
  
- [get.means, 12](#)
  
- [high.points, 13](#)
  
- [label.positions, 13](#)
- [last.points, 14](#)
- [latticedl-package, 2](#)
- [left.points, 15](#)
- [low.points, 15](#)
  
- [maxvar.points, 16](#)
  
- [need.trans, 16](#)
  
- [panel.superpose.dl, 17](#)
- [perpendicular.lines, 19](#)
- [Positioning.Function](#)
  - [\(positioning.functions\), 20](#)
- [positioning.function](#)
  - [\(positioning.functions\), 20](#)
- [Positioning.Functions](#)
  - [\(positioning.functions\), 20](#)
- [positioning.functions, 20](#)

- [right.points, 21](#)
- [rug.mean, 21](#)
  
- [top.points, 22](#)
- [trans.densityplot, 22](#)
- [trans.qqmath, 23](#)