Package ‘directlabels’

February 1, 2020

**Maintainer**  Toby Dylan Hocking <toby.hocking@r-project.org>

**Author**  Toby Dylan Hocking

**Version**  2020.1.31

**BugReports**  https://github.com/tdhock/directlabels/issues

**License**  GPL-3

**Title**  Direct Labels for Multicolor Plots

**Description**  An extensible framework

  for automatically placing direct labels onto multicolor 'lattice' or 'ggplot2' plots.

  Label positions are described using Positioning Methods

  which can be re-used across several different plots.

  There are heuristics for examining ``trellis'' and ```ggplot'' objects

  and inferring an appropriate Positioning Method.

**URL**  https://github.com/tdhock/directlabels

**LazyData**  true

**Suggests**  MASS, inlinedocs, ggplot2 (>= 2.0), rlang, lattice, alphahull, nlme, lars, latticeExtra, testthat

**Imports**  grid, quadprog


**NeedsCompilation**  no

**Repository**  CRAN

**Date/Publication**  2020-02-01 09:10:02 UTC

**R topics documented:**

ahull.grid ................................................................. 4
ahull.points ............................................................... 4
angled.boxes .............................................................. 4

1
R topics documented:

angled.endpoints .................................................. 5
apply.method ....................................................... 5
big.boxes .......................................................... 6
bottom.pieces ...................................................... 7
bumpup ............................................................... 7
calc.borders ....................................................... 8
calc.boxes ........................................................ 8
check.for.columns ................................................ 9
chull.grid ........................................................... 9
chull.points ....................................................... 10
default.ahull ..................................................... 10
default_picker .................................................... 11
defaultpf.ggplot ................................................ 11
defaultpf.trellis ................................................ 12
direct.label ....................................................... 12
direct.label.ggplot ............................................. 14
direct.label.trellis ............................................. 14
dl.combine ........................................................ 15
dl.jitter ........................................................... 17
dl.move ............................................................ 18
dl.summarize ...................................................... 19
dl.trans ............................................................ 19
dlcompare .......................................................... 20
dldoc ................................................................. 21
dlgrob ............................................................... 22
draw.polygons ..................................................... 23
draw.rects ........................................................ 23
drawDetails.dlgrob ............................................. 24
dedges.to.outside ............................................... 25
empty.grid .......................................................... 25
enlarge.box ........................................................ 26
extract.plot ........................................................ 26
extract.posfun .................................................... 27
extreme.grid ....................................................... 27
extreme.points ................................................... 28
far.from.others.borders ......................................... 28
filltemplate ........................................................ 29
first.bumpup ..................................................... 29
first.points ....................................................... 29
first.polygons ................................................... 30
first.qp ............................................................. 30
gapply ............................................................. 31
gapply.fun ....................................................... 31
GeomDL ............................................................. 32
gem._dl ............................................................ 32
get.means .......................................................... 34
getLegendVariables ............................................... 35
ignore.na ........................................................... 35


R topics documented:

in1box .................................................. 36
in1which ............................................... 36
inside ................................................... 37
iris.l1.cluster ........................................... 37
label.endpoints ........................................ 38
label.pieces ............................................ 39
lasso.labels ............................................. 39
last.bumpup .............................................. 39
last.points .............................................. 40
last.polygons ........................................... 40
last.qp .................................................. 40
lattice.translators ...................................... 41
legends2hide ............................................ 41
lines2 ................................................... 42
make.tiebreaker ........................................ 42
maxvar.points .......................................... 43
maxvar.qp ............................................... 43
merge_recurse .......................................... 44
midrange ............................................... 44
normal.l2.cluster ...................................... 45
only.unique.vals ....................................... 46
outside.ahull .......................................... 46
outside.chull .......................................... 47
panel.superpose.dl ..................................... 47
pkgFun .................................................. 49
polygon.method ......................................... 50
positioning.functions .................................. 51
project.onto.segments .................................. 55
projectionSeconds ..................................... 56
qp.labels .............................................. 57
reduce.cex.lr .......................................... 59
rhtmlescape ............................................ 60
SegCost ................................................ 61
smart.grid .............................................. 61
static.labels .......................................... 62
svmtrain ............................................... 62
top.bumptwice ......................................... 63
top.bumpup ............................................. 63
top.pieces .............................................. 64
top.points .............................................. 64
top.qp .................................................. 65
uselegend.ggplot ....................................... 65
uselegend.trellis ..................................... 65
vertical.qp ............................................. 66
visualcenter ............................................ 66
xlims .................................................... 67
ylims .................................................... 67

Index 68
ahull.grid  

**Description**
Label the closest point on the alpha hull of the data.

**Usage**
```
"ahull.grid"
```

ahull.points  

**Description**
Calculate the points on the ashape.

**Usage**
```
ahull.points(d, ..., 
ahull = default.ahull(d))
```

**Arguments**
- `d`
- `...`
- `ahull`

**Author(s)**
Toby Dylan Hocking

angled.boxes  

**Description**
Draw a box with the label `inside`, at the point furthest away from the plot border and any other curve.

**Usage**
```
"angled.boxes"
```
angled.endpoints  angled endpoints

Description
Useful for labeling lines that all end at the top.

Usage
"angled.endpoints"

apply.method  Apply a Positioning Method

Description
Run a Positioning Method list on a given data set. This function contains all the logic for parsing a Positioning Method and sequentially applying its elements to the input data to obtain the label positions.

Usage
apply.method(method, d, columns.to.check = c("x", "y", "groups"), ..., debug = FALSE)

Arguments
method  Direct labeling Positioning Method. Starting from the data frame of points to plot for the panel, the elements of the Positioning Method list are applied in sequence, and then each row of the resulting data frame is used to draw a direct label. The elements of a Positioning Method list can be

  • a Positioning Function is any function(d,...) which takes a data.frame d with columns x,y,groups and returns another data.frame representing the positions of the desired direct labels. For a description of all the columns that are interpreted for drawing direct labels, see drawDetails.dlgrob. For example, maxvar.points is a Positioning Function that returns a data.frame with columns x,y,groups,hjust,vjust.
  • a character vector of length 1 is treated as the name of an R object. For example, specifying "maxvar.points" means to look up the variable called maxvar.points and use that. Using the name of a Positioning Function is preferable to specifying the Positioning Function itself, since then the name is visible in the Positioning Method list, which is more interpretable when debugging.
• a named list element is used to add or update variables in the data.frame of direct labels to plot. For example list("first.points",cex=1.5) means take only the first points of every group and then set the cex column to 1.5.

• an element of a Positioning Method list can be another Positioning Method list, in which case the elements of the inner list are applied.

`d` Data frame to which we apply the Positioning Method. The x and y columns should be in centimeters (cm), so that Positioning Methods can easily calculate the L2/Euclidean/visual distance between pairs of points.

columns.to.check

After applying each Positioning Method list element, we check for the presence of these columns, and if not found we stop with an error.

... Named arguments, passed to Positioning Functions.

debug If TRUE, print each Positioning Method list element and the direct label data.frame that results from its evaluation.

**Value**

The final data frame returned after applying all of the items in the Positioning Method list, with x and y in units of cm.

**Author(s)**

Toby Dylan Hocking

---

**big.boxes**

**big boxes**

**Description**

Calculate big boxes around the means of each cluster.

**Usage**

"big.boxes"
### Description

Positioning Method for the bottom of a group of points.

### Usage

```r
bottom.pieces(d, ...)
```

### Arguments

- `d`
- `...`

### Author(s)

Toby Dylan Hocking

---

### Description

Sequentially bump labels up, starting from the bottom, if they collide with the label underneath.

### Usage

```r
bumpup(d, ...)
```

### Arguments

- `d`
- `...`

### Author(s)

Toby Dylan Hocking
### calc.borders

**Description**

Calculate bounding box based on newly calculated width and height.

**Usage**

```r
calc.borders(d, ...)
```

**Arguments**

- `d` Data frame of point labels, with new widths and heights in the `w` and `h` columns.
- `...` ignored.

**Author(s)**

Toby Dylan Hocking

### calc.boxes

**Description**

Calculate boxes around labels, for collision detection.

**Usage**

```r
calc.boxes(d, debug = FALSE, 
            ...)```

**Arguments**

- `d`
- `debug`
- `...`

**Author(s)**

Toby Dylan Hocking
check.for.columns

Description
Stop if a data.frame does not have some columns.

Usage
check.for.columns(d, must.have)

Arguments
- d: data.frame to check.
- must.have: column names to check.

Author(s)
Toby Dylan Hocking

chull.grid

Description
Label the closest point on the convex hull of the data.

Usage
“chull.grid”
chull.points

Description
Calculate the points on the convex hull.

Usage
chull.points(d, ...)

Arguments
d
...

Author(s)
Toby Dylan Hocking

default.ahull

Description
Calculate the default alpha parameter for ashape based on the average size of label boxes.

Usage
default.ahull(d, ...)

Arguments
d
...

Author(s)
Toby Dylan Hocking
**default picker**

---

**Description**

Look at options() for a user-defined default Positioning Method picker, and use that (or the hard-coded default picker), with the calling environment to figure out a good default.

**Usage**

```
default.picker(f)
```

**Arguments**

- **f**  
  Object class to look for (trellis or ggplot).

**Author(s)**

Toby Dylan Hocking

---

**defaultpf.ggplot**

---

**Description**

Default method selection method for ggplot2 plots.

**Usage**

```
defaultpf.ggplot(geom,  
    p, L, colvar, ...)
```

**Arguments**

- **geom**
- **p**
- **L**
- **colvar**
- **...**

**Author(s)**

Toby Dylan Hocking
**defaultpf.trellis** *defaultpf trellis*

**Description**

If no Positioning Method specified, choose a default using this function. The idea is that this is called with all the variables in the environment of `panel.superpose.dl`, and this can be user-customizable by setting the `directlabels.defaultpf.lattice` option to a function like this.

**Usage**

```r
defaultpf.trellis(lattice.fun.name, groups, type, ...)
```

**Arguments**

- `lattice.fun.name`
- `groups`
- `type`
- `...`

**Author(s)**

Toby Dylan Hocking

---

**direct.label** *Direct labels for color decoding*

**Description**

Add direct labels to a plot, and hide the color legend. Modern plotting packages like lattice and ggplot2 show automatic legends based on the variable specified for color, but these legends can be confusing if there are too many colors. Direct labels are a useful and clear alternative to a confusing legend in many common plots.

**Usage**

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

**Arguments**

- `p` The "trellis" or "ggplot" object with things drawn in different colors.
- `method` Positioning Method, which determines the positions of the direct labels as a function of the plotted data. If NULL, we examine the plot `p` and try to choose an appropriate default. See `apply.method` for more information about Positioning Methods.
- `debug` Show debug output?

**Value**

A plot with direct labels and no color legend.

**Author(s)**

Toby Dylan Hocking

**Examples**

```r
if(require(ggplot2)){
  ## Add direct labels to a ggplot2 scatterplot, making sure that each
  ## label is close to its point cloud, and doesn't overlap points or
  ## other labels.
  scatter <- qplot(jitter(hwy),jitter(cty),data=mpg,colour=class,
                   main="Fuel efficiency depends on car size")
  print(direct.label(scatter))
}

## direct labels for lineplots that do not overlap and do not go off
## the plot.
library(nlme)
library(lattice)
oldopt <- lattice.options(panel.error=NULL)
ratplot <-
  xyplot(weight~Time|Diet,BodyWeight,groups=Rat,type='l',layout=c(3,1))
## Using the default Positioning Method (maxvar.qp), the labels are
## placed on the side which is most spread out, so in multipanel
## plots they sometimes end up on different sides.
print(direct.label(ratplot))
## To put them on the same side, just manually specify the
## Positioning Method.
print(direct.label(ratplot,"last.qp"))

lattice.options(oldopt)
```
direct.label.ggplot

Description
Direct label a ggplot2 grouped plot.

Usage
```r
## S3 method for class 'ggplot'
direct.label(p,
    method = NULL, debug = FALSE)
```

Arguments
- `p`: The ggplot object.
- `method`: Method for direct labeling as described in `apply.method`.
- `debug`: Show debug output?

Value
The ggplot object with direct labels added.

Author(s)
Toby Dylan Hocking

direct.label.trellis

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
```r
## S3 method for class 'trellis'
direct.label(p,
    method = NULL, debug = FALSE)
```
dl.combine

Arguments

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

Value

  The lattice plot.

Author(s)

  Toby Dylan Hocking

---

dl.combine  

Combine output of several methods

Description

  Apply several Positioning methods to the original data frame.

Usage

  dl.combine(...)  

Arguments

  ...  Several Positioning Methods.

Value

  A Positioning Method that returns the combined data frame after applying each specified Positioning Method.

Author(s)

  Toby Dylan Hocking

Examples

  ## Simple example: label the start and endpoints
  library(nlme)
  library(lattice)
  ratplot <- xyplot(
    weight~Time|Diet,BodyWeight,groups=Rat,type='l',layout=c(3,1))
  both <- dl.combine("first.points","last.points")
  rat.both <- direct.label(ratplot,"both")
  print(rat.both)
```r
## same as repeated call to direct.label:
rat.repeated <-
  direct.label(direct.label(ratplot,"last.points"),"first.points")
print(rat.repeated)

## same with ggplot2:
if(require(ggplot2)){
  rp2 <- qplot(
    Time,weight,data=BodyWeight,geom="line",facets=.~Diet,colour=Rat)
  print(direct.label(direct.label(rp2,"last.points"),"first.points"))
  print(direct.label(rp2,"both"))
}

## more complex example: first here is a function for computing the
## lasso path.
mylars <- function
  ## Least angle regression algorithm for calculating lasso solutions.
  (x, y, epsilon=1e-6)
  ## If correlation < epsilon, we are done.
  (xscale <- scale(x) # need to work with standardized variables
  b <- rep(0,ncol(x))# coef vector starts at 0
  names(b) <- colnames(x)
  ycor <- apply(xscale,2,function(xj)sum(xj*y))
  j <- which.max(ycor) # variables in active set, starts with most correlated
  alpha.total <- 0
  out <- data.frame()
  while(1){## lar loop
    xak <- xscale[,j] # current variables
    r <- y-xscale%*%b # current residual
    ## direction of parameter evolution
    delta <- solve(t(xak)%*%xak)%*%t(xak)%*%r
    ## Current correlations (actually dot product)
    intercept <- apply(xscale,2,function(xk)sum(r*xk))
    ## current rate of change of correlations
    z <- xak%*%delta
    slope <- apply(xscale,2,function(xk)-sum(z*xk))
    ## store current values of parameters and correlation
    out <- rbind(out,data.frame(variable=colnames(x),
      coef=b, 
      corr=abs(intercept), 
      alpha=alpha.total, 
      arclength=sum(abs(b)), 
      coef.unscaled=b/attr(xscale,"scaled:scale")))
    if(sum(abs(intercept)) < epsilon)#corr==0 so we are done
      return(transform(out,s=arclength/max(arclength)))
    ## If there are more variables we can enter into the regression,
    ## then see which one will cross the highest correlation line
  }
  out
}
## dl.jitter

### Usage

```r
dl.jitter(d, ...)
```

### Description

Jitter the label positions.

### Examples

```r
dl.jitter()
```
dl.move

Manually move a direct label

Description

Sometimes there is 1 label that is placed oddly by another Positioning Function. This function can be used to manually place that label in a good spot.

Usage

dl.move(group, x, y,
   ...)

Arguments

- group: Group to change.
- x: Horizontal position of the new label.
- y: Vertical position of the new label. If missing(y) and !missing(x) then we will calculate a new y value using linear interpolation.
- ...: Variables to change for the specified group

Value

A Positioning Function that moves a label into a good spot.

Author(s)

Toby Dylan Hocking

Examples

```r
if(require(ggplot2)){
  library(lattice)
  scatter <- xyplot(jitter(cty)~jitter(hwy),mpg,groups=class,aspect=1)
  dcompare(list(scatter),
    list("extreme.grid",
      "+dl.move"=list(extreme.grid,dl.move("suv",15,15))))
  p <- qplot(log10(gamma),rate,data=svmtrain,group=data,colour=data,
    geom="line",facets=replicate~nu)
```
adjust.kif <- dl.move("KIF11", -0.9, hjust=1, vjust=1)
dlcompare(list(p+xlim(-8,7)),
  list("last.points",
    `+dl.move` = list(last.points, adjust.kif)))
}

dl.summarize          dl summarize

Description
summarize which preserves important columns for direct labels.

Usage
dl.summarize(OLD, ...)

Arguments
OLD               data frame
...

Author(s)
Toby Dylan Hocking

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.
dlcompare

Author(s)
Toby Dylan Hocking

Examples

```r
complicated <- list(dl.trans(x=x+10),
    gapply.fun(d[-2,]),
    rot=c(30,180))

library(lattice)
direct.label(dotplot(VADeaths,type="o"),complicated,TRUE)
```

dlcompare  

Direct label comparison plot

Description
Compare several plots and/or label placement methods. This creates a custom grid graphics display based on lattice and/or ggplot2 output. Plots will be on the columns and positioning methods will be on the rows.

Usage
```
dlcompare(plots, pos.funs,
    rects = TRUE, row.items = "plots",
    debug = FALSE)
```

Arguments
- **plots** List of ggplot2 or lattice plots. List names will be used to annotate the plot.
- **pos.funs** List of label placement methods to apply to each plot. List names, or function names if specified as character strings, will be used to annotate the plot.
- **rects** Draw rectangles around each plot, creating a grid?
- **row.items** If "plots" then put plots on the rows and method on the columns. Otherwise, do the opposite.
- **debug** Show debug output?

Author(s)
Toby Dylan Hocking
Examples

library(lattice)
oldopt <- lattice.options(panel.error=NULL)

## Compare two plots of the same data using lattice and ggplot2.
deads.by.sex <- list(male=mdeaths, female=fdeaths)
deads.list <- list()
for(sex in names(deaths.by.sex)){
  deaths.ts <- deaths.by.sex[[sex]]
  deaths.list[[sex]] <-
    data.frame(year=as.numeric(time(deaths.ts)),
                sex,
                deaths=as.integer(deaths.ts))
}
deaths <- do.call(rbind, deaths.list)
deads.plot.list <-
  list(lattice=xyplot(deaths~year,deaths,groups=sex,type="l"))
if(require(ggplot2)){
  deaths.plot.list$ggplot2 <-
    qplot(year,deaths,data=deaths,colour=sex,geom="line")
}

if(names(dev.cur())!="postscript"){
  ## to avoid error on pkg check.
  ## Use some exotic labeling options with different rotation, font
  ## face, family, and alpha transparency.
  exotic <- list("last.points",
                rot=c(0,180),
                fontsize=c(10,20),
                fontface=c("bold","italic"),
                fontfamily=c("mono","serif"),
                alpha=c(0.25,1))
  dlcompare(deaths.plot.list, list(exotic))
}

lattice.options(oldopt)

## Compare a legend with direct labels on the same plot.
library(nlme)
if(require(ggplot2)){
  ggrat <- qplot(Time,weight,data=BodyWeight,
                colour=Rat,geom="line",facets=.~Diet)
  pfuns <- list("legend","direct labels"="last.qp")
  dlcompare(list(ggrat),pfuns,rects=FALSE,row.items="posfuns")
}
Description
Positioning Methods for direct labels are supposed to work with only certain plot types. Each Positioning Method is defined in R/file.R and plot examples are found in tests/doc/file/*.R so that we can automatically assemble a database of example plots from the code.

Usage

dldoc(pkgdir = "..")

Arguments

dlgrob(data, method,
   debug = FALSE, axes2native = identity,
   ...
)

Arguments

data Data frame including points to plot in native coordinates.
method Positioning Method.
dbg
axes2native
...

Author(s)

Toby Dylan Hocking
**draw.polygons**

**Description**

Draw polygons around label positions.

**Usage**

```
draw.polygons(d, ...)```

**Arguments**

- `d`
- `...`

**Author(s)**

Toby Dylan Hocking

---

**draw.rects**

**Description**

Positioning Function that draws boxes around label positions. Need to have previously called `calc.boxes`. Does not edit the data frame.

**Usage**

```
draw.rects(d, ...)```

**Arguments**

- `d`
- `...`

**Author(s)**

Toby Dylan Hocking
Description

Process data points using the Positioning Method and draw the resulting direct labels. This is called for every panel with direct labels, every time the plot window is resized.

Usage

## S3 method for class 'dlgrob'
drawDetails(x,
recording)

Arguments

x

The `dlgrob` list object. `x$method` should be a Positioning Method list and `x$data` should be a data.frame with the following variables:

x, y numeric horizontal and vertical positions of direct labels, in native units. These are converted to cm units before applying the Positioning Method.

groups factor that indices the different groups, and colour indicates the corresponding group colour.

hjust and vjust (optional) numeric values usually in [0,1] that control the justification of the text label relative to the x,y position.

rot (optional) numeric value in [0,360] that specifies the degrees which the text should be rotated.

cex, alpha, fontface, fontfamily (optional) passed to gpar.

Additionally, `x$debug` should be set to TRUE or FALSE, and `x$saxestonative` should be a function that converts units shown on the axes to native units of `x$data[,c("x","y")].`

recording

Author(s)

Toby Dylan Hocking
edges.to.outside

Description
Given a list of edges from the convex or alpha hull, and a list of cluster centers, calculate a point near to each cluster on the outside of the hull.

Usage
edges.to.outside(edges, centers, debug = FALSE, ...)

Arguments
edges
centers
depug
...

Author(s)
Toby Dylan Hocking

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 the input data points. Makes use of attr(d,"orig.data").

Usage
empty.grid(d, debug = FALSE, ...)

Arguments
d Data frame of target points on the scatterplot for each label.
ddebug Show debugging info on the plot?
d...
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

---

**enlarge.box**

**enlarge box**

Description

Make text bounding box larger by some amount.

Usage

enlarge.box(d, ...)

Arguments

- d
- ...

Author(s)

Toby Dylan Hocking

---

**extract.plot**

*Extract plot and definition for documentation*

Description

Given an R code file, execute it, store the definition, and save the resulting plot in a variable.

Usage

extract.plot(f)

Arguments

- f R code file with plot example.

Author(s)

Toby Dylan Hocking
extract.posfun

**Extract Positioning Method for documentation**

**Description**

Use inlinedocs to extract comments and definitions from code, then for each item found add the value and its name to the list.

**Usage**

```
extract.posfun(f)
```

**Arguments**

- `f` R code file, which should contain only Positioning Methods that can be used with examples defined in the doc/ subdirectory with the same name.

**Value**

List of lists, each of which describes one Positioning Method defined in `f`.

**Author(s)**

Toby Dylan Hocking

---

extreme.grid

**extreme grid**

**Description**

Label each point cloud near the extremities of the plot region.

**Usage**

```
"extreme.grid"
```
**Description**

Label the points furthest from the middle for each group.

**Usage**

```r
extreme.points(d, ...)
```

**Arguments**

- `d`
- `...`

**Author(s)**

Toby Dylan Hocking

---

**Description**

Find the point on each curve which maximizes the distance to the plot border or to another curve.

**Usage**

```r
far.from.others.borders(all.groups, 
..., debug = FALSE)
```

**Arguments**

- `all.groups`
- `...`
- `debug`

**Author(s)**

Toby Dylan Hocking
**filltemplate**

**Description**

Fill in occurrences of OBJ$item in the file template with the value in R of L$item.

**Usage**

`filltemplate(L, template)`

**Arguments**

- **L**
- **template**

**Author(s)**

Toby Dylan Hocking

---

**first.bumpup**

**Description**

Label first points, bumping labels up if they collide.

**Usage**

```
"first.bumpup"
```

---

**first.points**

**Description**

Positioning Method for the first of a group of points.

**Usage**

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

d
...

Author(s)

Toby Dylan Hocking

first.polygons  first polygons

Description

Draw a speech polygon to the first point.

Usage

"first.polygons"

first.qp  first qp

Description

Label first points from QP solver that ensures labels do not collide.

Usage

"first.qp"
Description

apply a Positioning Method to every group. works like ddply from plyr package, but the grouping column is always called groups, and the Positioning Method is not necessarily a function (but can be).

Usage

gapply(d, method, ..., 
groups = "groups")

Arguments

d data frame with column groups.
method Positioning Method to apply to every group separately.
... additional arguments, passed to Positioning Methods.
groups can also be useful for piece column.

Value

data frame of results after applying FUN to each group in d.

Author(s)

Toby Dylan Hocking

gapply.fun Direct label groups independently

Description

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

Usage

gapply.fun(expr)

Arguments

eexpr 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

Examples

```r
complicated <- list(dl.trans(x=x+10),
                   gapply.fun(d[-2,]),
                   rot=c(30,180))
library(lattice)
direct.label(dotplot(VADeaths,type="o"),complicated,TRUE)
```

---

GeomDl

---

Description

ggproto object implementing direct labels.

Usage

```
"GeomDl"
```

---

geom_dl

---

Description

Geom that will plot direct labels.

Usage

```r
geom_dl(mapping = NULL,
        data = NULL, ...,
        method = stop("must specify method= argument"),
        debug = FALSE, na.rm = TRUE,
        parse = FALSE, stat = "identity",
        position = "identity",
        inherit.aes = TRUE)
```
Arguments

- **mapping**: aes(label=variable_that_will_be_used_as_groups_in_Positioning_Methods).
- **data**: data.frame to start with for direct label computation.
- **method**: Positioning Method for direct label placement, passed to apply.method.
- **debug**: Show directlabels debugging output?
- **na.rm**: passed to params.
- **parse**: parse text labels as plotmath expressions? not yet supported, but I would be open to accepting a PR if somebody wants to implement that.
- **stat**: passed to layer.
- **position**: passed to layer.
- **inherit.aes**: inherit aes from global ggplot definition?

Author(s)

Toby Dylan Hocking

Examples

```r
if(require(ggplot2)){
  vad <- as.data.frame.table(VADeaths)
  names(vad) <- c("age","demographic","deaths")
  ## color + legend
  leg <- ggplot(vad,aes(deaths,age,colour=demographic))+
         geom_line(aes(group=demographic))+
         xlim(8,80)
  print(direct.label(leg,list("last.points",rot=30)))
  ## this is what direct.label is doing internally:
  labeled <- leg+
             geom_dl(aes(label=demographic), method=list("last.points",rot=30))+
             scale_colour_discrete(guide="none")
  print(labeled)
  ## no color, just direct labels!
  p <- ggplot(vad,aes(deaths,age))+
       geom_line(aes(group=demographic))+
       geom_dl(aes(label=demographic),method="top.qp")
  print(p)
  ## add color:
  p+aes(colour=demographic)+
       scale_colour_discrete(guide="none")
  ## add linetype:
  p+aes(linetype=demographic)+
       scale_linetype(guide="none")
  ## no color, just direct labels
  library(nlme)
  bwbase <- ggplot(BodyWeight,aes(Time,weight,label=Rat))+
           geom_line(aes(group=Rat))+
           facet_grid(-Diet)
  ```
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
getLegendVariables

Description
get the aes which are variable in one legend.

Usage
getLegendVariables(mb)

Arguments
mb

Author(s)
Toby Dylan Hocking

ignore.na

Description
Remove rows for which either x or y is NA

Usage
ignore.na(d, ...)

Arguments
d
...

Author(s)
Toby Dylan Hocking
Description

Calculate how many points fall in a box.

Usage

\texttt{in1box}(p, box)

Arguments

\texttt{p}  
\texttt{box}

Author(s)

Toby Dylan Hocking

Description

Calculate which points fall in a box.

Usage

\texttt{in1which}(p, box)

Arguments

\texttt{p}  
\texttt{box}  
\texttt{data frame of points with columns x and y and many rows.}  
\texttt{box}  
\texttt{data frame of 1 row with columns left right top bottom.}

Author(s)

Toby Dylan Hocking
Description

Calculate for each box how many points are inside.

Usage

inside(boxes, points)

Arguments

boxes Data frame of box descriptions, each row is 1 box, need columns left right top bottom.
points Data frame of points, each row is 1 point, need columns x y.

Value

Vector of point counts for each box.

Author(s)

Toby Dylan Hocking

iris.l1.cluster  Clustering of the iris data with the l1 clusterpath

Description

The l1 clustering algorithm from the clusterpath package was applied to the iris dataset and the breakpoints in the solution path are stored in this data frame.

Usage

data(iris.l1.cluster)

Format

A data frame with 9643 observations on the following 8 variables.

row a numeric vector: row of the original iris data matrix
Species a factor with levels setosa versicolor virginica: Species from corresponding row
alpha a numeric vector: the value of the optimal solution.
lambda a numeric vector: the regularization parameter (ie point in the path).
col a factor with levels Sepal.Length Sepal.Width Petal.Length Petal.Width: column from the original iris data.
gamma a factor with levels 0: parameter from clustering.
norm a factor with levels 1 parameter from clustering.
solver a factor with levels path algorithm used for clustering.

Source
clusterpath package

References
clusterpath article

Examples
data(iris.l1.cluster,package="directlabels")
iris.l1.cluster$y <- iris.l1.cluster$alpha
if(require(ggplot2)){
  p <- ggplot(iris.l1.cluster,aes(lambda,y,group=row,colour=Species)) +
    geom_line(alpha=1/4) +
    facet_grid(col~.)
  p2 <- p+xlim(-0.0025,max(iris.l1.cluster$lambda))
  print(direct.label(p2,list(first.points,get.means)))
}

label.endpoints

Description
Make a Positioning Method that labels a certain x value.

Usage
label.endpoints(FUN,
    HJUST)

Arguments
FUN FUN(d$x) should return an index of which point to label. for example you can use which.min or which.max.
HJUST hjust of the labels.

Value
A Positioning Method like first.points or last.points.
**label.pieces**

**Author(s)**
Toby Dylan Hocking

**Description**
Make a Positioning Method that will, for every piece, select points and assign a vjust value.

**Usage**

```
label.pieces(FUN, VJUST)
```

**Arguments**

- **FUN**
- **VJUST**

**lasso.labels**

**Description**
Label points at the zero before the first nonzero y value.

**Usage**

```
"lasso.labels"
```

**last.bumpup**

**Description**
Label last points, bumping labels up if they collide.

**Usage**

```
"last.bumpup"
```
last.points

Description
Positioning Method for the last of a group of points.

Usage
last.points(d, ...)

Arguments
d
...

Author(s)
Toby Dylan Hocking

last.polygons

Description
Draw a speech polygon to the last point.

Usage
"last.polygons"

last.qp

Description
Label last points from QP solver that ensures labels do not collide.

Usage
"last.qp"
**Description**

Some lattice plot functions do some magic in the background to translate the data you give them into the data points that are plotted onscreen. We have to replicate this magic in native coordinate space before applying the Positioning Method in cm space. These functions accomplish this translation.

**Usage**

```
"lattice.translators"
```

**Description**

Extract guides to hide from a ggplot.

**Usage**

```
legends2hide(p)
```

**Arguments**

- `p`

**Value**

NULL if no legends with colour or fill to hide.

**Author(s)**

Toby Dylan Hocking
Description
Positioning Method for 2 groups of longitudinal data. One curve is on top of the other one (on average), so we label the top one at its maximal point, and the bottom one at its minimal point. Vertical justification is chosen to minimize collisions with the other line. This may not work so well for data with high variability, but then again lineplots may not be the best for these data either.

Usage
lines2(d, offset = 0.3, ...)
maxvar.points

Description

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

Usage

maxvar.points(d, ...)

Arguments

d

... 

Author(s)

Toby Dylan Hocking

maxvar.qp

Description

Label first or last points, whichever are more spread out, and use a QP solver to make sure the labels do not collide.

Usage

"maxvar.qp"
merge_recurse

Description
Copied from reshape.

Usage
merge_recurse(dfs, ...)

Arguments
dfs
...

Author(s)
Toby Dylan Hocking

midrange

Description
Point halfway between the min and max

Usage
midrange(x)

Arguments
x

Author(s)
Toby Dylan Hocking
normal.l2.cluster  Clustering of some normal data in 2d with the l2 clusterpath

Description
The l2 clustering algorithm from the clusterpath package was applied to some randomly generated data in 2 dimensions, and the solutions found using the descent algorithm are stored in this data frame.

Usage
data(normal.l2.cluster)

Format
The format is: List of 2 $ pts : data.frame': 320 obs. of 3 variables: ..$ class: Factor w/ 8 levels "1","2","3","4"...: 1 1 1 1 1 1 1 1 ... .. x : num [1:320] -2.73 -3.63 -2.13 -1.27 -2.98 ... .. y ... ..$ lambda: num [1:21760] 0 0 0 0 0 0 0 0 0 0 ... ..$ row ... ..$ gamma : Factor w/ 1 level "0.1": 1 1 1 1 1 1 1 1 1 1 ... ..$ norm : Factor w/ 1 level "2": 1 1 1 1 1 1 1 1 1 1 ... ..$ solver: Factor w/ 1 level "descent.nocheck": 1 1 1 1 1 1 1 1 1 1 ... ..$ attr(*, "data")= num [1:320, 1:2] -2.73 -3.63 ... .. ..- attr(*, "dimnames")=List of 2 .. .. ..$ : NULL .. .. ..$ : chr [1:2] "x" "y" ..- attr(*, "alphacolnames")= chr [1:2] "x" "y" ..- attr(*, "dimnames")= List of 2 .. ..$ : NULL .. ..$ : chr [1:2] "x" "y"

Source
clusterpath package

References
clusterpath article

Examples
data(normal.l2.cluster)
if(require(ggplot2)){
p <- ggplot(normal.l2.cluster$path,aes(x,y))+ geom_path(aes(group=row),colour="grey")+
 geom_point(aes(size=lambda),colour="grey")+
 geom_point(aes(colour=class),data=normal.l2.cluster$pts)+
 coord_equal()
 print(direct.label(p))
}
only.unique.vals  only unique vals

Description
Create a 1-row data.frame consisting of only the columns for which there is only 1 unique value.

Usage
only.unique.vals(d, ...)

Arguments
d
...

Author(s)
Toby Dylan Hocking

outside.ahull  outside ahull

Description
Calculate closest point on the alpha hull with size of the boxes, and put it outside that point.

Usage
outside.ahull(d, ...)

Arguments
d
...

Author(s)
Toby Dylan Hocking
outside.chull

Description
Calculate closest point on the convex hull and put it outside that point. Assume d is the center for each point cloud and then use orig.data to calculate hull.

Usage
outside.chull(d, ...)

Arguments
d
...

Author(s)
Toby Dylan Hocking

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 = lattice::panel.superpose,
type = "p", debug = FALSE,
...)

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        Positioning Method for direct labeling. NULL indicates to choose a Positioning Method based on the panel.groups function.
.panel.superpose
              The panel function to use for drawing data points.
type           Plot type, used for default method dispatch.
debug          passed to dlgrob.
...           passed to real panel function, and to translator.

Author(s)
Toby Dylan Hocking

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)))
## 3 equivalent ways to make the same plot:
library(lattice)
print(direct.label( ## most user-friendly
densityplot(~ppp,loci,groups=type,n=500)
))
print(direct.label( ## exactly the same as above but with specific panel fns
densityplot(~ppp,loci,groups=type,n=500,
            panel=lattice::panel.superpose,
            panel.groups="panel.densityplot")
))
## using panel.superpose.dl as the panel function automatically adds
## direct labels
print(densityplot(~ppp,loci,groups=type,n=500,
            panel=panel.superpose.dl,panel.groups="panel.densityplot")
)

## Exploring custom panel and panel.groups functions
library(nlme)
## Say we want to use a simple linear model to explain rat body weight:
fit <- lm(weight~Time+Diet+Rat,BodyWeight)
bw <- BodyWeight
bw$.fitted <- predict(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(par.settings=simpleTheme(col=c("red","black")))
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,...)  
    lines(x,bw[subscripts,".fitted"],col=col.line,lty=2)
}
pg <- ratxy(panel=lattice::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 Method, since the  
## name of panel.groups is used to infer a default:  
print(direct.label(pg,method="first.qp"))  
print(ratxy(panel=panel.superpose.dl,panel.groups="panel.model",  
    method="first.qp"))

## Custom panel function that draws a box around values:  
panel.line1 <- function(ps=lattice::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.qp"))
print(ratxy(panel=panel.line1(panel.superpose.dl),  
    panel.groups=panel.model,method="first.qp"))
Description

https://github.com/tdhock/directlabels/issues/2 CRAN won’t complain about this version of :::

Usage

pkgFun(fun, pkg = "ggplot2")

Arguments

fun
pkg

Author(s)

Toby Dylan Hocking

Description

Make a Positioning Method that places non-overlapping speech polygons at the first or last points.

Usage

polygon.method(method,
   space, data.col,
   na.col)

Arguments

method
space
data.col
na.col

Author(s)

Toby Dylan Hocking
**Description**

When adding direct labels to a grouped plot, label placement can be specified using a Positioning Method (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 Method(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 Method will return 1 row for each group. Several built-in Positioning Methods are discussed below, but you can also create your own, either from scratch or by using dl.indep and dl.trans.

**Author(s)**

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

**Examples**

```r
## Not run:
### contourplot Positioning Methods
for(p in list({
  ## Example from help(contourplot)
  require(stats)
  require(lattice)
  attach(environmental)
  ozo.m <- loess((ozone^(1/3)) ~ wind * temperature * radiation,
               parametric = c("radiation", "wind"), span = 1, degree = 2)
  w.marginal <- seq(min(wind), max(wind), length.out = 50)
  t.marginal <- seq(min(temperature), max(temperature), length.out = 50)
  r.marginal <- seq(min(radiation), max(radiation), length.out = 4)
  wtr.marginal <- list(wind = w.marginal, temperature = t.marginal,
                       radiation = r.marginal)
  grid <- expand.grid(wtr.marginal)
  grid[, "fit"] <- c(predict(ozo.m, grid))
  detach(environmental)
  library(ggplot2)
  p <- ggplot(grid,aes(wind,temperature,z=fit)) +
      stat_contour(aes(colour=..level..)) +
      facet_wrap(~radiation)
},{
  ## example from help(stat_contour)
  library(reshape2)
  volcano3d <- melt(volcano)
  names(volcano3d) <- c("x", "y", "z")
  library(ggplot2)
  p <- ggplot(volcano3d, aes(x, y, z = z)) +
      stat_contour(aes(colour = ..level..))
}))
```
### densityplot Positioning Methods

```r
for(p in list(
  data(Chem97, package="mlmRev")
  library(lattice)
  p <- densityplot(~gcsescore | gender, Chem97,
    groups = factor(score), layout = c(1, 2),
    n = 500, plot.points = FALSE)
),
  {  
    library(reshape2)
    iris2 <- melt(iris, id = "Species")
    library(lattice)
    p <- densityplot(~value | variable, iris2, groups = Species, scales = "free")
  },
  {  
    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(ggplot2)
    p <- qplot(ppp, data = loci, colour = type, geom = "density")
  })){
  print(direct.label(p, "top.bumptwice"))
  print(direct.label(p, "top.bumpup"))
  print(direct.label(p, "top.points"))
}
```

### dotplot Positioning Methods

```r
for(p in list(
  library(lattice)
  p <- dotplot(VADeaths, xlim = c(8, 85), type = "o")
),
  {  
    vad <- as.data.frame.table(VADeaths)
    names(vad) <- c("age", "demographic", "deaths")
    library(ggplot2)
    p <- qplot(deaths, age, data = vad, group = demographic, geom = "line", colour = demographic) +
      xlim(8, 80)
  })){
  print(direct.label(p, "angled.endpoints"))
  print(direct.label(p, "top.qp"))
}
```

### lineplot Positioning Methods

```r
for(p in list(
  data(BodyWeight, package="nlme")
  library(lattice)
  p <- xyplot(weight ~ Time | Diet, BodyWeight, groups = Rat, type = "l",
    layout = c(3, 1), xlim = c(-10, 75))
),
  {  
    
  })
```
```r
data(Chem97, package = "mlmRev")
library(lattice)
p <- qqmath(~gcsescore | gender, Chem97, groups = factor(score),
     type = c("l", "g"), f.value = ppoints(100))
}
{ data(Chem97, package = "mlmRev")
library(lattice)
p <- qqmath(~gcsescore, Chem97, groups = gender,
     type = c("l", "g"), f.value = ppoints(100))
}
data(prostate, package = "ElemStatLearn")
pros <- subset(prostate, select = -train, train == TRUE)
ycol <- which(names(pros) == "lpsa")
x <- as.matrix(pros[[-ycol]])
y <- pros[[ycol]]
library(lars)
fit <- lars(x, y, type = "lasso")
beta <- scale(coef(fit), FALSE, 1/fit$normx)
arclength <- rowSums(abs(beta))
library(reshape2)
path <- data.frame(melt(beta), arclength)
names(path)[1:3] <- c("step", "variable", "standardized.coef")
library(ggplot2)
p <- ggplot(path, aes(arclength, standardized.coef, colour = variable)) +
     geom_line(aes(group = variable)) +
     ggtitle("LASSO path for prostate cancer data calculated using the LARS") +
xlim(0, 20)
}
data(projectionSeconds, package = "directlabels")
p <- ggplot(projectionSeconds, aes(vector.length / 1e6)) +
     geom_ribbon(aes(ymin = min, ymax = max, fill = method, group = method), alpha = 1/2) +
     geom_line(aes(y = mean, group = method, colour = method)) +
     ggtitle("Projection Time against Vector Length (Sparsity = 10") +
guides(fill = "none") +
ylab("Runtime (s)")
}
#
## complicated ridge regression lineplot ex. fig 3.8 from Elements of
## Statistical Learning, Hastie et al.
myridge <- function(f, data, lambda = c(exp(-seq(-15, 15, l = 200)), 0)) {
    require(MASS)
    require(reshape2)
    fit <- lm.ridge(f, data, lambda = lambda)
    X <- data[!which(names(data) == as.character(f[[2]]))]
    Xs <- svd(scale(X)) ## my d's should come from the scaled matrix
dsq <- Xs$d^2
    ## make the x axis degrees of freedom
    df <- sapply(lambda, function(l) sum(dsq / (dsq + l)))
    D <- data.frame(t(fit$coef), lambda, df) # scaled coefs
```
molt <- melt(0,id=c("lambda","df"))
## add in the points for df=0
limpts <- transform(subset(molt,lambda==0),lambda=Inf,df=0,value=0)
rbind(limpts,molt)
}
data(prostate,package="ElemStatLearn")
pros <- subset(prostate,train==TRUE,select=-train)
m <- myridge(lpsa~.,pros)
library(lattice)
p <- xyplot(value~df,m,groups=variable,type="o",pch="+",
  panel=function(...){
    panel.xyplot(...)
    panel.abline(h=0)
    panel.abline(v=5,col="grey")
  },
xlim=c(-1,9),
  main="Ridge regression shrinks least squares coefficients",
  ylab="scaled coefficients",
  sub="grey line shows coefficients chosen by cross-validation",
  xlab=expression(df(lambda)))

library(ggplot2)
tx <- time(mdeaths)
Time <- ISOdate(floor(tx),round(tx)
uk.lung <- rbind(data.frame(Time,sex="male",deaths=as.integer(mdeaths)),
  data.frame(Time,sex="female",deaths=as.integer(fdeaths)))
p <- qplot(Time,deaths,data=uk.lung,colour=sex,geom="line")+
  xlim(ISOdate(1973,9,1),ISOdate(1980,4,1))
  )(}
  print(direct.label(p,"angled.boxes"))
  print(direct.label(p,"first.bumpup"))
  print(direct.label(p,"first.points"))
  print(direct.label(p,"first.polygons"))
  print(direct.label(p,"first.qp"))
  print(direct.label(p,"lasso.labels"))
  print(direct.label(p,"last.bumpup"))
  print(direct.label(p,"last.points"))
  print(direct.label(p,"last.polygons"))
  print(direct.label(p,"last.qp"))
  print(direct.label(p,"lines2"))
  print(direct.label(p,"maxvar.points"))
  print(direct.label(p,"maxvar.qp"))
)
### scatterplot Positioning Methods
for(p in list(
  data(mpg,package="ggplot2")
  m <- lm(cty~disp,mpg)
  mpgf <- fortify(m,mpg)
  library(lattice)
  library(latticeExtra)
  p <- xyplot(cty~hwy|manufacturer,mpgf,groups=class,aspect="iso",
  )
  print(direct.label(p,"angled.boxes"))
  print(direct.label(p,"first.bumpup"))
  print(direct.label(p,"first.points"))
  print(direct.label(p,"first.polygons"))
  print(direct.label(p,"first.qp"))
  print(direct.label(p,"lasso.labels"))
  print(direct.label(p,"last.bumpup"))
  print(direct.label(p,"last.points"))
  print(direct.label(p,"last.polygons"))
  print(direct.label(p,"last.qp"))
  print(direct.label(p,"lines2"))
  print(direct.label(p,"maxvar.points"))
  print(direct.label(p,"maxvar.qp"))
)
project.onto.segments

---

**Description**

Given a point and a set of line segments representing a convex or alpha hull, calculate the closest point on the segments.
Usage

project.onto.segments(m, h, debug = FALSE, ...)

Arguments

  m  m is 1 row, a center of a point cloud, we need to find the distance to the closest point on each segment of the convex hull.
  h  Data frame describing the line segments of the convex or alpha hull.
  debug
  ... ignored

Author(s)

Toby Dylan Hocking

---

projectionSeconds  Timings of projection algorithms

Description

Timings of seconds for 3 projection algorithms.

Usage

data(projectionSeconds)

Format

A data frame with 603 observations on the following 6 variables.

  vector.length  a numeric vector
  method  a factor with levels Heap Random Sort
  mean  a numeric vector
  sd  a numeric vector
  min  a numeric vector
  max  a numeric vector

Source

Mark Schmidt’s prettyPlot code for MATLAB http://www.di.ens.fr/~mschmidt/Software/prettyPlot.html
qp.labels

Make a Positioning Method for non-overlapping lineplot labels

Description

Use a QP solver to find the best places to put the points on a line, subject to the constraint that they should not overlap.

Usage

qp.labels(target.var, lower.var, upper.var, 
          order.labels = function(d) order(d[, target.var]),
          limits = NULL)

Arguments

target.var Variable name of the label target.
lower.var Variable name of the lower limit of each label bounding box.
upper.var Variable name of the upper limit of each label bounding box.
order.labels Function that takes the data.frame of labels and returns an ordering, like from the order function. That ordering will be used to reorder the rows. This is useful to e.g. break ties when two groups have exactly the same value at the endpoint near the label.
limits Function that takes the data.frame of labels and returns a numeric vector of length 2. If finite, these values will be used to add constraints to the QP: limits[1] is the lower limit for the first label's lower.var, and limits[2] is the upper limit for the last label's upper.var. Or NULL for no limits.

Value

Positioning Method that adjusts target.var so there is no overlap of the label bounding boxes, as specified by upper.var and lower.var.

Author(s)

Toby Dylan Hocking

Examples

SegCost$error <- factor(SegCost$error,c("FP","FN","E","I"))
if(require(ggplot2)){
  fp.fn.colors <- c(FP="skyblue",FN="#E41A1C",I="black",E="black")
  fp.fn.sizes <- c(FP=2.5,FN=2.5,I=1,E=1)
  fp.fn.linetypes <- c(FP="solid",FN="solid",I="dashed",E="solid")
  err.df <- subset(SegCost,type!="Signal")
kplot <- ggplot(err.df,aes(segments,cost)) +
  geom_line(aes(colour=error,size=error,linetype=error)) +
  facet_grid(type~bases.per.probe) +
  scale_linetype_manual(values=fp.fn.linetypes) +
  scale_colour_manual(values=fp.fn.colors) +
  scale_size_manual(values=fp.fn.sizes) +
  scale_x_continuous(limits=c(0,20),breaks=c(1,7,20),minor_breaks=NULL) +
  theme_bw() + theme(panel.margin=grid::unit(0,"lines"))

## The usual ggplot without direct labels.
print(kplot)

## Get rid of legend for direct labels.
no.leg <- kplot+guides(colour="none",linetype="none",size="none")

## Default direct labels.
direct.label(no.leg)

## Explore several options for tiebreaking and limits. First let's
## make a qp.labels Positioning Method that does not tiebreak.
no.tiebreak <- list("first.points",
  "calc.boxes",
  qp.labels("y","bottom","top"))
direct.label(no.leg, no.tiebreak)

## Look at the weird labels in the upper left panel. The E curve is
## above the FN curve, but the labels are the opposite! This is
## because they have the same y value on the first points, which are
## the targets for qp.labels. We need to tiebreak.
qp.break <- qp.labels("y","bottom","top",make.tiebreaker("x","y"))
tiebreak <- list("first.points",
  "calc.boxes",
  "qp.break")
direct.label(no.leg, tiebreak)

## Enlarge the text size and spacing.
tiebreak.big <- list("first.points",
  cex=2,
  "calc.boxes",
  dl.trans(h=1.25*h),
  "calc.borders",
  "qp.break")
direct.label(no.leg, tiebreak.big)

## Even on my big monitor, the FP runs off the bottom of the screen
## in the top panels. To avoid that you can specify a limits
## function.

## Below, the ylims function uses the limits of each panel, so
## labels appear inside the plot region. Also, if you resize your
## window so that it is small, you can see that the text size of the
## labels is decreased until they all fit in the plotting region.
```r
tiebreak.lim <- list("first.points",
    cex=2,
    "calc.boxes",
    dl.trans(h=1.25*h),
    "calc.borders",
    "qp.limited")
direct.label(no.leg, tiebreak.lim)
}
```
rhtmlescape

Description

for standards compliance we should escape &<

Usage

rhtmlescape(code)

Arguments

code R code to be displayed on a HTML page between pre tags.

Value

Standards compliant HTML to display.

Author(s)

Toby Dylan Hocking
Description

20 segmentation models were fit to 2 simulated signals, and several different error measures were used to quantify the model fit.

Usage

data(SegCost)

Format

A data frame with 560 observations on the following 5 variables.

- bases.per.probe: a factor with levels 374 7: the sampling density of the signal.
- segments: numeric: the model complexity measured using number of segments.
- cost: numeric: the cost value.
- error: a factor with levels E FP FN I: what kind of error? FP = False Positive, FN = False Negative, I = Imprecision, E = Error (sum of the other terms).

Source


Description

Search the plot region for a label position near the center of each point cloud.

Usage

"smart.grid"
static.labels

Description
to hard-code label positions...

Usage
static.labels(x, y, groups, ...)  

Arguments
x
y
groups
...

Author(s)
Toby Dylan Hocking

svmtrain

False positive rates from several 1-SVM models

Description
Support Vector Machine density estimation (1-SVM) was applied to a set of negative control samples, and then used to test on a positive control.

Usage
data(svmtrain)

Format
A data frame with 378 observations on the following 5 variables.
replicate a factor with levels 1 2 3, the experimental replicate. We fit 1-SVM models to each replicate separately.
ratem a numeric vector, the percent of observations that were outside the trained model.
data a factor with levels KIF11 test train, which set of observations did we measure. test and train are each 50% random splits of the negative controls in the experiment, and KIF11 is the positive control in the experiment.
gamm a numeric vector, the tuning parameter of the radial basis function kernel.
u a numeric vector, the regularization parameter of the 1-SVM.
top.bumptwice

description

label the tops, bump labels up to avoid other labels, then to the side to avoid collisions with points.

usage

top.bumptwice(d, debug = FALSE, ...)

arguments

  d
  debug
  ...

author(s)

  toby dylan hocking

top.bumpup

description

label the tops, but bump labels up to avoid collisions.

usage

  "top.bumpup"
Description

Positioning Method for the top of a group of points.

Usage

top.pieces(d, ...)

Arguments

d
...

Author(s)

Toby Dylan Hocking

Description

Positioning Method for the top of a group of points.

Usage

top.points(d, ...)

Arguments

d
...

Author(s)

Toby Dylan Hocking
Description
Label points at the top, making sure they don’t collide.

Usage
“top.qp”

Description
Show the ggplot2 legend, for comparison.

Usage
uselegend.ggplot(p, ...)

Arguments
p The ggplot object.
... Ignored.

Author(s)
Toby Dylan Hocking

Description
Add a legend to a trellis plot, for comparison.

Usage
uselegend.trellis(p, ...)
Arguments

p The trellis object.
... Ignored.

Author(s)

Toby Dylan Hocking

Description

Make a Positioning Function from a set of points on a vertical line that will be spaced out using \texttt{qp.labels}.

Usage

vertical.qp(M)

Arguments

M

Author(s)

Toby Dylan Hocking

Description

Point in the middle of the min and max for each group.

Usage

visualcenter(d, ...)

Arguments

d ...

Author(s)

Toby Dylan Hocking
Description

Return the positions of the plot horizontal limits in cm, for use as the limit argument to `qp.labels`.

Usage

`xlimits(...)`

Arguments

...

Author(s)

Toby Dylan Hocking

Description

Return the positions of the plot vertical limits in cm, for use as the limit argument to `qp.labels`.

Usage

`ylimits(...)`

Arguments

...

Author(s)

Toby Dylan Hocking
Index

*Topic datasets
   iris.l1.cluster, 37
   normal.l2.cluster, 45
   projectionSeconds, 56
   SegCost, 61
   svmtrain, 62

ahull.grid, 4
ahull.points, 4
angled.boxes, 4
angled.endpoints, 5
apply.method, 5, 13–15, 33

big.boxes, 6
bottom.pieces, 7
bumpup, 7
calc.borders, 8
calc.boxes, 8, 23, 59
check.for.columns, 9
chull.grid, 9
chull.points, 10
default.ahull, 10
default_picker, 11
defaultpf.ggplot, 11
defaultpf.trellis, 12
direct.label, 12
direct.label.ggplot, 14
direct.label.trellis, 14
directlabels(direct.label), 12
dl.combine, 15
dl.jitter, 17
dl.move, 18
dl.summarize, 19
dl.trans, 19
dlcompare, 20
dl_doc, 21
dl_grob, 22, 24, 48
draw.polygons, 23
draw.rects, 23
drawDetails.dlgrob, 5, 24
drawDetails.dlgrob, 5, 24
edges.to.outside, 25
empty.grid, 25
enlarge.box, 26
extract.plot, 26
extract.posfun, 27
extreme.grid, 27
extreme.points, 28
far.from.others.borders, 28
filltemplate, 29
first.bumpup, 29
first.points, 29, 38
first.polygons, 30
first.qp, 30
gapply, 31
gapply.fun, 31
geom_dl, 32
GeomDl, 32
get.means, 34
getLegendVariables, 35
ignore.na, 35
in1box, 36
in1which, 36
inside, 4, 37, 59
iris.l1.cluster, 37
label.endpoints, 38
label.pieces, 39
lasso.labels, 39
last.bumpup, 39
last.points, 38, 40
last.polygons, 40
last.qp, 40
lattice.translators, 41
legends2hide, 41
lines2, 42

68
INDEX

make.tiebreaker, 42
maxvar.points, 5, 43
maxvar.qp, 43
merge_recurse, 44
midrange, 44

normal.l2.cluster, 45
only.unique.vals, 46
outside.ahull, 46
outside.chull, 47

panel.superpose.dl, /2, 47
pkgFun, 49
polygon.method, 50
positioning.functions, 51
positioning.methods
  (positioning.functions), 51
project.onto.segments, 55
projectionSeconds, 56

qp.labels, 42, 57, 66, 67
reduce.cex.lr, 59
rhtmlescape, 60

SegCost, 61
smart.grid, 61
static.labels, 62
svmtrain, 62

top.bumptwice, 63
top.bumpup, 63
top.pieces, 64
top.points, 64
top.qp, 65

uselegend.ggplot, 65
uselegend.trellis, 65

vertical.qp, 66
visualcenter, 66

xlimits, 67

ylimits, 67