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

Werner A. Stahel, ETH Zurich

September 18, 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.

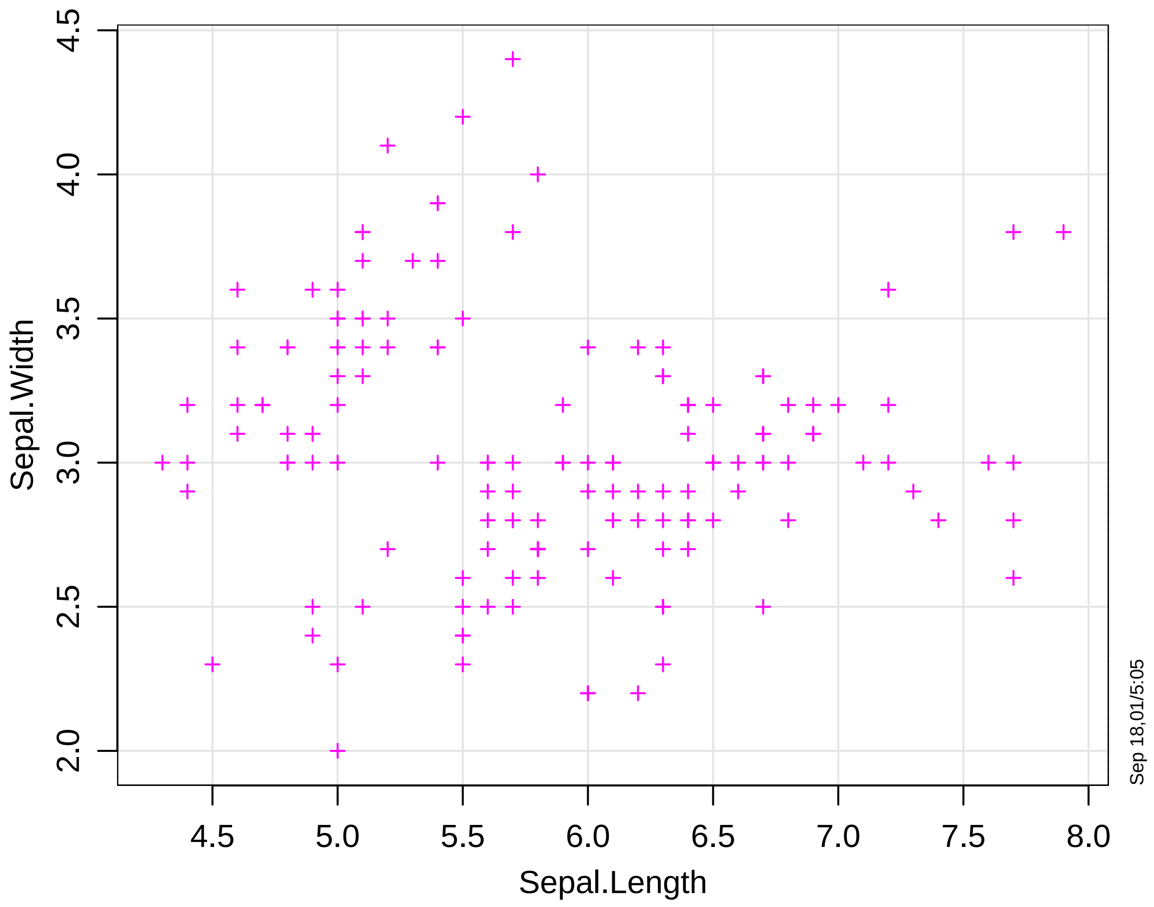
The package *will be* available from `R-forge`, e.g. by calling
`install.packages("plgraphics", repos="http://r-forge.r-project.org")`.

2 Scatterplots

2.1 The basic scatterplot

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

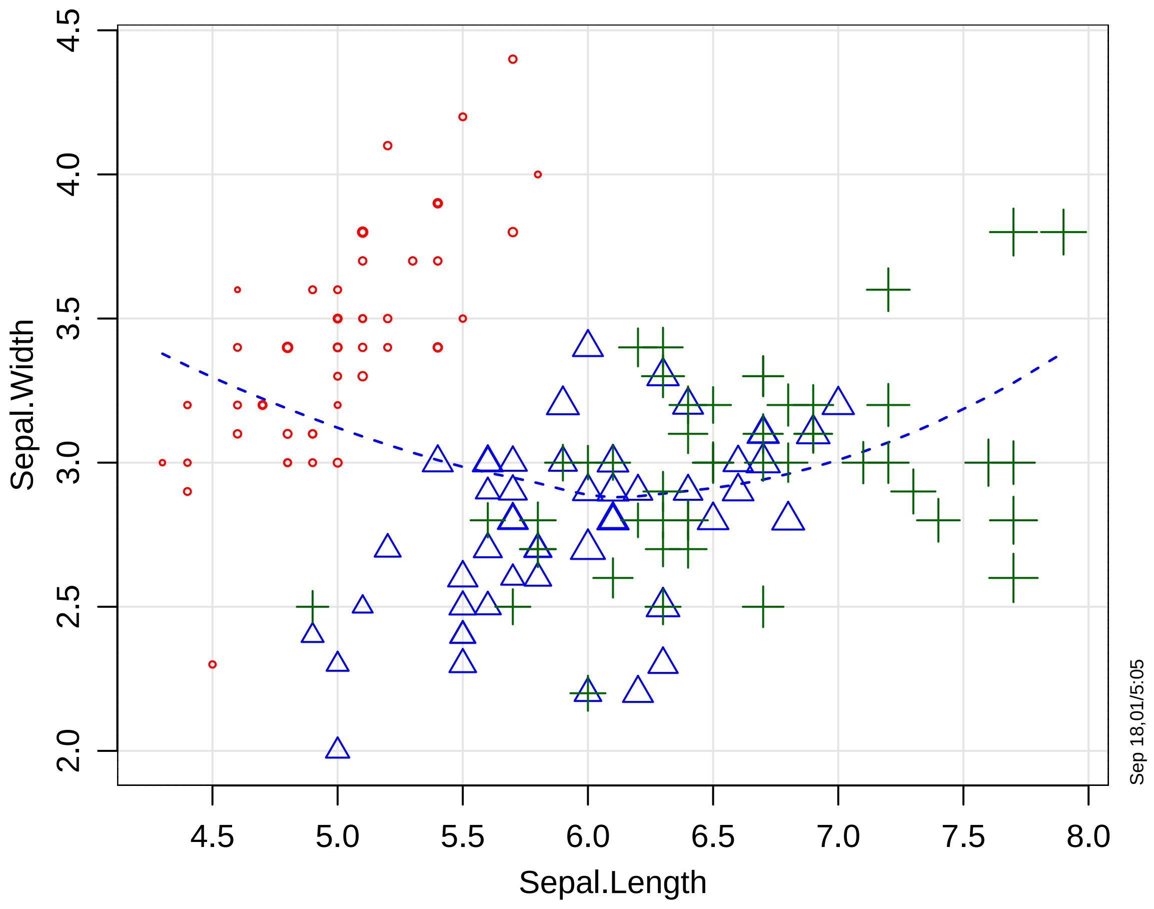
```
plyx(Sepal.Width~Sepal.Length, data=iris, smooth=FALSE)
```



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. Without `smooth=FALSE`, a smooth line is added, see below.

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)
```



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.

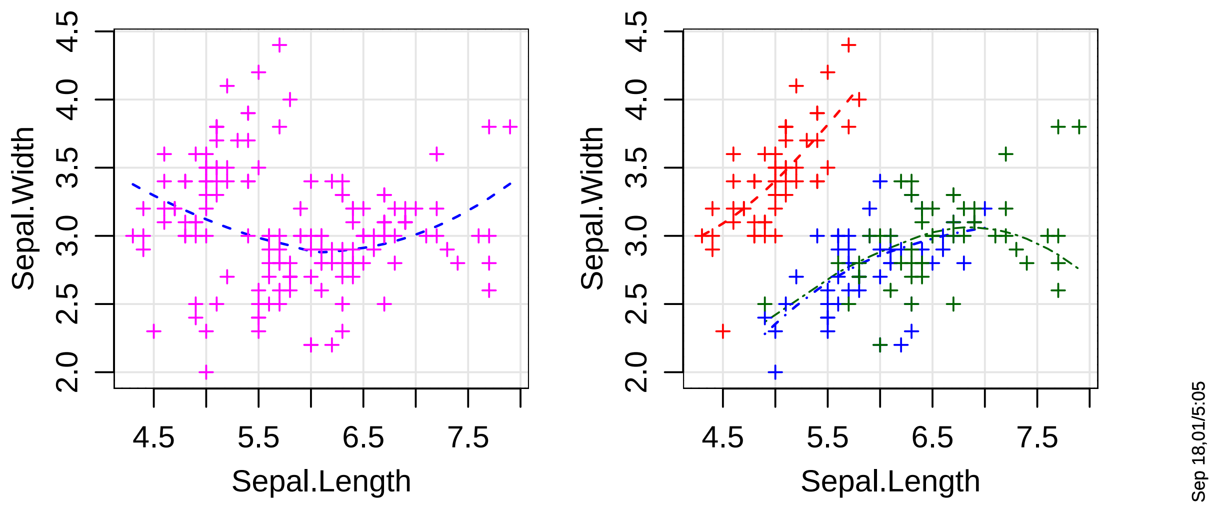
See ... for details.

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)
```

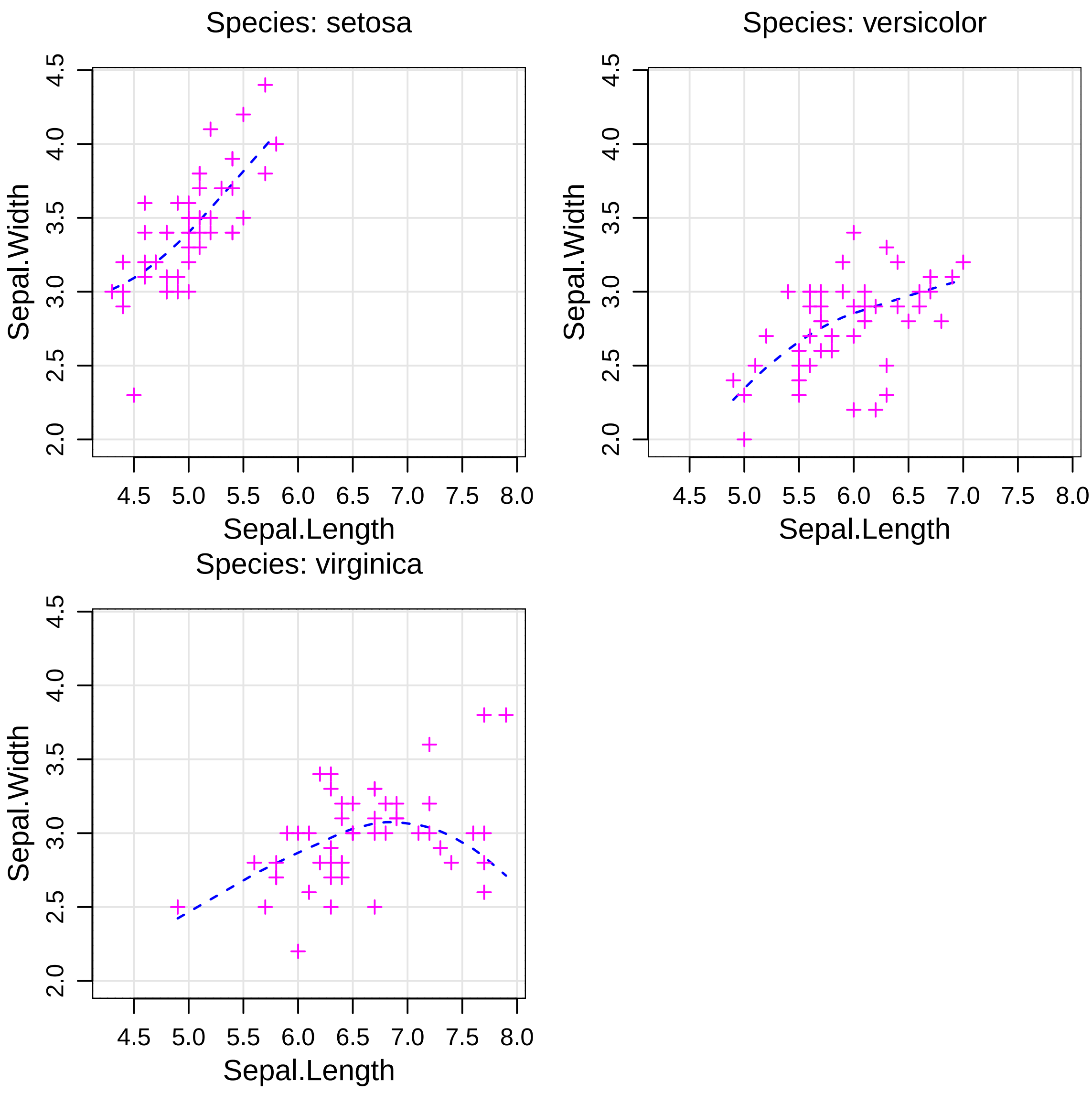


Setting `pcol=Species` allowed the color of plotting symbols and smooth lines to be the same. The call to `plmfg` splits the screen essentially like `par(mfrow=c(1,2))`.

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)
```

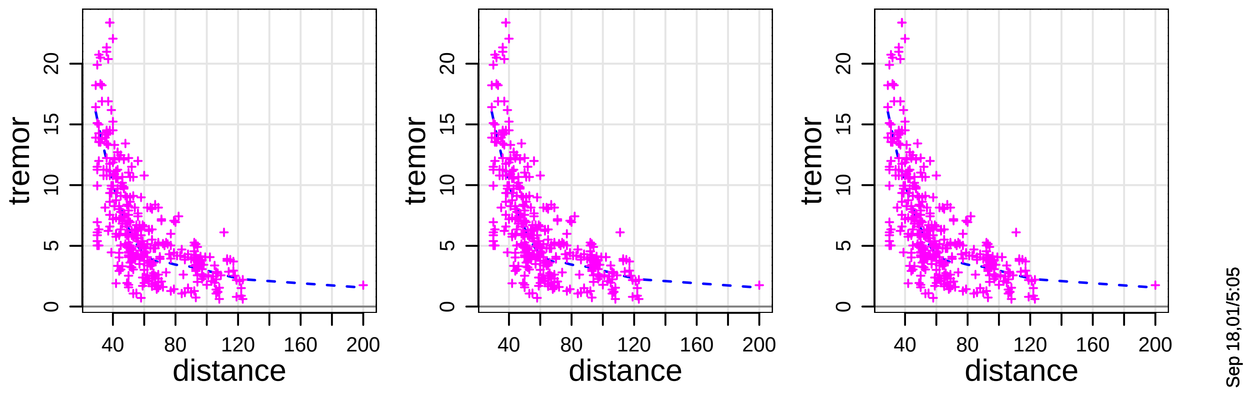


Sep 18,01/5:05

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.

2.2 Multiple y and x

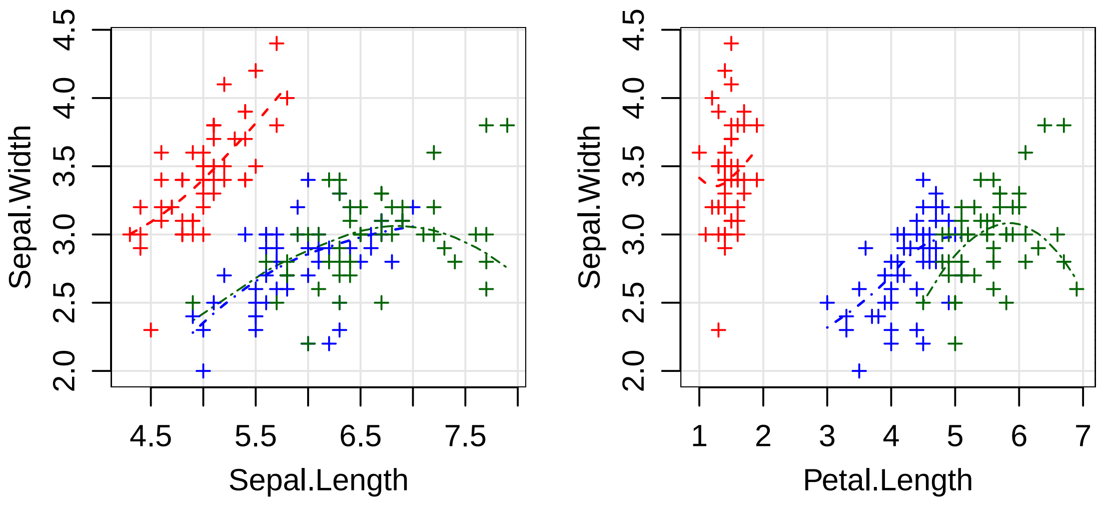
Two or more variables may be given to be plotted on the vertical axis, in the sense of `matplot` of R. Often, these are parallel time series, and it is convenient to ask for lines connecting the points, either `type="l"` or `type="b"`. `plyx` will choose different scales for the different variables unless `rescale=FALSE`.

```
plyx(1:40, EuStockMarkets[1:40,], type="b")

## Error in xy.coords(x, y, xlabel, ylabel, log):  'x' and 'y' lengths differ
```

If multiple x variables are given, a separate plot is drawn for each of them.

```
plmfg(1,2)
plyx(Sepal.Width~Sepal.Length+Petal.Length, data=iris,
     smooth.group=Species, pcol=Species)
```

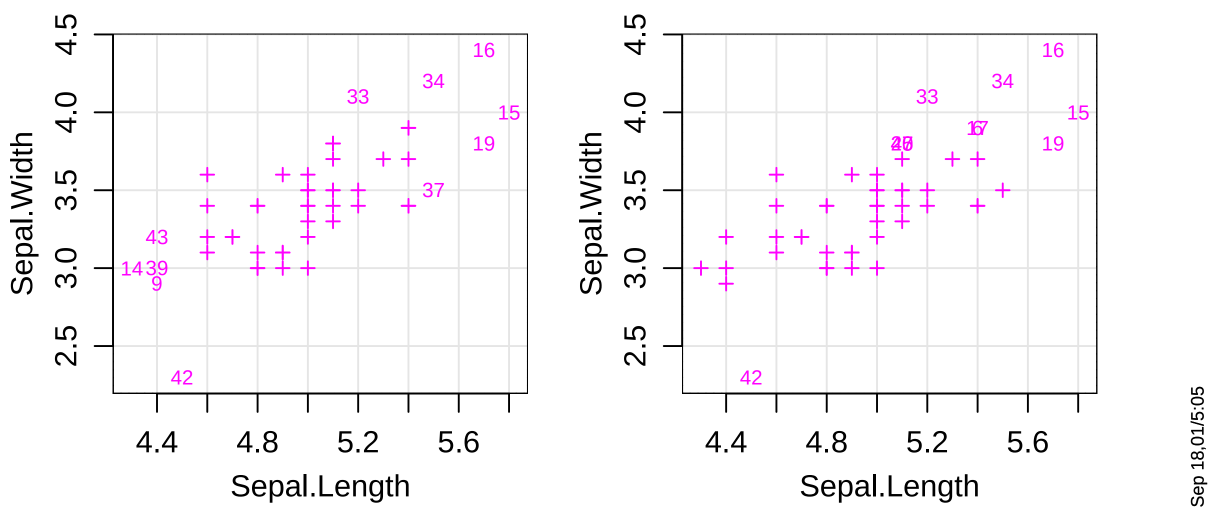


Sep 18,01/5:05

2.3 Marking extreme points

Extreme points are often of interest. They can easily be identified if they are labelled. This is achieved by setting the argument `markextremes`.

```
plmfg(1,2)
plyx(Sepal.Width~Sepal.Length, data=iris[1:50,], smooth=F,
     markextremes=0.1, cex=0.7)
plyx(Sepal.Width~Sepal.Length, data=iris[1:50,], smooth=F,
     markextremes=list(0,c(0.02,0.2)), cex=0.7)
```



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

2.4 Factors, multibox plot

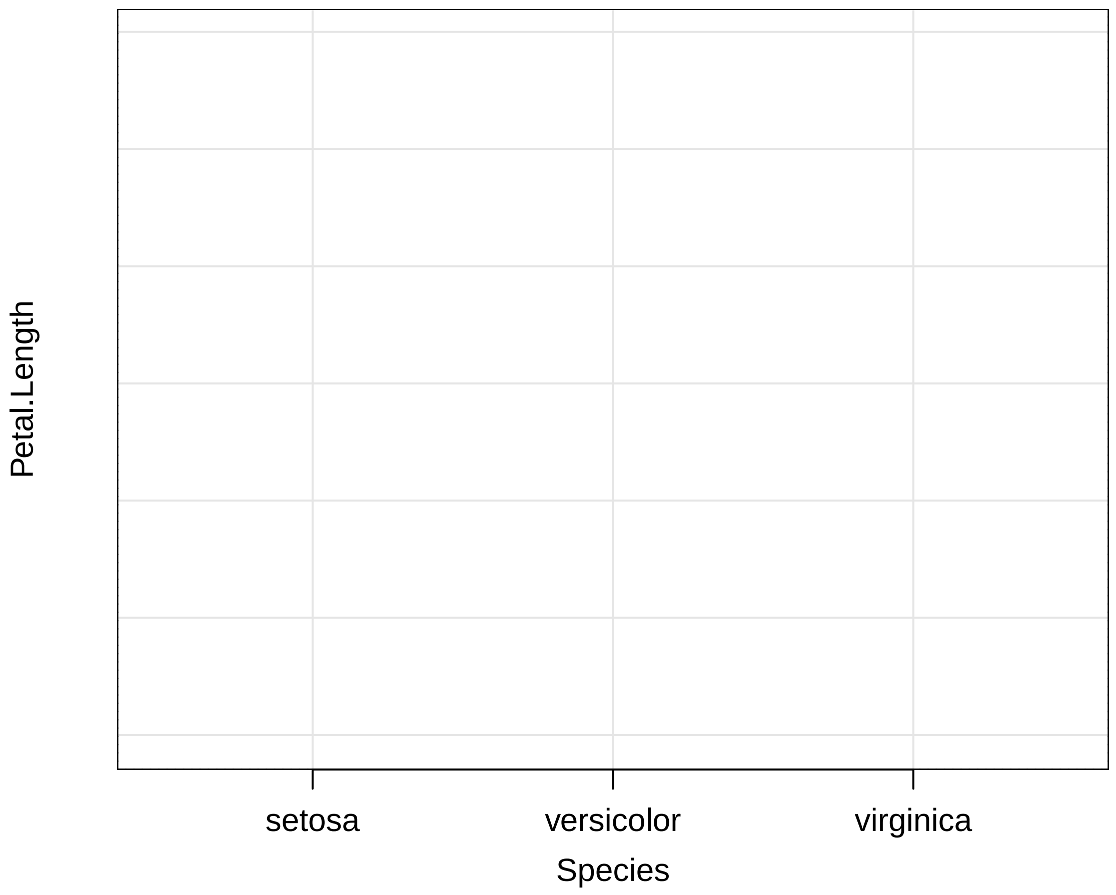
If the x variable is a factor, R's generic plot function draws box plots. Since this often results in too much simplification, `plyx` shows a “multibox plot”, which is a refinement of a boxplot, to be described in more detail below.

[to be implemented]

The multibox plot can also be called directly.

```
plmfg()
plmboxes(Petal.Length~Species, data=iris)


## Error in axis(axis, at = lat, labels = llab, col = col, ...):  'labels' is supplied
and not 'at'
```



Censored data

Raw or transformed variables? max(x1,x2)

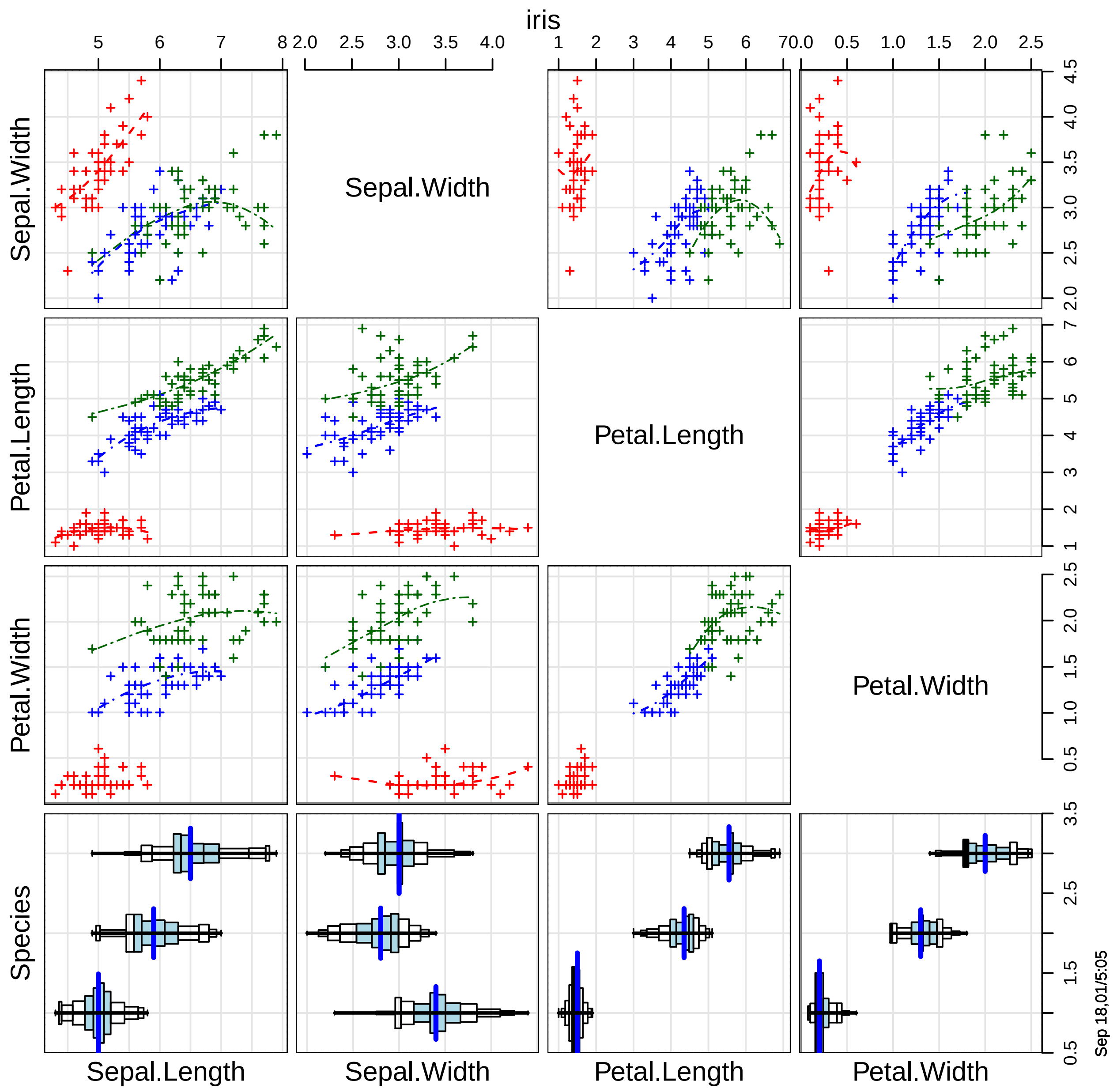
3 Scatterplot matrix

The `pairs` plotting function of R has some inconvenient restrictions. If the number of variables is larger than about 8, the panels become so small that hardly anything can be observed. Furtherore, factors are simply converted to numeric.

The function `plmatrix` has much more flexibility. If used for a small number of variables, it does a similar job as `pairs`, but also provides the flexibility for the panels that have been described above.

`plmatrix`

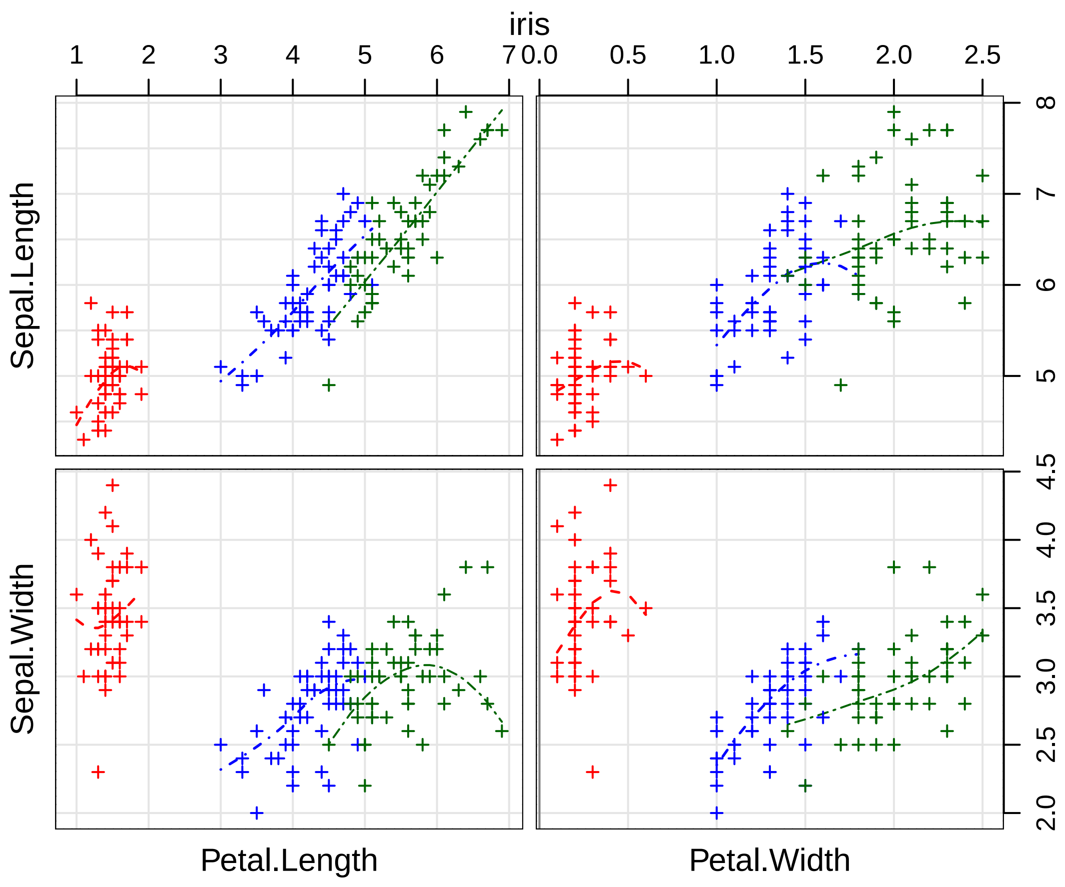
```r
plmatrix(iris, smooth.group=Species, pcol=Species)
```



```
## [1] "plmatrix: done"
```

`plmatrix` can also show any submatrix of the full scatterplot matrix.

```
plmatrix(~Petal.Length+Petal.Width, ~Sepal.Length+Sepal.Width, data=iris,
         smooth.group=Species, pcol=Species)
```



```
## [1] "plmatrix: done"
```

4 Regression diagnostic plots

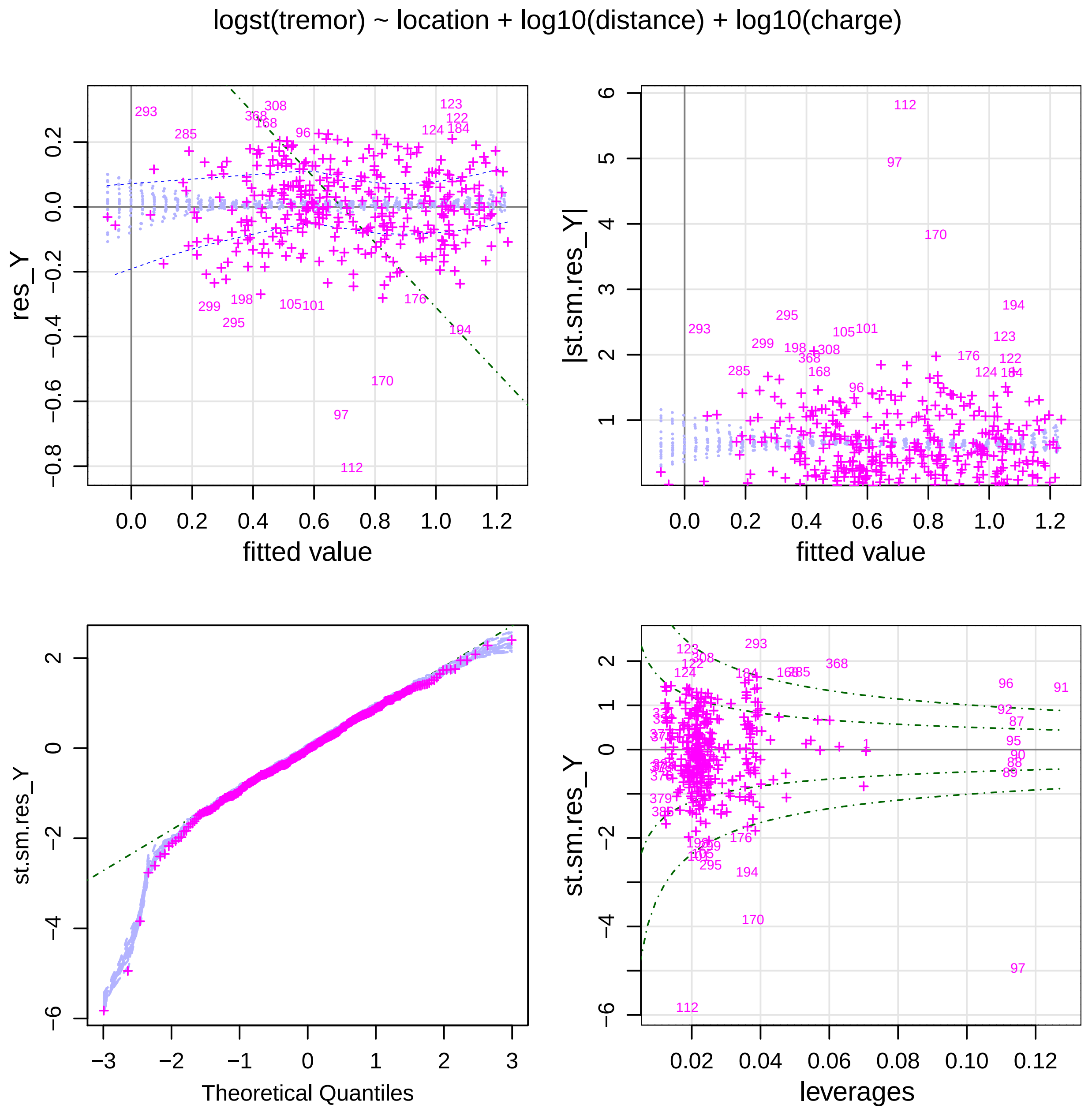
The primary purpose of developing `plgraphics` has been to improve regression diagnostic plots. The features are obtained by using `plot.regr`.

```
data(d.blast)
rr <-
  lm(logst(tremor)~location+log10(distance)+log10(charge), data=d.blast)
```

```
plot.regr(rr, xvar=FALSE)

## Warning in max(abs(as.numeric(lattrj$innerrange))):  no non-missing arguments to max;
returning -Inf

## Warning in par(loldpar):  "mfig" is not a graphical parameter
```



Before we describe the plots in some detail, let us first explain a principle guiding the design of diagnostics. Each diagnostic (plot) should be specific for a well-identified potential deficiency of the model.

Residuals against fit: the Tukey-Anscombe plot. By default, the scatterplot of residuals against fitted values shows the points with the feature of outlier margins and marking of extremes in the residual direction. It adds a smooth line to show deviations from the linearity assumption. Another 19 smooth lines are shown to mimik the variability of this smooth line under the hypothesis that the model is correct. It also adds a reference line indicating the direction of constant observed response values Y. This helps to see whether a transformation of Y could help to avoid any significant curvature.

Absolute residuals against fit. As a second diagram, the plot of absolute residuals against fitted values is shown. Note that the absolute residuals shown here in this plot are not the absolute values of the residuals used in the first plot. They differ in two ways:

- They are standardized to have the same variances. ... weighted
- By default, they are modified because in the following way. Note first that the plot should show any dependence of the scale of the random errors on the model value. If the plot of residuals against fit shows a clear curvature, the residuals do not show only the random errors but also the bias of the regression function, which should be best approximated by the smooth line in that first plot. Therefore, the residuals from the smooth line are used in the plot of absolute residuals against fit. Additionally, they are standardized using the same factor that is commonly used for standardizing the ordinary residuals.

censored: no intervals

QQ-plot only for Gaussian

Residuals against leverage

The argument `xvar=FALSE` in the statement generating the last plot indicates that by default, `plot.regr` shows more diagrams: The plots of residuals against explanatory variables.

4.1 Residuals against input variables

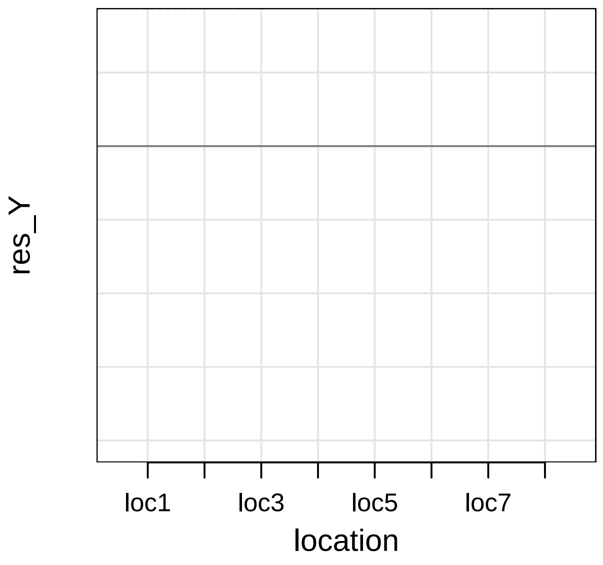
Since the "x" variables in a regression model cannot always be interpreted as explaining the variability of the response Y, we call them "input" variables here.

The plots of residuals against these variables are important regression diagnostics. They are often neglected since the ordinary plot function for models does not show them. `plot.regr` does, unless `xvars=FALSE` is used as it was above. It does so by calling `plresx`, which can also be done directly.

```
plresx(rr)

## Error in axis(axis, at = lat, labels = llab, col = col, ...):  'labels' is supplied
```

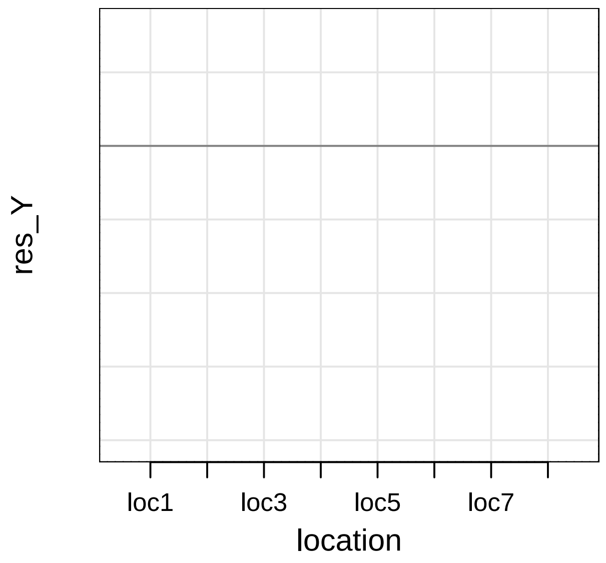
```
and not 'at'
## Warning in par(loldpar):  "mfig" is not a graphical parameter
```



The input variables are often transformed before they are used in the linear predictor, and the main purpose of showing a plot of residuals against them is to possibly find a (more) adequate transformation. For those that have been transformed already, the adequate transformation may be more easily guessed if the untransformed version is used in the plot. The transformed variables can be called for by setting `transformed=TRUE`.

```
plresx(rr, transformed=TRUE)

## Error in axis(axis, at = lat, labels = llab, col = col, ...):  'labels' is supplied
and not 'at'
## Warning in par(loldpar):  "mfig" is not a graphical parameter
```



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

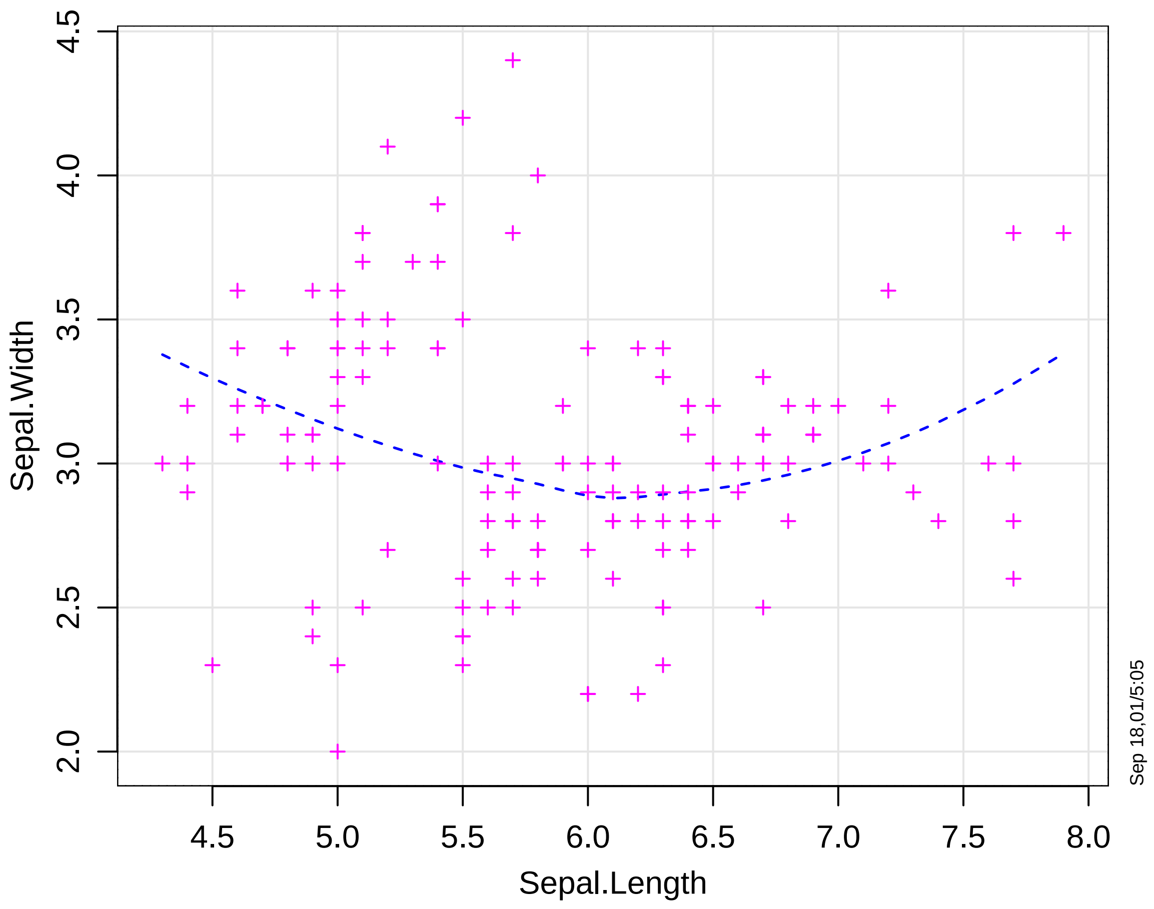
plot.regr(rr, transformed=TRUE, reflinesband=TRUE)

plrex2x

5 Options

The graphical elements, like plotting character, color, line types, etc. to be used in pl graphics are specified in the pl options.

```
t.plopt <- ploptions(basic.col="magenta", basic.pch=3, smoothlines.lty=3)
plyx(Sepal.Width~Sepal.Length, data=iris)
```



```
## restore the old optios
ploptions(list=attr(t.plopt, "old"))
```

They are analogous to the ordinary options, with differences:

- The pl options are stored in a list `.ploptions` in the global environment and are therefore not erased when leaving the R session.
- 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`.
- Both of these lists can be overriden by objects with the same name that appear earlier in the search list than the `plgraphics` package.

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 in the global environment,
4. the list `ploptionsDefault` in the package `plgraphics` or in an environment hiding it.

The components of these lists include

- `colors`, the palette of colors to be used,
- `linewidth`, the linewidths used for the different line types. If the line types are shown with the same `lwd`, they are perceived with different intensity. `linewidth` intends to compensate this effect.
- `cex`, the median character expansion. The default is the function `cexSize` with an argument `n`, defined as `min(1.5/log10(n), 2)`, that is called when the number `n` of observations is available. Alternatively, a fixed scalar can be given.

- a group of components with “group name” `basic`: `basic.pch`, ... `basic.cex` is a factor which will be applied to `cex` above for showing points by a single symbol (`basic.pch`), `basic.cex.plab` is an additional factor applied for the points that are shown by `plab`.
- a group of compnents starting by `group`. They characterize how different groups will be displayed. Thus, `group.pch` should be a vector defining the plotting symbol for the first, second, ... group (when there are groups in the data).
- a group `variables`, defining the elements to be used when different variables should be distinguishable.
- `censored.pch` and `censored.cex` used to show censored observations.
- `mar, oma, mgp, thickintervals`
- `stamp`, logical value determining if a stamp should be added to each plotting page,
- a group `innerrange`, determining if and how an inner plotting range should be used and generated.
- `plext`: percentage by which the range of the data should be extended unless an inner range is active (in which case the extension is determined by `innerrange.ext`), and
  `plextext`: further extension to allow for large symbols near the limits of the plotting range.
- `markextremes` sets the proportion of extreme points that are shown with labels to help identify them. If set to `TRUE`, the proportion depends on the number of observations through `ceiling(sqrt(n)/2)/n`.
- `title.cex` determines the character expansion of the plot title. By default, it adapts to the length of the title. For long titles, it will however never be smaller than `title.cexmin`. If `title.cex` has 2 elements, the second refers to the subtitle.
- a group `gridlines`. If `gridlines` is a list of two vectors, it contains the values where vertical and horizontal thin lines are drawn. If it is `TRUE`, the gridlines correspond to the tickmarks of the two axes. `gridlines.lty, gridlines.lwd, gridlines.col` set the respective properties for the gridlines.
- a group `zeroline`, which is analogous to `gridlines`, but the default is a value of 0 for both axes, and the properties are independent of those for `gridlines`
- a group `refline`. `refline` can be set in high level pl functions as a vector giving intercept and slope of a straight line, or a function that generates this list, such as `lm`.
- a group `smoothline`, analogous to `refline`, usually generated by setting `smooth` to `TRUE`.
- a group `smooth`. If `smooth` is `TRUE`, a smoothing function with default `ploptions("smooth.function")` is called to calculate a smooth line and show it according toe the `smoothline` properties.

- a group `bar` needed to show reference values for levels of factors used as explanatory variables in regression diagnostic plots.
- `jitter`: logical indicating if factors should be shown with jittering. A named vector may be given that defines the jittering for each variable.
  `jitter.factor` is the jittering factor used, see `?jitter`.
- `condprobrange` is used to determine which bars should be shown in the case of censored data.
- `functionxvalues` contains the number of values used to calculate smooth functions and fitting component for diagnostic plots.
- `leveragelim` determines the range used for leverage values when plotting residuals against leverages.

5.1 pl.control, pldata, plargs

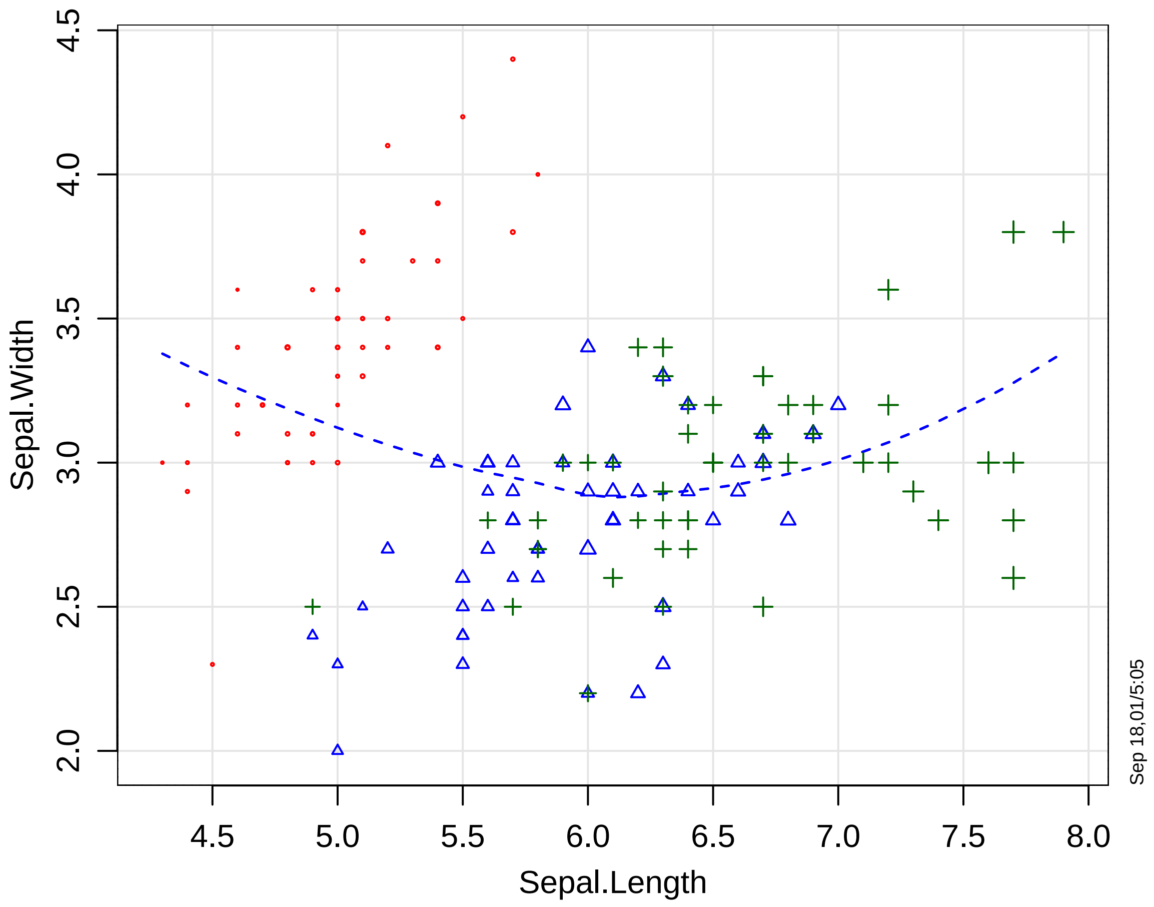
High level pl functions call the function `pl.control` first. It generates the "plotting dataset" `pldata`, which collects data dependent information needed for plotting in an enriched, standardized form. It also takes any futher arguments to be passed on to `ploptions`. The result is stored as `.plargs` in the global environment. This allows for inspection of the plotting data `.plargs$pldata` and the active `ploptions` (`.plargs$ploptions` and thereby helps debugging.

Plotting data There are plotting elements that are useful to represent individual observations or individual variables. Those related to the observations include:

- plotting symbol (character) `pch`,
- plotting label, an extension of `pch` to more than one symbol, often used to identify observations, `plab`,
- plotting size `psize`, scaled by the pl option `cex`,
- color of the symbol, `pcol`

These elements are stored in `pldata` as columns with names `(pch)`, `(plab)`, ... They are generated in `pl.control` when the respective arguments `pch`, `plab`, ... are given to the high level pl function.

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



The elements attached to variables are

- a variable name and a variable label (to be used for labelling the axis on which the variable is shown), typically identical to the name of the variable in the data.frame it comes from,
- the values for which tick marks and labels should be shown in plots, `axisat`,
- an inner and an outer plotting range, `innerrange` and `plrange`,
- the extension `innerrange.ext` used to calculate `plrange` from `innerrange`, if the latter does not cover all points,
- the number of points modified at each end of the inner range, `nmod`,
- coordinates, possibly different from the variable's data values, typically when an inner plotting range or jittering is active,
- line type `lty`, color `col` to be used if multiple y's are shown in a plot.

These elements are stored as attributes of the variables, e.g., `attr(var, "axisat")`. They can be set or generated by the function `genvarattributes` and then modified before calling the high level pl function, such as `plyx`. Those that are needed and have not been stored beforehan will be generated by `pl.control` when calling such a function.

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

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 (excemption, 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`