Package 'plotcli'

May 9, 2026

Title Command Line Interface Plotting

Version 0.2.0

Date 2025-11-27

Description The 'plotcli' package provides terminal-based plotting in R. It supports colored scatter plots, line plots, bar plots, boxplots, histograms, density plots, and more. The 'ggplotcli()' function is a universal converter that renders any 'ggplot2' plot in the terminal using Unicode Braille characters or ASCII. Features include support for 15+ geom types, faceting (facet_wrap/facet_grid), automatic theme detection, legends, optimized color mapping, and multiple canvas types.

License LGPL-3

URL https://github.com/cheuerde/plotcli

Encoding UTF-8

VignetteBuilder knitr

Depends R6, ggplot2

Imports crayon, stringr, rlang, grDevices, stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown

NeedsCompilation yes

RoxygenNote 7.3.2

Config/testthat/edition 3

Author Claas Heuer [aut, cre]

Maintainer Claas Heuer <claasheuer@gmail.com>

Repository CRAN

Date/Publication 2025-11-27 10:40:02 UTC

Contents

+.plotcli *Overload the "+" operator for plotcli objects*

Description

This function overloads the "+" operator to merge two plotcli objects.

Usage

```
## S3 method for class 'plotcli'
plot1 + plot2
```

Arguments

plot1	A plotcli object to be merged.
plot2	A plotcli object to be merged.

Value

A new plotcli object containing the combined data from both objects.

AsciiCanvas *ASCII Canvas Class*

Description

ASCII Canvas Class

ASCII Canvas Class

Details

Simple canvas using basic ASCII characters. Resolution: 1x1 (one pixel per character)

Super class

plotcli::Canvas -> AsciiCanvas

Public fields

point_char Character used for points

Methods

Public methods:

- `AsciiCanvas$new()`
- `AsciiCanvas$set_pixel()`
- `AsciiCanvas$clone()`

Method `new()`**:** Initialize ASCII canvas

Usage:

```
AsciiCanvas$new(width, height, point_char = "*")
```

Arguments:

`width` Character width

`height` Character height

`point_char` Character to use for points (default "*")

Method `set_pixel()`**:** Set a pixel

Usage:

```
AsciiCanvas$set_pixel(x, y, color = NULL)
```

Arguments:

`x` X coordinate (1-based)

`y` Y coordinate (1-based, 1 = top)

`color` Optional color name

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
AsciiCanvas$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`BlockCanvas` *Block Canvas Class*

Description

Block Canvas Class

Block Canvas Class

Details

Canvas using Unicode block elements for 2x vertical resolution. Uses half-block characters: upper half (U+2580), lower half (U+2584), full block (U+2588).

Resolution: 1x2 (1 horizontal, 2 vertical pixels per character)

Super class

plotcli::Canvas -> BlockCanvas

Public fields

upper_block Upper half block character

lower_block Lower half block character

full_block Full block character

pixel_state Matrix tracking which half-pixels are set (0=none, 1=upper, 2=lower, 3=both)

Methods

Public methods:

- BlockCanvas$new()
- BlockCanvas$set_pixel()
- BlockCanvas$clear()
- BlockCanvas$clone()

Method new(): Initialize Block canvas

Usage:

```
BlockCanvas$new(width, height)
```

Arguments:

width Character width

height Character height

Method set_pixel(): Set a pixel in Block space

Usage:

```
BlockCanvas$set_pixel(x, y, color = NULL)
```

Arguments:

x X coordinate (1-based, in pixel space: 1 to width)

y Y coordinate (1-based, in pixel space: 1 to height*2, 1 = top)

color Optional color name

Method clear(): Clear the canvas

Usage:

```
BlockCanvas$clear()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
BlockCanvas$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`BrailleCanvas` *Braille Canvas Class*

Description

Braille Canvas Class

Braille Canvas Class

Details

High-resolution canvas using Unicode Braille patterns. Resolution: 2x4 (2 horizontal, 4 vertical dots per character = 8x resolution)

Braille dot layout (dot numbers and bit values):

```
       Col 0  Col 1     Bit values
Row 0:   1      4        0x01  0x08
Row 1:   2      5        0x02  0x10
Row 2:   3      6        0x04  0x20
Row 3:   7      8        0x40  0x80
```

Super class

`plotcli::Canvas` -> `BrailleCanvas`

Public fields

`braille_base` Unicode code point for empty Braille character

Methods

Public methods:

- `BrailleCanvas$new()`
- `BrailleCanvas$get_dot_bit()`
- `BrailleCanvas$set_pixel()`
- `BrailleCanvas$clone()`

Method `new()`**:** Initialize Braille canvas

Usage:

```
BrailleCanvas$new(width, height)
```

Arguments:

`width` Character width

`height` Character height

Method `get_dot_bit()`**:** Get the bit value for a dot position

Usage:

```
BrailleCanvas$get_dot_bit(dot_row, dot_col)
```

Arguments:

`dot_row` Row within Braille cell (0-3)

`dot_col` Column within Braille cell (0-1)

Returns: Bit value

Method `set_pixel()`**:** Set a pixel in Braille space

Usage:

```
BrailleCanvas$set_pixel(x, y, color = NULL)
```

Arguments:

`x` X coordinate (1-based, in pixel space: 1 to width*2)

`y` Y coordinate (1-based, in pixel space: 1 to height*4, 1 = top)

`color` Optional color name

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
BrailleCanvas$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`braille_dot_bit` *Get Braille dot bit value*

Description

Returns the bit value for a specific dot position in a Braille cell. Braille cells have a 2x4 dot matrix with the following bit values:

Usage

```
braille_dot_bit(dot_row, dot_col)
```

Arguments

`dot_row`	Row within the Braille cell (0-3, top to bottom)
`dot_col`	Column within the Braille cell (0-1, left to right)

Details

```
       Col 0  Col 1
Row 0:  0x01   0x08
Row 1:  0x02   0x10
Row 2:  0x04   0x20
Row 3:  0x40   0x80
```

Value

The bit value for that dot position

`braille_set_dot` *Set a dot in a Braille character*

Description

Set a dot in a Braille character

Usage

```
braille_set_dot(current_char, dot_row, dot_col)
```

Arguments

`current_char`	The current character (can be space or existing Braille)
`dot_row`	Row within the Braille cell (0-3)
`dot_col`	Column within the Braille cell (0-1)

Value

The updated Braille character

`bresenham` *Bresenham's line algorithm*

Description

This function generates a list of points that form a line between two given points using Bresenham's line algorithm.

Usage

```
bresenham(x0, y0, x1, y1)
```

Arguments

`x0`	The x-coordinate of the starting point.
`y0`	The y-coordinate of the starting point.
`x1`	The x-coordinate of the ending point.
`y1`	The y-coordinate of the ending point.

Value

A list of points that form a line between the two given points.

Examples

```
bresenham(0, 0, 5, 5)
bresenham(0, 0, -5, -5)
```

Canvas *Canvas Classes for Terminal Plotting*

Description

Abstract canvas system that provides different rendering backends: - AsciiCanvas: Basic ASCII characters (*, -, |, +) - BrailleCanvas: Unicode Braille patterns (2x4 dots per cell = 8x resolution) - BlockCanvas: Unicode block elements (half-blocks for 2x vertical resolution)

Details

Abstract base class for all canvas types. Provides the interface that all canvas implementations must follow.

Public fields

`width` Character width of the canvas

`height` Character height of the canvas

`pixel_width` Pixel width (may be higher than char width for Braille/Block)

`pixel_height` Pixel height (may be higher than char height for Braille/Block)

`matrix` Character matrix for rendering

`color_matrix` Parallel matrix tracking colors per cell

`x_mult` Horizontal multiplier (pixels per character)

`y_mult` Vertical multiplier (pixels per character)

Methods

Public methods:

- `Canvas$new()`
- `Canvas$set_pixel()`
- `Canvas$draw_line()`
- `Canvas$draw_polyline()`
- `Canvas$draw_points()`
- `Canvas$fill_rect()`
- `Canvas$fill_bar()`
- `Canvas$draw_text()`

- Canvas$apply_colors()
- Canvas$render()
- Canvas$print()
- Canvas$clear()
- Canvas$draw_rect()
- Canvas$fill_area()
- Canvas$draw_segment()
- Canvas$draw_hline()
- Canvas$draw_vline()
- Canvas$draw_circle()
- Canvas$fill_circle()
- Canvas$draw_polygon()
- Canvas$clone()

Method `new()`**:** Initialize the canvas

Usage:

```
Canvas$new(width, height)
```

Arguments:

`width` Character width

`height` Character height

Method `set_pixel()`**:** Set a pixel at the given coordinates

Usage:

```
Canvas$set_pixel(x, y, color = NULL)
```

Arguments:

`x` X coordinate (1-based, in pixel space)

`y` Y coordinate (1-based, in pixel space, 1 = top)

`color` Optional color name

Method `draw_line()`**:** Draw a line between two points

Usage:

```
Canvas$draw_line(x0, y0, x1, y1, color = NULL)
```

Arguments:

`x0` Start X coordinate

`y0` Start Y coordinate

`x1` End X coordinate

`y1` End Y coordinate

`color` Optional color name

Method `draw_polyline()`**:** Draw multiple connected line segments

Usage:

```
Canvas$draw_polyline(xs, ys, color = NULL)
```

Arguments:

`xs` Vector of X coordinates

`ys` Vector of Y coordinates

`color` Optional color name

Method `draw_points()`**:** Draw points (scatter)

Usage:

```
Canvas$draw_points(xs, ys, color = NULL)
```

Arguments:

`xs` Vector of X coordinates

`ys` Vector of Y coordinates

`color` Optional color name

Method `fill_rect()`**:** Fill a rectangle

Usage:

```
Canvas$fill_rect(x0, y0, x1, y1, color = NULL)
```

Arguments:

`x0` Left X coordinate

`y0` Top Y coordinate

`x1` Right X coordinate

`y1` Bottom Y coordinate

`color` Optional color name

Method `fill_bar()`**:** Fill a vertical bar from bottom up to a height

Usage:

```
Canvas$fill_bar(x, height, bar_width = 2, color = NULL)
```

Arguments:

`x` X coordinate (center of bar in pixel space)

`height` Height in pixels from bottom

`bar_width` Width of bar in pixels (default 2)

`color` Optional color name

Method `draw_text()`**:** Place text at a position

Usage:

```
Canvas$draw_text(x, y, text, color = NULL)
```

Arguments:

`x` X coordinate (character position)

`y` Y coordinate (character position)

`text` Text string to place

`color` Optional color name

Method `apply_colors()`**:** Apply colors to the matrix

Usage:

```
Canvas$apply_colors()
```

Returns: Matrix with ANSI color codes applied

Method `render()`**:** Get the rendered matrix (with colors)

Usage:

```
Canvas$render()
```

Returns: Character matrix

Method `print()`**:** Print the canvas to console

Usage:

```
Canvas$print()
```

Method `clear()`**:** Clear the canvas

Usage:

```
Canvas$clear()
```

Method `draw_rect()`**:** Draw a rectangle outline

Usage:

```
Canvas$draw_rect(x0, y0, x1, y1, color = NULL)
```

Arguments:

`x0` Left X coordinate (pixel space)

`y0` Top Y coordinate (pixel space)

`x1` Right X coordinate (pixel space)

`y1` Bottom Y coordinate (pixel space)

`color` Optional color name

Method `fill_area()`**:** Fill an area between a polyline and the bottom

Usage:

```
Canvas$fill_area(xs, ys, color = NULL)
```

Arguments:

`xs` Vector of X coordinates

`ys` Vector of Y coordinates

`color` Optional color name

Method `draw_segment()`**:** Draw a segment (line with optional arrowhead)

Usage:

```
Canvas$draw_segment(x0, y0, x1, y1, arrow_end = FALSE, color = NULL)
```

Arguments:

`x0` Start X coordinate

`y0` Start Y coordinate

`x1` End X coordinate

`y1` End Y coordinate

`arrow_end` Add arrowhead at end (default FALSE)

`color` Optional color name

Method `draw_hline()`**:** Draw a horizontal line

Usage:

```
Canvas$draw_hline(y, x0 = 1, x1 = NULL, color = NULL)
```

Arguments:

`y` Y coordinate
`x0` Start X (default 1)
`x1` End X (default pixel_width)
`color` Optional color name

Method `draw_vline()`**:** Draw a vertical line

Usage:

```
Canvas$draw_vline(x, y0 = 1, y1 = NULL, color = NULL)
```

Arguments:

`x` X coordinate
`y0` Start Y (default 1)
`y1` End Y (default pixel_height)
`color` Optional color name

Method `draw_circle()`**:** Draw a circle outline

Usage:

```
Canvas$draw_circle(cx, cy, r, color = NULL)
```

Arguments:

`cx` Center X coordinate
`cy` Center Y coordinate
`r` Radius in pixels
`color` Optional color name

Method `fill_circle()`**:** Fill a circle

Usage:

```
Canvas$fill_circle(cx, cy, r, color = NULL)
```

Arguments:

`cx` Center X coordinate
`cy` Center Y coordinate
`r` Radius in pixels
`color` Optional color name

Method `draw_polygon()`**:** Draw a polygon outline

Usage:

```
Canvas$draw_polygon(xs, ys, closed = TRUE, color = NULL)
```

Arguments:

`xs` Vector of X coordinates
`ys` Vector of Y coordinates
`closed` Whether to close the polygon (default TRUE)
`color` Optional color name

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Canvas$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`cat_plot_matrix` *Print plot matrix*

Description

This function prints a plot matrix to the console.

Usage

```
cat_plot_matrix(plot_matrix)
```

Arguments

`plot_matrix`	The plot matrix to be printed.

Examples

```
cat_plot_matrix(matrix(c("a", "b", "c", "d"), nrow = 2, ncol = 2))
```

`cbind.plotcli` *Generic function for combining plotcli objects horizontally*

Description

Generic function for combining plotcli objects horizontally

Usage

```
## S3 method for class 'plotcli'
cbind(..., deparse.level = 1)
```

Arguments

`...`	plotcli objects to be combined.
`deparse.level`	The deparsing level for the arguments.

Value

A combined plot matrix.

`cbind_plots` *Combine plot matrices horizontally*

Description

This function combines multiple plot matrices horizontally, centering them vertically.

Usage

```
cbind_plots(...)
```

Arguments

`...`	A list of plot matrices to be combined.

Value

A combined plot matrix.

`color_to_term` *Convert ggplot2 color to terminal color name*

Description

If init_color_mapping() was called, uses the pre-computed mapping. Otherwise falls back to simple hue-based matching.

Usage

```
color_to_term(color)
```

Arguments

`color`	A color value (hex, name, or R color)

Value

A terminal color name (blue, red, green, etc.) or NULL

create_canvas *Create a canvas of the specified type*

Description

Create a canvas of the specified type

Usage

```
create_canvas(width, height, type = "braille")
```

Arguments

width	Character width
height	Character height
type	Canvas type: "ascii", "braille", or "block"

Value

A Canvas object

create_scales *Create Scale Object from ggplot_build data*

Description

Create Scale Object from ggplot_build data

Usage

```
create_scales(built, plot_width, plot_height, has_border = FALSE)
```

Arguments

built	Result from ggplot_build()
plot_width	Canvas pixel width
plot_height	Canvas pixel height
has_border	Whether a border will be drawn (adds padding)

Value

List with x_scale and y_scale functions

format_four_chars *Format number to four characters*

Description

This function formats a number to a string of exactly four characters.

Usage

```
format_four_chars(num)
```

Arguments

num	The number to be formatted.

Value

A string representation of the number with exactly four characters.

Examples

```
format_four_chars(123)
format_four_chars(-12.34)
```

GeomRegistry *Geom Registry and Dispatch System*

Description

This module provides a registry for geom rendering functions and a dispatch system for converting ggplot2 geoms to terminal plots.

get_data_subset *Get data subset for a specific geom*

Description

This function returns a subset of the data for a specific geom.

Usage

```
get_data_subset(geom_name, data, aes, p_build)
```

Arguments

geom_name	The name of the geom for which the data subset is needed.
data	The data to be subsetted.
aes	The aesthetic mappings for the geom.
p_build	The ggplot build object.

Value

A list containing the data subset for the specified geom.

---

get_geom_handler *Get a Geom Handler*

Description

Retrieve the registered handler for a geom, or NULL if not found.

Usage

```
get_geom_handler(geom_name)
```

Arguments

geom_name	Name of the geom

Value

The handler function or NULL

---

get_term_colors *Get terminal colors*

Description

This function returns a vector of terminal colors.

Usage

```
get_term_colors(n = NULL)
```

Arguments

n	The number of colors to return.

Value

A vector of terminal colors.

Examples

```
get_term_colors(5)
get_term_colors(10)
```

---

ggplotcli *ggplotcli - Render ggplot2 objects in the terminal*

Description

Convert any ggplot2 plot to a terminal-based visualization using Unicode Braille characters or ASCII. Supports 15+ geom types, faceting, themes, and color aesthetics.

Usage

```
ggplotcli(
  p,
  width = 60,
  height = 20,
  canvas_type = "braille",
  border = "auto",
  grid = "none",
  show_axes = TRUE,
  axis_labels = TRUE,
  legend = "auto",
  title_align = "center",
  subtitle = TRUE,
  caption = TRUE,
  title = NULL,
  boxplot_style = "ascii"
)
```

Arguments

p	A ggplot2 object to render
width	Character width of the plot (default: 60)
height	Character height of the plot (default: 20)
canvas_type	Type of canvas: "braille" (high-res), "block" (medium), or "ascii" (basic). Default: "braille"
border	Draw border around plot. "auto" uses ggplot theme, or TRUE/FALSE (default: "auto")
grid	Grid lines: "none", "major", "minor", "both", or "auto" (default: "none")

show_axes	Whether to show axis values (default: TRUE)
axis_labels	Whether to show axis labels from ggplot (default: TRUE)
legend	Legend display: "auto", "right", "bottom", "none" (default: "auto")
title_align	Title alignment: "center" or "left" (default: "center")
subtitle	Whether to show subtitle (default: TRUE)
caption	Whether to show caption (default: TRUE)
title	Optional title override (NULL uses ggplot title)
boxplot_style	Style for boxplots: "ascii" uses box-drawing characters (default), "braille" uses Braille dots like other geoms

Value

Invisibly returns the canvas object

Examples

```
library(ggplot2)

# Basic scatter plot
p <- ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
ggplotcli(p)

# With styling
ggplotcli(p, border = TRUE, grid = "major")

# Faceted plot
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  facet_wrap(~cyl)
ggplotcli(p, width = 70, height = 16)

# Multiple geoms
p <- ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(aes(y = after_stat(density)), bins = 10) +
  geom_density(color = "red")
ggplotcli(p)
```

---

init_color_mapping *Initialize color mapping for a set of ggplot colors*

Description

This function takes all unique colors from a ggplot and assigns terminal colors to minimize repetition while respecting hue similarity.

Usage

```
init_color_mapping(ggplot_colors)
```

Arguments

ggplot_colors	Vector of unique colors from ggplot

---

is_braille *Check if a character is a Braille character*

Description

This function checks if a given character is a Braille character.

Usage

```
is_braille(char)
```

Arguments

char	The character to be checked.

Value

A boolean value indicating whether the character is a Braille character or not.

Examples

```
is_braille("A")
```

---

is_geom_registered *Check if a Geom is Registered*

Description

Check if a Geom is Registered

Usage

```
is_geom_registered(geom_name)
```

Arguments

geom_name	Name of the geom

Value

Logical

list_registered_geoms *List Registered Geoms*

Description

List Registered Geoms

Usage

```
list_registered_geoms()
```

Value

Character vector of registered geom names

make_colored *Make colored text*

Description

This function applies a specified color to a given text string.

Usage

```
make_colored(x, color = NULL)
```

Arguments

x	The text string to be colored.
color	The color to be applied to the text. If NULL, the color codes will be removed.

Value

A colored text string or a text string with color codes removed.

Examples

```
make_colored("Hello, world!", "blue")
make_colored("Hello, world!", NULL)
```

make_unique_names *Make unique names*

Description

This function takes a vector of names and ensures that each name is unique by appending a number if necessary.

Usage

```
make_unique_names(names)
```

Arguments

names	A character vector of names.

Value

A character vector of unique names.

Examples

```
make_unique_names(c("apple", "apple", "banana", "apple"))
```

normalize_data *Normalize data*

Description

This function normalizes the given data to a specified plot range.

Usage

```
normalize_data(data, data_min, data_max, plot_range)
```

Arguments

data	The data to be normalized.
data_min	The minimum value of the data.
data_max	The maximum value of the data.
plot_range	The range to normalize the data to.

Value

The normalized data.

Examples

```
normalize_data(c(1, 2, 3, 4, 5), 1, 5, 10)
normalize_data(c(10, 20, 30, 40, 50), 10, 50, 100)
```

---

pclib *Short version of plotcli_bar*

Description

Short version of plotcli_bar function.

Usage

```
pclib(
  y,
  x = NULL,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "barplot",
  ...
)
```

Arguments

y	A numeric vector of values
x	A vector of categories
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "y")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "barplot")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- 1:5
y <- c(10, 15, 8, 12, 6)
pclib(x, y)
```

pclibx *Short version of plotcli_box*

Description

Short version of plotcli_box function.

Usage

```
pclibx(
  y,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "boxplot",
  ...
)
```

Arguments

y	A list of numeric vectors of values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "y")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "boxplot")
...	Additional arguments passed to the plotcli$new() function
x	A vector of categories

Examples

```
y <- rnorm(50, mean = 0)
pclib(y)
```

pclid *Short version of plotcli_density*

Description

Short version of plotcli_density function.

Usage

```
pclid(
  x,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "Density",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "density",
  ...
)
```

Arguments

x	A numeric vector of values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "Density")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "density")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- rnorm(100)
pclid(x)
```

pclih *Short version of plotcli_histogram*

Description

Short version of plotcli_histogram function.

Usage

```
pclih(
  x,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "Frequency",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  bin_width = NULL,
  ylim = NULL,
  name = "histogram",
  ...
)
```

Arguments

x	A numeric vector of values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "Frequency")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
bin_width	Width of the bins (default: NULL)
ylim	y limits (default: NULL)
name	Name of the plot element (default: "histogram")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- rnorm(100)
pclih(x)
```

pclil *Short version of plotcli_line*

Description

Short version of plotcli_line function.

Usage

```
pclil(
  y,
  x = NULL,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "line",
  ...
)
```

Arguments

y	A numeric vector of y values
x	A numeric vector of x values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "y")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "line")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- 1:10
y <- x^2
pclil(x, y)
```

pclis *Short version of plotcli_scatter*

Description

Short version of plotcli_scatter function.

Usage

```
pclis(
  y,
  x = NULL,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "scatter",
  ...
)
```

Arguments

y	A numeric vector of y values
x	A numeric vector of x values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "y")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "scatter")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- rnorm(100)
y <- rnorm(100)
pclis(x, y)
```

pixel_to_braille *Convert pixel coordinates to Braille cell and dot position*

Description

Convert pixel coordinates to Braille cell and dot position

Usage

```
pixel_to_braille(px, py, canvas_rows, canvas_cols)
```

Arguments

px	Pixel x coordinate (1-based)
py	Pixel y coordinate (1-based, from top)
canvas_rows	Number of character rows in canvas
canvas_cols	Number of character columns in canvas

Value

List with cell_row, cell_col, dot_row, dot_col

plotcli *plotcli R6 Class*

Description

plotcli R6 Class

plotcli R6 Class

Details

This class provides a set of methods to create and customize command-line plots using R6. It supports various plot types, such as scatter, line, bar, and box plots, and allows customization of plot elements, such as title, axis labels, ticks, and legend.

Usage

```
plotcli <- plotcli$new()
plotcli$add_data(data)
plotcli$print_plot()
```

Methods

initialize() Initializes the PlotCLI object with parameters.

initialize_plot_matrix() Initializes the plot matrix with the plot canvas.

print() Default print method for PlotCLI object.

add_row() Adds a single row to the plot matrix.

add_col() Adds a single column to the plot matrix.

add_borders() Adds borders around the plot canvas.

add_row_col_index() Adds row and column index to the plot matrix.

add_title() Adds a title to the plot matrix.

add_y_ticks() Adds y-axis tick labels to the plot matrix.

add_y_label() Adds a y-axis label to the plot matrix.

add_x_ticks() Adds x-axis tick labels to the plot matrix.

add_x_label() Adds an x-axis label to the plot matrix.

add_legend() Adds a legend to the plot matrix.

add_data() Adds data to the object.

get_min_max() Gets minimum and maximum values for x and y.

remove_out_of_range_data() Removes out of range data points if xlim and ylim were given.

draw_scatter_plot() Draws a scatter plot on the plot canvas.

draw_line_plot() Draws a line plot on the plot canvas.

draw_barplot() Draws a bar plot on the plot canvas.

draw_barplot_braille() Draws a bar plot with braille characters on the plot canvas.

draw_boxplot() Draws a box plot on the plot canvas.

print_plot() Assembles all plot elements and prints the plot to the console.

Public fields

`plot_width` The width of the plot

`plot_height` The height of the plot

`plot_canvas` The canvas for drawing the plot

`plot_matrix` The matrix containing the entire plot, including borders, labels, and title

`data` A list containing the data sets to be plotted

`title` The title of the plot

`x_label` The label for the x-axis

`y_label` The label for the y-axis

`ylim` The limits for the y-axis

`xlim` The limits for the x-axis

`x_min` The minimum value of the x-axis

`x_max` The maximum value of the x-axis

`y_min` The minimum value of the y-axis

`y_max` The maximum value of the y-axis

`plot_matrix_canvas_row_start` The starting row of the plot canvas within the plot matrix

`plot_matrix_canvas_col_start` The starting column of the plot canvas within the plot matrix

`is_boxplot` A logical value indicating if the plot is a boxplot

`draw_legend` A logical value indicating if the legend should be drawn

Methods

Public methods:

- `plotcli$new()`
- `plotcli$initialize_plot_matrix()`
- `plotcli$print()`
- `plotcli$add_row()`
- `plotcli$add_col()`
- `plotcli$add_borders()`
- `plotcli$add_row_col_index()`
- `plotcli$add_title()`
- `plotcli$add_y_ticks()`
- `plotcli$add_y_label()`
- `plotcli$add_x_ticks()`
- `plotcli$add_x_label()`
- `plotcli$add_legend()`
- `plotcli$add_data()`
- `plotcli$get_min_max()`
- `plotcli$remove_out_of_range_data()`
- `plotcli$draw_scatter_plot()`
- `plotcli$draw_line_plot()`
- `plotcli$draw_barplot()`
- `plotcli$draw_barplot_braille()`
- `plotcli$draw_boxplot()`
- `plotcli$draw_colors()`
- `plotcli$draw_plot()`
- `plotcli$make_plot_matrix()`
- `plotcli$export_plot_matrix()`
- `plotcli$print_plot()`
- `plotcli$merge()`
- `plotcli$clone()`

Method `new()`**:** Initialize object

Usage:

```
plotcli$new(
  plot_width = 60,
  plot_height = 20,
  x_label = "x",
  y_label = "y",
  ylim = NULL,
  xlim = NULL,
  title = NULL,
  is_boxplot = FALSE,
  draw_legend = TRUE
)
```

Arguments:

`plot_width` integer, width of the plot canvas

`plot_height` integer, height of the plot canvas

`x_label` character, label for the x-axis

`y_label` character, label for the y-axis

`ylim` numeric vector, limits for the y-axis

`xlim` numeric vector, limits for the x-axis

`title` character, title of the plot

`is_boxplot` logical, whether the plot is a boxplot

`draw_legend` logical, whether to draw the legend This function initializes the plot matrix based on the plot canvas.

Method `initialize_plot_matrix()`**:** Initialize the plot matrix

Usage:

```
plotcli$initialize_plot_matrix()
```

Arguments:

`plot_width` The width of the plot

`plot_height` The height of the plot

Returns: A plot matrix object

Method `print()`**:** Default print method for plotcli object

Usage:

```
plotcli$print(...)
```

Arguments:

`...` Additional arguments passed to the print method

Returns: The plotcli object, invisibly

Method `add_row()`**:** Add a single row to the plot matrix

Usage:

```
plotcli$add_row(bottom = FALSE)
```

Arguments:

`bottom` logical, if TRUE, add row to the bottom of the matrix, otherwise add to the top (default: FALSE)

Method `add_col()`: Add a single column to the plot matrix

Usage:

```
plotcli$add_col()
```

Method `add_borders()`: Add borders to the plot matrix

Usage:

```
plotcli$add_borders()
```

Method `add_row_col_index()`: Add row and column index to the plot matrix Add title to the plot matrix

Usage:

```
plotcli$add_row_col_index()
```

Method `add_title()`:

Usage:

```
plotcli$add_title()
```

Arguments:

`title` character, title of the plot Add y-ticks label to the plot matrix

Method `add_y_ticks()`:

Usage:

```
plotcli$add_y_ticks(n_ticks = 5)
```

Arguments:

`n_ticks` numeric, number of ticks Add y-axis label to the plot matrix

Method `add_y_label()`: Add a y-axis label to the plot matrix

Usage:

```
plotcli$add_y_label(y_label = self$y_label)
```

Arguments:

`y_label` character, the y-axis label to be added Add x-ticks label to the plot matrix

Method `add_x_ticks()`:

Usage:

```
plotcli$add_x_ticks(n_ticks = 5)
```

Arguments:

`n_ticks` numeric, number of ticks Add x-axis label to the plot matrix

Method `add_x_label()`: Add x-axis label to the plot matrix

Usage:

```
plotcli$add_x_label(x_label = self$x_label)
```

Arguments:

`x_label` x label Add legend to the plot matrix

Method `add_legend()`**:** Add legend to the plot matrix Add data to the object.

Usage:

```
plotcli$add_legend()
```

Method `add_data()`**:**

Usage:

```
plotcli$add_data(data)
```

Arguments:

`data` list, list with elements: x, y, type, color, braille, name Get minimum and maximum values for x and y

Method `get_min_max()`**:** Calculate the minimum and maximum values for x and y Function to remove out of range data points if xlim and ylim were given

Usage:

```
plotcli$get_min_max()
```

Method `remove_out_of_range_data()`**:** Remove data points that are outside the specified xlim and ylim Draw a scatter plot to the plot canvas.

Usage:

```
plotcli$remove_out_of_range_data()
```

Method `draw_scatter_plot()`**:** Draw a scatter plot of the specified data set on the plot canvas.

Usage:

```
plotcli$draw_scatter_plot(set_idx)
```

Arguments:

`set_idx` numeric, the data element index to be drawn Draw a line plot to the plot canvas.

Method `draw_line_plot()`**:**

Usage:

```
plotcli$draw_line_plot(set_idx)
```

Arguments:

`set_idx` numeric, the data element index to be drawn Draw a barplot to the plot canvas.

Method `draw_barplot()`**:**

Usage:

```
plotcli$draw_barplot(set_idx)
```

Arguments:

`set_idx` numeric, the data element index to be drawn Draw a barplot to the plot canvas with braille characters.

Method `draw_barplot_braille()`**:**

Usage:

```
plotcli$draw_barplot_braille(set_idx)
```

Arguments:

`set_idx` numeric, the data element index to be drawn Draw a boxplot to the plot canvas.

Method `draw_boxplot()`**:**

Usage:

```
plotcli$draw_boxplot(set_idx)
```

Arguments:

`set_idx` numeric, the data element index to be drawn Draw colors to the canvas

Method `draw_colors()`**:** In the draw_ functions we have been keeping track of the locations of the colored matrix elements. These are now being colored. Draw the different plots types from all data elements to the canvas

Usage:

```
plotcli$draw_colors()
```

Method `draw_plot()`**:** This function iterates through all data elements and calls the appropriate draw_ function based on the plot type (scatter, line, boxplot, or barplot). Make plot matrix: assembles all plot elements (canvas + borders + title + axes + legend)

Usage:

```
plotcli$draw_plot()
```

Method `make_plot_matrix()`**:** This function assembles all plot elements (canvas + borders + title + axes + legend) and creates the final plot matrix. Export plot matrix

Usage:

```
plotcli$make_plot_matrix()
```

Method `export_plot_matrix()`**:** This function exports the plot matrix.

Usage:

```
plotcli$export_plot_matrix()
```

Returns: The plot matrix. Main plotting function: assembles all plot elements (canvas + borders + title + axes + legend) and prints the plot by 'cat'ing the plot matrix to the console.

Method `print_plot()`**:** This function assembles all plot elements (canvas + borders + title + axes + legend) and prints the final plot by 'cat'ing the plot matrix to the console. Merge two plotcli objects

This method combines the data from two plotcli objects into a single plotcli object. It takes the maximum of the plot_width and plot_height, combines the titles, and sets the xlim and ylim to the minimum and maximum values of both objects.

Usage:

```
plotcli$print_plot()
```

Method `merge()`**:**

Usage:

```
plotcli$merge(other)
```

Arguments:

`other` A plotcli object to be merged with the current object.

Returns: A new plotcli object containing the combined data from both objects.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
plotcli$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Create a new plotcli object
plotcli <- plotcli$new()

# Add data for a scatter plot
plotcli$add_data(list(x = 1:10, y = rnorm(10), type = "scatter", color = "red"))

# Print the plot
plotcli$print_plot()
```

`plotcli_bar` *Bar plot using plotcli*

Description

Create a bar plot using plotcli. Short alias: `pclb`.

Usage

```
plotcli_bar(
  y,
  x = NULL,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "barplot",
  ...
)
```

Arguments

y	A numeric vector of values
x	A vector of categories
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "y")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "barplot")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- 1:5
y <- c(10, 15, 8, 12, 6)
plotcli_bar(x, y)
```

---

plotcli_box *Box plot using plotcli*

Description

Create a box plot using plotcli. Short alias: pclbx.

Usage

```
plotcli_box(
  y,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "boxplot",
  ...
)
```

Arguments

y	A list of numeric vectors of values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "y")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "boxplot")
...	Additional arguments passed to the plotcli$new() function
x	A vector of categories

Examples

```
y <- rnorm(50, mean = 0)
plotcli_box(y)
```

---

plotcli_density *Density plot using plotcli*

Description

Create a density plot using plotcli. Short alias: pcld.

Usage

```
plotcli_density(
  x,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "Density",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "density",
  ...
)
```

Arguments

x	A numeric vector of values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "Density")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "density")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- rnorm(100)
plotcli_density(x)
```

---

plotcli_histogram *Histogram plot using plotcli*

Description

Create a histogram plot using plotcli. Short alias: `pclih`.

Usage

```
plotcli_histogram(
  x,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "Frequency",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  bin_width = NULL,
  ylim = NULL,
  name = "histogram",
  ...
)
```

Arguments

x	A numeric vector of values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "Frequency")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
bin_width	Width of the bins (default: NULL)
ylim	y limits (default: NULL)
name	Name of the plot element (default: "histogram")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- rnorm(100)
plotcli_histogram(x)
```

---

plotcli_line *Line plot using plotcli*

Description

Create a line plot using plotcli. Short alias: `pcli`.

Usage

```
plotcli_line(
  y,
  x = NULL,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "line",
  ...
)
```

Arguments

y	A numeric vector of y values
x	A numeric vector of x values
plot_width	Width of the plot (default: 80)
plot_height	Height of the plot (default: 40)
x_label	Label for the x-axis (default: "x")
y_label	Label for the y-axis (default: "y")
color	Color of the plot elements (default: NULL)
braille	Use Braille characters for the plot (default: TRUE)
name	Name of the plot element (default: "line")
...	Additional arguments passed to the plotcli$new() function

Examples

```
x <- 1:10
y <- x^2
plotcli_line(x, y)
```

---

plotcli_options *Set global options for plotcli*

Description

Set global options for plotcli

Usage

```
plotcli_options(plot_width = 60, plot_height = 20, braille = FALSE)
```

Arguments

plot_width	Default plot width (default: 60)
plot_height	Default plot height (default: 20)
braille	Default braille setting (default: FALSE)

`plotcli_scatter` *Scatter plot using plotcli*

Description

Create a scatter plot using plotcli. Short alias: `pclis`.

Usage

```
plotcli_scatter(
  y,
  x = NULL,
  plot_width = getOption("plotcli.plot_width", 80),
  plot_height = getOption("plotcli.plot_height", 40),
  x_label = "x",
  y_label = "y",
  color = NULL,
  braille = getOption("plotcli.braille", TRUE),
  name = "scatter",
  ...
)
```

Arguments

`y`	A numeric vector of y values
`x`	A numeric vector of x values
`plot_width`	Width of the plot (default: 80)
`plot_height`	Height of the plot (default: 40)
`x_label`	Label for the x-axis (default: "x")
`y_label`	Label for the y-axis (default: "y")
`color`	Color of the plot elements (default: NULL)
`braille`	Use Braille characters for the plot (default: TRUE)
`name`	Name of the plot element (default: "scatter")
`...`	Additional arguments passed to the plotcli$new() function

Examples

```
x <- rnorm(100)
y <- rnorm(100)
plotcli_scatter(x, y)
```

rbind.plotcli *Generic function for combining plotcli objects vertically*

Description

Generic function for combining plotcli objects vertically

Usage

```
## S3 method for class 'plotcli'
rbind(..., deparse.level = 1)
```

Arguments

`...`	plotcli objects to be combined.
`deparse.level`	The deparsing level for the arguments.

Value

A combined plot matrix.

rbind_plots *Combine plot matrices vertically*

Description

This function combines multiple plot matrices vertically, centering them horizontally.

Usage

```
rbind_plots(...)
```

Arguments

`...`	A list of plot matrices to be combined.

Value

A combined plot matrix.

register_geom *Register a Geom Handler*

Description

Register a function that can render a specific ggplot2 geom to a canvas.

Usage

```
register_geom(geom_name, handler)
```

Arguments

geom_name	Name of the geom (e.g., "GeomPoint", "GeomLine")
handler	Function that takes (data, canvas, scales, params) and draws to canvas

Examples

```
register_geom("GeomPoint", function(data, canvas, scales, params) {
  # Draw points on canvas
})
```

remove_color_codes *Remove color codes from a string*

Description

This function removes ANSI color codes from a given text string.

Usage

```
remove_color_codes(s)
```

Arguments

s	The text string containing ANSI color codes.

Value

A text string with ANSI color codes removed.

Examples

```
colored_text <- make_colored("Hello, world!", "blue")
remove_color_codes(colored_text)
```

Index