Package 'precisePlacement'

October 29, 2025

Title Suite of Functions to Help Get Plot Elements Exactly Where You Want Them

Version 0.2.0

Description Provides a selection of tools that make it easier to place elements onto a (base R) plot exactly where you want them. It allows users to identify points and distances on a plot in terms of inches, pixels, margin lines, data units, and proportions of the plotting space, all in a manner more simple than manipulating par().

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, testthat, rmarkdown

VignetteBuilder knitr

BugReports https://github.com/rntq472/precisePlacement/issues

Language en-GB

NeedsCompilation no

Author Jasper Watson [aut, cre]

Maintainer Jasper Watson <jasper.g.watson@gmail.com>

Repository CRAN

Date/Publication 2025-10-29 12:00:02 UTC

Contents

`convertUnits` *Convert Between Different Available Units That Measure Points on a Plot*

Description

Convert Between Different Available Units That Measure Points on a Plot

Usage

```
convertUnits(from, value, to, side = NULL, axis = NULL, region = "plot")
```

Arguments

`from`	Units one wishes to convert from. Allowed values are "line", "data", and "proportion".
`value`	Numeric value(s) of the coordinate(s) one wishes to convert.
`to`	Units one wishes to convert to. Allowed values are "line", "data", and "proportion".
`side`	Integer giving the side of the plot to count lines from.
`axis`	One of "x" or "y", giving the axis a proportion should be calculated from.
`region`	Required when either from or to has the value "proportion". Must be one of "device", "figure", "plot", or "data". Defaults to "plot".

Value

Numeric value(s) of the input coordinates converted into the new units.

Author(s)

Jasper Watson

Examples

```
## Not run:

## Illustrate where the lines fall when using mtext:

plot(1:10)
mtext(1:26, line = -(1:26), side = 1, col = 'blue')
x <- convertUnits('line', 0:(-26), 'data', side = 1)
abline(h = x, col = 'red', lty = 2)

## Show how proportions of a plot can be identified:

plot(seq(as.Date('2018-01-01'), as.Date('2019-01-01'), length.out = 10), 1:10,
pch = 19)

## Identify the "center" of the plot.
abline(h = convertUnits('proportion', 0.5, 'data', axis = 'y'),
col = 'red', lwd = 4)
abline(v = convertUnits('proportion', 0.5, 'data', axis = 'x'),
col = 'blue', lwd = 4)

print(convertUnits('proportion', 0.5, 'data', axis = 'y'))
## as.Date is needed because convertUnits returns a numeric value.
print(as.Date(convertUnits('proportion', 0.5, 'data', axis = 'x'),
origin = '1970-01-01'))

## Change the region we are defining the proportions from.
abline(v = convertUnits('proportion', 0.75, 'data', axis = 'x', region = 'plot'),
col = 'darkgreen', lwd = 4)
abline(v = convertUnits('proportion', 0.75, 'data', axis = 'x', region = 'device'),
col = 'orange', lwd = 4)


## End(Not run)
```

`getBoundaries` *Determine the Boundaries of a Plot in Terms of the Data Units*

Description

Determine the Boundaries of a Plot in Terms of the Data Units

Usage

```
getBoundaries(region, units = "data", sides = 1:4)
```

Arguments

region	The region of the plot to use for defining the boundaries. Must be one of "device", "figure", "plot", or "data".
units	Character string giving the units in which to define the range. Must be either "data" or "lines".
sides	Numeric vector giving the four sides to uses as a reference if the requested units are "lines". Defaults to 1:4.

Value

A numeric vector of length four giving the coordinates of the plotting boundary, in the order of bottom, left, top, right.

Author(s)

Jasper Watson

Examples

```
## Not run:

par(oma = 1:4)
plot(1:10)
print(getBoundaries('data'))
print(getBoundaries('plot'))
print(getBoundaries('figure'))
print(getBoundaries('device'))

print(getBoundaries('data', units = 'lines'))
print(getBoundaries('plot', units = 'lines'))
print(getBoundaries('figure', units = 'lines'))
print(getBoundaries('device', units = 'lines'))


## End(Not run)
```

---

getDataPerInch *Find the Number of Data Points Per Inch in a Plot*

Description

Find the Number of Data Points Per Inch in a Plot

Usage

```
getDataPerInch()
```

Value

Numeric vector of length two giving the number of data points per inch, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getDataPerInch())


## End(Not run)
```

`getDataPerLine` *Find the Number of Data Points Per Margin Line in a Plot*

Description

Find the Number of Data Points Per Margin Line in a Plot

Usage

```
getDataPerLine()
```

Value

Numeric vector of length two giving the number of data points per line, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getDataPerLine())


## End(Not run)
```

getDataPerPixel *Find the Number of Data Points Per Pixel in a Plot*

Description

Find the Number of Data Points Per Pixel in a Plot

Usage

```
getDataPerPixel()
```

Value

Numeric vector of length two giving the number of data points per pixel, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getDataPerPixel())


## End(Not run)
```

getInchesPerDatum *Find the Number of Inches Per Data Point in a Plot*

Description

Find the Number of Inches Per Data Point in a Plot

Usage

```
getInchesPerDatum()
```

Value

Numeric vector of length two giving the number of inches per data point, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getInchesPerDatum())


## End(Not run)
```

`getInchesPerLine` *Find the Number of Inches Per Margin Line in a Plot*

Description

Find the Number of Inches Per Margin Line in a Plot

Usage

```
getInchesPerLine()
```

Value

Numeric vector of length two giving the number of inches per line, for the x axis and y axis, respectively.

Note

The number of lines per inch is the same for both the x and y axes but we return a vector of length two to maintain consistency with all of the other getXperY style functions.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getInchesPerLine())


## End(Not run)
```

getInchesPerPixel *Find the Number of Inches Per Pixel in a Plot*

Description

Find the Number of Inches Per Pixel in a Plot

Usage

```
getInchesPerPixel()
```

Value

Numeric vector of length two giving the number of inches per pixel, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getInchesPerPixel())


## End(Not run)
```

getLinesPerDatum *Find the Number of Margin Lines Per Data Point in a Plot*

Description

Find the Number of Margin Lines Per Data Point in a Plot

Usage

```
getLinesPerDatum()
```

Value

Numeric vector of length two giving the number of lines per data point, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getLinesPerDatum())


## End(Not run)
```

`getLinesPerInch` *Find the Number of Margin Lines Per Inch in a Plot*

Description

Find the Number of Margin Lines Per Inch in a Plot

Usage

```
getLinesPerInch()
```

Value

Numeric vector of length two giving the number of lines per inch, for the x axis and y axis, respectively.

Note

The number of lines per inch is the same for both the x and y axes but we return a vector of length two to maintain consistency with all of the other getXperY style functions.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getLinesPerInch())


## End(Not run)
```

getLinesPerPixel *Find the Number of Margin Lines Per Pixel in a Plot*

Description

Find the Number of Margin Lines Per Pixel in a Plot

Usage

```
getLinesPerPixel()
```

Value

Numeric vector of length two giving the number of lines per pixel, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getLinesPerPixel())


## End(Not run)
```

getPixelsPerDatum *Find the Number of Pixels Per Data Point in a Plot*

Description

Find the Number of Pixels Per Data Point in a Plot

Usage

```
getPixelsPerDatum()
```

Value

Numeric vector of length two giving the number of pixels per data point, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getPixelsPerDatum())


## End(Not run)
```

`getPixelsPerInch` *Find the Number of Pixels Per Inch in a Plot*

Description

Find the Number of Pixels Per Inch in a Plot

Usage

```
getPixelsPerInch()
```

Value

Numeric vector of length two giving the number of pixels per inch, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getPixelsPerInch())


## End(Not run)
```

getPixelsPerLine *Find the Number of Pixels Per Margin Line in a Plot*

Description

Find the Number of Pixels Per Margin Line in a Plot

Usage

```
getPixelsPerLine()
```

Value

Numeric vector of length two giving the number of pixels per line, for the x axis and y axis, respectively.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getPixelsPerLine())


## End(Not run)
```

getRange *Determine the Width and Height of a Plot*

Description

Determine the Width and Height of a Plot

Usage

```
getRange(region, units)
```

Arguments

region	The region of the plot to use for defining the boundaries. Must be one of "device", "figure", "plot", or "data".
units	Character string giving the units in which to define the range. Must be one of "in", "px", "data", or "lines".

Value

Numeric vector of length two giving the range of the plotting region, in the order of x-axis, y-axis.

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
print(getRange('data', 'in'))
print(getRange('plot', 'px'))
print(getRange('figure', 'data'))
print(getRange('device', 'lines'))


## End(Not run)
```

---

highlightDataRegion *Highlight the Data Region of a Plot*

Description

Highlight the Data Region of a Plot

Usage

```
highlightDataRegion(border = "green", col = adjustcolor(border, 0.1))
```

Arguments

border	Parameter to be passed to graphics::rect. Defaults to "green".
col	Parameter to be passed to graphics::rect. Defaults to adjustcolor(border, 0.1).

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
highlightDataRegion()


## End(Not run)
```

`highlightDeviceRegion` *Highlight the Device Region of a Plot*

Description

Highlight the Device Region of a Plot

Usage

```
highlightDeviceRegion(border = "skyblue", col = adjustcolor(border, 0.1))
```

Arguments

`border`	Parameter to be passed to graphics::rect. Defaults to "green".
`col`	Parameter to be passed to graphics::rect. Defaults to adjustcolor(border, 0.1).

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
par(xpd = NA)
highlightDeviceRegion()


## End(Not run)
```

highlightFigureRegion *Highlight the Figure Region of a Plot*

Description

Highlight the Figure Region of a Plot

Usage

```
highlightFigureRegion(border = "orange", col = adjustcolor(border, 0.1))
```

Arguments

border	Parameter to be passed to graphics::rect. Defaults to "green".
col	Parameter to be passed to graphics::rect. Defaults to adjustcolor(border, 0.1).

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
par(xpd = NA)
highlightFigureRegion()


## End(Not run)
```

highlightPlotRegion *Highlight the Plotting Region of a Plot*

Description

Highlight the Plotting Region of a Plot

Usage

```
highlightPlotRegion(border = "red", col = adjustcolor(border, 0.1))
```

Arguments

border	Parameter to be passed to graphics::rect. Defaults to "green".
col	Parameter to be passed to graphics::rect. Defaults to adjustcolor(border, 0.1).

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
highlightPlotRegion()


## End(Not run)
```

`lineLocations` *Wrapper Function for convertUnits Focused on Identifying the Locations of the Margin Lines of a Plot*

Description

Wrapper Function for convertUnits Focused on Identifying the Locations of the Margin Lines of a Plot

Usage

```
lineLocations(side, line)
```

Arguments

`side`	Integer giving the side of the plot to count lines from.
`line`	Numeric vector giving margin lines one wishes to find the data coordinates of.

Note

No attempt is made to limit the returned values to the device region.

Author(s)

Jasper Watson

Examples

```
## Not run:

## Illustrate where the lines fall when using mtext:
plot(1:10)
mtext(1:26, line = -(1:26), side = 1, col = 'blue')
x <- lineLocations(1, 0:(-26))
```

```
abline(h = x, col = 'red', lty = 2)

## End(Not run)
```

`omiForSubFigure` *Generate Values for par("omi") That Will Place a New Plot in a Sub-Region of an Existing One*

Description

Generate Values for par("omi") That Will Place a New Plot in a Sub-Region of an Existing One

Usage

```
omiForSubFigure(
  bottom,
  left,
  top,
  right,
  units = "proportion",
  region = "device"
)
```

Arguments

`bottom`	Boundary value for the bottom edge.
`left`	Boundary value for the left edge.
`top`	Boundary value for the top edge.
`right`	Boundary value for the right edge.
`units`	The units in which the boundary parameters are defined. Must be one one of "proportion" or "data". Defaults to "proportion".
`region`	The region of the plot to use for defining the boundaries. Must be one of "device", "figure", "plot", or "data". Only necessary when using units of "proportion".

Details

The choice of accepting proportions instead of data units by default is to more easily handle empty devices, otherwise there is a risk of getting confused by the fact that par("usr") defaults to c(0, 1, 0, 1).

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10, pch = 19, col = 'black')
oldPar = par()
par(omi = omiForSubFigure(0.6, 0.25, 0.8, 0.45, region = 'device'))
par(mar = c(0,0,0,0))
plot(1:10, pch = 19, col = 'red')
par(oldPar)
par(omi = omiForSubFigure(2, 6, 5, 10, units = 'data'))
par(mar = c(0,0,0,0))
plot(1:10, pch = 19, col = 'blue')
par(oldPar)

## Illustrates how the proportions line up:

plot(1:10, pch = 19)
par(xpd = NA)
oldPar = par()

## Show where the propotions are as a reference:
abline(v = convertUnits('proportion', seq(0, 1, by = 0.1), 'data',
region = 'device', axis = 'x'), lty = 2, col = 'red')
abline(h = convertUnits('proportion', seq(0, 1, by = 0.1), 'data',
region = 'device', axis = 'y'), lty = 2, col = 'red')

## Create a new sub-plot.
par(omi = omiForSubFigure(0.2, 0.2, 0.8, 0.8, region = 'device'))

plot(1:10, pch = 19, col = 'red')
highlightFigureRegion()

par(oldPar)


## End(Not run)
```

`precisePlacement` *Suite of Functions To Get Plot Elements Exactly Where You Want Them*

Description

This package provides a selection of tools that make it easier to place elements onto a (base R) plot exactly where you want them. It allows users to identify points and distances on a plot in terms of inches, pixels, margin lines, data units, and proportions of the plotting space, all in a manner more simple than manipulating par(). It helps users identify where exactly the margin lines of a plot fall and to create sub-figures.

Author(s)

Jasper Watson

See Also

Useful links:

- Report bugs at https://github.com/rntq472/precisePlacement/issues

---

`repelAxisLabels` *Distribute Axis Label Locations to Avoid Overlap.*

Description

Distribute Axis Label Locations to Avoid Overlap.

Usage

```
repelAxisLabels(side, at, labels, ..., spacing = "", reduceCex = TRUE)
```

Arguments

`side`	Integer giving the side of the plot, as in graphics::axis.
`at`	The points at which the axis labels and tickmarks would ordinarily be plotted.
`labels`	Character vector of axis labels.
`...`	Other arguments passed to strheight and strwidth, for example font.
`spacing`	Character string giving any spacing desired between the adjacent labels. Defaults to an empty string.
`reduceCex`	Logical indicating whether axis labels should be shrunk in size if they don't all fit. Defaults to TRUE.

Value

List containing the fields "originalAt", "at", and "cex.axis". "originalAt" is the input parameter "at" that has been sorted, "at" is a vector of new axis locations, and "cex.axis" is the cex value used to fit the labels without overlap (only necessary if reduceCex is TRUE).

Note

This function is experimental and just uses simple heuristics instead of any physics logic. It worked the one time I wanted it but if it doesn't generalise well I may not invest the time to try improving it.

Also note that this function does not work when a plot is resized; you will have to re-run it after resizing.

Author(s)

Jasper Watson

showMarginLines *Add Lines to a Plot to Indicate Where the Margins Are*

Description

Add Lines to a Plot to Indicate Where the Margins Are

Usage

```
showMarginLines(sides = 1:4, lty = 2, ...)
```

Arguments

sides	Numeric vector giving the sides of the plot to annotate. Defaults to 1:4.
lty	Argument to be passed to graphics::abline. Defaults to 2.
...	Other arguments to be passed to graphics::abline

Author(s)

Jasper Watson

Examples

```
## Not run:

plot(1:10)
par(xpd = NA)
showMarginLines()


## End(Not run)
```

showOuterMarginLines *Add Lines to a Plot to Indicate Where the Outer Margins Are*

Description

Add Lines to a Plot to Indicate Where the Outer Margins Are

Usage

```
showOuterMarginLines(sides = 1:4, lty = 3, col = "purple", ...)
```

Arguments

sides	Numeric vector giving the sides of the plot to annotate. Defaults to 1:4.
lty	Argument to be passed to graphics::abline. Defaults to 3.
col	Argument to be passed to graphics::abline. Defaults to "purple".
...	Other arguments to be passed to graphics::abline

Author(s)

Jasper Watson

Examples

```
## Not run:

par(oma = 1:4, mfrow = 2:1)
plot(1:10)
plot(1:10)
par(xpd = NA)
showOuterMarginLines()


## End(Not run)
```

Index