About

iPlots is a package for **R** statistical environment (see www.r-project.org) which provides high interaction statistical graphics, written in Java. It offers a wide variety of plots, such as histograms, barcharts, scatterplots, boxplots, fluctuation diagrams, parallel coordinates plots or spineplots. All plots support many interactive features, such as linked highlighting, color brushing, multiple views and interactive change of parameters. The **iPlots** was introduced at the DSC-2003 (http://www.ci.tuwien.ac.at/Conferences/DSC-2003/). The various additions of Version 2.0 have been introduced at the userR!2006 (http://www.r-project.org/useR-2006/) Conference.

Beside interactive plots, iPlots also provide API for managing plots and adding user-defined objects, such as lines or polygons to the plot.

Version 2.0 not only adds new multivariate plots and interactive features throughout all plots, but also lays the foundations for interactive **customizable** plots (ICP). A preview version of ICPs will be available soon!

News:

- **2006/10/03** Launch of new website, more documentation will be added gradually.
- **2006/08/20** Released **iplots_1.0-3** (Version 2.0) and binaries for R 2.3.1. iPlots are now also officially available from CRAN.
- **2006/05/05** Released **iplots_0.2-1** (including Windows and Mac binaries for R 2.3.0) to go along with JGR. Please note that iplots are under heavy construction now and features are popping up on a daily basis, so you may want to update the package evey now and then even if there is no official release. Next major release is expected for the useR!2006.
- **2005/01/25** Updated web-pages; latest versions of iPlots are available from our R repositories and binaries as a part of JGR distribution.
- **2004/06/26** iPlots are now by default delivered with JGR (Java GUI for R). It is recommended to use iPlots with JGR, because iPlots are seamlessly integrated into JGR.

Further Reading:

- **DSC 2003**
  - Talk
  - Paper
- **Newsletter**
  - Paper
- **useR!2006**
  - Talk

Known Bugs and Features:

- View menus need to be cleaned up and consolodated with the context menus.
- More plot parameters need to be exposed on R level.
- Some plots on some platform don't update decently after creation.
- …

Plots

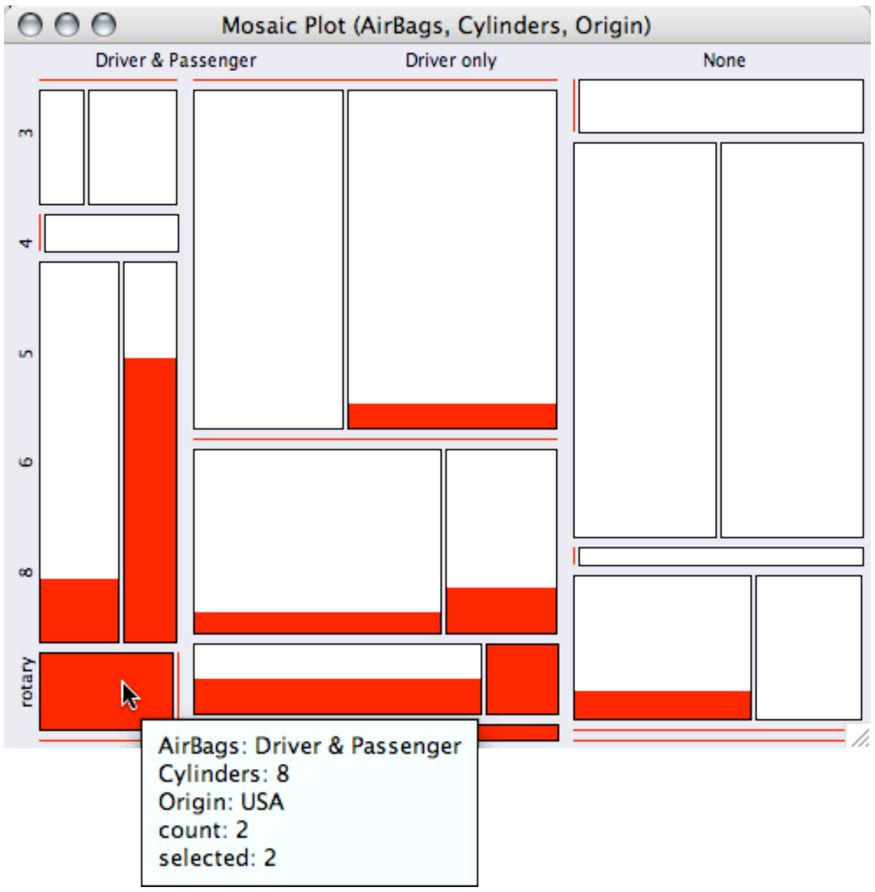
(Right now, we mostly use the help-file examples - feel free to contribute!)

Mosaic Plot

Mosaic plots in iPlots are fully interactive, i.e. they not only allow the standard selection, highlighting and color brushing but also to rearrange variables and to include and exclude variables.

```
> library(MASS)
> data(Cars93)
> attach(Cars93)
> imosaic(data.frame(AirBags,Cylinders,Origin))
```

Here is what you will get:



AirBags: Driver & Passenger
Cylinders: 8
Origin: USA
count: 2
selected: 2

(to make the plot more interesting, we linked all cars with more then 200 hp. The query allows to get exact info on the cell)

Besides the standard mosaic plot three further variations are implemented in iplots, which are:

1. Same Binsize
2. Fluctuation Diagram
3. Multiple Barchart
4. (Model View)

To switch between the different representations, use the context menu. All the variations have their particular strengths and weaknesses, so you will need some experience to use them most efficiently.

To rearrange the variabels use the four arrow key ... you will soon get the idea how to manage the variables!

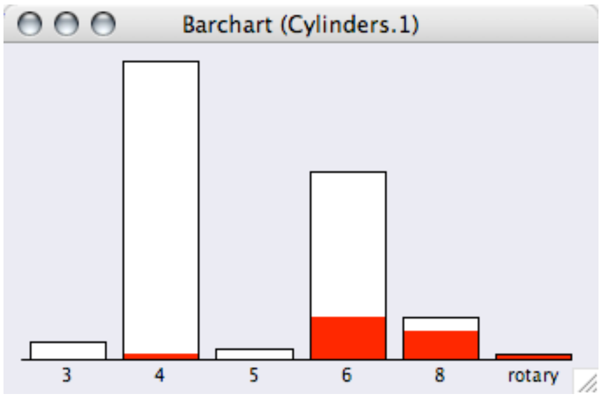
Since most attempts to optimally place labels in mosaic plots fail sooner or later, we use a simple linear spacing for the first variables. Any further information should be queried using <ctrl>-mouse-over.

Barcharts

Barcharts in iPlots also feature Spineplots (use ctrl-s or the "View" menu to switch between the two representations)

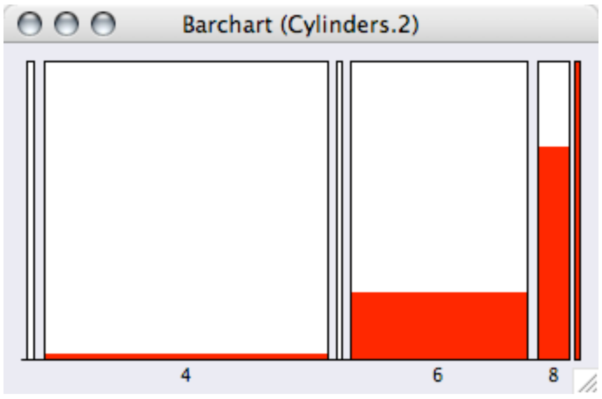
```
> library(MASS)
> data(Cars93)
> attach(Cars93)
> ibar(Cylinders)
```

yields



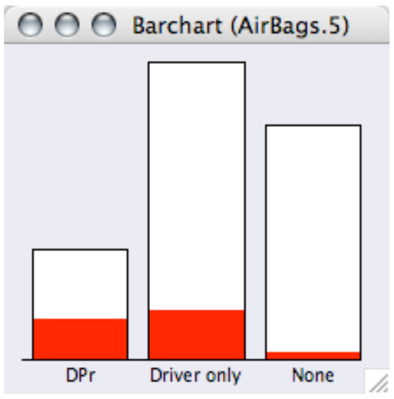
To get a spineplot via the command line, there are two version:

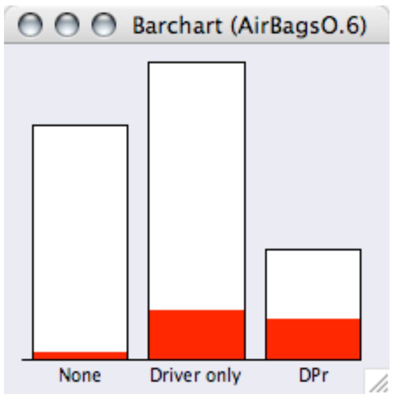
```
# directly upon creation

> ibar(Cylinders, isSpine=T)

# or switching by changing the plot options of the barchart
# (if no plot is specified, the ibar must be the current plot)

> iplot.opt(isSpine=T)
```



Bars can be reordered by either using the options in the "View" menu or by simply dragging the bars to the new position.
iPlots respect the order of a factor, so orderings from within R can be used to order the categories in a barchart.

```
> levels(AirBags)
[1] "Driver & Passenger" "Driver only"    "None"

> AirBagsO <- ordered(AirBags,
            c("None", "Driver only", "Driver & Passenger"))

> levels(AirBagsO)
[1] "None"                "Driver only"    "Driver & Passenger"

> ibar(AirBags)
ID:9 Name: "Barchart (AirBags.5)"

> ibar(AirBagsO)
ID:10 Name: "Barchart (AirBagsO.6)"
```





Maps

Interactive maps - which use the data-format of the **maptools** package - will be availabe soon!

Parallel Plots

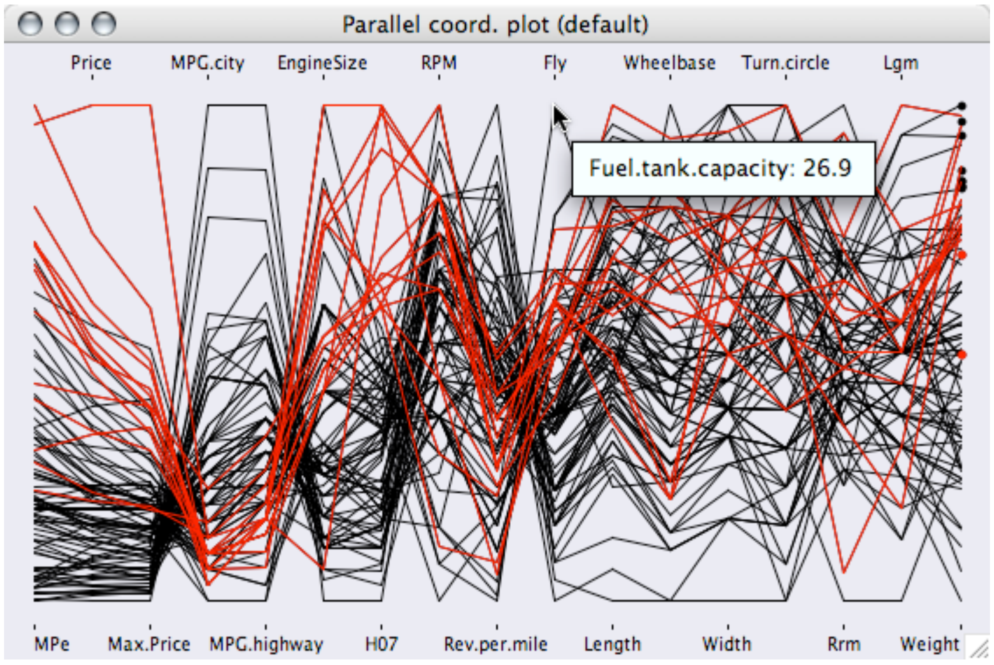
There is a whole family of parallel plots in iPlots.

I. Parallel Coordinate Plot

A parallel coordinate plot connects all cases by lines.

```
# Make a PCP for all continuous variables ...

> ipcp(Cars93[c(4:8,12:15,17,19:25)])
```

The default plot will scale all variables between min and max.



There are various options in the "View" menu like

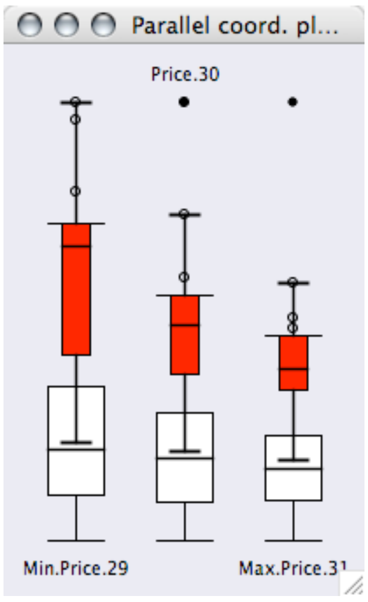
- Common and individual scale
- Show or hide dots or lines (this allows to draw a parallel dotplot)
- Show only selected cases

Most other options are general options and will be discussed in the conventions section.

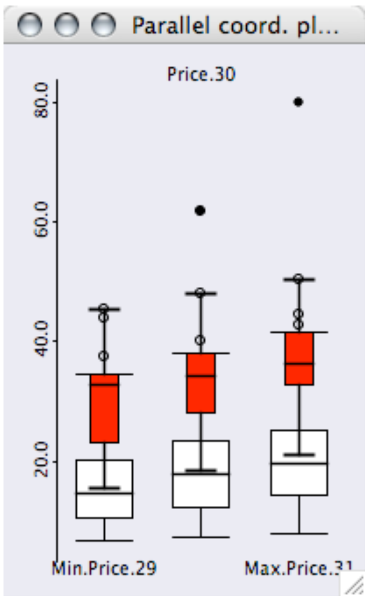
II. Parallel Boxplot

Parallel boxplots are quite similar to PCPs but feature statistics like the median and the hinges, which makes them more useful for comparing variables.

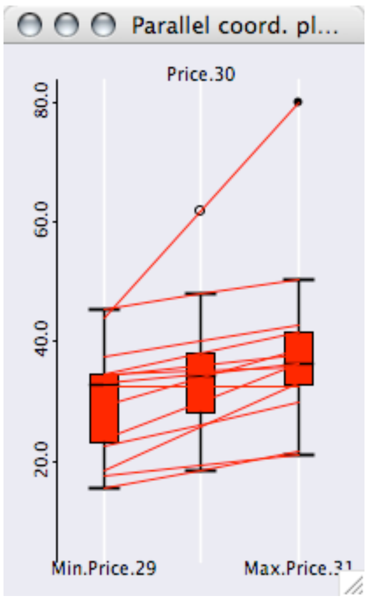
```
# Make a parallel boxplot for all price variables ...

> ibox(Cars93[4:6])
```



Parallel boxplot for the three prices in the dataset. Each variable uses the full range.



Using a common scale in the parallel boxplot gives sensible results:
max price > price > min price



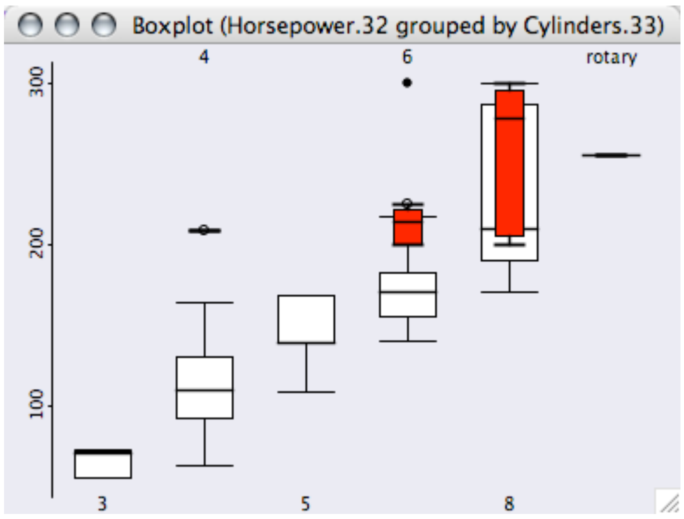
Showing only selected cases allows to additionally display the corresponding lines.

All options can be found in the "View" menu of the plot.
(Note, that the scale is only displayed when all variables share the same scale!)

III. Boxplot y by x

The boxplot y by x is quite different compared to the two other parallel plots, as it is a condition plot, which shows boxplots by group. If ibox() is called with a continuous variable and a factor, a boxplot y by x is created

```
# split the boxplot for horsepower by number of cylinders

> ibox(Horsepower, Cylinders)
```

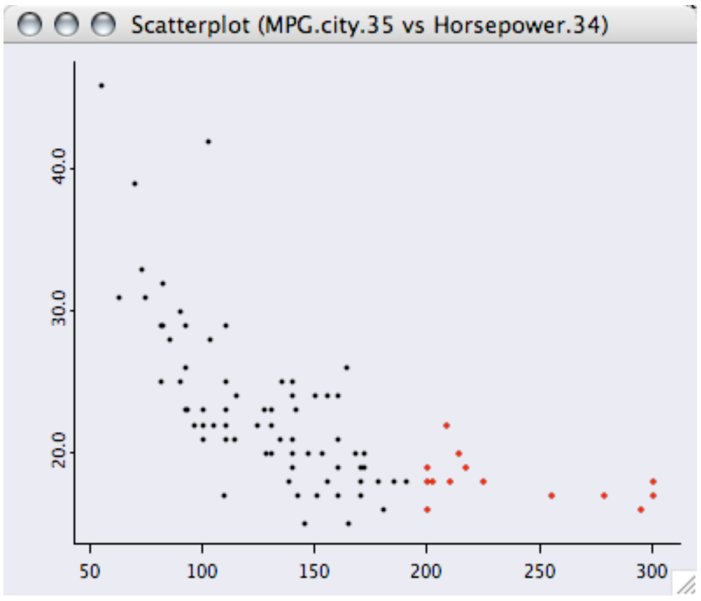


This example shows nicely that all cars having more than 200hp (this is the sected group) have either 6 ore more cylinder, or a rotary engine. Obviously, a boxplot y by x uses always the same scale for all boxplot for a proper comparison!

Scatterplots

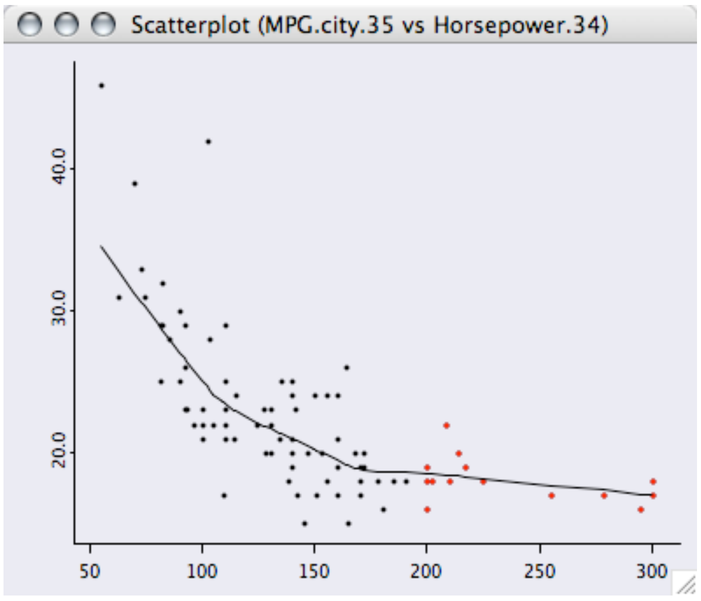
The scatterplot is probably the most often used plot of all statistical graphics. Scatterplots in iPlots can be used in the same way as the standard scatterplot in R.

```
> iplot(Horsepower, MPG.city)
```



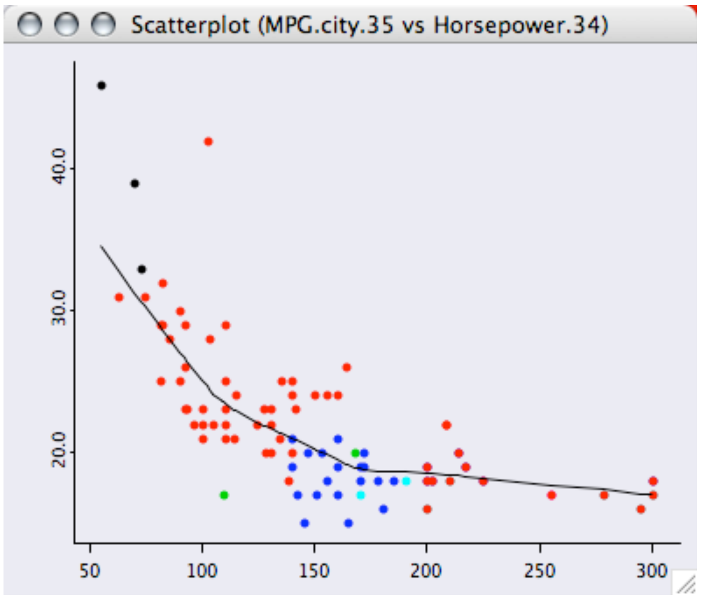
Looking at the above example, a natural task would be to add a scatterplot smoother to the plot.

```
# create a default lowess smoother

> l <- lowess(Horsepower, MPG.city)

# use ilines() to add the smoother to the iplot()

> ilines(l)
PlotPolygon(coord=1:1,dc=PlotColor(black),fc=none,points=93,...)

# we like to have it a bit rougher, so we remove the first smooth
# and rerun the lowess and plot it again.

> iobj.rm()

> l <- lowess(Horsepower, MPG.city, f=0.5)

> ilines(l)
PlotPolygon(coord=1:1,dc=PlotColor(black),fc=none,points=93,...)
```



We finally want some other cosmetics for the plot:

```
# set point size to 5 and color the points by number of cylinders

> iplot.opt(ptDiam=5, col=unclass(Cylinders))
```

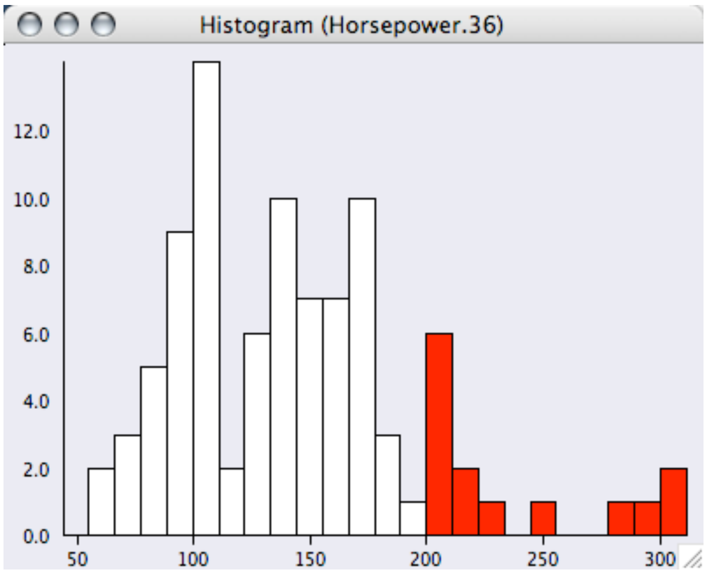


Many other options for the scatterplot within iPlots can be found in the view menu.

Histograms

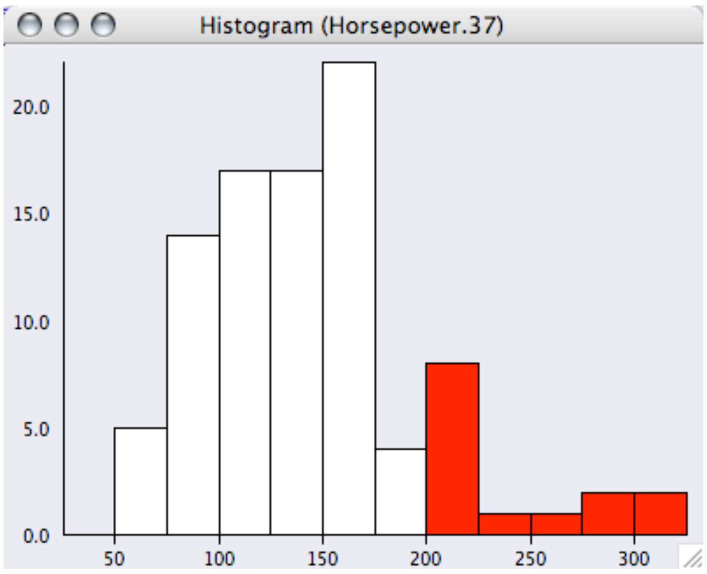
Histograms in iPlots are fully interactive and can also be switched to the spineplot view.

```
> ihist(Horsepower)
```



Binwith and anchorpoint can either be changed interactively by <alt>>-dragging the leftmost interval border (anchorpoint) or any other bin break (binwidth), or by explicitly specifying the two parameters either via iplot.opt() or upon creation of the plot.

```
> iplot.opt(anchor=25, binw=25)
```



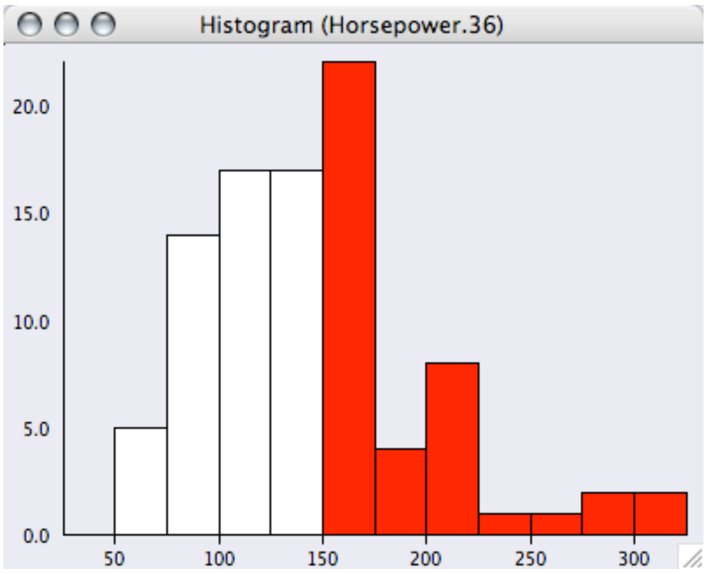
Histogram rescale by default when their parameters are changed, which can be controlled by two flags (see the help pages for details).

Selections

Selection in all iPlots can be done via either selecting single objects like a point, a bar/rectangle or a line, or by creating a drag-box, which selects all points within the rectangle.

Via the iset-functions, the selection state of an iplot session can be queried and modified.

```
# get selected indices

> iset.selected()
 [1] 76 63 59 52 30 10  5  2 50 57 48 11 28 19

# what proportion of the dataset is selcted?

> sum(sign(iset.selected()))/length(Horsepower)
[1] 0.1505376

# select all cases for cars with more than 150hp

> iset.select(Horsepower >= 150)
```

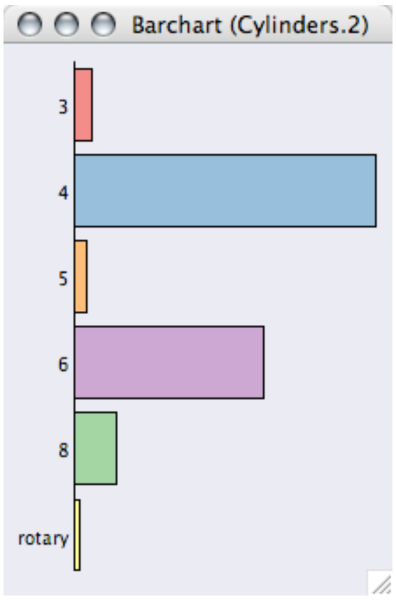


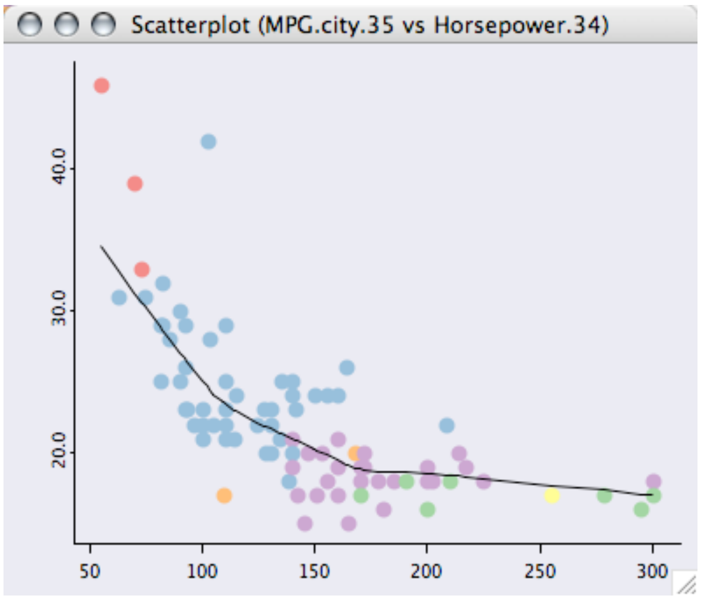
All plots, which share the same iset are updated immediately.

To perform more complex selections, pressing the <shift>-modifier will select data in XOR mode.

Color Brush

Selections are a transient attribute of the data. Whenever a new selection is defined, the old selection state is overwritten. A persistent color attribute can be defined via the so called **color brushing** within iPlots.
Color brushing can either be applied via the col=... option in iplot.opt() or any other iPlot graphics, or via the "View > Set Colors (CB)" menu command.

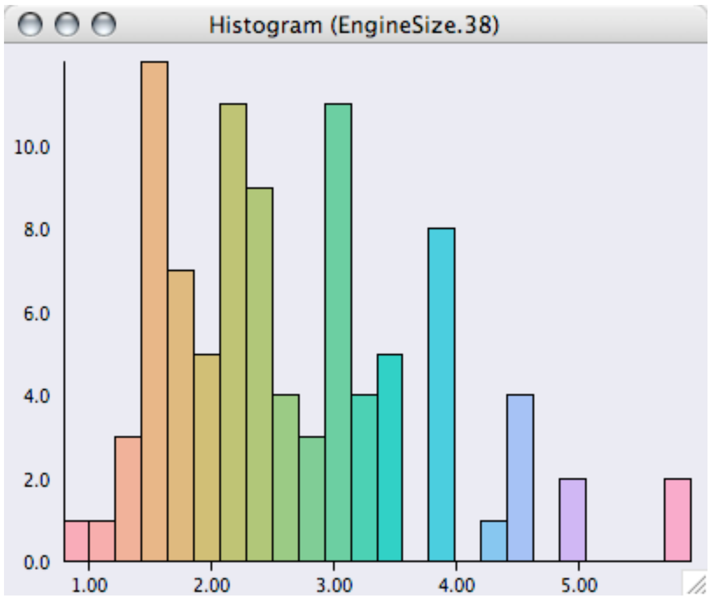


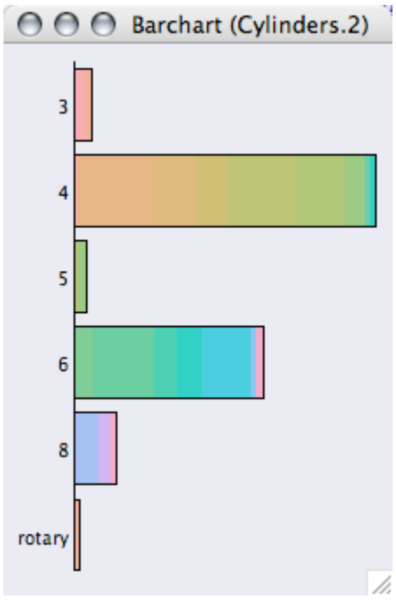


The same could be done from within R (though using the standard R colors) via:

```
# set colors according to Cylinders (using standard R colors)

> iplot.opt(col=unclass(Cylinders))
```

A second option for using color brushing is to apply rainbow colors over the range of a continuous variable. The menu command "View > Set Colors (rainbow)" will set the color scheme.





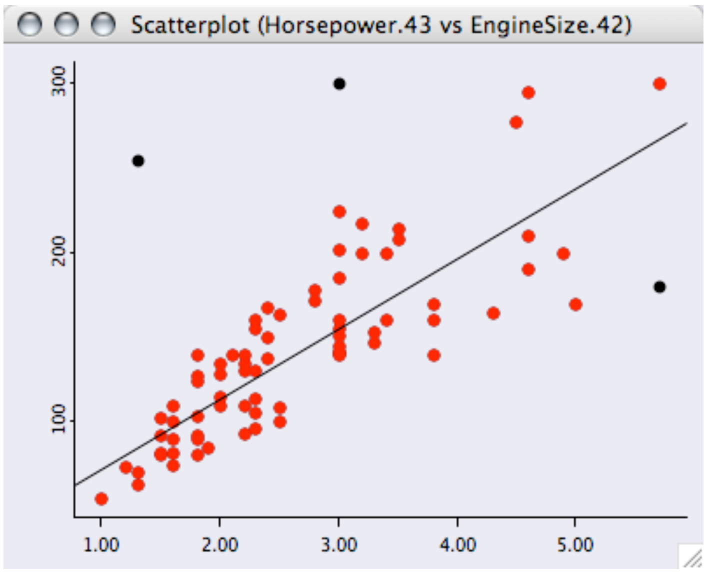
iObjects

iObjects allow to annotate plots with further graphical elements. Right now, iObjects are

- ilines()
- itext()
- iabline()

Here is a simple example

```
# create the scatterplot

iplot(EngineSize, Horsepower)

# select the outliers and get the selected subset for the regression

> subs <- iset.selected()

# add the linear regression to the scatterplot

> iabline(lm(Horsepower ~ EngineSize, subset=subs))
```



Not really a good model, but at least we got rid of the outliers very easily.

There are a couple of housekeeping functions for iObjects (please refer to the help files for details)

- **`iobj.cur()`**
- **`iobj.get()`**
- **`iobj.list()`**
- **`iobj.next()`**
- **`iobj.opt()`**
- **`iobj.prev()`**
- **`iobj.rm()`**
- **`iobj.set()`**

Conventions

Interactive graphics benefits strongly from an interface which is highly consistent and has thus a flat learning curve. The most important functionality within iPlots are the keyboard shortcuts and modifier keys.

- <shift>-selection is always in XOR-mode
- to query an object or plot canvas, mouse-over while <ctrl> is pressed
- <command>-drag-box (MAC), middle mouse button zooms in and out (to zoom out, just click, i.e. a zero size drag-box)
- <command>-r / <ctrl>-r rotates plots
- <command>-s / <ctrl>-s switches between absolute and relative view (is applicable)

Many other common functionality can be found in the "View" menu.

The handling of iplots is done via a set of maintenance functions (see help files for details). The basic concept follows the handling of devices within R.

- `iplot.cur()`
- `iplot.data()`
- `iplot.list()`
- `iplot.new()`
- `iplot.next()`
- `iplot.off()`
- `iplot.opt()`
- `iplot.prev()`
- `iplot.set()`

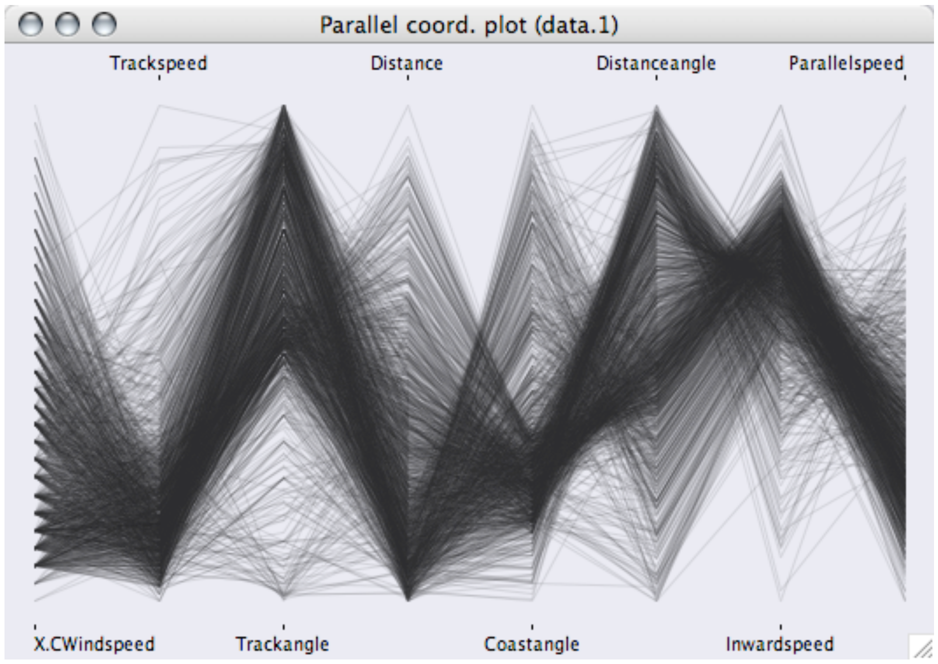
The same principles are used for maintaining the so called isets, which are the dataset, which are used in iPlots to link cases.

- `iset.brush()`
- `iset.col()`
- `iset.cur()`
- `iset.df()`
- `iset.list()`
- `iset.new()`
- `iset.next()`
- `iset.prev()`
- `iset.sel.changed()`
- `iset.select()`
- `iset.selectAll()`
- `iset.selected()`
- `iset.selectNone()`
- `iset.set()`

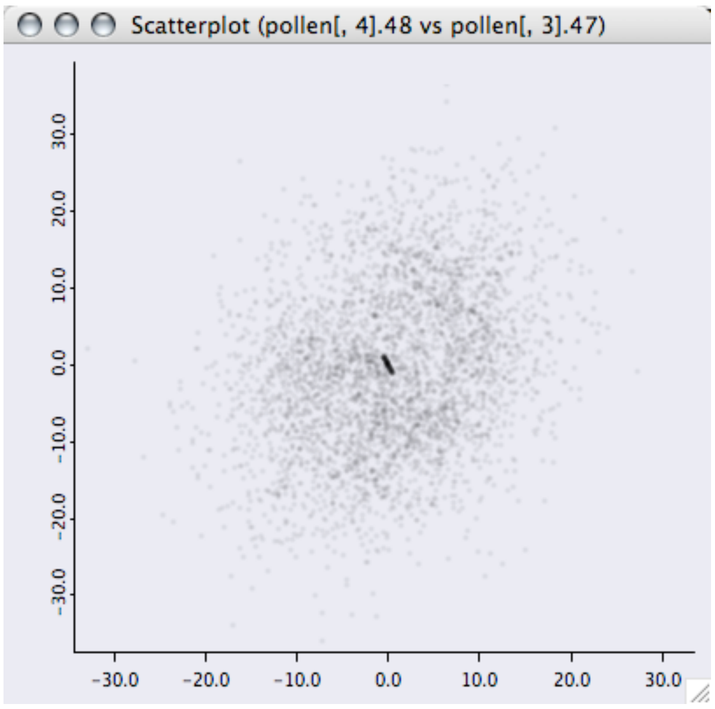
α-Channel

The α-channel can be used to specify the transparency of an object painted. This is very useful, when plotting really many objects , which would result in heavy overplotting. Thus the density of objects can be easily displayed.

All glyph based plot in iPlots offer α-channel transparency to handle large data and avoid overplotting.



(α-transparency in parallel coordinate plots)



(α-transparency in a scatter plot - you remember the dataset?!)

Use the arrow keys (left and right) to interactively increase or decrease the transparency.

FAQ

Where can I download iPlots?

- The current version of iPlots for R can simply be downloaded from CRAN. Use either the package manager from your favorite GUI or type:
  `install.packages("iplots",,"http://www.rosuda.org/R/")`

- The latest source packages can be found in our R repositories:

  - Release repository: http://www.rosuda.org/R/
    This repository also feature binary packages for Windows and Mac OS X
  - Development repository: http://www.rosuda.org/R/nightly

Is there anything else I need to install in order to use iPlots?

iPlots needs SUN's JRE to run. For Windows and OS X it is recommended to use at least Sun Java 1.4 or higher.
For other UNIX operating systems (in particular Linux) Sun Java 1.5 or higher is highly recommended. For the best user-experience iPlots can be run from within JGR.

Other Problems

Please post questions and bug reports to the mailing list stats-rosuda-devel.

Issues of general interest will be posted on this page.

Martin Theus, 10/03/2006