Adding direct labels to plots

Toby Dylan Hocking
toby.hocking@inria.fr
http://directlabels.r-forge.r-project.org

INRIA Sierra, Laboratoire d'Informatique de l'ENS
Mines ParisTech CBIO
INSERM U900 - Institut Curie
Paris, France

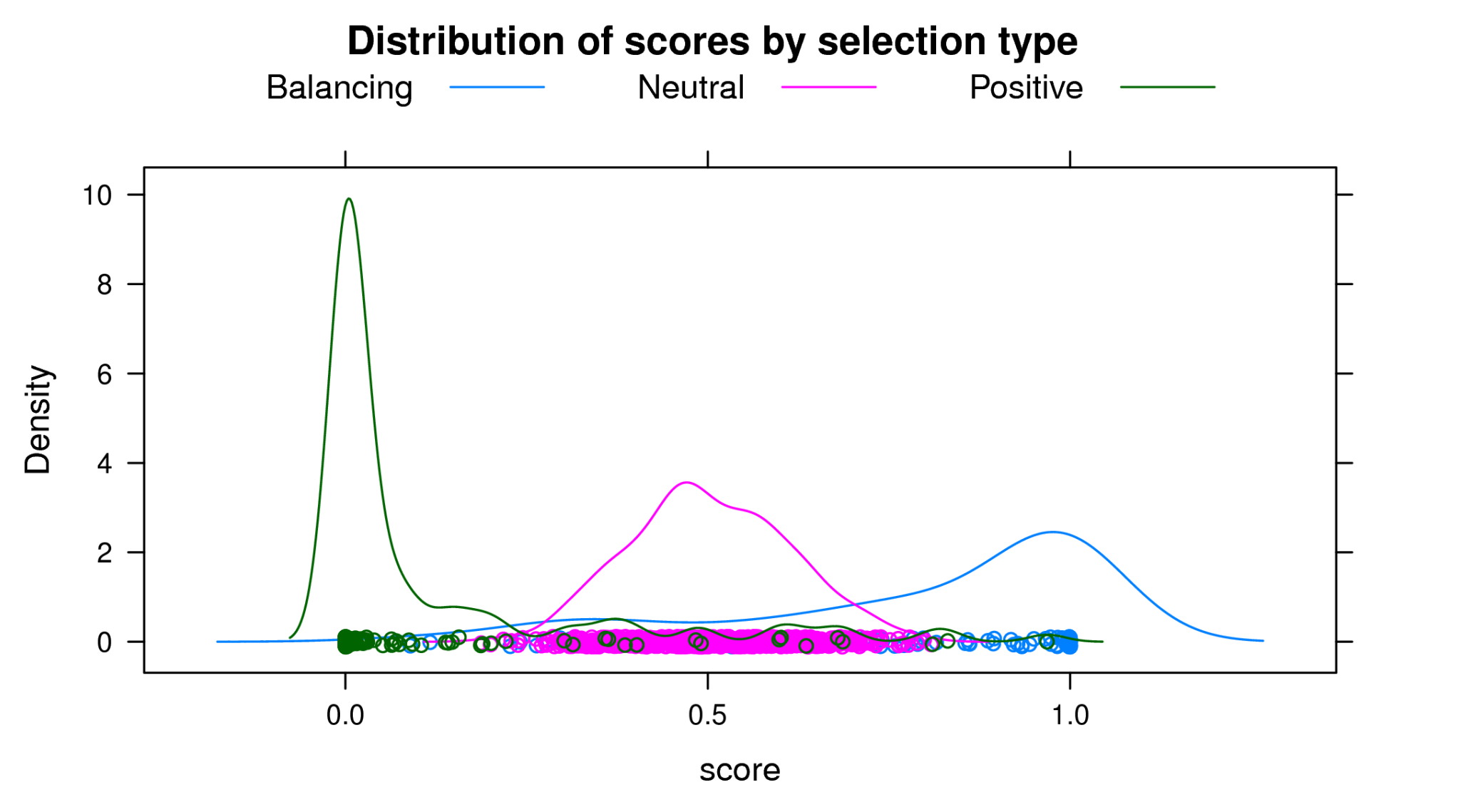
Motivation

- Direct labels are useful when legends are confusing, as below.
- The **`directlabels`** package makes it easy to use direct labels in everyday statistical plots with lattice [1] and ggplot2 [2].

[1] Deepayan Sarkar. Lattice: Multivariate Data Visualization with R. Springer, New York, 2008.
[2] Hadley Wickham. ggplot2: elegant graphics for data analysis. Springer, New York, 2009.

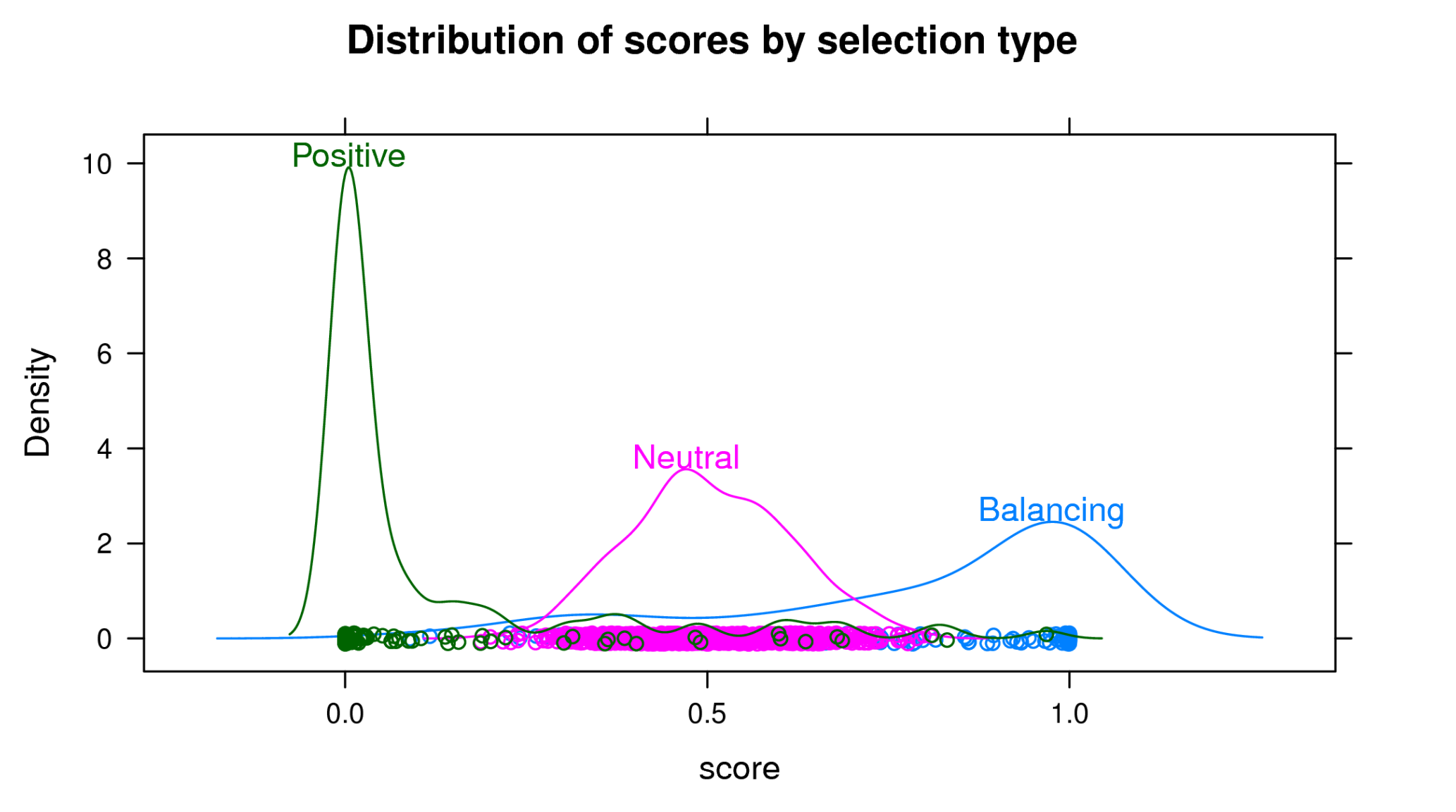
Problem 1: confusing legend!

```
library(lattice)
dens <- densityplot(~score,loci,groups=type,
  auto.key=list(space="top",columns=3))
```



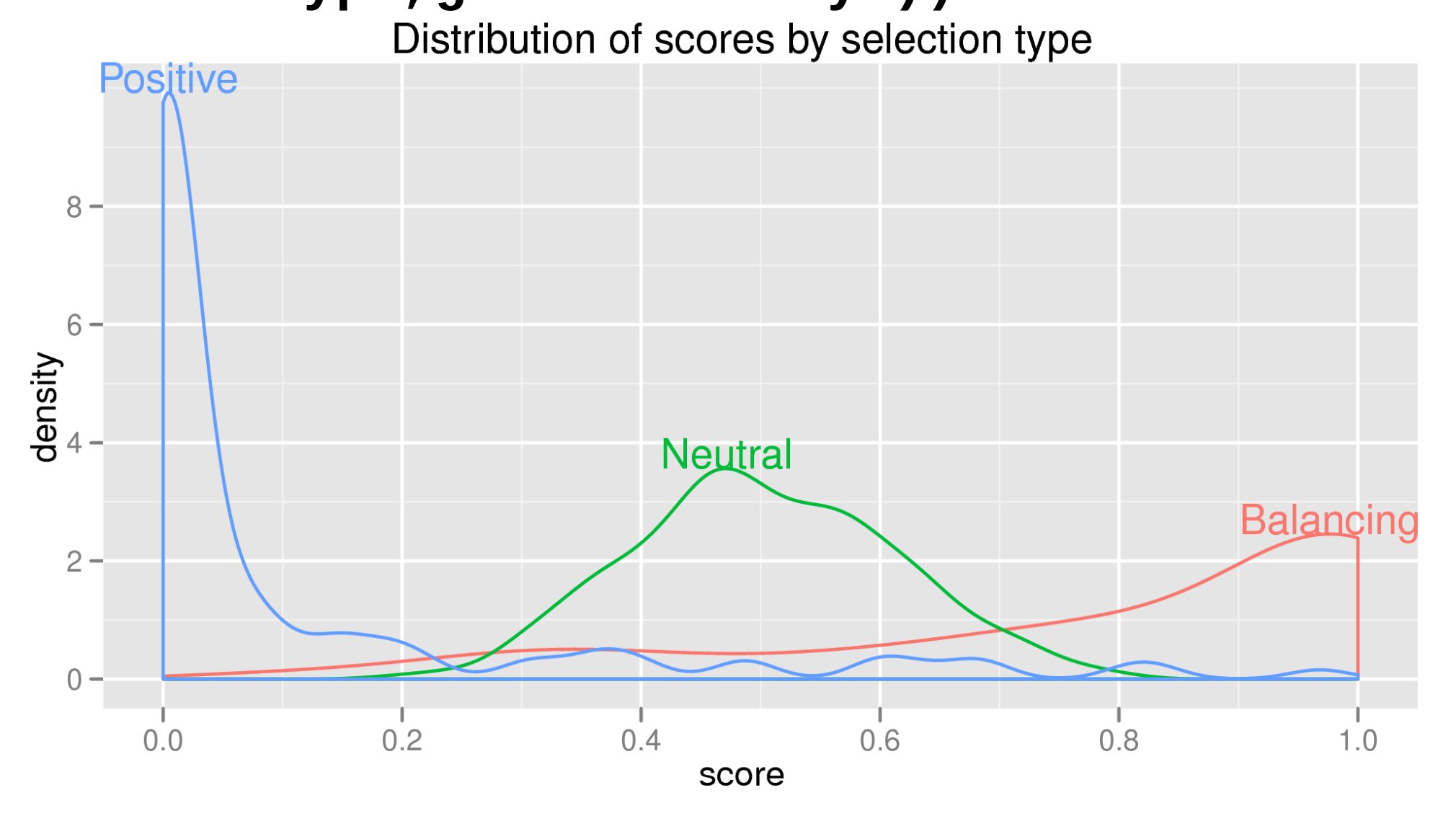
Solution: direct labels.

```
install.packages("directlabels")
library(directlabels)
direct.label(dens)
```



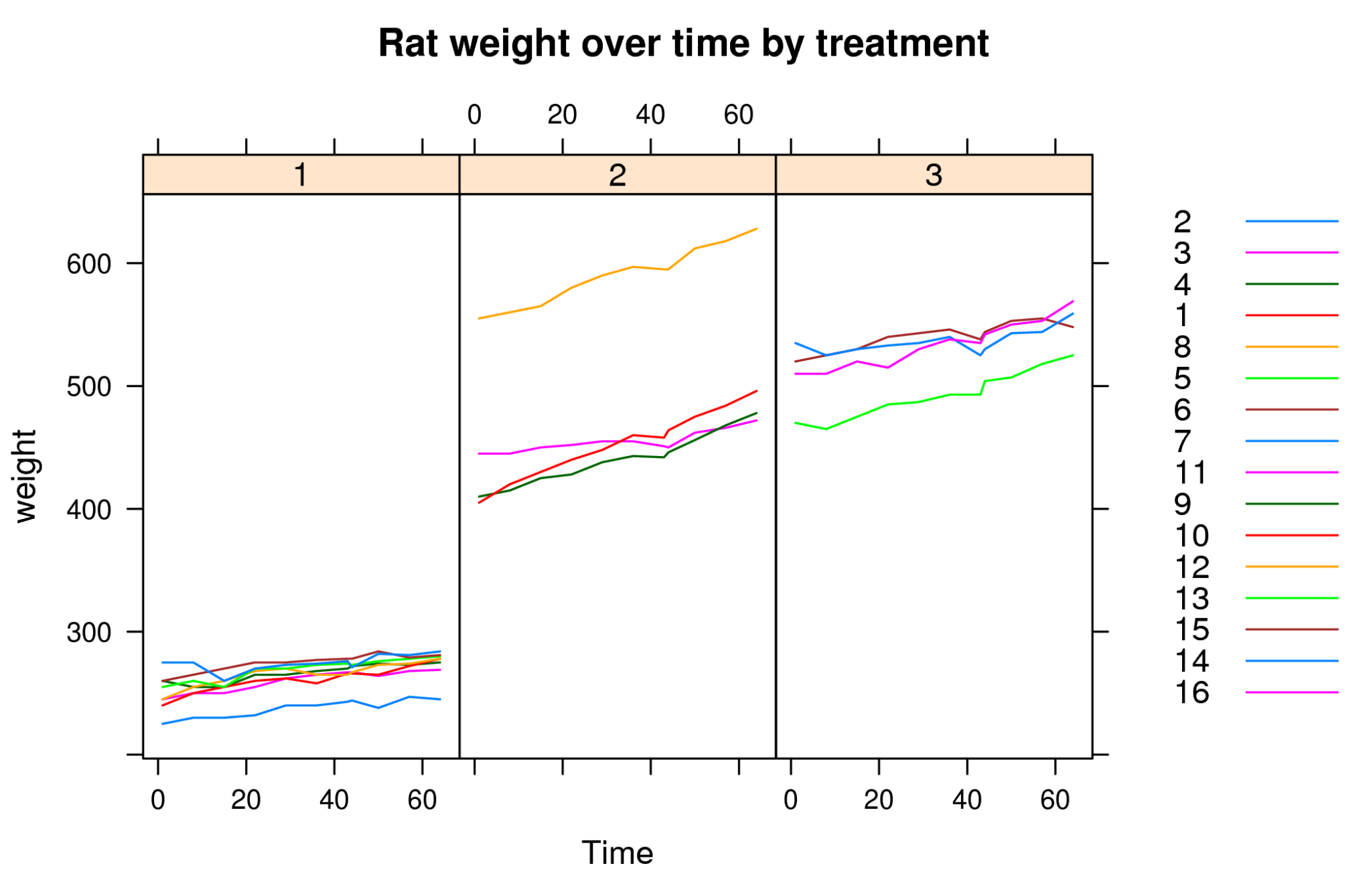
Also works with ggplot2!

```
library(ggplot2)
direct.label(qplot(score,data=loci,
  color=type,geom="density"))
```



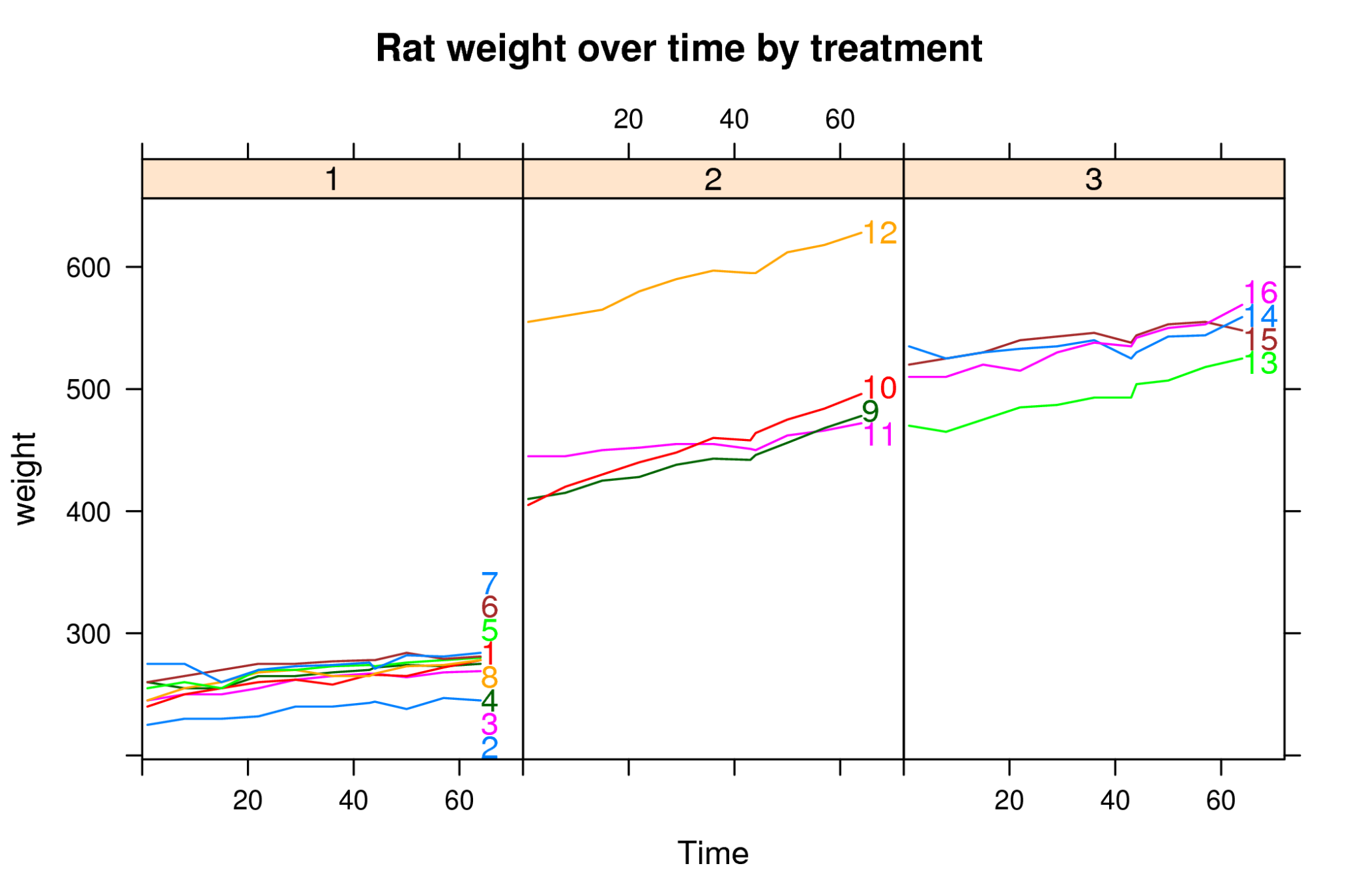
Problem 2: too many legend classes!

```
data(BodyWeight,package="nlme")
ratplot <- xyplot(weight~Time|Diet,BodyWeight,
  groups=Rat,type="l",layout=c(3,1),auto.key=
  list(space="right",points=FALSE,lines=TRUE))
```



Solution: direct labels.

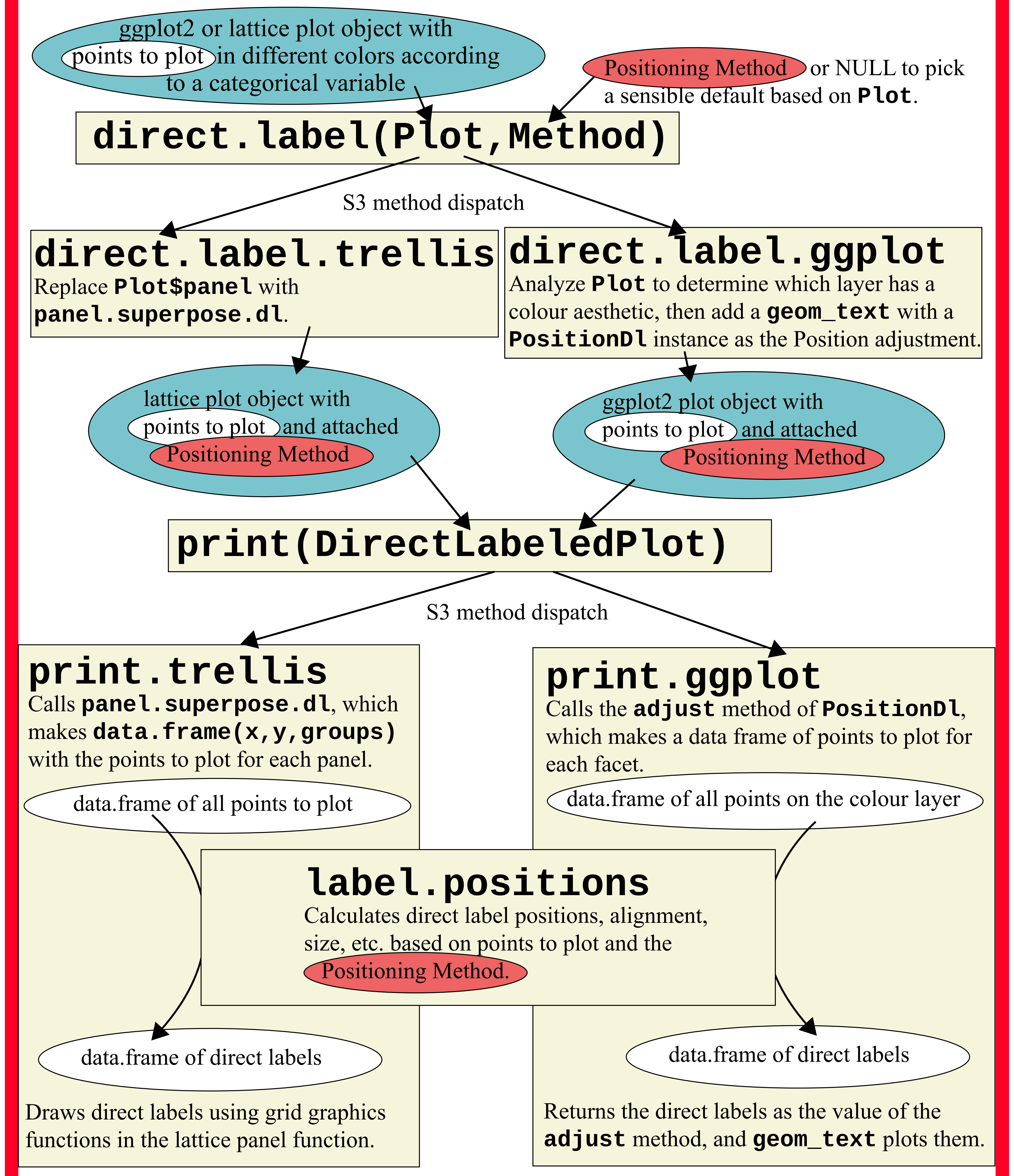
```
direct.label(update(ratplot,xlim=c(0,72)),
                    last.qp)
```



- labels are unambiguous and do not overlap.
- allows perception of group order.

Modular package design

- The **`directlabels`** package assumes the plot is an object and we can extract the data to plot.
- **`direct.label`** S3 methods are implemented for lattice and ggplot2. These methods implement framework-specific plot analysis and drawing.
- Positioning Methods calculate label positions, independently of the plot framework.



ggplot2 or lattice plot object with points to plot in different colors according to a categorical variable

Positioning Method or NULL to pick a sensible default based on **`Plot`**.

`direct.label(Plot,Method)`

S3 method dispatch

`direct.label.trellis`
Replace **`Plot$panel`** with **`panel.superpose.dl`**.

`direct.label.ggplot`
Analyze **`Plot`** to determine which layer has a colour aesthetic, then add a **`geom_text`** with a **`PositionDl`** instance as the Position adjustment.

lattice plot object with points to plot and attached Positioning Method

ggplot2 plot object with points to plot and attached Positioning Method

`print(DirectLabeledPlot)`

S3 method dispatch

`print.trellis`
Calls **`panel.superpose.dl`**, which makes **`data.frame(x,y,groups)`** with the points to plot for each panel.

data.frame of all points to plot

`print.ggplot`
Calls the **`adjust`** method of **`PositionDl`**, which makes a data frame of points to plot for each facet.

data.frame of all points on the colour layer

`label.positions`
Calculates direct label positions, alignment, size, etc. based on points to plot and the Positioning Method.

data.frame of direct labels

Draws direct labels using grid graphics functions in the lattice panel function.

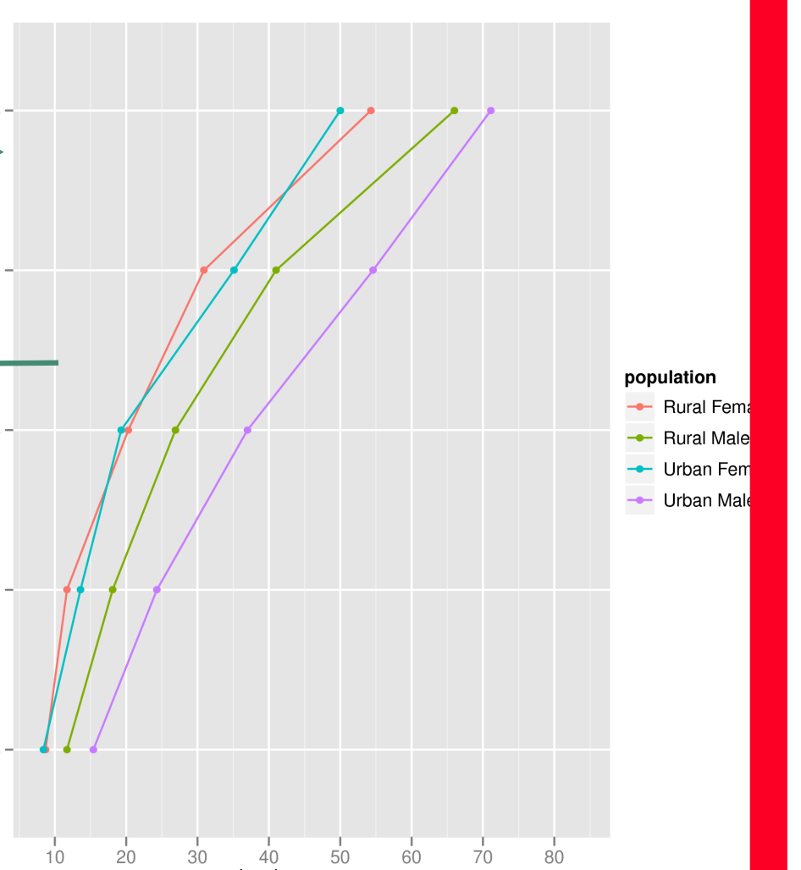
data.frame of direct labels

Returns the direct labels as the value of the **`adjust`** method, and **`geom_text`** plots them.

Positioning Methods

A Positioning Method is a list that describes where to draw the direct labels, based on the data.
- Elements of the Positioning Method are applied sequentially to the data, starting with the data.frame of all points to plot.
- Elements can be functions or named constants.
- The functions must apply some transformation to the data.frame, i.e. the **`endpoints`** function below simply returns the points with the largest x value.
- The **`gapply`** function can be used to apply a Positioning Method to each group of points independently, as in the **`group.endpoints`** function below.
- Named constants are written to the data.frame, as **`rot`** and **`hjust`** below.

```
library(reshape)
VADF <- melt(VADeaths)
names(VADF) <-
  c("age","population","death.rate")
dp <- ggplot(VADF)+
  aes(death.rate,age,colour=population)+
  geom_line(aes(group=population))+
  geom_point()+xlim(8,84)
```



```
  age   population death.rate
50-54   Rural Male       11.7
55-59   Rural Male       18.1
60-64   Rural Male       26.9
65-69   Rural Male       41.0
70-74   Rural Male       66.0
50-54 Rural Female        8.7
  ...          ...        ...
```

adjust

```
  y       groups    x
1.0   Rural Male 11.7
2.0   Rural Male 18.1
3.0   Rural Male 26.9
4.0   Rural Male 41.0
5.0   Rural Male 66.0
1.0 Rural Female  8.7
...          ...  ...
```

```
endpoints <- function(Points,...){
  subset(Points,x==max(x))
}
group.endpoints <- function(Points,...){
  gapply(Points,endpoints)
}
```

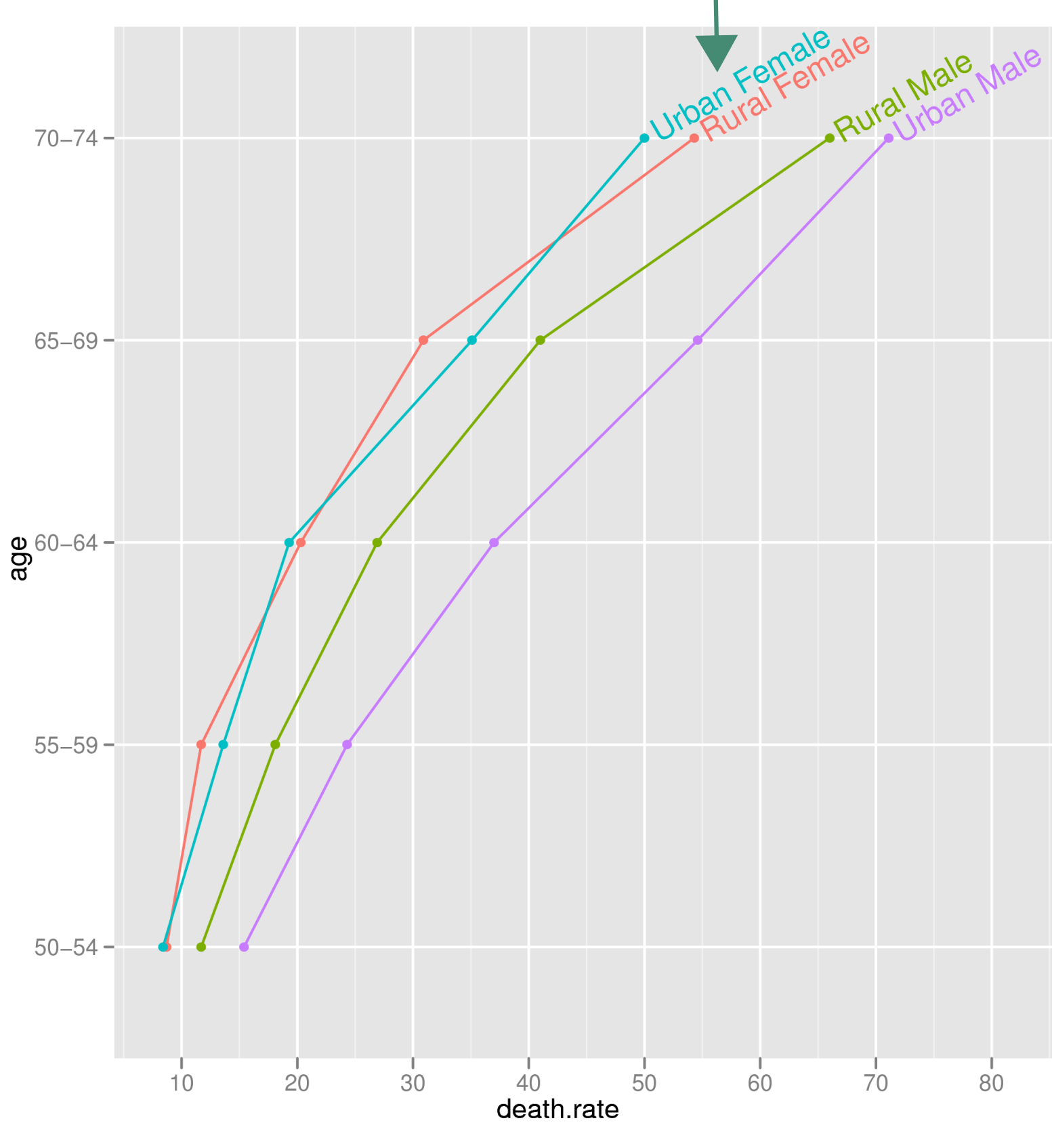
group.endpoints

```
   x       groups y
54.3 Rural Female 5
66.0   Rural Male 5
50.0 Urban Female 5
71.1   Urban Male 5
```

```
direct.label(dp,list("group.endpoints",
  rot=30,hjust=-0.1))
```

list items

```
   x       groups y rot hjust
54.3 Rural Female 5  30  -0.1
66.0   Rural Male 5  30  -0.1
50.0 Urban Female 5  30  -0.1
71.1   Urban Male 5  30  -0.1
```



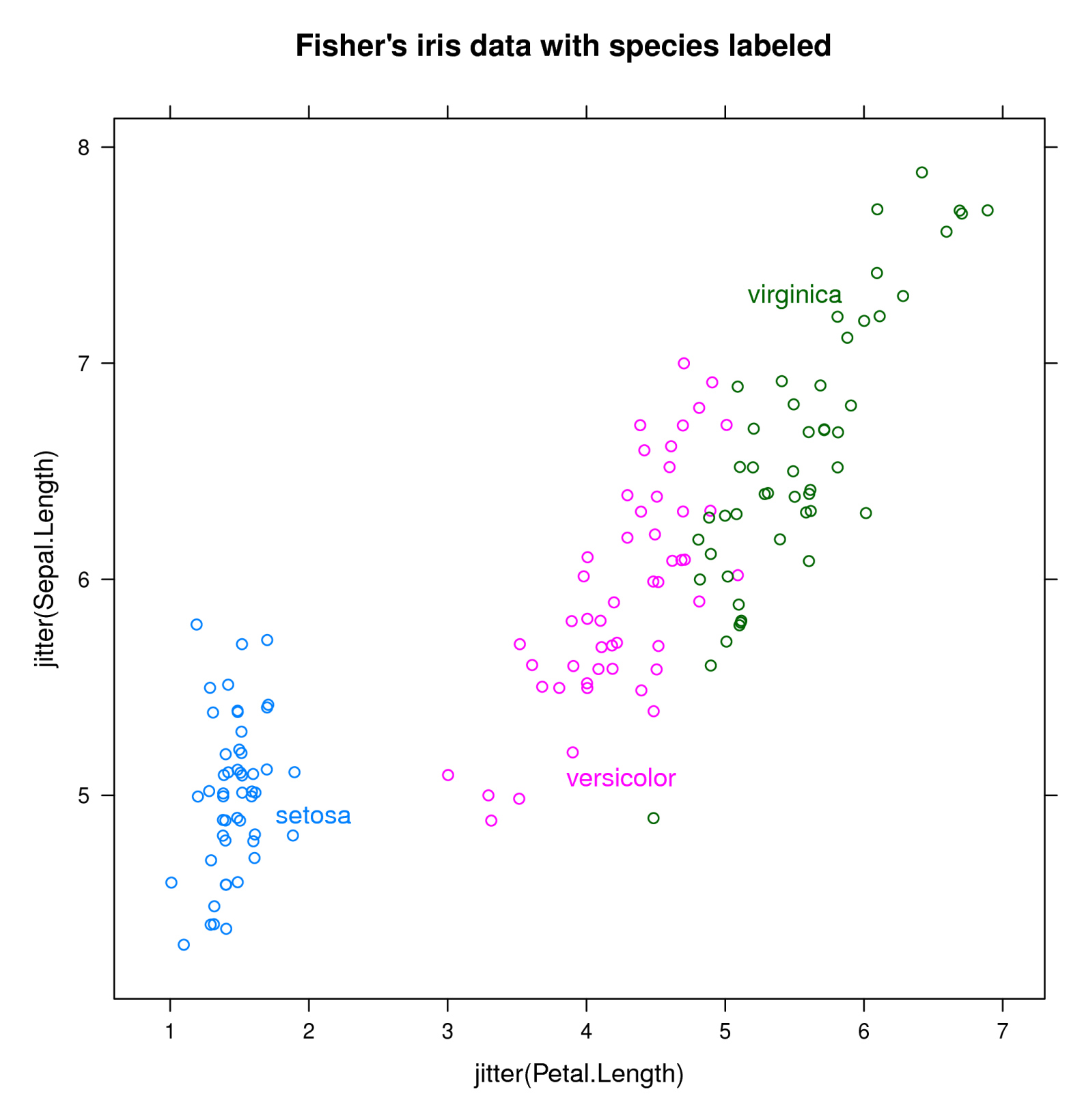
Helper functions

- **`dl.trans(x=x+0.1)`** shifts direct labels to the right.
- **`dl.move("suv",x=20,y=10,hjust=0)`** updates the direct label for the "suv" group.
- **`dl.combine(method1,method2)`** provides direct labels from both methods.
- **`calc.boxes`** calculates bounding boxes for labels, adding columns **`w`**, **`h`**, **`top`**, **`bottom`**, **`right`**, and **`left`**.
Caveat: currently only works for lattice!
- **`draw.rects`** draws grey boxes around the current labels. This is useful for debugging.
- **`ahull.points`** and **`chull.points`** calculate the alpha-hull and convex hull of some points.
- **`project.onto.segments`** finds the closest point on the hull from the mean of a point cloud.

Smart labels

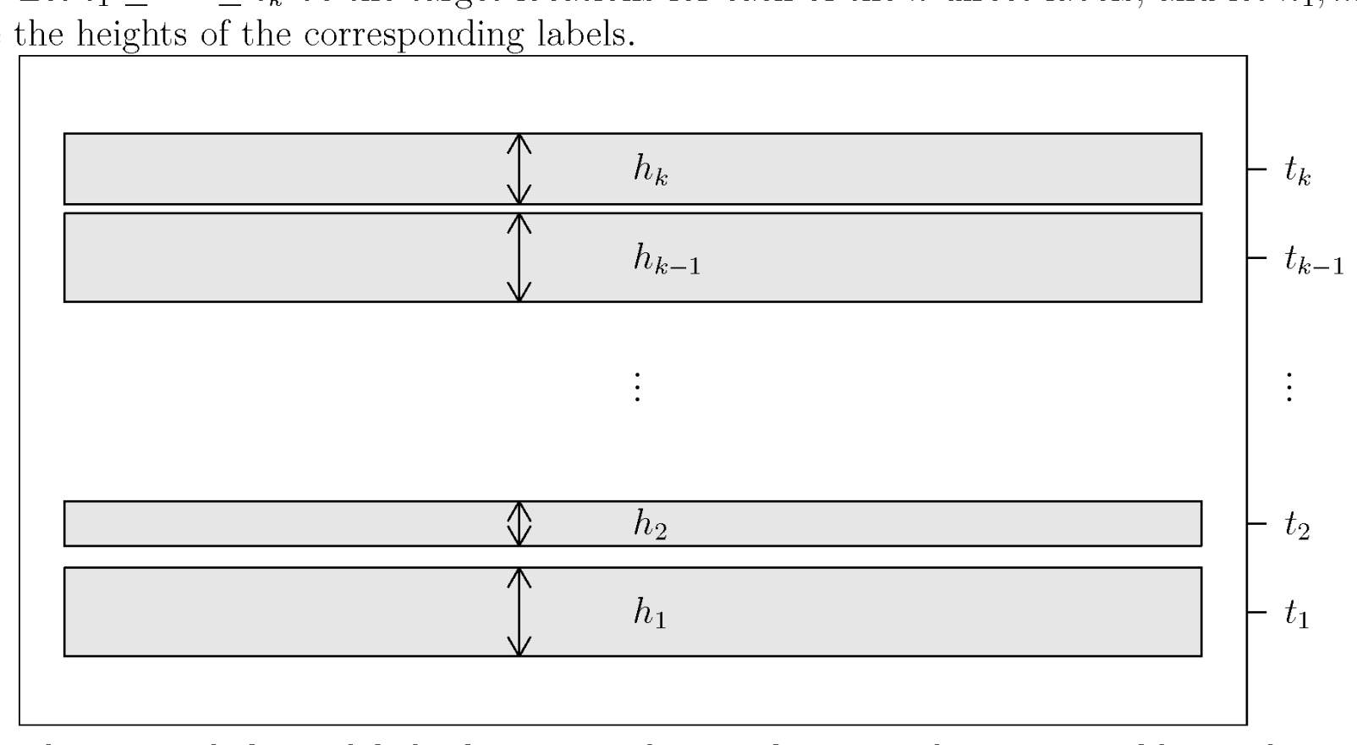
- In lattice panel functions we can calculate the bounding box of each label using grid functions **`stringHeight`**, **`stringWidth`**, **`convertHeight`**, and **`convertWidth`**.
- Smart Positioning Methods take advantage of this information to avoid collisions with points and other labels.
- For example, in the scatterplot below we recursively perform a grid search to find good label positions.

```
iplot <- xyplot(jitter(Sepal.Length)~
  jitter(Petal.Length),iris,
  groups=Species)
direct.label(iplot,smart.grid)
```



Optimal labels

Let $t_1 \leq \cdots \leq t_k$ be the target locations for each of the k direct labels, and let $h_1, ..., h_k$ be the heights of the corresponding labels.



The optimal direct labels do not overlap, and are as close as possible to the target locations:

$$\min_{b\in\mathbb{R}^k} \quad \sum_{i=1}^k (b_i - t_i)^2 = ||b-t||^2 \qquad (1)$$
$$\text{subject to} \quad b_{i+1} \geq b_i + h_{i+1}/2 + h_i/2,\ \forall\ i = 1, ..., k-1$$

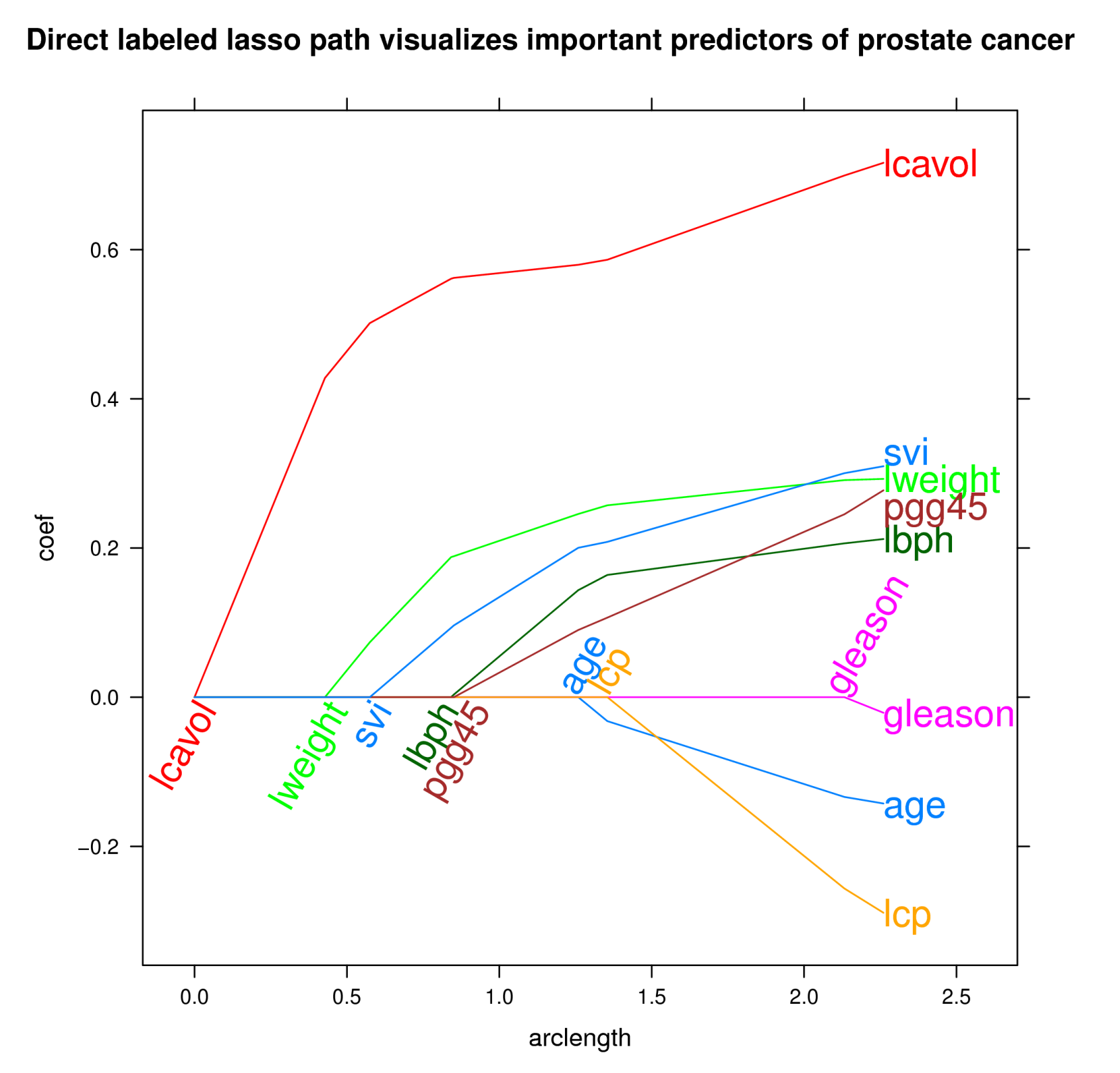
This is a quadratic program (QP) that we can solve using `quadprog::solve.QP()` and we can use the optimal b for the direct label positions. To use the solver, we must write the QP in standard form:

$$\min_{b\in\mathbb{R}^k} \quad \frac{1}{2}b'b - t'b$$
$$\text{subject to} \quad A'b = \begin{bmatrix} -1 & 1 & 0 & & & \\ 0 & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & 0 \\ & & & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} \geq \underbrace{\begin{bmatrix} (h_1+h_2)/2 \\ \vdots \\ (h_{k-1}+h_k)/2 \end{bmatrix}}_{\texttt{(h[-k]+h[-1])/2}} \qquad (2)$$

where A is the $k \times k-1$ constraint coefficient matrix:

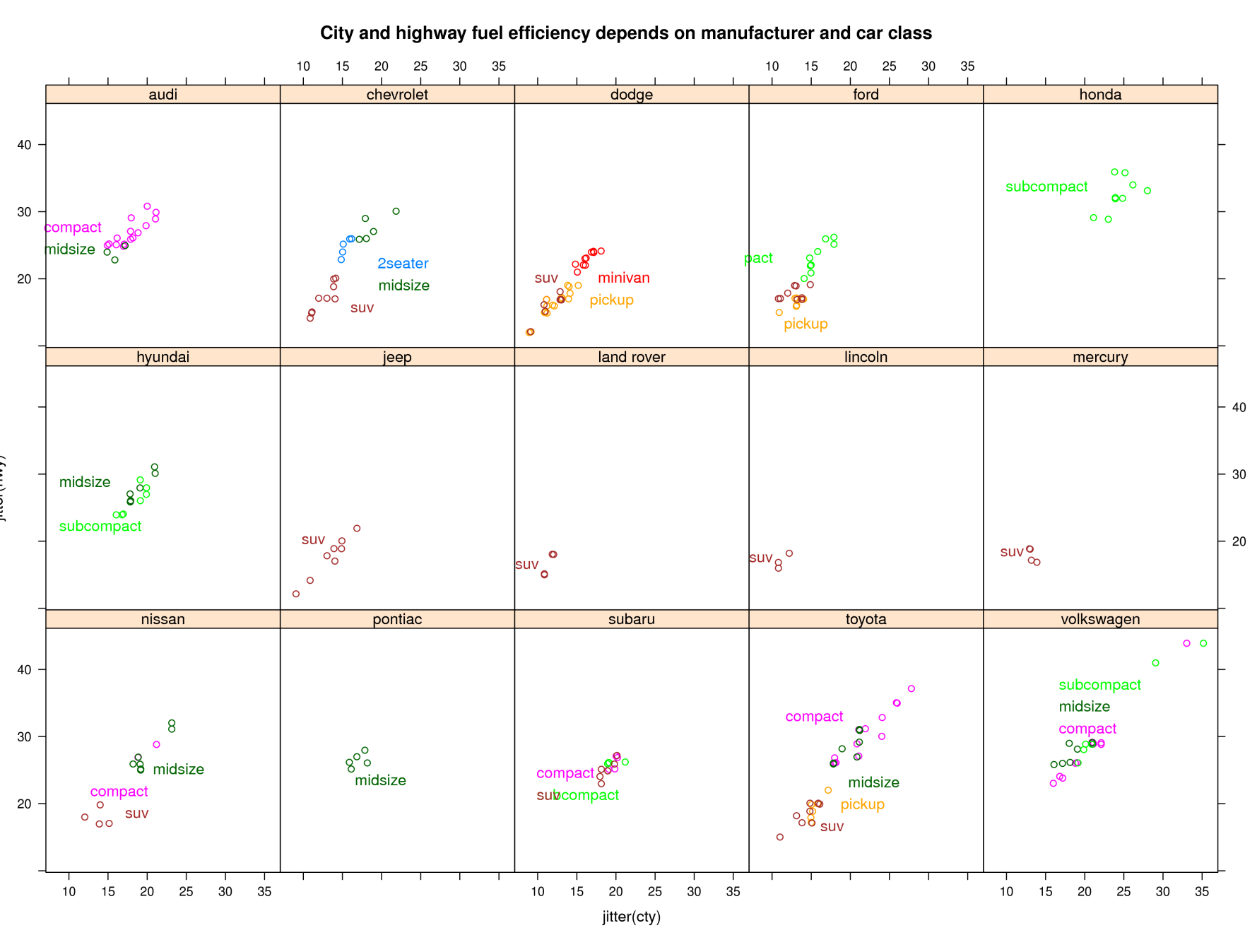
$$A = \begin{bmatrix} -1 & 0 & & & \\ 1 & -1 & & & \\ 0 & 1 & \ddots & & \\ & & \ddots & -1 & 0 \\ & & & 1 & -1 \\ & & & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \cdots & 0 \\ 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}}_{\texttt{rbind(0,}I_{k-1}\texttt{)}} - \underbrace{\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \\ 0 & \cdots & 0 \end{bmatrix}}_{\texttt{rbind(}I_{k-1}\texttt{,0)}} \qquad (3)$$

```
path <- xyplot(coef~arclength,
  prostate.path,groups=variable,type="l")
direct.label(path,
  dl.combine(lasso.labels,last.qp))
```



Limitation: cluttered panels

Legends preferable for multipanel displays, unless only a subset of groups appears in every panel.



Future work

- ggplot2 support for fontface and fontfamily options?
- ggplot2 support for Smart Positioning Methods? Write a custom grid grob that recalculates position when redrawn?
- automatic scale adjustment for direct label visibility?