'plgraphics': A user-oriented collection of graphical R-functions based on the 'pl' concept

Werner A. Stahel, ETH Zurich

August 27, 2018

Abstract

The package, `plgraphics`, collects enhanced versions of basic plotting functions. It is based on a paradigm between the basic R graphics elements and the more computer science oriented ggplot concepts. The intention is to furnish user-oriented functions that allow efficient production of useful graphics.

1 Introduction

The plotting functionality is the historical origin of the R package. It has been introduced half a century ago and has grown for a while. For the sake of upward compatibility, it has been essentially unchanged for several decades.

New graphical concepts, adjusted to the development of graphical devices and computer science ideas have been implemented in new packages, notably `ggplot` ...

The intention of the package `plgraphics` is to implement some functions that provide efficient production of simple to rather sophisticated plots, but are still based on the core R functionality. They have been developed over a long time with a focus on allowing for readily interpretable graphical diagnostics for regression model development.

The general idea is that it should be easy to produce standard plots by calling `plot(x,y)` or `plot(y~x, data=dd)`, as well as enhancing the plot by adding an argument `smooth=T` to ask for a smoother or specifying a column in the dataset that drives the color or yields labels to mark the points to be shown. The plots should remain useful if there are outliers of one of the two variables is a grouping factor instead of a quantitative variable.

Asking that a basic function provides many variations under the control of the user means that a large list of arguments must be available. Some of these variations depend on the taste of the user. They can be specified in a kind of "style" list, analogous to `options` and `par`, which is called `userOptions`.

The package also provides enhanced low level graphical functions like `plpoints`, which extends the functionality of `points`. This leads to a very short basic scatterplot function `plyx` that can easily be modified by the user.

This document presents the main features of the package `plgraphics` and explains the concepts behind them.

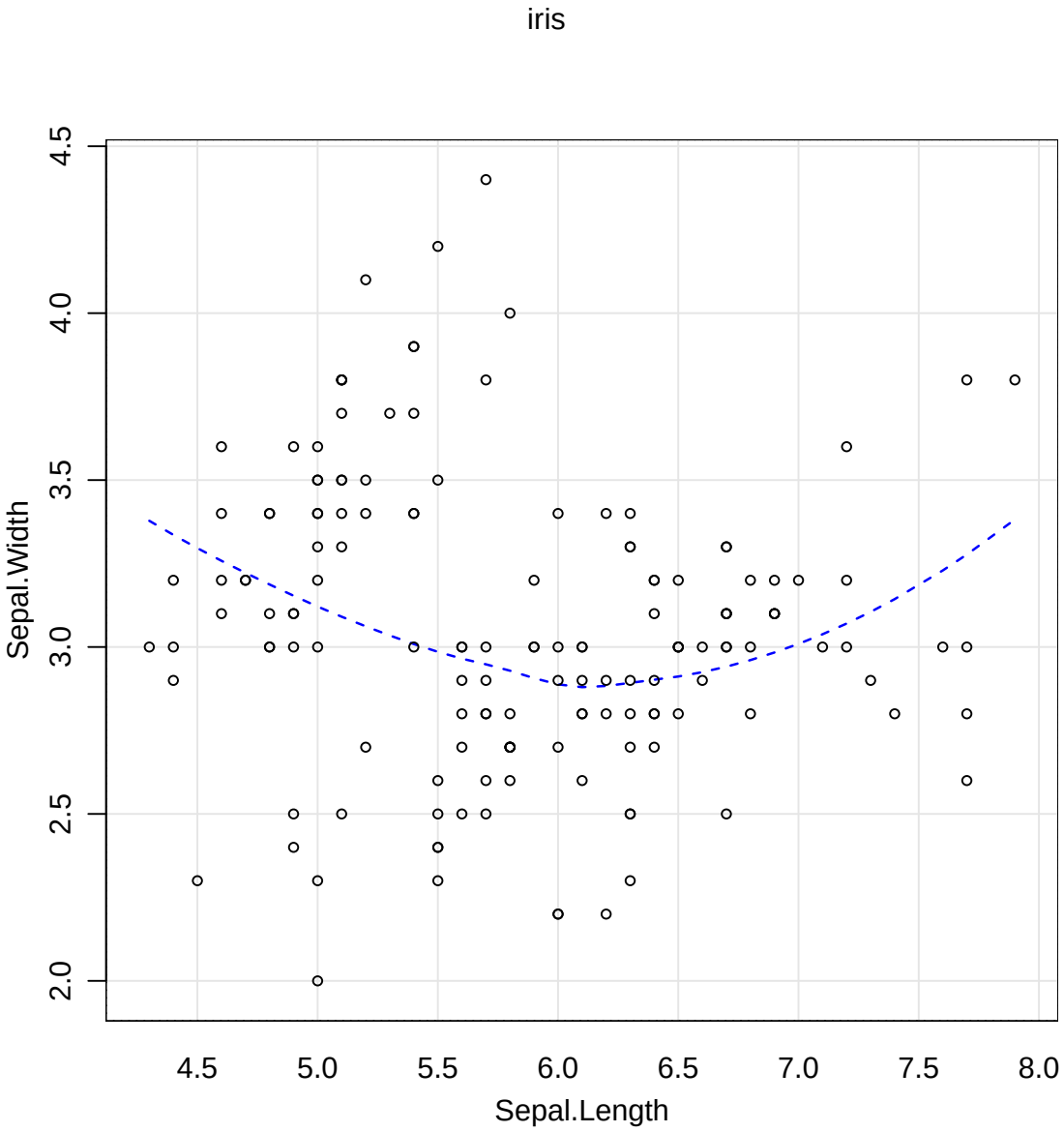
The package *will be* available from `R-forge`, e.g. by calling
`install.packages("plgraphics", repos="http://r-forge.r-project.org")`.

2 Scatterplots

2.1 The basic scatterplot

A basic scatterplot is generated by calling `plyx` (plot y on x).

```
plyx(Sepal.Width~Sepal.Length, data=iris)
```

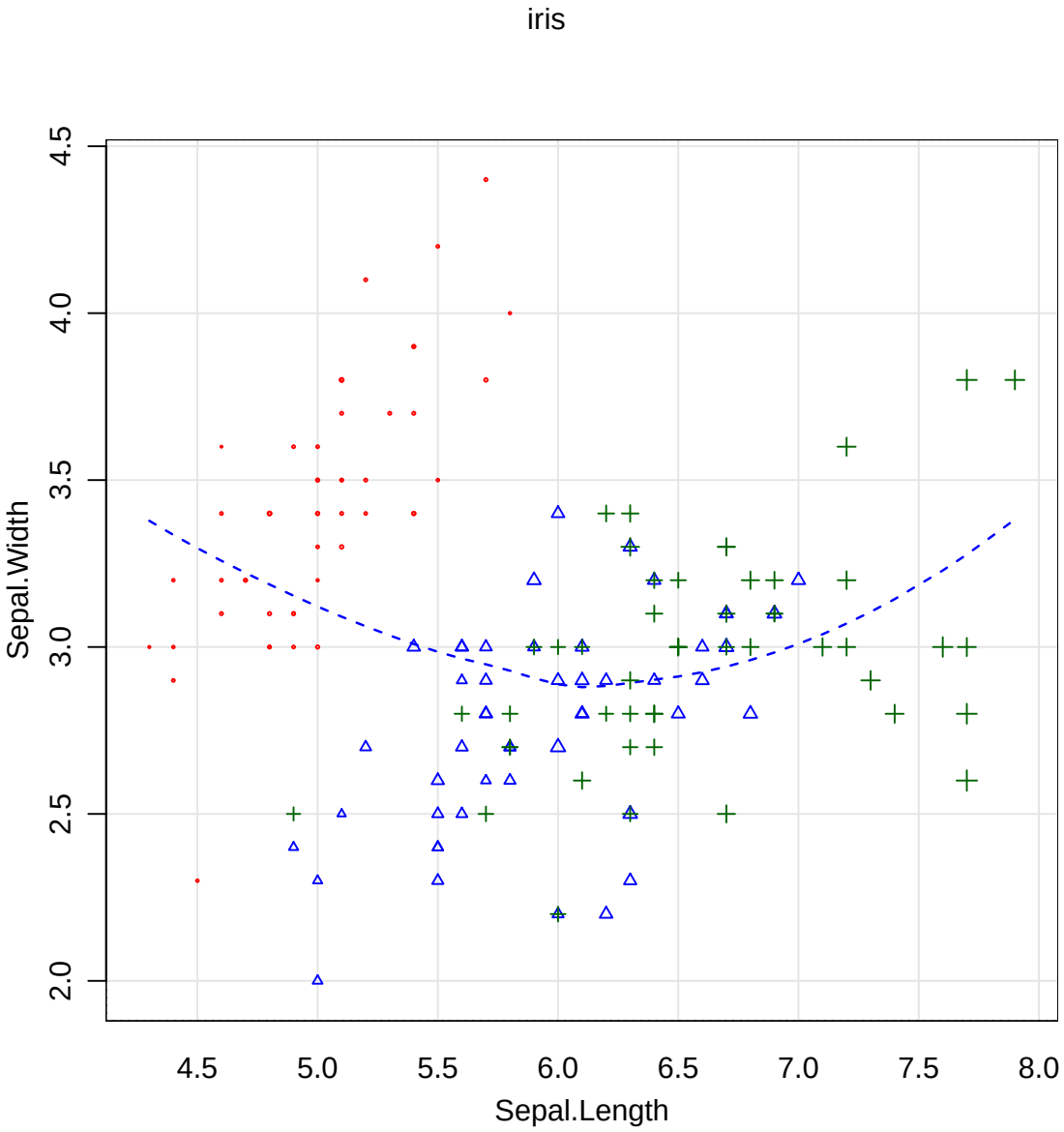


Aug 27 01:0:48

Clearly, this stongly resembles the result of simply calling `plot`, except for the thin gridlines and some documentation added by default: The name of the dataset is shown as a (sub-) title, and there is a small text in the lower right corner that shows the date.

More arguments allow to specify many aspects of the plot.

```
plyx(Sepal.Width~Sepal.Length, data=iris,
     psize=Petal.Length^2, pcol=Species, pch=Species, cex=1.5)
```



Aug 27.01/0:48

The argument `psize`) sets the size of the plotting symbols such that their area is proportional to it. The median size is determined by `cex`. By default, this value adjusts to the number of observations.

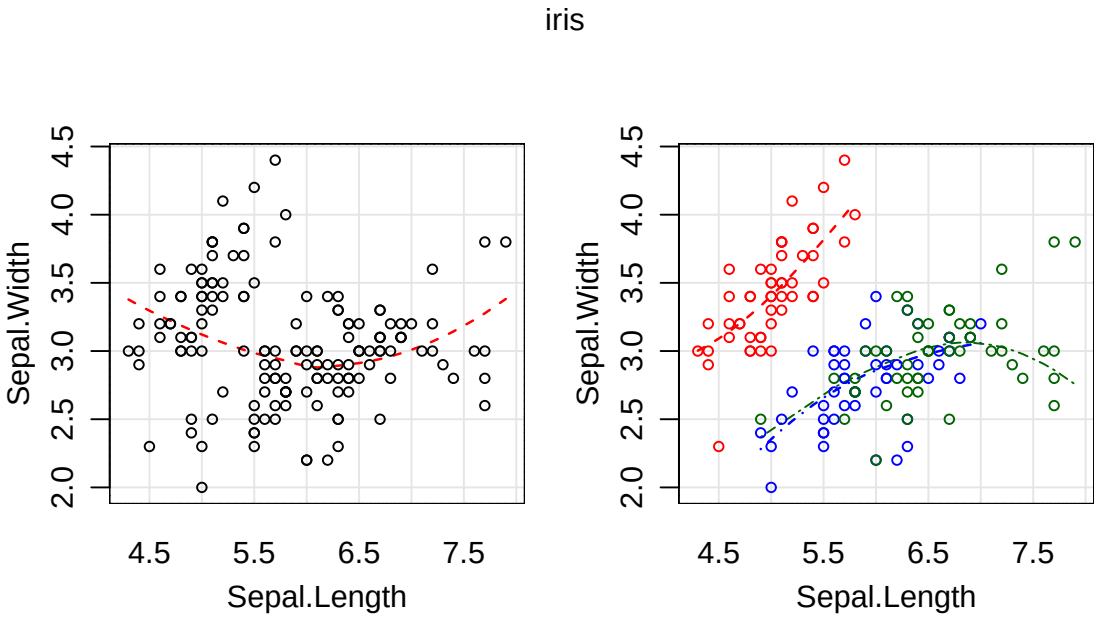
See ... for details.

Smooth. A smooth line fitting the data in the plot is requested by typing `smooth=TRUE`. The line type, width and color are modified by respective arguments.

Smooths can also be fitted groupwise by specifying `smooth.group`.

```
plmfg(1,2)
plyx(Sepal.Width~Sepal.Length, data=iris, smooth=TRUE, smoothlines.col="red")

plyx(Sepal.Width~Sepal.Length, data=iris, smooth=TRUE, smooth.group=Species,
     pcol=Species)
```



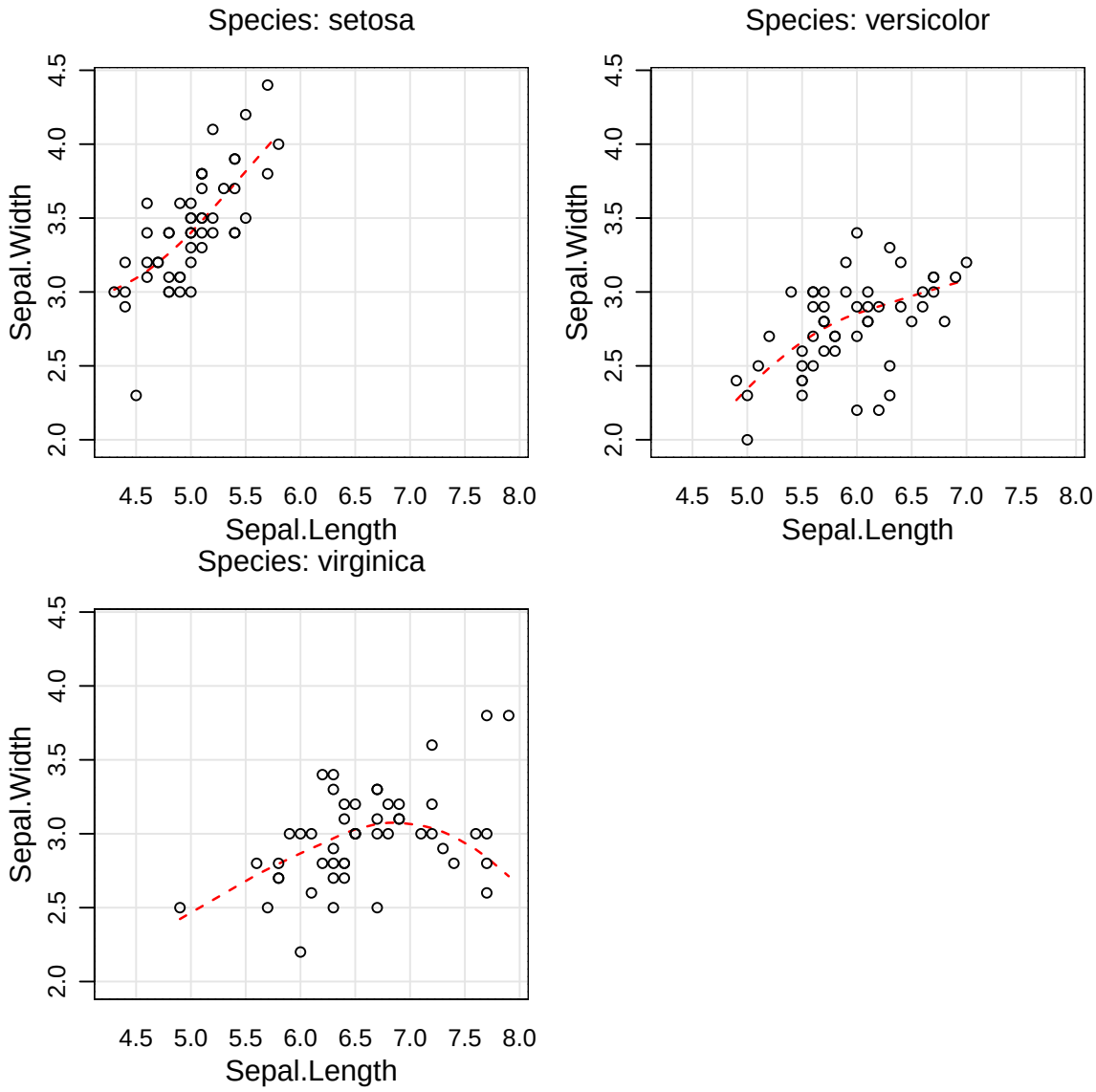
Aug 27 01:0:48

Setting `pcol=Species` allowed the color of plotting symbols and smooth lines to be the same.

If the argument `group` is specified, separate plots will be generated for the different groups, thereby maintaining the plot ranges.

```
plmfg(2,2)

plyx(Sepal.Width~Sepal.Length, data=iris, smooth=TRUE, group=Species)
```

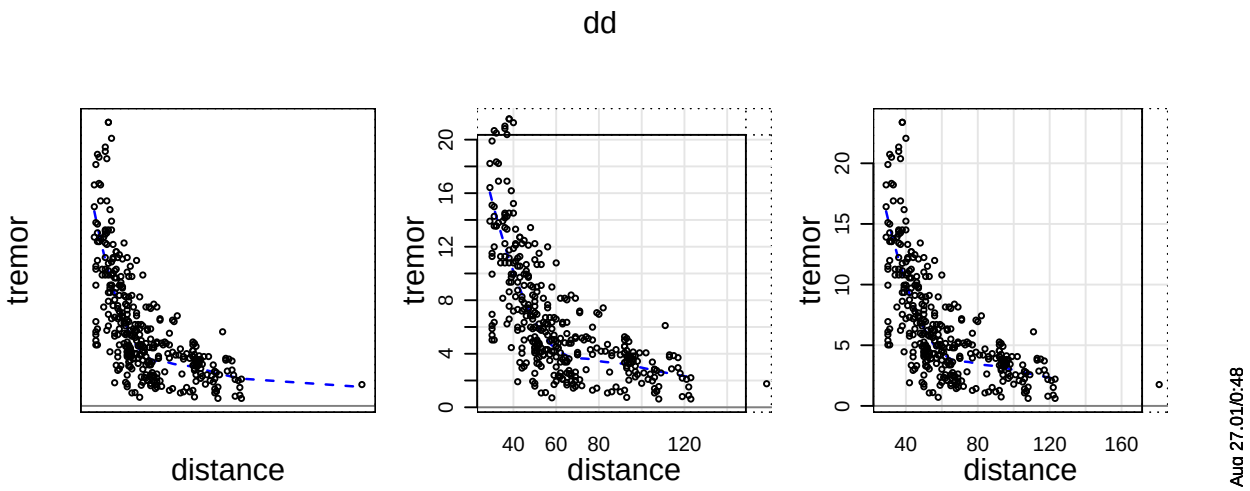


Inner range of plots. When there are outliers in the data, plots are dominated by their effect of determining the plotting range. This means that the user who would like to see more detail about the "regular" observations needs to gnerate a new plot, specifying the limits of the plotting range by `xlim` and `ylim`.

In order to avoid the urge for two versions of the plot, an "inner plotting range" is determined, based on robust measures of location and scale. Outside this range, there is a plotting margin where coordinates are transformed with a highly nonlinear function in order to accomodate all outliers. In these margins, the order of coordinates is still maintained, thus allowing to see which points are further

out than others, but quantitative information is distorted by the transformation. The figure shows data from the blasting example with an added outlier, plotted without and with inner plotting limits.

```
data(d.blast)
dd <- d.blast
dd$distance[2] <- 200
plmfg(1,3)
plyx( tremor~distance, data=dd, innerrange=FALSE)
plyx( tremor~distance, data=dd)
plyx( tremor~distance, data=dd, innerrange.factor=5)
```



If `innerrange=TRUE`, which is the default, the `plgraphics` functions will determine an "inner plotting range" based on the 20% trimmed mean and a 20% trimmed scale by default.

The function `robrange`, which is called by `plinnerrange`, determines the α-trimmed mean with $\alpha = 0.2$ as the location and the (one-sided) trimmed mean of the deviations from it. It adjusts this latter mean to obtain an approximately consistent estimate of the standard deviation for normal observations to obtain the scale estimate. It then calculates the location plus-minus `innerrangeFactor` times the scale to get a potential inner range. The final inner plotting range will be the intersection of this and the ordianry range of the values.

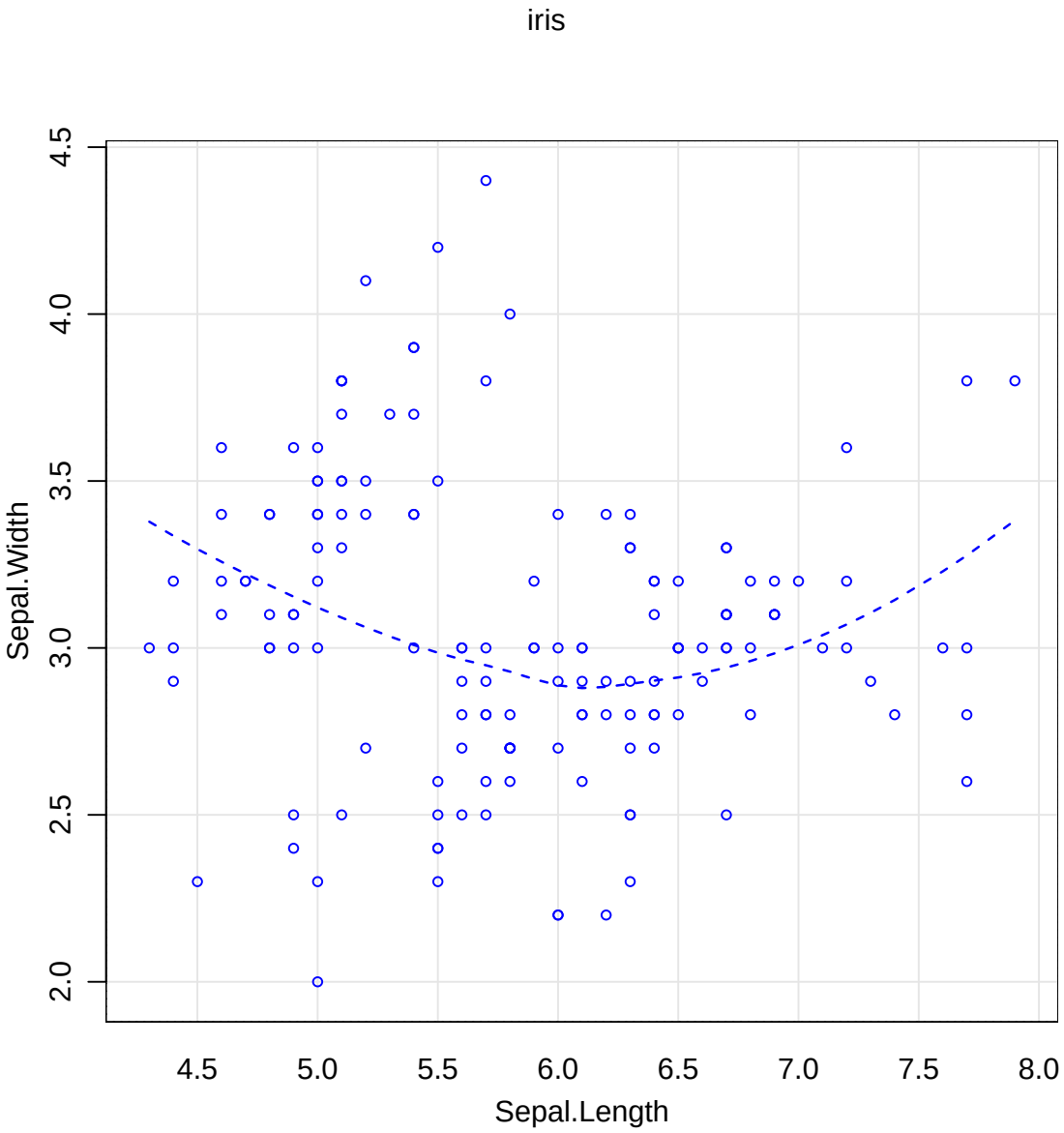
The default value of `markextremes` is 0 for `plyx`. If the argument is `NA`, it depends on the number of observations: It is $1/(2\sqrt{n})$.

Factors, `mboxes`

Censored data

Raw or transformed variables? max(x1,x2)

```
t.plopt <- ploptions(basic.col="blue")
plyx(Sepal.Width~Sepal.Length, data=iris, ploptions=t.plopt)
```



Aug 27.01/0:48

3 Scatterplot matrix

Multiple x

Groups

Same scales

`plmatrix`

4 Regression diagnostic plots

diagnostics should be specific scale diagnostic should work even when bias is signalled

`plot.regr`

Residuals against fit

Absolute residuals against fit residuals from smooth censored: no intervals

QQ-plot only for Gaussian

markextremes

plresx This function plots the residuals against the input variables, either in raw or in transformed form.

The raw input variables are those appearing in the formula, as delivered by `all.vars(formula)`.

The transformed input variables are those appearing in the terms of the formula, as delivered by `rownames(attr(terms(formula[1:2]), "factors"))`.

fitcomp: use model.frame -¿ model.matrix -¿ lm.fit

plrex2x

5 Options

ploptions The graphical elements, like plotting character, color, line types, etc. to be used in pl graphics are specified in the pl options. They are analogous to the ordinary options, with two differences:

- The pl options are stored in a list `.ploptions` which is not erased when leaving the R session. It can therefore be stored with the global environment.
- There is a list `ploptionsDefault` in the package. It collects the packages default settings and is used as a backup if some components should not be contained in `.ploptions`.

The function `ploptions` is used to set and get pl options, in the same way that `options` sets and gets the basic R options.

The basic concept behind the pl option list is that all pl functions use it as a resource to find the graphical elements.

Remark: This concept is a version of a more general idea, saying that the default values of any "high level" R function should have an associated list of default arguments, which is not contained in the function definition, but stored separately. This allows the user to specify his own style by changing these defaults and storing them in a kind of style file to be loaded at the start of each session. Here, there is only one list because the pl functions need the same graphical elements.

Thus, a graphical element like a plotting character is generally searched in

1. the argument list of the calling function
2. the `ploptions` argument of the calling function
3. the `.ploptions` list
4. the list `ploptionsDefault` in the package `plgraphics`

Plotting data

pdata attributes of points, grouping

plotting elements for grouping

attributes of variables label axis colors lty, lwd?) to be used if variable is plotted vertically (only if multiple vars?)

very often NULL, adaptation to data

.parguments

Some graphical elements will depend on the number of observations default: cex

title: cex, cexmin

6 Low level graphics and auxiliary functions

plframe,

plpoints, plab; pch for is.na(plab) or plab=="" cex: plappearance$cex * plappearance$default$cex (priorities for) plotting character censored

plsmooth, plsmlines,

plreflines

pltitle adaptation of cex

pllimits, plcoord

7 Details

7.1 plargs, ploptions, default values

(if needed, see above)

Default values `i.def` `i.getploption` and `i.getplopt`

Some arguments to low level pl functions need to be set by changing the `ploptions` argument. Example:

residuals in pargs are data.frame

variable colors, ... stored in pdata generated in pl.control avoiding elements already in use

7.2 Components of plptions

innerrange Usually, `innerrange` is a logical, indicating if inner ranges should be determined `innerrange.limits` is a vector of length 2 giving the range to be applied.

?? Both can also be a named list of such objects, where the names reflect the variables.

set innerrange.limits by calling genvarattributes

7.3 Point labelling and plotting character

Priorities:

1. If they are specified by the respective argument to high level `pl` function (and evaluated by `pl.control`), this has priority (exemption, see 2.).
2. In the case of multiple ys, colors are determined primarily by the argument `ycol` of the high level `pl` function, scondarily by the `col` attribute of the variables. Thirdly, the `variables` component of `plappearance` is used, avoiding colors that are already specified for some variables by the foregoing steps. [See `i.getPlattributes`, called by `genvarattributes` in `pl.control`.]
   If `pch` is not determined otherwise (argument, see 1., or group, see 3.) it is set in the same way.
   For plots of type `l` or `b`, the line type `lty` is determined in the same way as the color.
3. If there is grouping and only a single y, the group determines `pch` and its color by the `group` component of `plappearance` unless set by 1. above.
4. In other cases, the `default` component of `plappearance` is used.

7.4 Groups

Color If color (`pcol`) is a factor, it will be converted into `ploptions("group.col")[as.numeric(pcol)]`. In order to give color by color names, make sure that `pcol` is a character variable.

7.5 Standardized residuals

$$R_i^* = R_i \Big/ \left(\widehat{\sigma}\sqrt{w_i}\sqrt{1-H_{ii}}\right)$$

Standardization ratio: $\texttt{stratio}_i = R_i^*/R_i$

`i.stres` calculates leverages, standardized residuals, and `strratio` according to this formula. For binary and Poisson models, ...

Cook's distance:

$$d_i^{(C)} = \frac{R_i^2\, H_{ii}}{p\widehat{\sigma}^2\,(1-H_{ii})^2} = (1/p)\, R_i^{*2}\, H_{ii}/(1-H_{ii})\ ,$$

It is constant, $= d$, on the curve

$$R_i^{*2} = d\, p\, (1-H_{ii})/H_{ii}$$

A rule suggests $d = 4/(n-p)$ as a warning level. Curves are drawn for $d = \texttt{cookdistlines}^2/(n-p)$.

This is the end of the story for the time being. I hope that you will get into using `regr` and have good success with your data analyses. Feedback is highly appreciated.

Werner Stahel, `stahel at stat.math.ethz.ch`