Package ‘cols4all’

March 12, 2024

License GPL-3

Title Colors for all

Type Package

LazyLoad yes

Description Color palettes for all people, including those with color vision deficiency. Popular color palette series have been organized by type and have been scored on several properties such as color-blind-friendliness and fairness (i.e. do colors stand out equally?). Own palettes can also be loaded and analysed. Besides the common palette types (categorical, sequential, and diverging) it also includes bivariate color palettes. Furthermore, a color for missing values is assigned to each palette.

Version 0.7-1

Date 2024-03-12

Encoding UTF-8

Depends R (>= 3.5.0),

Imports methods, grDevices, stats, abind, png, stringdist, colorspace (>= 2.1), spacesXYZ

Suggests colorblindcheck, kableExtra, knitr, shiny, shinyjs, ggplot2, scales, rmarkdown, bibtex

URL `https://mtennekes.github.io/cols4all/`,
`https://github.com/mtennekes/cols4all`

BugReports `https://github.com/mtennekes/cols4all/issues`

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation no

Author Martijn Tennekes [aut, cre],
Marco Puts [ctb],
Achim Zeileis [ctb],
Jakub Nowosad [ctb],
Robin Lovelace [ctb],
Helgasoft [ctb],

Matthew Petroff [ctb],
Olivier Roy [ctb]

Maintainer Martijn Tennekes <mtennekes@gmail.com>

Repository CRAN

Date/Publication 2024-03-12 11:20:08 UTC

R topics documented:

`cols4all-package` *cols4all overview*

Description

cols4all stands for: color palettes for all people, including those with color vision deficiency. Popular color palette series, such as ColorBrewer, have been organized by type and have been scored on several properties such as color-blind-friendliness and fairness (i.e. do colors stand out equally?). Own palettes can also be loaded and analysed. Besides the common palette types (categorical, sequential, and diverging) it also includes bivariate color palettes. ggplot2 scales are included.

Details

This page provides a brief overview of all package functions.

Main functions

`c4a_gui`	Dashboard for analyzing the palettes
`c4a`	Get the colors from a palette (`c4a_na` for the associated color for missing values)
`c4a_plot`	Plot a color palette

Palette names and properties

`c4a_palettes`	Get available palette names
`c4a_series`	Get available series names
`c4a_overview`	Get an overview of palettes per series x type
`c4a_citation`	Show how to cites palettes (with bibtex code)
`c4a_info`	Get information from a palette, such as type and maximum number of colors)
`.P`	Environment via which palette names can be browsed with auto-completion (using `$`)

Importing and exporting palettes

`c4a_data`	Build color palette data
`c4a_load`	Load color palette data
`c4a_sysdata_import`	Import system data
`c4a_sysdata_export`	Export system data

Author(s)

Maintainer: Martijn Tennekes <`mtennekes@gmail.com`>

Other contributors:

- Marco Puts <`mputs@acm.org`> [contributor]
- Achim Zeileis <`Achim.Zeileis@R-project.org`> [contributor]
- Jakub Nowosad <`nowosad.jakub@gmail.com`> [contributor]
- Robin Lovelace <`rob00x@gmail.com`> [contributor]
- Helgasoft <`contact@helgasoft.com`> [contributor]
- Matthew Petroff <`matthew@mpetroff.net`> [contributor]
- Olivier Roy [contributor]

See Also

Useful links:

- `https://mtennekes.github.io/cols4all/`
- `https://github.com/mtennekes/cols4all`
- Report bugs at `https://github.com/mtennekes/cols4all/issues`

`c4a` *Get a cols4all color palette*

Description

Get a cols4all color palette: `c4a` returns the colors of the specified palette, and `c4a_na` returns the color for missing value that is associated with the specified palette. Run `c4a_gui` to see all available palettes, which are also listed with `c4a_palettes`.

Usage

```
c4a(
  palette = NULL,
  n = NA,
  m = NA,
  type = c("cat", "seq", "div", "bivs", "bivc", "bivd", "bivg"),
  reverse = FALSE,
  order = NULL,
  range = NA,
  format = c("hex", "RGB", "HCL"),
  nm_invalid = c("error", "repeat", "interpolate"),
  verbose = TRUE
)

c4a_na(palette = NULL, type = c("cat", "seq", "div"), verbose = TRUE)
```

Arguments

`palette`	name of the palette. See `c4a_palettes` for options. If omitted, the default palette is provided by `c4a_default_palette`. The palette name can be prefixed with a `"-"` symbol, which will reverse the palette (this can also be done with the `reverse` argument).
`n`	number of colors. If omitted then: for type `"cat"` the maximum number of colors is returned, for types `"seq"` and `"div"`, 9 colors.
`m`	number of rows in case type is `"bivs"`, `"bivc"`, `"bivd"` or `"bivg"` (which stand for respectively sequential, categorical, diverging and desaturated (g for 'grayscale')).
`type`	type of color palette, in case `palette` is not specified: one of `"cat"` (categorical/qualitative palette), `"seq"` (sequential palette), `"div"` (diverging palette), and `"bivs"`/`"bivc"`/`"bivd"`/`"bivg"` (bivariate: respectively seq-seq seq-cat, seq-div, and seq-desaturated).
`reverse`	should the palette be reversed?
`order`	order of colors. Only applicable for `"cat"` palettes

`range`	a vector of two numbers between 0 and 1 that determine the range that is used for sequential and diverging palettes. The first number determines where the palette begins, and the second number where it ends. For sequential `"seq"` palettes, 0 means the leftmost (normally lightest) color, and 1 the rightmost (often darkest) color. For diverging `"seq"` palettes, 0 means the middle color, and 1 both extremes. If only one number is provided, this number is interpreted as the endpoint (with 0 taken as the start).
`format`	format of the colors. One of: `"hex"` character vector of hex color values, `"RGB"` 3 column matrix of RGB values, or `"HCL"` 3-column matrix of HCL values
`nm_invalid`	what should be done in case `n` or `m` is larger than the maximum number of colors or smaller than the minimum number? Options are `"error"` (an error is returned), `"repeat"`, the palette is repeated, `"interpolate"` colors are interpolated. For categorical `"cat"` palettes only.
`verbose`	should messages be printed?

Value

A vector of colors (`c4a`) and a color (`c4a_na`)

Examples

```
c4a_palettes("div")

c4a(type = "cat")

(pal = c4a("tol.sunset", n = 7, range = c(0, .6)))

c4a_plot(pal)

c4a("set2")

c4a("hcl.set2")

c4a("hcl.set2", n = 8)

# reversed palette
c4a("hcl.set2", reverse = TRUE, n = 8)

# handy shortcut
c4a("-hcl.set2", n = 8)

# the color for missing values is white:
c4a_na("hcl.set2")
```

`c4a_citation` *Show how to cite palettes*

Description

Show how to cite palettes

Usage

```
c4a_citation(name, verbose = TRUE)
```

Arguments

`name`	name of a palette or series
`verbose`	should text be printed (if FALSE only a `utils::bibentry` object is returned)

Value

`utils::bibentry` object

Examples

```
c4a_citation("hcl")

c4a_citation("poly.glasbey")
```

`c4a_data` *Build and load palette data*

Description

Build palette data. Both `c4a_data` and `c4a_data_as_is` build data palette. The difference is that the former may restructure the palette colors (see details) whereas the latter takes the palette colors as they are. Data can subsequently be loaded into cols4all via `c4a_load`. The `c4a_data` function can also be used to read `c4a_info` objects, which contain data for a single palette.

Usage

```
c4a_data(
  x,
  xNA = NA,
  types = "cat",
  series = "x",
  nmin = NA,
  nmax = NA,
  ndef = NA,
  mmin = NA,
  mmax = NA,
  mdef = NA,
  format.palette.name = TRUE,
  remove.blacks = TRUE,
  take.gray.for.NA = TRUE,
  remove.other.grays = FALSE,
  light.to.dark = TRUE,
  remove.names = TRUE,
  biv.method = "byrow",
  space = "rgb",
  range_matrix_args = list(NULL),
  bib = NA,
  description = NA
)

c4a_load(data, overwrite = FALSE)

c4a_data_as_is(
  ...,
  format.palette.name = FALSE,
  remove.blacks = FALSE,
  take.gray.for.NA = FALSE,
  remove.other.grays = FALSE,
  light.to.dark = FALSE,
  remove.names = FALSE
)
```

Arguments

`x`	either a named list of color palettes or a `c4a_info` object. For the first case: see details for indexing. The second case will bypass the other arguments.
`xNA`	colors for missing values. Vector of the same length as x (or length 1). For `NA` values, the color for missing values is automatically determined (preferable a light grayscale color, but if it is indistinguishable by color blind people, a light color with a low chroma value is selected)
`types`	character vector of the same length as x (or length 1), which determines the type of palette: `"cat"`, `"seq"`, `"div"`, `"bivs"`, `"bivc"`, `"bivd"`, or `"bivg"`. See details.
`series`	a character vector of the same length as x (or length 1), which determines the series.
`nmin, nmax, ndef`	minimum / maximum / default number of colors for the palette. By default: `nmin` = 1, for `"cat"` `nmax` and `ndef` the number of supplied colors. For the other types, `nmax` is `Inf`. `ndef` is 7 for `"seq"`, 9. For diverging palettes, these numbers refer to the number of columns. (See `mmin`, `mmax`, `mdef` for the rows)
`mmin, mmax, mdef`	minimum / maximum / default number of rows for bivariate palettes.
`format.palette.name`	should palette names be formatted to lowercase/underscore format?
`remove.blacks, take.gray.for.NA, remove.other.grays`	These arguments determine the processing of grayscale colors for categorical `"cat"` palettes: if `remove.blacks` and there are (near) blacks, these are removed

	first. Next, if `take.gray.for.NA`, `xNA` is `NA`, and a palette contains at least one grayscale color (which can also be white), this is used as color for missing values. In case there are more than one grayscale color, the lightest is taken. `remove.other.grays` determines what happens with the other grays.
`light.to.dark`	should sequential `"seq"` palettes be automatically ordered from light to dark?
`remove.names`	should individual color names be removed?
`biv.method`	method to a create bivariate palette. Options are `"byrow"` means that the colors are wrapped row-wise to a color matrix where the number of rows and columns is automatically determined, `"byrowX"` the same but with X (integer between 2 and 9) columns, `"bycol"` and `"bycolX` similar but wrapped column-wise. `"div2seqseq"` and `"div2catseq` means that colors are extracted from a divering palette. The former translates colors into a matrix with the neutral color in the diagonal, while the latter places the neutral color in the middle column. `"seq2uncseq"`
`space`	color space in which interpolated colors are determined. Options: `"rgb"` (RGB) and `"Lab"` (CIE Lab).
`range_matrix_args`	list of lists, one for each palette. Each such list specifies the range of sequential and diverging palettes, in case they are not indexed. See details.
`bib`	bibtex reference in the form of a `utils::bibentry` object.
`description`	description of the series. If `series` contains multiple series (rather than one value), please specify a vector of the same length as `series`. See `c4a_series` for the descriptions of the currently loaded series.
`data`	cols4all data created with `c4a_data`
`overwrite`	in case the palettes already exist (i.e. the full names), should the old names be overwritten?
`...`	passed on to `c4a_data`

Details

In cols4all, palettes are organized by series and by type. The **series** or 'family' specifies where the palettes belong to. For instance `"brewer"` stands for the color palettes from ColorBrewer. Run `c4a_series` to get an overview of loaded series. The **type** specifies what kind of palette it is; see `c4a_types` for a description of the implemented ones.

This function structures the palette data, such that it is consistent with the other palette data. This includes:

- Palette names are made consistent. We use the convention `"my_series.my_palette"`, so all lower case, a period to separate the series name from the palette name, and underscores to separate words.
- (Only for `c4a_data`, bypassed for `c4a_data_as_is`) Categorical palettes: black is removed from categorical palettes, and a grayscale color is assigned to be used for missing values (other grayscale colors are removed). Sequential palettes are sorted from light to dark.

Indexing: for a categorical `"cat"` palette, an optional `"index"` attribute determines which colors to use for which lengths: if the palette consists of k colors, index should be a list of k, where the i-th element is an integer vector of length i with values 1,2,...,k. See `c4a_info("rainbow")` and for an example.

Range: sequential and diverging palettes are usually defined for 9+ colors. The optional `"range_matrix"` attribute determines that range is used for less colors. It is a n x 2 matrix where row i defines the applied range of a palette of length i. For sequential palettes a range `c(0,1)` means that the palette is generated (via a color ramp) between the two outermost colors. For diverging palettes, a range `c(x, y)` means that both sides of the palette are generated (via a color ramp) from `x`, which is the distance to the center color, to `y` which represents both outermost colors.

The range is automatically set for sequential and diverging palettes that have no `"index"` or `"range_matrix"` attribute via the parameter `range_matrix_args`, which is a list per palette. The arguments for a sequential palette are: `nmin` the minimum number of colors for which the range is reduced, `nmax`, the number of colors for which the range is set to `c(0,1)`, `slope_min` and `slope_max` determine the slopes of range reduction from a palette of length `nmax` to `nmin`, and `space` sets the color space for which the color ramp is applied (`"rgb"` or `"Lab"`). The arguments for a diverging palette are the same, but only one `slope` is used (namely for the outermost colors).

It may take some time to process, especially large categorical palettes, because of calculations of the color blind checks.

Value

`c4a_data` object, which is a list of four items: `data`, `s`, `citation`, and `description`

Examples

```
# palettes extracted Pink Floyd albums
pf = list(piper = c("#391C1C", "#C6C6AA", "#713939", "#C6391C",
    "#C6E3C6", "#AA7155", "#AA8E71", "#C68E71"),
  saucerful = c("#000000", "#1C1C1C", "#393939", "#FFFFFF",
    "#555555", "#8E8E71", "#E3C6AA", "#715539"),
  atom = c("#C6E3FF", "#397139", "#557139", "#E3E3C6",
    "#1C1C1C", "#1C551C", "#AAAA8E", "#8EC6E3"),
  meddle = c("#715539", "#553939", "#8E7155", "#71AAAA",
    "#8E8E71", "#1CAAE3", "#55C6E3", "#AA7155"),
  obscured = c("#000000", "#1C1C1C", "#393939", "#717155",
    "#8E8E71", "#715539", "#C6AA8E", "#E3C6AA"),
  moon = c("#000000", "#FF0000", "#FF9224", "#FFFF00",
    "#71C600", "#00C6FF", "#8E398E", "#FFFFFF"),
  wish = c("#FFFFFF", "#AAC6E3", "#8E8E8E", "#717155",
    "#555539", "#8E8E71", "#555555", "#8E7155"),
  animals = c("#391C39", "#393955", "#E3C671", "#718E8E",
    "#AAAA8E", "#C67139", "#AA5539", "#E3AA39"),
  wall = c("#FFFFFF", "#E3E3E3", "#C6C6C6", "#AAAAC6",
    "#1C1C1C", "#000000", "#8E8E8E", "#E3C6E3"),
  cut = c("#000000", "#E30000", "#AA0000", "#391C55",
    "#FFE3E3", "#1C1C00", "#FFAA55", "#8E8E55"),
  lapse = c("#000000", "#8E8EC6", "#8E8E71", "#7171AA",
    "#39391C", "#717171", "#AAAAAA", "#E3E3E3"),
  division = c("#000000", "#FFFFC6", "#00398E", "#AA8E55",
    "#39558E", "#C6AA71", "#39391C", "#555571"),
  more = c("#0055AA", "#FFAA1C", "#1C71AA", "#003971",
    "#E38E55", "#E3AAAA", "#718EAA", "#71718E"),
  umma = c("#AA8E71", "#555539", "#39391C", "#1C1C1C",
    "#E3E3C6", "#715539", "#391C1C", "#8E7155"),
  relics = c("#3955AA", "#1C3971", "#5571C6", "#715555",
    "#8E7155", "#E3AA71", "#8E8EAA", "#E3FFFF"),
  river = c("#393939", "#555555", "#39558E", "#C6C6C6",
    "#718EAA", "#1C1C1C", "#717171", "#E3C68E"))

if (requireNamespace("colorblindcheck", quietly = TRUE)) {
pfdata = c4a_data_as_is(pf, series = "pinkfloyd",
description = "Palettes extracted from Pink Floyd album covers")
c4a_load(pfdata)

c4a_series()
c4a_overview()

if (requireNamespace("shiny") &&
requireNamespace("shinyjs") &&
requireNamespace("kableExtra") &&
requireNamespace("colorblindcheck") &&
interactive()) {
c4a_gui(series = "pinkfloyd", n = 8)
}
}
```

`c4a_gui` *Graphical user interface to analyse palettes*

Description

Graphical user interface to analyse palettes. `c4a_table` shows a table that can be opened in the browser. `c4a_gui` is a graphical user interface (shiny app) around this table.

Usage

```
c4a_gui(type = "cat", n = NA, series = "all")

c4a_table(
  type = c("cat", "seq", "div", "bivs", "bivc", "bivd", "bivg"),
  n = NULL,
  m = NULL,
  cvd.sim = c("none", "deutan", "protan", "tritan"),
  sort = "name",
  text.format = "hex",
  text.col = "same",
  series = "all",
  range = NA,
  include.na = FALSE,
  show.scores = FALSE,
  columns = NA,
  verbose = TRUE
)
```

Arguments

`type`	type of palette. Run `c4a_types` to see the implemented types and their description. For `c4a_gui` it only determines which type is shown initially.
`n, m`	`n` is the number of displayed colors. For bivariate palettes `"biv"`, `n` and `m` are the number of columns and rows respectively. If omitted: for `"cat"` the full palette is displayed, for `"seq"` and `"div"`, 9 colors, and for `"bivs"`/`"bivc"`/`"bivd"`/`"bivg"` 4 columns and rows. For `c4a_gui` it only determines which number of colors initially.
`series`	Series of palettes to show. See `c4a_series` for options. By default, `"all"`, which means all series. For `c4a_gui` it only determines which series are shown initially.
`cvd.sim`	color vision deficiency simulation: one of `"none"`, `"deutan"`, `"protan"`, `"tritan"`
`sort`	column name to sort the data. The available column names depend on the arguments `type` and `show.scores`. They are listed in the warning message. Use a `"-"` prefix to reverse the order.
`text.format`	The format of the text of the colors. One of `"hex"`, `"RGB"` or `"HCL"`.
`text.col`	The text color of the colors. By default `"same"`, which means that they are the same as the colors themselves (so invisible, but available for selection). `"auto"` means automatic: black for light colors and white for dark colors.
`range`	vector of two numbers that determine the range that is used for sequential and diverging palettes. Both numbers should be between 0 and 1. The first number determines where the palette begins, and the second number where it ends. For sequential palettes, 0 means the leftmost (normally lightest) color, and 1 the rightmost (often darkest) color. For diverging palettes, 0 means the middle color, and 1 both extremes. If only one number is provided, this number is interpreted as the endpoint (with 0 taken as the start). By default, it is set automatically, based on `n`.
`include.na`	should color for missing values be shown? `FALSE` by default
`show.scores`	should scores of the quality indicators be printed? See details for a description of those indicators.
`columns`	number of columns. By default equal to `n` or, if not specified, 12. Cannot be higher than the palette lengths.
`verbose`	should messages and warnings be printed?

Value

An HMTL table (`kableExtra` object)

See Also

References of the palettes: `cols4all-package`.

Examples

```
if (requireNamespace("shiny") &&
  requireNamespace("shinyjs") &&
  requireNamespace("kableExtra") &&
  requireNamespace("colorblindcheck") &&
  interactive()) {

c4a_gui()

# categorical palettes with maximum number of colors
c4a_table(type = "cat")

# sort sequential palettes by hue
c4a_table(type = "seq", n = 7, sort = "H")

# sort sequential palettes by hue type (how many hues are used)
c4a_table(type = "seq", n = 5, sort = "hueType")
}
```

`c4a_info` *Get information from a cols4all palette*

Description

Get information from a cols4all palette

Usage

```
c4a_info(palette, no.match = c("message", "error", "null"), verbose = TRUE)
```

Arguments

`palette`	name of the palette
`no.match`	what happens is no match is found? Options: `"message"`: a message is thrown with suggestions, `"error"`: an error is thrown, `"null"`: `NULL` is returned
`verbose`	should messages be printed?

Value

list with the following items: name, series, fullname, type, palette (colors), na (color), nmax, and reverse. The latter is `TRUE` when there is a `"-"` prefix before the palette name.

`c4a_modify` *Edit cols4all palettes (in development)*

Description

Edit cols4all palettes. c4a_duplicate duplicates an existing cols4all palette, and c4a_modify is used to change the colors. Use c4a_data to craete palettes from scratch.

Usage

```
c4a_modify(palette, x = NULL, xNA = NULL)

c4a_duplicate(palette, name = NA)
```

Arguments

`palette`	name of the palette
`x`	vector of the new colors. It should either the same length, or a named vector, where the names correspond to the index numbers. E.g. `c("3" = "#AABBCC")` will replace the third color with the color `"#AABBCC"`.
`xNA`	the new color for missing values.
`name`	name of new palette

See Also

`c4a_data()`

Examples

```
c4a_duplicate("brewer.set2", "set2_mod")
c4a_modify("set2_mod", c("4" = "#EA8AB8"))
```

`c4a_options` *Set cols4all options*

Description

Get or set global options for c4a. Works similar as the base function `options`

Usage

```
c4a_options(...)
```

Arguments

`...`	Use character values to retrieve options. To set options, either use named arguments (where the names refer to the options), a list that consists of those options.

Details

Option	Description
defaults	Default palettes per type
CBF_th	Parameters that label a palette as color blind friendly
CBU_th	Parameters that label a palette as color blind unfriendly
CrangeFair	Maximum chroma range for which a palette is considered harmonic
CrangeUnfair	Minimum chroma range for which a palette is considered disharmonic
LrangeFair	Maximum luminance range for which a palette is considered harmonic
LrangeUnfair	Minimum luminance range for which a palette is considered disharmonic
Cintense	Chroma of colors that are considered intense
Cpastel	Chroma of colors that are considered 'pastel'
HwidthDivRainbow	A diverging palette is labeled as 'rainbow hue' if HwidthL or HwidthR are at least `HwidthDivRainbow`
HwidthDivSingle	A diverging palette is labeled as 'single hue' if HwidthL and HwidthR are at most `HwidthDivSingle`
HwidthSeqRainbow	A sequential palette is labeled as 'rainbow hue' if Hwidth is at least `HwidthSeqRainbow`
HwidthSeqSingle	A sequential palette is labeled as 'single hue' if Hwidth is at most `HwidthSeqSingle`

Value

A list of options

`c4a_palettes` *Get available palette names and series*

Description

`c4a_palettes` lists all available cols4all color palettes. Palettes are organized by series. The available series are listed with `c4a_series`. Palettes are also organized per functional type, where we currently support: categorical `"cat"`, sequential `"seq"`, and diverging `"div""` palette types. The function `c4a_types` lists all available types. The function `c4a_overview` gives an overview table of the number of palette per series and type. In an IDE with auto-completion (such as RStudio) it is possible to browse through the palette names with `.P` (using `$` like in lists).

Usage

```
c4a_palettes(
  type = c("all", "cat", "seq", "div"),
  series = NULL,
  full.names = TRUE
)

c4a_series(type = c("all", "cat", "seq", "div"), as.data.frame = TRUE)

c4a_types(series = NULL, as.data.frame = TRUE)

c4a_overview()

.P
```

Arguments

`type`	type of color palette: one of `"all"` (all palettes), `"cat"` (categorical/qualitative palettes), `"seq"` (sequential palettes) and `"div"` (diverging palettes).
`series`	series to list the palettes from. Run `c4a_series` to see the options.
`full.names`	should full names, i.e. with the prefix "series."? By default `TRUE`.
`as.data.frame`	should `c4a_series` and `c4a_types` return the result as a data.frame, with description included as a column?

Format

An object of class `environment` of length 17.

Value

names of the loaded color palettes

See Also

References of the palettes: `cols4all-package`.

Examples

```
c4a_series()

c4a_types()

c4a_overview()

c4a_palettes(type = "cat", series = "tol")

c4a_palettes(type = "seq", series = "kovesi")

# handy when auto-completion is available:
.P$kovesi$seq$linear_terrain
```

c4a_plot *Plot a color palette*

Description

Plot a color palette, either a cols4all palette, or a color vector. `c4a_plot_cvd` is a shortcut to include color-blind simulated colors, ‘c4a_plot_hex is a shortcut to print hex codes instead of labels.

Usage

```
c4a_plot(
  palette,
  ...,
  dark = FALSE,
  include.na = FALSE,
  hex = FALSE,
  include.cvd = FALSE,
  nrows = NA,
  ncols = NA
)

c4a_plot_cvd(...)

c4a_plot_hex(...)
```

Arguments

`palette`	Palette name (see `c4a`) or a color vector
`...`	arguments passed on to `c4a`
`dark`	dark (black) background?
`include.na`	should a color for missing values be included?
`hex`	should hex codes be printed instead of color labels (or numbers)?
`include.cvd`	should color deficiency simulated colors be included?
`nrows, ncols`	Number of rows and columns. Ignored if `include.cvd = TRUE` (in that case, rows are used for the simulated colors). By default automatically calculated based on aspect ratio of the device.

Value

Besides the plot, a `gTree` is returned silently

Examples

```
c4a_plot("brewer.set1", nrows=1)

c4a_plot("greens", nrows=1)

c4a_plot("tol.pu_gn", nrows=1)

c4a_plot(.P$c4a$bivs$pu_gn_bivs, n = 5)

c4a_plot(.P$c4a$bivd$pu_gn_bivd, n = 5)

c4a_plot(.P$c4a$bivg$gn_bivg, n = 5)
```

`c4a_scores` *Get information from a cols4all palette*

Description

Get information from a cols4all palette

Usage

```
c4a_scores(
  palette,
  n = NA,
  no.match = c("message", "error", "null"),
  verbose = TRUE
)
```

Arguments

`palette`	name of the palette
`n`	number of colors
`no.match`	what happens is no match is found? Options: `"message"`: a message is thrown with suggestions, `"error"`: an error is thrown, `"null"`: `NULL` is returned
`verbose`	should messages be printed?

Value

list with the following items: name, series, fullname, type, palette (colors), na (color), nmax, and reverse. The latter is `TRUE` when there is a `"-"` prefix before the palette name.

c4a_sysdata_import *Import and export system data*

Description

Import and export system data. `c4a_sysdata_import` will import system data and overwrite the current system data, `c4a_sysdata_export` will export the current system data, and `c4a_sysdata_remove` (partly) removes system data.

Usage

```
c4a_sysdata_import(data)

c4a_sysdata_export()

c4a_sysdata_remove(fullnames = NULL, series = NULL, are.you.sure = NA)
```

Arguments

`data`	cols4all data (see `c4a_data`)
`fullnames`	full palette names (so in the format `series.palette_name`)
`series`	a character vector of series names that should be removed (use `"all"` to remove all).
`are.you.sure`	are you sure you want to remove series?

Value

`c4a_sysdata_export` returns the system data (a list)

Examples

```
x = c4a_sysdata_export()
c4a_sysdata_import(x)
y = c4a_sysdata_export()
identical(x, y)
```

scale_color_discrete_c4a_cat *col4all scales for ggplot2*

Description

col4all scales for ggplot2. The scale functions are organized as `scale_<aesthetic>_<mapping>_c4a_<type>`, where the `<aesthetic>` should be either `colo(u)r` or `fill`, `<mapping>` refers to the mapping that is applied (`discrete`, `continuous` or `binned`), and `<type>` is the palette type: `cat`, `seq`, or `div`.

Usage

```
scale_color_discrete_c4a_cat(
  palette = NULL,
  reverse = FALSE,
  order = NULL,
  ...
)

scale_colour_discrete_c4a_cat(
  palette = NULL,
  reverse = FALSE,
  order = NULL,
  ...
)

scale_fill_discrete_c4a_cat(palette = NULL, reverse = FALSE, order = NULL, ...)

scale_color_discrete_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  ...
)

scale_colour_discrete_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  ...
)

scale_fill_discrete_c4a_seq(palette = NULL, reverse = FALSE, range = NULL, ...)

scale_color_discrete_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  ...
)

scale_colour_discrete_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  ...
)

scale_fill_discrete_c4a_div(palette = NULL, reverse = FALSE, range = NULL, ...)
```

```
scale_color_continuous_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_colour_continuous_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_fill_continuous_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_color_continuous_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_colour_continuous_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_fill_continuous_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_color_binned_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_colour_binned_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_fill_binned_c4a_seq(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_color_binned_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_colour_binned_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)

scale_fill_binned_c4a_div(
  palette = NULL,
  reverse = FALSE,
  range = NULL,
  mid = 0,
  n_interp = 11,
  ...
)
```

Arguments

`palette, reverse, order, range`	See `c4a`.
`...`	parameters passed on to the underlying scale functions: `discrete_scale`, `continuous_scale`, and `binned_scale`.
`mid`	data value that should be mapped to the mid-point of the diverging color scale
`n_interp`	number of discrete colors that should be used to interpolate the continuous color scale. Recommended to use an odd number to include the midpoint

Value

A ggplot2 component that defines the scale

Examples

```
if (require("ggplot2")) {
data("diamonds")
diam_exp = diamonds[diamonds$price >= 15000, ]
diam_exp$clarity[1:500] = NA

# discrete categorical scale
ggplot(diam_exp, aes(x = carat, y = price, color = color)) +
geom_point(size = 2) +
scale_color_discrete_c4a_cat("carto.safe") +
theme_light()

# missing values
c4a_plot("tol.muted", 8)
ggplot(diam_exp, aes(x = carat, y = price, fill = clarity)) +
geom_point(size = 2, shape = 21) +
scale_fill_discrete_c4a_cat("tol.muted") +
theme_light()

# discrete sequential scale
ggplot(diam_exp, aes(x = carat, y = price, color = cut)) +
geom_point(size = 2) +
scale_color_discrete_c4a_seq("hcl.blues2") +
theme_light()

# continuous sequential scale
ggplot(diam_exp, aes(x = carat, y = price, color = depth)) +
geom_point(size = 2) +
scale_color_continuous_c4a_seq("hcl.blues2", range = c(0.4, 1)) +
theme_light()

# continuous diverging scale
ggplot(diam_exp, aes(x = carat, y = depth, color = price)) +
geom_point(size = 2) +
scale_color_continuous_c4a_div("wes.zissou1", mid = mean(diam_exp$price)) +
theme_light()

# binned sequential scale
ggplot(diam_exp, aes(x = carat, y = price, color = depth)) +
geom_point(size = 2) +
scale_color_binned_c4a_seq("scico.batlow", range = c(0.4, 1)) +
theme_light()
}
```

Index