

# Package ‘iplots’

May 1, 2022

**Version** 1.1-8

**Title** iPlots - Interactive Graphics for R

**Author** Simon Urbanek <simon.urbanek@r-project.org>, Tobias Wichtrey <tobias@tarphos.de>

**Maintainer** Simon Urbanek <simon.urbanek@r-project.org>

**Depends** R (>= 1.5.0), methods, rJava (>= 0.5-0)

**Suggests** maps, MASS

**Imports** grDevices, png

**License** GPL-2

**Description** Interactive plots for R.

**URL** <http://www.iPlots.org/>

**BugReports** <https://github.com/s-u/iplots/issues>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-05-01 14:58:01 UTC

## R topics documented:

iabline . . . . .	2
ibar . . . . .	3
ibox . . . . .	4
ievent.wait . . . . .	5
ihammock . . . . .	6
ihist . . . . .	7
ilines . . . . .	8
imap . . . . .	9
imosaic . . . . .	10
iobj.list . . . . .	11
iobj.opt . . . . .	12
ipcp . . . . .	13
iplot . . . . .	14
iplot.data . . . . .	15

iplot.list . . . . .	16
iplot.manip . . . . .	17
iplot.opt . . . . .	18
iraster . . . . .	20
iset . . . . .	21
iset.col . . . . .	22
iset.list . . . . .	23
iset.selected . . . . .	24
itext . . . . .	25
ivar . . . . .	26

## Index 29

---

iabline	<i>Add a straight line to the current iPlot.</i>
---------	--

---

### Description

This function adds one straight line through the current iPlot.

### Usage

```
iabline(a = NULL, b = NULL, reg = NULL, coef = NULL, ..., plot = iplot.cur())
```

### Arguments

a, b	the intercept and slope
coef	a vector of length two giving the intercept and slope
reg	an object with a coef component
...	object options for <a href="#">iobj.opt</a>
plot	plot to which this line is to be added

### Details

Typical usages are

```
iabline(a, b, ...)
iabline(coef=, ...)
iabline(reg=, ...)
```

### Value

Resulting iObject.

### See Also

[ilines](#), [iobj.opt](#)

---

ibar	<i>Interactive Bar Chart</i>
------	------------------------------

---

**Description**

This function creates a new interactive bar chart from the given data.

**Usage**

```
ibar(var, ...)
```

**Arguments**

var	Factor, vector of strings or variable to use.
...	All additional parameters are passed to <a href="#">iplot.opt</a> .

**Details**

Creates an interactive barchart or spline plot.

Additional parameters (also accessible via [iplot.opt](#)):

`drawTicks` Whether ticks should be drawn.

`isSpine` Whether a spineplot should be created instead of a normal barchart.

`borderColorSel` Here this color defaults to black. See [iplot.opt](#) for details.

**Value**

Resulting plot object.

**See Also**

[iplot](#), [ihist](#), [iplot.list](#), [iplot.opt](#)

**Examples**

```
data(iris)
attach(iris)
ibar(Species)
```

---

`ibox`*Interactive Box Plot*

---

## Description

This function creates a new interactive box plot from the given data.

## Usage

```
ibox(x, y=NULL, ...)
```

## Arguments

<code>x</code>	Vector of numbers or data frame containing the variables
<code>y</code>	A factor to specify groups for y-by-x boxplot
<code>...</code>	All additional parameters are passed to <code>iplot.opt</code> . Many of the parameters used in <code>plot</code> are supported.

## Details

Creates either a regular boxplot (if `y` is not specified) or 'x-by-y' boxplot (if `y` is specified). In the latter case both `x` and `y` must be of the same length.

Additional parameters:

**alterningLabels** Whether labels should be placed alternately at the top and at the bottom.

## Value

Resulting plot object.

## See Also

[ihist](#), [ibar](#), [iplot.list](#), [iplot.opt](#)

## Examples

```
data(iris)
attach(iris)
ibox(Petal.Length)
ibox(Sepal.Length, Species)
```

ievent.wait

*Interactive events***Description**

This function provides a way of building interactive event loops in R. Currently SJava interface has no stable callbacks, therefore this is the only way of defining new interactions in iPlots.

`ievent.wait` waits until some iPlots event (iEvent) occurs. The most commonly used events include selection change and the "break" event. It is possible to generate user-definable events (such as events sent by custom buttons or menu entries) with low-level API (via SJava)

Note that in the current implementation the Windows GUI "freezes" until the function returns.

**Usage**

```
ievent.wait()
```

**Value**

NULL is the break event occurred or a ievent object if some other event occurred.

**Examples**

```
data(iris)
attach(iris)
iplot(Sepal.Length, Petal.Length)

d<-iplot.data()

iabline(lm(d$y ~ d$x), col = "red")
ilines(lowess(d$x,d$y), col = "#0000c0")
ilines(c(0,0),c(0,0), col = "marked", visible = FALSE)

cat("Select 'Break' from the menu of any plot to return back to R.\n")

while (!is.null(ievent.wait())) {
  if (iset.sel.changed()) {
    s <- iset.selected()
    if (length(s) > 1)
      iobj.opt(x=lowess(d$x[s],d$y[s]),visible = TRUE)
    else iobj.opt(visible = FALSE)
  }
}
for(i in 1:3) iobj.rm()
iplot.off()
```

---

`ihammock`*Interactive Hammock Plot*

---

## Description

This function creates a new interactive hammock plot. Please note that a hammock plot was not necessarily designed to support highlighting, so it may be of limited use. It was created as a proof of the iBase concept which makes it very easy to implement new interactive plots.

## Usage

```
ihammock(...)
```

## Arguments

... All unnamed parameters are treated as variables to display. All additional parameters are passed to [iplot.opt](#).

## Details

The plot can be used in either in the form `ihammock(df)` to plot a list or data frame of variables or `ihammock(x, y, z)` to plot factors `x`, `y` and `z`.

## Value

Resulting plot object.

## See Also

[ihist](#), [ibar](#), [iplot](#), [iplot.opt](#)

## Examples

```
library(MASS)
data(Cars93)
attach(Cars93)
ihammock(AirBags, Cylinders, Origin)
```

---

ihist	<i>Interactive Histogram</i>
-------	------------------------------

---

## Description

This function creates a new interactive histogram from the given data.

## Usage

```
ihist(var, ...)
```

## Arguments

<code>var</code>	Vector of numbers or variable to use.
<code>...</code>	All additional parameters are passed to <a href="#">iplot.opt</a> .

## Details

Creates an interactive histogram.

Additional parameters (also available via [iplot.opt](#)):

**anchor** Anchor point for the histogram.

**binw** Bin width.

**autoScaleXAxis** Whether the x axis should be automatically rescaled.

**autoScaleYAxis** Whether the y axis should be automatically rescaled.

## Value

Resulting plot object.

## See Also

[iplot](#), [ibar](#), [iplot.list](#), [iplot.opt](#)

## Examples

```
data(iris)
ihist(iris$Sepal.Width)
```

---

`ilines`*Add connected lines or polygon to the current iPlot.*

---

**Description**

A generic function taking coordinates of points in data space and creating corresponding connected lines or polygon in the current iPlot.

**Usage**

```
ilines(x, y, col=NULL, fill=NULL, visible=NULL, plot = iplot.cur())
```

**Arguments**

<code>x, y</code>	Coordinate vectors of points to join. <code>xy.coords</code> is used to obtain the coordinates for plotting.
<code>col</code>	Drawing color of the lines. Currently only "#rrggbb" notation and named colors are supported. NULL means that color of the object is not explicitly specified.
<code>fill</code>	Color of the polygon area or NA if no filling should be performed.
<code>visible</code>	If set to FALSE the lines/polygon won't be visible initially.
<code>plot</code>	parent plot for the lines

**Details**

The point vectors `x` and `y` can contain NA values, in which case each sequence of points separated by NAs will be treated as a separate polygon. NAs must be present in both coordinates at the same index, otherwise the behavior is undefined.

**Value**

Resulting iObject.

**See Also**

[ihist](#), [ibar](#), [iplot.list](#), [iobj.opt](#)

**Examples**

```
data(iris)
attach(iris)
iplot(Sepal.Width, Petal.Width)
l<-lowess(Sepal.Width, Petal.Width)
ilines(l)
```

---

`imap`*Interactive Map*

---

## Description

This function creates a new interactive map from the given data.

## Usage

```
imap(x, y=NULL, ...)
```

## Arguments

<code>x</code>	either an object of the class "map" as created by the <code>map</code> function or a vector of the x-coordinates of the map polygons
<code>y</code>	y-coordinates of the map polygons
<code>...</code>	All additional parameters are passed to <code>iplot.opt</code> .

## Details

Creates an interactive map plot.

The input can be either an object of the class "map" passed directly to the `x` parameter or two vectors of matching coordinates passed to `x` and `y`. The format for polygons is the same as used by the `map` function.

Each polygon should correspond to a case and it will be linked correspondingly. See [ivar.new.map](#) for details on map variables.

Note: this function is currently experimental and it may change in the future.

## Value

Resulting plot object.

## See Also

[iplot](#), [ivar.new.map](#)

## Examples

```
library(maps)
m <- map('state', plot=FALSE)
imap(m)
```

---

`imosaic`*Interactive Mosaic Plot*

---

## Description

This function creates a new interactive mosaic plot from the given data.

## Usage

```
imosaic(...)
```

## Arguments

...

All unnamed parameters are treated as variables to display. Additional parameters are passed to `iplot.opt`. The parameter "type" selects the variation of the mosaic plot. Valid values are "observed", "expected", "fluctuation", "same.bin.size" and "multiple.barchart". Partial matching is used, so first letter is sufficient.

## Details

The plot can be used either in the form `imosaic(df)` to plot a list or data frame of variables contained in `df` or `imosaic(x, y, z)` to plot variables `x`, `y` and `z`.

Additional parameters:

**rotateYLabelsBy** The default value of this variable is changed here. For a description see [iplot.opt](#).

## Value

Resulting plot object.

## See Also

[ihist](#), [ibar](#), [iplot.list](#), [iplot.opt](#)

## Examples

```
library(MASS)
data(Cars93)
attach(Cars93)
imosaic(AirBags,Cylinders,Origin)
imosaic(AirBags,Cylinders,Origin,type="mul")
iplot.location(300,100,TRUE)
```

iobj.list

*Interactive objects (iObjects) management functions.***Description**

These functions are used to manage iObjects of an iPlot. Exactly one of the iObjects is the current one (for each iPlot). Every newly created iObject automatically becomes current.

Please note that both iPlots and iObjects can be also used directly - each function creating an iObject or iPlot returns the newly created object which can be used in calls to functions requiring plot parameter (for iPlots) or iobj.opt, iobj.rm and similar (iObjects). The object list management functions below are provided for convenience only - the direct use of objects is encouraged instead.

[iobj.list](#) returns all iObjects of the current iPlot.

[iobj.cur](#) returns the current iObject.

[iobj.next](#) and [iobj.prev](#) return the ID of the next resp. previous object in the list relative to the object specified by the argument.

[iobj.set](#) makes the object with the specified ID current.

[iobj.get](#) returns the object specified by its ID.

[iobj.rm](#) removes the object

== and != operators can be used to compare two iObjects

**Usage**

```
iobj.list(plot = iplot.cur())
iobj.cur(plot = iplot.cur())
iobj.next(which=iobj.cur(), plot = iplot.cur())
iobj.prev(which=iobj.cur(), plot = iplot.cur())
iobj.set(which=iobj.next())
iobj.get(pos, plot=iplot.cur())
iobj.rm(which=iobj.cur(), plot = iplot.cur())
## S3 method for class 'iobj'
a == b
## S3 method for class 'iobj'
a != b
```

**Arguments**

which	An object or an integer specifying the object number.
pos	ID of an object
plot	plot to operate on (either as integer ID or plot object itself).
a	object to compare
b	object to compare

**See Also**

[ilines](#), [iabline](#)

---

`iobj.opt`*Modify parameters of an iObject*

---

### Description

This function modifies parameters of an iObject.

### Usage

```
iobj.opt(o=iobj.cur(),...)
```

### Arguments

<code>o</code>	object whose options are to be set
<code>...</code>	options to be set

### Value

If no parameters (except for the object) are specified, a list of the current parameters is returned (if supported by the object).

The following common parameters are used by most iObjects:

`visible` visibility flag. The default is TRUE.

`layer` layer in which the object is placed. Currently the following layers are supported: 0 - background, 1 - data points, 2 - selection, 3 - drag boxes. The value -1 has a special meaning by denoting the topmost layer (which is the default)

`col` drawing color of the object

`fill` filling color of the object (where applicable). If set to NA then no filling is performed.

`coord` coordinates system to use. 0=graphical coordinates, 1=data space coordinates (default), 2=relative coordinates.

`update` if set to FALSE then no plot refresh is done after updating the options. When modifying multiple objects it is common practice to set update to FALSE for all but the last updated object. The default is TRUE.

### See Also

[iobj.list](#)

---

`ipcp`*Interactive Parallel Coordinates Plot*

---

## Description

This function creates a new interactive parallel coordinates plot from the given data.

## Usage

```
ipcp(...)
```

## Arguments

... All unnamed parameters are treated as variables to display. All additional parameters are passed to [ipplot.opt](#).

## Details

Creates an interactive parallel coordinates plot.

The plot can be used either in the form `ipcp(df)` to plot a list or data frame of variables contained in `df` or `ipcp(x, y, z)` to plot variables `x`, `y` and `z`.

Additional parameters:

`alternatingLabels` Whether labels should be placed alternately at the top and at the bottom.

`COL_AXES` Color of the (optional) axes.

## Value

Resulting plot object.

## See Also

[ihist](#), [ibar](#), [ipplot.list](#), [ipplot.opt](#)

## Examples

```
data(iris)
ipcp(iris)
```

iplot

*Interactive Scatterplot***Description**

This function creates a new interactive scatterplot from the given data.

**Usage**

```
iplot(x, y=NULL, xlab=NULL, ylab=NULL, ...)
```

**Arguments**

x	Data for the x axis. It can be either a vector of values or a variable of an iset. If y is not given, this must be a list of one of these.
y	Data for the y axis. It can be either a vector of values or a variable of an iset.
xlab	Name for x variable.
ylab	Name for y variable.
...	All additional parameters are passed to <code>iplot.opt</code> . Many of the parameters used in <code>plot</code> are supported.

**Details**

Creates an interactive scatterplot.

Additional parameters:

**changePtDiamBy** Number of pixels the point diameter should be changed when in-/decreasing it via keyboard or menu.

**customFieldBg** Whether to use a custom background color.

`COL_CUSTOMBG` The custom background color.

**drawAxes** Whether axes should be drawn.

**equiscale** Whether the same scale should be applied to both axes.

**minimalDiam** Minimal point diameter.

**ptDiam** Point diameter.

**spaceprop** Gives the amount of space around the data points. 1.0 means no space, 1.5 means half as much space around the data as is used for the data itself. This resets zoom.

Default values:

```
changePtDiamBy=2, customFieldBg=FALSE, COL_CUSTOMBG="white",
drawAxes=TRUE, equiscale=FALSE, minimalDiam=1, ptDiam=3,
spaceprop=1.1
```

**Value**

Resulting plot object.

**See Also**

[ihist](#), [ibar](#), [iplot.list](#), [iplot.opt](#)

**Examples**

```
data(iris)
attach(iris)
iplot(Sepal.Width,Petal.Width)
iplot(Sepal.Width/Sepal.Length, Species)
```

---

iplot.data

*Retireve data from a plot.*

---

**Description**

This function retrieves the associated data from the current plot.

**Usage**

```
iplot.data(id=NULL)
```

**Arguments**

`id` number of the variable to retrieve. If omitted a list of all associated varaibles (and their contents) is returned.

**Value**

A vector representaing a variable or a list of variable contents. The first two vecctors in the list are traditionally named x and y. The number of variables depends on the plot used, e.g. scatter plot has two, histogram or bar chart return one.

**See Also**

[iplot](#), [ihist](#), [ibar](#)

**Examples**

```
data(iris)
attach(iris)
iplot(Sepal.Width,Petal.Width)
iplot(Sepal.Width/Sepal.Length, Species)
```

---

`iplot.list`*Interactive plots management functions.*

---

## Description

These functions are used to manage currently open iPlots. Exactly one of the open iPlots is the current plot. Every newly created iPlot automatically becomes the current plot. Any plot specific functions, such as `ilines` operate on the current plot.

Please note that the functions below are provided for convenience only. It is also possible to use plot objects directly without using the plot list. Each function creating a new iPlot directly returns the plot object which can then be used to any subsequent calls to `ilines`, `iplot.opt` etc.

`iplot.list` returns all currently registered iPlots (even if they are hidden).

`iplot.cur` returns the ID of the current plot.

`iplot.next` and `iplot.prev` return the ID of the next resp. previous plot in the list relative to the plot specified by the argument.

`iplot.set` makes the plot with the specified ID current.

`iplot.off` closes the plot.

## Usage

```
iplot.list()
iplot.cur()
iplot.next(which=iplot.cur())
iplot.prev(which=iplot.cur())
iplot.set(which=iplot.next())
iplot.off(plot=iplot.cur())
```

## Arguments

<code>which</code>	An integer specifying a plot number.
<code>plot</code>	Plot object or plot number of a plot to close.

## See Also

`ilines`, `iplot`, `ihist`, `ibar`

## Examples

```
data(iris)
attach(iris)
iplot(Sepal.Width, Petal.Width)
ibar(Species)
iplot.list()
```

---

iplot.manip

*iPlot manipulation functions*


---

## Description

The following functions are used to manipulate iplots. They are NOT part of the official API and may disappear without warning. Most of them are legacy functions introduced before `iplot.opt` was available.

`iplot.backend` retrieves or sets the iPlots back-end.

`iplot.resetZoom` reset zoom

`iplot.rotate` set plot rotation

`iplot.setExtendedQuery` set text for extended query

`iplot.zoomIn` zoom into specified area

`iplot.zoomOut` zoom out (the coordinates are ignored as the zoom is hierarchical)

`iplot.location` get or set the location of the iplot

`iplot.size` get or set the size of an iplot plot

## Usage

```

iplot.backend(type = NULL)
iplot.resetZoom()
iplot.rotate(i)
iplot.setExtendedQuery(str, plotID=.iplot.curid)
iplot.zoomIn(x1, y1, x2, y2)
iplot.zoomOut(x, y)
iplot.location(x, y, relative=FALSE, plot=iplot.cur())
iplot.size(width, height, plot=iplot.cur())
## S3 method for class 'iobj'
print(x, ...)
## S3 method for class 'iplot'
print(x, ...)

```

## Arguments

<code>i</code>	rotation orientation
<code>plotID</code>	plot ID (number)
<code>str</code>	string to show on extended query or FALSE or NULL to disable extended query
<code>type</code>	back-end type - one of "awt", "swing" or "opengl" (or any unambiguous first part hereof) to set the type or NULL to retrieve the current back-end type
<code>x1</code>	basis coordinate for the x axis
<code>y1</code>	basis coordinate for the y axis
<code>x2</code>	edge coordinate for the x axis

y2	edge coordinate for the y axis
x	object to print or x-coordinate
y	y-coordinate
plot	plot to query or move (id or object)
relative	can be TRUE, FALSE or another plot
width	width of the plot (in pixels)
height	height of the plot (in pixels)
...	additional parameters

### Details

`iplot.location`, `iplot.size`: those functions either query or set the location or size of the plot. If either of the coordinates is missing, the size/location in that coordinate will not be changed. If both coordinates are missing, the functions have no side effect and just return the current size (`iplot.size`) or location and size (`iplot.location`) as named vectors (`x`, `y` for location and `width`, `height` for size).

`iplot.backend` determines the back-end used by iplots. The choices are `awt`, `swing` and `dopengl`. Not all back-ends are available on all platforms. AWT is the most compatible back-end and available on all platforms, Swing has more modern look and feel and OpenGL sacrifices text rendering quality for speed (i.e. OpenGL backend is best for large data). When no backend is specified, the function has no side effect and just returns the currently requested back-end. Currently there is no indication whether the back-end request was honored or not. If a backend is not supported, the request is silently ignored.

---

<code>iplot.opt</code>	<i>Modify parameters of an interactive plot</i>
------------------------	---

---

### Description

This function modifies parameters of an `iPlot`.

### Usage

```
iplot.opt(..., plot=iplot.cur())
```

### Arguments

...	Parameters to modify
plot	Plot whose parameters are to be modified

**Details**

The following parameters are common to all plots. See help pages of individual plots for plot-specific parameters.

`xlim` Range of the X axis (vector of two numbers).

`ylim` Range of the Y axis (vector of two numbers).

`col` Colors of the points. See [iset.brush](#) for details.

`autoAdjustMargins` Whether iPlots should try to determine appropriate margin sizes. Set this to `FALSE` if you want to set the margins manually.

`defaultMargins` Vector of default values for left, right, top and bottom margins.

`mLeft` Size of left margin.

`mRight` Size of right margin.

`mTop` Size of top margin.

`mBottom` Size of bottom margin.

`fillColor` Fill color.

`borderColor` Border color.

`fillColorSel` Fill color when selected.

`borderColorSel` Border color when selected.

`fillColorDrag` Fill color of dragged bars.

`COL_INVALID` Color of invalid elements.

`COL_OUTLINE` Default line color.

`COL_SELBG` Background color of selection rectangle. Defaults to selection color with alpha value 0.298.

`COL_ZOOMBG` Background color of selection rectangle.

`horizontalMedDist` Mean horizontal label distance.

`horizontalMinDist` Minimal horizontal label distance.

`verticalMedDist` Mean vertical label distance.

`verticalMinDist` Minimal vertical label distance.

`extQueryString` Extended query string.

`rotateYLabels` Whether labels for the y axis should be rotated by the angle given by `rotateYLabelsBy`.

`rotateYLabelsBy` The labels for the y axis are rotated by this amount of degrees if `rotateYLabels` is set to true.

`title` Frame title.

**Value**

If no parameters (except for the plot) are specified, a list of the current parameters is returned.

**See Also**

[iplot.list](#)

**Examples**

```
data(iris)
attach(iris)
iplot(Sepal.Width,Petal.Width)
iplot.opt(xlim=c(1.5,5),col=Species)
```

---

iraster

---

*Add a bitmap (raster) image to the current iPlot.*


---

**Description**

iraster adds a raster image as an iObj to the given iPlot. Position in the plot is specified by bottom-left and top-right points of the image.

**Usage**

```
iraster(x1, y1, x2, y2, img, ..., plot = iplot.cur())
```

**Arguments**

x1, y1, x2, y2	coordinates of the bottom-left (x1, y1) and top-right (x2, y2) corner. Alternatively, x1 can be a vector of length 4 specifying the four values, or x1 and x2 can be vectors of length 2 specifying one point each
img	image to draw. It can be either a raster, file name, binary connection or a raw vector containing an image in a common image format such as PNG or JPEG. If it is a raster, then the raster is first encoded into PNG format and passed as raw vector.
...	additional arguments that will be passed to <a href="#">iobj.opt</a> if present.
plot	parent plot for the image

**Details**

The current implementation uses Java's ImageIO API to read the image, so the supported formats will depend on your Java implementation. Raster objects (i.e., of class "raster", "nativeRaster"), matrices and arrays) are simply passed to `png::writePNG(img)` so for anything other than computationally constructed objects it is more efficient to use the encoded image.

**Value**

Resulting iObject.

**See Also**

[iobj.opt](#)

## Examples

```
## very silly example ...
iplot(0:20/20, 0:20/20)
## get a sample image (R logo) from the png package
fn <- system.file("img", "Rlogo.png", package="png")
## put this image behind all points
iraster(0, 0, 1, 1, fn, layer=-2)
## you can use a raster but it's less efficient
## this one goes to the top layer where iObjs reside normally
iraster(0, 0, 0.5, 0.5, png::readPNG(fn))
```

---

 iset

*iSet and iVar - managing data in iPlots*


---

## Description

iPlots maintain a separate copy of all data that are displayed in the iPlots. This allows iPlots to operate even after the underlying data has been deleted in R. It also allows iPlots to use hot linking in all aspects, including update of plots on data changes.

This data management consists mainly of two classes: *iset* and *ivar*. The *iset* object encapsulates an *iSets* which can be thought of as a kind of special data frame that allows linking of all variables contained therein. Each variable (or column in data.frame-speech) is represented by an *ivar* object encapsulating an *iVar*.

An instance of a *ivar* class can be used to create new iplots or update data in the existing iplots. Convenience operators on both *ivar* and *iset* objects include those such as `length`, `subsetting` and `subassignment` and therefore from user's point of view it is possible to use them transparently a data frames (*iSet*) or vectors (*iVar*). In addition, *iSet* supports methods such as `names` or `dim`.

`iset` returns the object corresponding to a given *iSet*.

`isets` returns objects for all *iSets*

## Usage

```
iset(which=iset.cur())
isets()
## S3 method for class 'iset'
x[[i]]
## S3 method for class 'iset'
x$name
## S3 replacement method for class 'iset'
x$name <- value
## S3 replacement method for class 'iset'
x[[i]] <- value
## S3 method for class 'iset'
x[i = 1:(dim(x)[1]), j = 1:length(x)]
## S3 replacement method for class 'iset'
x[i = 1:(dim(x)[1]), j = 1:length(x)] <- value
```

```
## S3 replacement method for class 'iset'
names(x) <- value
## S3 method for class 'iset'
names(x)
## S3 method for class 'iset'
length(x)
## S3 method for class 'iset'
dim(x)
```

### Arguments

x	iset object
i	row (case) index
j	column (variable) index
value	replacement value
which	An integer specifying an iSet ID or a string specifying a name of the iSet.
name	name of the variable

### See Also

[ivar](#), [iset.cur](#)

### Examples

```
data(iris)
s <- iset.new("iris", iris)
rm(iris)
s[]
names(s)
dim(s)
length(s)
s[[1]]
s$Species
s[1:5,c("Sepal.Length", "Petal.Length")]
rm(s)
# even if we get rid of the reference object,
# we can get it back
s <- iset()
s
# see ?ivar for more examples
```

---

iset.col

*Set color of cases in an iSet.*

---

### Description

This function sets the colors of cases of an iSet. Every iPlot is free to use the corresponding representation individually. Color of a case is a iSet-global property, that is all iPlots associated with an iSet will reflect the change.

**Usage**

```
iset.col(col)
```

**Arguments**

`col` Vector of integers specifying colors or a factor. The values are re-cycled if necessary. 0 has a special meaning denoting "no color", i.e. the plot is free to use the default color. This is also the default state of a newly created iSet.

**See Also**

[iset.new](#)

**Examples**

```
data(iris)
attach(iris)
iplot(Sepal.Width,Petal.Width)
ibar(Species)
iset.col(Species)
```

---

iset.list

*iSet management functions.*

---

**Description**

These functions are used to manage iSets. An iSet groups all iPlots which use the same data as a basis. All iPlots belonging to an iSet are linked, that is selecting cases in one iPlot cause the same cases to be highlighted in all other iPlots based on the same iSet. The linking is done on case-level, therefore the variables should have the same number of cases.

You can get the iSet object by calling the `iset` function. An iSet object can be used in a way similar to data frames, so subsetting and other operations work as expected. Subassignment is always done "live", so changing parts of the data will cause plots to be updated accordingly.

There is always exactly one current iSet. An initial iSet called "default" is created when the `iplots` library is loaded. All new iPlots are created using the current iSet. The set of functions described here allow the manipulation of iSets.

Note that changing the iSet also has an effect on `iplot.list` and `iobj.list` since iPlots are linked to their iSets.

`iset.new` creates a new iSet and makes it current

`iset.list` returns all iSets

`iset.cur` returns the ID of the current iSet

`iset.next` and `iset.prev` return the ID of the next resp. previous iSet in the list relative to the iSet specified by the argument.

`iset.set` makes the iSet with the specified ID current.

`iset.rm` removes the given iSet and all associated plots. Note that this may cause the current set ID may change (even if some other iSet is removed).

`print.iset` prints an iSet object.

### Usage

```
iset.new(name=NULL, payload=NULL)
iset.list()
iset.cur()
iset.next(which=iset.cur())
iset.prev(which=iset.cur())
iset.set(which=iset.next())
iset.rm(which=iset.cur())
## S3 method for class 'iset'
print(x, ...)
```

### Arguments

name	Name of the new iSet. If no name is specified, an automatic name of the form "data.X" is created where "X" is an integer. The name can be used instead on an ID in many cases.
payload	Initial contents of the iSet in the form of a list/data.frame or NULL if an empty iSet is desired.
which	An integer specifying an iSet ID or a string specifying a name of the iSet.
x	An iSet object to print
...	further parameters to be passed to print

### Methods

`show` signature(object = "iset"): is mapped to `print.iset`

### See Also

[iset](#), [iplot.list](#), [iobj.list](#)

---

iset.selected	<i>Selection / linked highlighting</i>
---------------	--

---

### Description

These functions modify the selection or return the currently selected (highlighted) cases.

`iset.selected` returns a vector of IDs of all currently selected cases (in the current iSet)

`iset.select` changes the selection of the current iSet. All plots are updated immediately to reflect the new selection.

`iset.sel.changed` returns TRUE if the selection has changed since last call to this function.

`iset.selectAll` selects everything

`iset.selectNone` clears all selections (yes, not a good name)

**Usage**

```

iset.selected()
iset.select(what, mode="replace", mark=TRUE, batch=FALSE)
iset.sel.changed(iset=iset.cur())
iset.selectAll(batch=FALSE)
iset.selectNone(batch=FALSE)

```

**Arguments**

what	specification of the new selection. This can be either a vector of case IDs or a logical vector.
mode	mode to be used when combining the previous selection and the current one. Supported modes are "replace", "union" and "intersect".
mark	mark to be used.
iset	iSet to query
batch	when set to TRUE dependents (e.g. plots) are not notified. This is useful for performing many complex updates at once without the need to re-draw all plots. Use with care as the system usually relies on event propagation.

**Value**

List of IDs of selected cases (`iset.selected`), boolean value (`iset.sel.changed`).

**See Also**

[iset.list](#)

**Examples**

```

data(iris)
attach(iris)
iplot(Sepal.Length, Petal.Length)
iset.select(Species=="virginica")

```

---

itext

---

*Add text to the current iPlot.*


---

**Description**

`text` adds the strings given in the vector `labels` at the coordinates given by `x` and `y` to the current `iPlot`. `y` may be missing since `xy.coords(x, y)` is used for construction of the coordinates.

**Usage**

```
itext(x, y=NULL, labels=seq(along=x), ax=NULL, ay=NULL, ..., plot = iplot.cur())
```

**Arguments**

<code>x, y</code>	Coordinate vectors of the text. <code>xy.coords</code> is used to obtain the coordinates.
<code>labels</code>	one or more character strings or expressions specifying the text to be written.
<code>ax, ay</code>	anchor coordinates of the text. They specify where is the point specified by <code>x</code> and <code>y</code> relative to the text. 0 means left resp. top and 1 means right/bottom. Therefore vertically and horizontally centered text can be obtained by using <code>ax=0.5</code> and <code>ay=0.5</code> .
<code>...</code>	any additional options are passed to <code>iobj.opt</code>
<code>plot</code>	parent plot

**Value**

Resulting `iObject`.

**See Also**

[ilines](#), [iobj.opt](#)

---

ivar

*Data (ivar) manipulation functions*

---

**Description**

The following functions are used to manipulate variables for iplots:

`ivar.data` returns the data associated with an `iVar` variable

`ivar.new` creates a new `iVar` variable in the current `iSet` using the supplied data

`ivar.new.map` same as `ivar.new` but for map (polygon) data

`ivar.update` replaces the content of a variable

`iVar` variables (deprecated - use `iset.new` instead)

`[.ivar` allows data subsetting of `ivar` contents

`[<- .ivar` allows partial replacement of `ivar` contents

`length.ivar` returns the length of a variable

`iset.updateVars` notifies all plots that variables have been changed. This function should not be normally used, because notification happens automatically, unless batch updates are performed.

**Usage**

```

ivar.data(var)
ivar.new(name = deparse(substitute(cont)), cont)
ivar.new.map(name, x, y)
ivar.update(var, cont, batch = FALSE)
## S3 method for class 'ivar'
length(x)
## S3 method for class 'ivar'
x[i, ...]
## S3 replacement method for class 'ivar'
x[...] <- value
## S3 method for class 'ivar'
print(x, ...)

iset.updateVars()

```

**Arguments**

<code>var</code>	an existing <code>iVar</code> variable
<code>cont</code>	desired contents - usually a numeric vector or a factor
<code>name</code>	variable name in an <code>iSet</code>
<code>batch</code>	if set to <code>TRUE</code> then plots are not automatically notified about the change. This allows an update of multiple variables in batches without constant re-painting of the plots.
<code>x</code>	object to subset or x-coordinate of the map data
<code>y</code>	y-coordinate of the map data
<code>i</code>	index - if missing all data are returned
<code>...</code>	indices
<code>value</code>	replacement value

**Details**

All data that will be displayed in an `iPlot` are organized in `iSets`. Each `iSet` contains variables, called `iVars`, that represent the displayed data. Typically an `iSet` corresponds to a data frame and an `iVar` corresponds to a column in a data frame. All variables in one `iSet` must have the same length and indexing order. This allows `iPlots` to perform proper linking of all plots.

Before some data can be displayed in an `iPlot`, they are put into an `ivar` using the `ivar.new` function. Each variable has a name that is unique within an `iSet`. This process is automatic if you pass arbitrary vectors to the `iPlots` plotting functions. However, it is possible (and desirable) to register variables beforehand using `iset.new`. It is much faster to use `iVars` in the plots instead of the raw data.

Once an `iVar` is created, it is merely a reference to the data in the `iSet`. The regular subsetting operators `[]` or `ivar.data` function can be used to retrieve the data back to R.

In addition, it is possible to update the contents of an `iVar` using the `ivar.update` function. Once the variable is updated, all plots that use the variable will be updated, too, and reflect the change, unless `batch` was set to `TRUE`. The notification is performed using the `iset.updateVars` function.

**Methods**

`show` signature(object = "ivar"): is mapped to `print.ivar`

**See Also**

[iset.new](#), [ibar](#), [iplot.list](#), [iplot.opt](#)

**Examples**

```
data(iris)
s <- iset.new("iris", iris)
rm(iris) # we don't need iris anymore
s[1:5,]
s$Species
s$Species[]
s[1:5,c("Sepal.Width", "Sepal.Length")]
s$Test <- factor(c("yes", "no")[runif(dim(s)[1])+1.5])
s[1:5,]
iplot(s$Sepal.Length, s$Petal.Length)
ibar(s$Species)
iplot.location(400, , TRUE)
iset.select(s$Test[]=="yes")
ibar(s$Test)
iplot.location(400, 250, TRUE)
# use iset.rm() to remove everything
```

# Index

`!=.iobj (iobj.list)`, 11

**\* hplot**

- `iabline`, 2
- `ibar`, 3
- `ibox`, 4
- `ievent.wait`, 5
- `ihammock`, 6
- `ihist`, 7
- `ilines`, 8
- `imap`, 9
- `imosaic`, 10
- `ipcp`, 13
- `ipplot`, 14
- `ipplot.manip`, 17
- `ipplot.opt`, 18
- `iraster`, 20
- `ivar`, 26

**\* iplot**

- `iobj.list`, 11
- `iobj.opt`, 12
- `ipplot.data`, 15
- `ipplot.list`, 16
- `iset`, 21
- `iset.col`, 22
- `iset.list`, 23
- `iset.selected`, 24
- `itext`, 25

`==.iobj (iobj.list)`, 11

`[.iset (iset)`, 21

`[.ivar (ivar)`, 26

`[<-.iset (iset)`, 21

`[<-.ivar (ivar)`, 26

`[[.iset (iset)`, 21

`[[<-.iset (iset)`, 21

`$.iset (iset)`, 21

`$<-.iset (iset)`, 21

`dim.isset (iset)`, 21

`iabline`, 2, 11

`ibar`, 3, 4, 6–8, 10, 13, 15, 16, 28

`ibox`, 4

`ievent.wait`, 5, 5

`ihammock`, 6

`ihist`, 3, 4, 6, 7, 8, 10, 13, 15, 16

`ilines`, 2, 8, 11, 16, 26

`imap`, 9

`imosaic`, 10

`iobj.cur`, 11

`iobj.cur (iobj.list)`, 11

`iobj.get`, 11

`iobj.get (iobj.list)`, 11

`iobj.list`, 11, 11, 12, 23, 24

`iobj.next`, 11

`iobj.next (iobj.list)`, 11

`iobj.opt`, 2, 8, 12, 20, 26

`iobj.prev`, 11

`iobj.prev (iobj.list)`, 11

`iobj.rm`, 11

`iobj.rm (iobj.list)`, 11

`iobj.set`, 11

`iobj.set (iobj.list)`, 11

`ipcp`, 13

`ipplot`, 3, 6, 7, 9, 14, 15, 16

`ipplot.backend (ipplot.manip)`, 17

`ipplot.cur`, 16

`ipplot.cur (ipplot.list)`, 16

`ipplot.data`, 15

`ipplot.list`, 3, 4, 7, 8, 10, 13, 15, 16, 16, 19, 23, 24, 28

`ipplot.location (ipplot.manip)`, 17

`ipplot.manip`, 17

`ipplot.next`, 16

`ipplot.next (ipplot.list)`, 16

`ipplot.off`, 16

`ipplot.off (ipplot.list)`, 16

`ipplot.opt`, 3, 4, 6, 7, 9, 10, 13–15, 18, 28

`ipplot.prev`, 16

`ipplot.prev (ipplot.list)`, 16

`iplot.resetZoom(iplot.manip)`, 17  
`iplot.rotate(iplot.manip)`, 17  
`iplot.set`, 16  
`iplot.set(iplot.list)`, 16  
`iplot.setExtendedQuery(iplot.manip)`, 17  
`iplot.size(iplot.manip)`, 17  
`iplot.zoomIn(iplot.manip)`, 17  
`iplot.zoomOut(iplot.manip)`, 17  
`iraster`, 20  
`iSet(iset)`, 21  
`iset`, 21, 24  
`iset-class(iset)`, 21  
`iset.brush`, 19  
`iset.brush(iset.col)`, 22  
`iset.col`, 22  
`iset.cur`, 22, 23  
`iset.cur(iset.list)`, 23  
`iset.list`, 23, 23, 25  
`iset.new`, 23, 26–28  
`iset.new(iset.list)`, 23  
`iset.next`, 23  
`iset.next(iset.list)`, 23  
`iset.prev`, 23  
`iset.prev(iset.list)`, 23  
`iset.rm`, 24  
`iset.rm(iset.list)`, 23  
`iset.sel.changed`, 24  
`iset.sel.changed(iset.selected)`, 24  
`iset.select`, 24  
`iset.select(iset.selected)`, 24  
`iset.selectAll(iset.selected)`, 24  
`iset.selected`, 24, 24  
`iset.selectNone(iset.selected)`, 24  
`iset.set`, 23  
`iset.set(iset.list)`, 23  
`iset.updateVars(ivar)`, 26  
`isets(iset)`, 21  
`itext`, 25  
`iVar(iset)`, 21  
`ivar`, 22, 26  
`ivar-class(iset)`, 21  
`ivar.new.map`, 9  
  
`length.iset(iset)`, 21  
`length.ivar(ivar)`, 26  
  
`names.iset(iset)`, 21  
`names.iset<-(iset)`, 21  
`names<- .iset(iset)`, 21  
  
`plot`, 4, 14  
`print.iobj(iplot.manip)`, 17  
`print.iplot(iplot.manip)`, 17  
`print.iset(iset.list)`, 23  
`print.ivar(ivar)`, 26  
  
`show,iset-method(iset.list)`, 23  
`show,ivar-method(ivar)`, 26  
  
`text`, 25  
  
`xy.coords`, 8, 26