DSC 2003 Working Papers
(Draft Versions)
http://www.ci.tuwien.ac.at/Conferences/DSC-2003/

iPlots
High interaction graphics for R

Simon Urbanek & Martin Theus

Department of Computer Oriented Statistics and Data Analysis
University of Augsburg

Abstract

This paper describes the integration of interactive statistical graphics within R using state-of-the art R-connectivity. The iPlots project uses SJava to deliver interactive plots for R and an associated framework for their customization. The graphical part of iPlots is implemented in Java. The design is discussed and some practical examples of iPlots are illustrated.

1 Introduction

The gap between interactive statistical graphics software and command-line driven analysis tools is still unbridged. Both kinds of software developed further and further over the last years, basically ignoring the other side. Projects like R/ggobi [4] connect both types of software, but still run two applications alongside, forcing the user to be familiar with two environments to work in. Tools such as KLIMT[5] use R[3] just as a calculation engine, completely avoiding a direct use of the command-line interface. iPlots go the other way round. High interaction plots implemented in JAVA are directly accessible within R, using SJava interface.

Interactivity comprises the standards of linked highlighting between graphical objects in the plots as well the interactive modification of plot parameters.

2 Design

One of the main concerns in the design of iPlots was to retain a flat learning curve, while providing most features expected to be supported by interactive graphics. The key feature of interactive plots is the ability to use linked highlighting and color brushing. Linked highlighting is the concept of reflecting the selection of cases

in one plot in all other linked plots. Within iPlots all linked plots are updated immediately after the selection was performed. Color brushing provides a way of changing the color of symbols representing individual cases. Typical use is to set colors corresponding to groups of interest for easy identification.

In order to be able to link plots, we need to introduce a concept of a data set. All plots working on the same data set are automatically linked. Since linked highlighting and brushing work on case level, it is necessary to distinguish which plots contain data that can be linked together. The classical R plot commands don't support any such concept, because all plots are unrelated to each other. Changes in one plot have no effect on others. Furthermore values of any origin, different data sets or ad-hoc created values, can be used to construct a single plot.

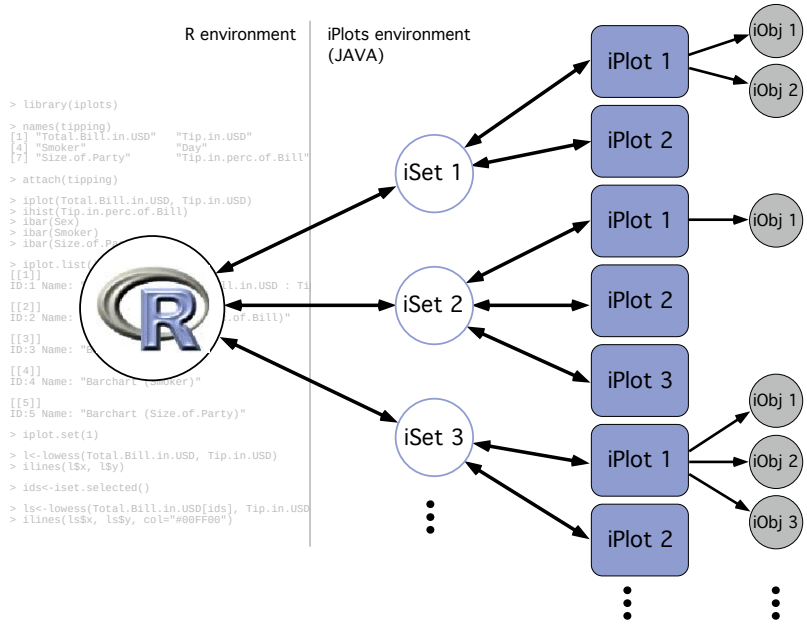

Figure 1: Within iPlots every iPlot is linked to (exactly) one iSet. iSets build the framework for linked (attribute) highlighting.

In iPlots every plot is linked to exactly one data set called iSet. Each iSet consists of an arbitrary number of variables, each represented by a vector. Each iSet requires that all variables of the iSet are of the same length to allow linking at case level. The iSets are managed automatically upon the use of any iPlot. Every iPlot command creates variables in the current iSet . The iSets can be explicitly changed in a similar way devices are managed in R. The current iSet can be changed by the `iset.set` command. The iPlots library automatically creates a default iSet upon startup. The user can explicitly create a new iSet with the `iset.new` command, otherwise a new iSet is created implicitly only when the data vector passed to a new iPlot is of different length than the current iSet contents.

The iPlots package offers a variety of interactive plots, including histograms, bar charts and scatter plots in the first release of iPlots. Other plots like mosaic

plots, parallel coordinates and maps will follow in future releases. All plots support interactive features, such as linked highlighting, interactive modification of plot parameters or querying. Every iPlot is represented by an object in R. This object can be used to modify plot parameters programatically. Typical examples are changes of the axis scales, setting plot parameters, assigning case colors or selecting cases. This enables the user to perform complex manipulations or create animations.

All currently displayed plots are accessible by the values of the list returned by the `iplots.list` command. For convenience the concept of a current plot was implemented as well. If no plot object is specified while performing plot modification operations, the modification is applied to the current plot. A newly created plot becomes automatically the current plot. The `iplots.set` command can also be used to make any specified plot current.

The actual implementation of the iPlots is done in Java. The iPlots library utilizes the SJava interface, which is a part of the Omegahat[2] project. That interface allows the use of Java code in R. We chose Java for the implementation, because of its platform independence, object orientation and graphics capabilities. The interface between Java and iPlots is completely hidden from the user, therefore no Java knowledge is necessary to use iPlots.

Another key aspect of the iPlots is the extensibility. All iPlots allow the user to modify the plots by adding new graphical objects such as lines, polygons, points or text. The interfaces to all these features is entirely defined in R, again no Java knowledge is necessary. This toolkit allows the user to enhance the plots in many ways, such as adding regression or smoother curves to scatter plots, or adding extra labels. The customized graphics are persistent and respect interactive updates of the plots, such as the change of the plot size or axis coordinates. All added objects can be accessed from R, modified or removed.

Although the main purpose of iPlots is to introduce exploratory data analysis capabilities to R, each plot can be exported in publication-quality vector-based representation to Encapsulated PostScript (EPS) format. Such files can be included in documents or even further edited by several graphics or design software.

3 On the interface

The implementation of iPlots pursues two main goals:

- Seamless integration in R
- Consistent user interface for iPlots with a flat learning curve.

3.1 Seamless Integration

iPlots should be used in R just as any other plot, with the additional benefit of interactivity. The naming convention for iPlots follows the basic scheme of just adding an "i" as a prefix to the known plots in R. For instance to plot an interactive histogram we issue the command `ihist(...)` instead of `hist(...)`. In the first release of iPlots, only the most important options of the plot commands are supported. Additionally not all plot options may be sensible any more in an interactive environment. The most important feature, supported by all iPlots are:

1. Setting scales and titles of axes and plot.

2. Specification of colors for subgroups with the `col=...` option.

3. Adding lines, points and labels to plots

A crucial point of iPlots is the consistent mapping between case-ids when using linked highlighting. Since the relationship between plots based on the same data is not known in R, the user may open different iPlots which do not belong into the same relation. One solution to this inconsistency would be to force the user to gather all data which might be used within the same relation in a dataframe-like structure. This would obviously guarantee the consistency of linked highlighting, but leads to a certain administrative overhead on the user side.

We chose a different way, in order to make iPlots as easy to use as possible. Whenever the user opens an iPlot, the length of the data vector is compared to the length of the data vector of the current iPlot. Are the vectors of identical length, the new iPlot is added to the relation of the current iPlot. To avoid errors, a dialog is presented, whenever the length of the vectors differ, asking whether a new iPlot relation shall be created or not.

3.2 Flat learning curve

The interactive features inside iPlots are not part of any R functionality and thus new to the user. To avoid lengthy user manuals and help files iPlots comply to a very strict user interface convention. Thus the user only needs to learn very few interactions in order to be able to use any plot of the iPlots suite interactively. These interactions comprise:

- Simple mouse click: Select (highlight) object at mouse position
- Drag-box: Highlight all objects in the selected rectangle
- *Control*-mouse-over: Display information of an object (query operation). Add *Shift* to obtain extended query revealing more details.
- *Meta* drag-box: Zoom in on this region (on system not supporting a meta key, i.e. Windows, the middle mouse button is used)
- *Meta* click: Zoom out to the view displayed before last zoom in
- *Alt*-drag: Move selected object to its new position

For the selection interactions, the following modifiers are defined, i.e. a selection performed with the modifier key pressed, will add this selection to the current selection using a binary logical operator.

- *Shift*-select: The newly selected points are combined with the currently selected points via XOR.
- *Alt-Shift*-select: The newly selected points are combined with the currently selected points via NAND, that is all newly selected points are removed from the previous selection.

Note, that not all interactions may be available for all plots. There may be iPlots where the zooming operation is not sensible at all. Some platforms don't support *Alt*+click, because the combination is used internally by the windows manager. Therefore the functionality of *Alt* and *Ctrl* is swapped for platforms using X11.

4 A Sample Session

We will use a very brief case study of the tipping data set[1] to illustrate the use of iPlots. The data set consists of 244 tips recorded by a waiter over the course of 4 days at a restaurant in a shopping mall. The following transcript of an R session shows how the relationship between bill size and tip is investigated with iPlots. A lowess smoother is superimposed on the scatterplot for the complete data and the selected subset of all smoker parties.

```
> library(iplots)

> names(tipping)
[1] "Total.Bill.in.USD"   "Tip.in.USD"          "Sex"
[4] "Smoker"              "Day"                 "Time"
[7] "Size.of.Party"       "Tip.in.perc.of.Bill"

> attach(tipping)

> iplot(Total.Bill.in.USD, Tip.in.USD)
> ihist(Tip.in.perc.of.Bill)
> ibar(Sex)
> ibar(Smoker)
> ibar(Size.of.Party)

> iplot.list()
[[1]]
ID:1 Name: "Scatterplot (Total.Bill.in.USD : Tip.in.USD)"

[[2]]
ID:2 Name: "Histogram (Tip.in.perc.of.Bill)"

[[3]]
ID:3 Name: "Barchart (Sex)"

[[4]]
ID:4 Name: "Barchart (Smoker)"

[[5]]
ID:5 Name: "Barchart (Size.of.Party)"

> iplot.set(1)

> l<-lowess(Total.Bill.in.USD, Tip.in.USD)
> ilines(l$x, l$y)

> ids<-iset.selected()

> ls<-lowess(Total.Bill.in.USD[ids], Tip.in.USD[ids])
> ilines(ls$x, ls$y, col="marked")
```

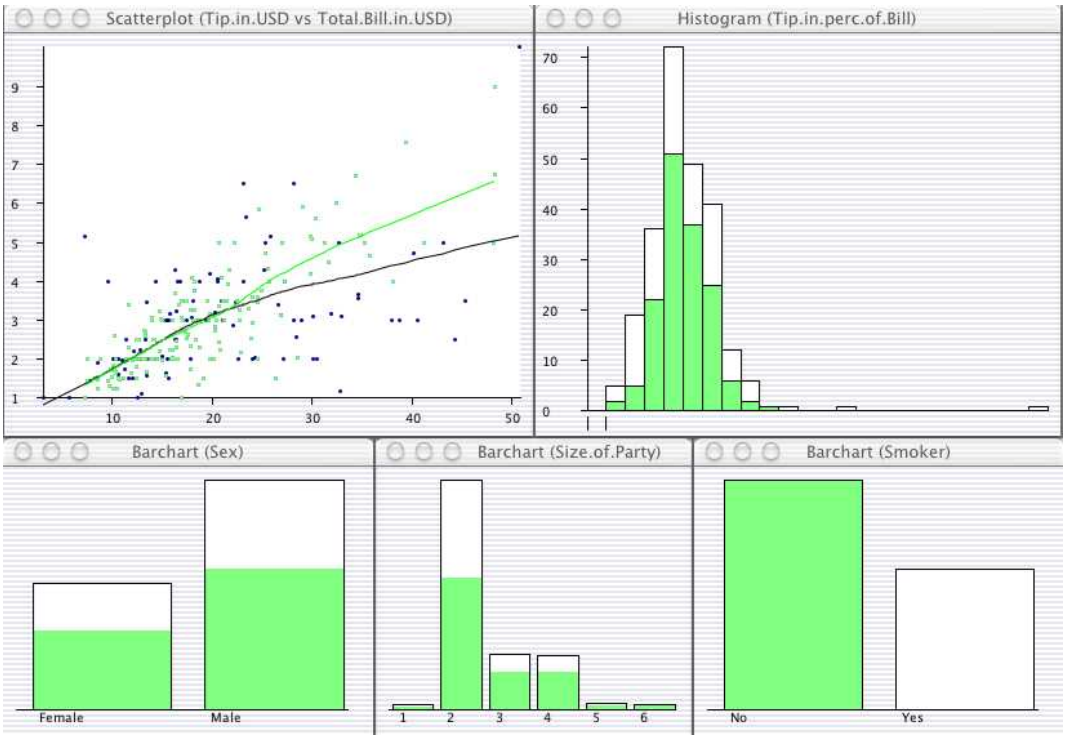

Figure 2: A set of 5 iPlots. The group of all smokers has bee selected in the lower right barchart. All other plots reflect this selection. The scatterplot has a lowess smoother superimposed for all points and the selected points.

5 Function Summary

5.1 Plots

The syntax of the iPlot functions was designed to be as close as possible to the syntax of the corresponding R plot. Parameters of the same name as in R-plots have the same meaning, unless stated otherwise. All iPlot functions listed below can be used without data specification, in which case the parameters of the current iPlot (see `iplots.cur()`) are changed. Most parameters are optional and have default values, which are often omitted here for brevity reasons. Consult the supplied online documentation for details. The goal of this section is to give a quick overview of the features and the interface.

- Histogram:
  `ihist(x, breaks, col, bwidth, anchor, right, border, main, xlab, ylab, xlim, ylim)`
  Plots an interactive histogram. `bwidth` and `anchor` allow to specifically set the bin width and anchor point of the histogram in data coordinates. `bwidth` and `breaks` are mutually exclusive, `bwidth` has higher priority. Only first two values in the vector version of `breaks` are used which corresponds to setting `anchor=breaks[1]` and `bwidth=breaks[2]-breaks[1]`.

- Scatter plot:

`iplot(x, y, col, main, xlab, ylab, xlim, ylim, log)`
Plots an interactive scatter plot. Currently the type is always "p".

- Bar chart:
  `ibar(data, col)`
  Plots an interactive bar chart. Unlike the `barplot` R function, this iPlot requires the entire factor as the `data` parameter, because the highlighting uses linking at the case level.

5.2 Manipulation of plots and data sets

- `iplot.set(which = iplot.next())`
  Change the current iplot.
- `iplot.cur()`
  Returns the current iplot.
- `iplot.list()`
  Lists all iplots.
- `iplot.next(which = iplot.cur())`
  Returns the next iplot in the list. The list is circular.
- `iplot.prev(which = iplot.cur())`
  Returns the previous iplot in the list.
- `iplot.data(id=NULL)`
  Returns the content of the plot data variables. If `id` is `NULL` then the content of all variables is returned form of a list, otherwise the content of a variable at the index `id` is returned (e.g. for scatter plot `id=1` is x and `id=2` is y).
- `iplot.param(...)` Returns a list of plot parameters and their values. The implementation is plot-dependent.
- `iset.new(name = NULL, data = NULL)`
  Creates a new iset and makes it current. The iset is assigned a name of the form "data.#", where # is an unique ID. If a data frame is specified as `data` then all variables from the data frame are put in the iset along with their names. It is the preferred method of creating isets if the data for the plots are organized in a data frame.
- `iset.set(which = iset.next())`
  Change the current iset.
- `iset.cur()`
  Returns the current iset.
- `iset.list()`
  Lists all isets.
- `iset.next(which = iset.cur())`
  Returns the next iset in the list. The list is circular.
- `iset.prev(which = iset.cur())`
  Returns the previous iset in the list.

5.3 Selection and color brushing

- `iset.select(what, mode="replace")`
  Selects cases specified by `what`, which can be either a logical vector used as a mask or a numeric vector used as list of IDs. `mode` specifies which logical operation should be applied relative to the existing selection.

- `iset.selected()`
  Returns a vector of selected cases as IDs.

- `iset.col(color=NULL, what="all")`
  Brushes specified cases with `color`. `what` can be either a list of IDs, a logical vector or the string `"all"`. If `color` is a scalar or shorter that the number of cases to paint, then it is repeated in a circular fashion. 0 means no color assigned. `NULL` is used when no parameters are supplied and results in returning a vector of case colors.

5.4 iObjects Toolkit

Every iPlot can have an arbitrary number of additional graphical objects attached, such as lines, rectangles, polygons or labels. Those objects, called iObjects, appear in the plot graphics and are programmable as of shape, position and color.

- `ilines(x, y=NULL, col=NULL, fill=NULL, ...)`
  Takes given coordinates and joins the corresponding points with line segments. The resulting iObject is a polygon or a polyline. The color of the lines can be specified as well as the fill color of the enclosed area.

- `iabline(a, b, ...)`
  Creates a line with the specified intercept and slope.

- `itext(x, y=NULL, labels = seq(along = x), ...)`
  Adds given text labels as an iObject to the plot at specified coordinates.

- `ipoints(x, y=NULL, col=NULL, ...)`
  Creates a new iObject which draws a sequence of points at the specified coordinates.

- `iobj.list(plot = iplot.cur())`
  Lists all iObjects of the plot.

- `iobj.cur(plot = iplot.cur())`
  Returns the ID of the current iObject of the plot.

- `iobj.rm(obj = iobj.cur())`
  Removes the iObject `obj` from the associated plot.

- `iobj.set(obj)`
  Make `obj` the current iObject.

- `iobj.opt(...,obj=iobj.cur())`
  Sets options and properties of the given iObject. The parameters are object-dependent. Usually most parameters available at creation time are supported as well.

5.5 Event handling

iPlots offer some basic functions to make interaction between iPlots and R more flexible. These functions can be used to build animations and small interaction loops in R.

- `ievent.wait(...)`
  This function waits until an iPlots event occurs. The result is `NULL` for the *break* event.
- `iset.sel.changed(iset=iset.cur(),...)`
  Returns `TRUE` if the selection of the specified iSet changed since the last call of this function.

The following example illustrates the functionality:

```
iplot(x, y)
iabline(lm(y ~ x), col="black")
iabline(0, 0, col="marked", visible=FALSE)
while (!is.null(ievent.wait())) {
  if (iset.sel.changed()) {
    s <- iset.selected()
    if (length(s)>0)
      iobj.opt(reg=lm(y[s] ~ x[s]), visible=TRUE)
    else
      iobj.opt(visible=FALSE)
  }
}
```

This piece of R code creates an interactive scatterplot of x and y with a linear regression line. Until the user uses "break" in iPlots, the inner loop re-calculates and re-displays a regression line for the selected points each time the user selects points in *any* of the iPlots. If no points are selected, the second regression line disappears.

6 Outlook

Future releases of iPlots will implement more plots like mosaic plots, parallel coordinate plots and maps.

Another extension is to offer the ability to add not only static objects, as in the current release, but interactive objects. Those objects such as points, lines and rectangles have the following basic properties: Interactive objects ...

- are associated to an iPlot, and keep a list of case ids, they represent.
- can be selected
- support linked highlighting

It is possible to build simple interactive graphics with these interactive objects. In contrast to the built-in iPlots, plots created with the help of such interactive objects do not offer additional interactive features beyond the standard selection and highlighting capabilities.

7 Conclusion

iPlots are a first step towards a seamless integration of interactive statistical graphics into R. The primary design goal for iPlots was a flat learning curve for users acquainted with R. The commands of the iPlot API follow the common R structures as close as possible. In order to facilitate selection and linked highlighting among different plots the concept of iSets has been introduced. Commonly used modifications to R plots such as `lines, points` and `abline` are implemented for iPlots as well.

In contrast to the static plots in R, iPlots can not be fine-tuned using various parameters of the `par()` command. We deliberately tried to avoid the concept of global graphical states. Only the most important options like `xlim, ylim, main, xlab, ylab` and `col` are supported on per-plot basis. The number of currently available plots is restricted to scatterplots, barcharts and histograms, but the list will be extended in near future.

8 Acknowledgements

Parts of this paper and work on iPlots has been supported by the statistics department of Iowa State University.

References

[1] P.G. Bryant and M.A. Smith. *Practical Data Analysis: Case Studies in Business Statistics*. Richard D. Irwin Publishing, Monewood, IL, 1995.

[2] Duncan Temple Lang. The omegahat environment: New possibilities for statistical computing. *JCGS*, 9(3), 2000.

[3] The R Project. `http://www.r-project.org`, 2003.

[4] D. Swayne, D. Temple, A. Buja, and D. Cook. Ggobi: Xgobi redesigned and extended. In *Proc. of the 33th Symposium on the Interface: Computing Science and Statistics*, 2001.

[5] Simon Urbanek. No need to talk to strangers - cooperation of interactive software with r as moderator. In *Proc. of the 34th Symposium on the Interface: Computing Science and Statistics*, 2002.