**Motivation**

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


**Problem 1: confusing legend!**

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

**Solution:**
Library `directlabels`.

```r
library(directlabels)
direct.label(dens)
```

**Also works with ggplot2!**

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

**Problem 2: too many legend classes!**

```r
catplot <- xyplot(weight~Time| Diet, BodyWeight, groups=Mat, type~"l", layout=c(3,1), auto.key=list(space="right", point="FACE", lines="TNULL")
```

**Solution:**
Direct labels.

```r
direct.label(updata=catplot, all=TRUE, last=gp)
```

**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.

```r
direct.label(Plot, Method)
```

**Smart labels**

- In lattice panel functions we can calculate the bounding box of each label using grid functions: `stringWeight`, `stringLength`, `convertString`, `convertStringWidth`.
- 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.

```r
print(adjust)
```

**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 `gapply` function below.
- Named constants are written to the data frame, as `rot` and `hjust` below.

```r
newpoints <- function(x)
  c(x=max(x, na.rm=T), y=0, labels=c("Max", "Min")
```

**Helper functions**

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

**Optimal labels**

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