Package 'unikn'

November 11, 2026

Type Package

Title Graphical Elements of the University of Konstanz's Corporate Design

Version 1.1.0

Date 2026-11-11

Maintainer Hansjoerg Neth <h.neth@uni.kn>

Description Define and use graphical elements of corporate design manuals in R. The 'unikn' package <doi:10.5281/zenodo.7096191> provides color functions (by defining dedicated colors and color palettes, and commands for finding, changing, viewing, and using them) and styled text elements (e.g., for marking, underlining, or plotting colored titles). The pre-defined range of colors and text decoration functions is based on the corporate design of the University of Konstanz <https://www.uni-konstanz.de/>, but can be adapted and extended for other purposes or institutions.

Depends R (>= 3.4.0)

Imports cli, ggplot2

Suggests knitr, rmarkdown, spelling

Collate 'color_def_1.R' 'color_def_2.R' 'util.R' 'util_color.R' 'util_plot.R' 'color_fun_1.R' 'color_fun_2.R' 'plot_box.R' 'plot_box_calls.R' 'plot_text.R' 'plot_text_calls.R' 'plot_themes.R' 'plot_cube.R' 'plot_demo.R' 'plot_kn.R' 'start_unikn.R'

Encoding UTF-8

License CC BY-SA 4.0

URL https://CRAN.R-project.org/package=unikn, https://hneth.github.io/unikn/, https://hneth.github.io/unikn//

BugReports https://github.com/hneth/unikn/issues

VignetteBuilder knitr

Language en-US

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Hansjoerg Neth [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5427-3141>),
Nico Gradwohl [aut] (ORCID: <https://orcid.org/0000-0002-8703-905X>)

Repository CRAN

Date/Publication 2026-11-11 10:30:02 UTC

Contents

ac . . . 3
Bordeaux . . . 4
demopal . . . 5
get_col_names . . . 6
Grau . . . 7
grepal . . . 7
heading . . . 9
Karpfenblau . . . 11
mark . . . 11
newpal . . . 13
pal_bordeaux . . . 16
pal_grau . . . 16
pal_karpfenblau . . . 17
pal_peach . . . 18
pal_petrol . . . 19
pal_pinky . . . 20
pal_seeblau . . . 20
pal_seegruen . . . 21
pal_signal . . . 22
pal_unikn . . . 23
pal_unikn_dark . . . 24
pal_unikn_light . . . 25
pal_unikn_pair . . . 26
pal_unikn_ppt . . . 27
pal_unikn_pref . . . 28
pal_unikn_web . . . 29
Peach . . . 30
Petrol . . . 30
Pinky . . . 31
plot_cube . . . 32
post . . . 39
Seeblau . . . 41
seecol . . . 42
Seegruen . . . 45
shades_of . . . 46
Signal . . . 47
simcol . . . 48
slide . . . 49

theme_bwkn 50
theme_grau 52
theme_minikn 53
theme_unikn 54
uline 56
unikn.guide 58
url_unikn 58
usecol 59
xbox 60

Index **62**

---

ac *Adjust color transparency*

Description

ac adjusts the transparency of a color or color palette col to an opacity level alpha (in [0, 1]).

Usage

```
ac(col, alpha = 0.5, use_names = TRUE)
```

Arguments

col	A (required) color or color palette (as a vector).
alpha	A factor modifying the opacity alpha (as alpha.f in adjustcolor) to a value in [0, 1]. Default: alpha = .50 (i.e., medium opacity).
use_names	A logical value indicating whether color names should be adjusted to include the values of alpha. Default: use_names = TRUE.

Details

ac is mostly a wrapper for adjustcolor of the **grDevices** package, but allows for more flexible combinations of (multiple) col and alpha values.

The name ac is an abbreviation of "adjust color", but is also a mnemonic aid for providing "air conditioning".

Value

A color vector of the same length as col, transformed by adjustcolor.

See Also

seecol for plotting/seeing color palettes; usecol for using color palettes; simcol for finding similar colors; newpal for defining new color palettes; grepal for finding named colors.

Other color functions: demopal(), grepal(), newpal(), plot_cube(), seecol(), shades_of(), simcol(), usecol()

Examples

```
ac("black")  # using alpha = .5 by default

# multiple colors:
cols <- ac(c("black", "gold", "deepskyblue"), alpha = .50)
seecol(cols, main = "Transparent colors")

# multiple alphas:
blacks <- ac("black", alpha = 5:0/5)
seecol(blacks, main = "One col several alpha values")

bgc <- ac(c("black", "gold"), alpha = 1:6/6)
seecol(bgc, main = "More alpha values than cols")

# Using a color palette:
seecol(ac(pal_unikn_pref, 2/3), main = "Adding color transparency by ac()")

# Color names:
seecol(ac(col = pal_unikn_pref, alpha = c(1/5, 4/5), use_names = TRUE))
seecol(ac(col = pal_unikn_pref, alpha = c(1/5, 4/5), use_names = FALSE))
```

Bordeaux *uni.kn color Bordeaux*

Description

Bordeaux provides the preferred color of pal_bordeaux (as an atomic HEX character value) and is defined as pal_bordeaux[[4]].

Usage

```
Bordeaux
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_bordeaux for the corresponding color palette; pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Grau, Karpfenblau, Peach, Petrol, Pinky, Seeblau, Seegruen, Signal

Examples

```
Bordeaux  # HEX character "#8E2043" (as value)
all.equal(Bordeaux, pal_bordeaux[[4]])  # TRUE (same HEX values)

seecol(Bordeaux)  # view color and details
```

---

demopal *Demonstrate a color palette (in a plot)*

Description

demopal provides an example plot of some type to illustrate a color palette pal.

Usage

```
demopal(pal = pal_unikn, type = NA, pal_name = NULL, ...)
```

Arguments

pal	A color palette (to be illustrated). Default: pal = pal_unikn.
type	The type of plot to be used (as character string or integer value). Permissible types are "bar", "curve", "mosaic", "polygon", or "scatter" (or an integer value from 1 to 5, respectively).
pal_name	A name for the input color palette pal (shown on bottom-right margin). Default: pal_name = NULL (deparsing to input name).
...	Auxiliary arguments passed to type-specific plots (see details).

Details

The demopal wrapper function passes a range of arguments to more specific functions. Common arguments include:

- col_par Default color for par(col);
- alpha Default value for color transparency (in 0:1);
- n A scaling parameter (for random data generation);
- main plot title (on top);
- sub plot subtitle (on right margin);
- seed A random seed value (for reproducible randomness).

The fit between a color palette pal and plot type depends on the uses of colors in a plot. For instance, overlaps of transparent color areas can be evaluated with plot type = "curve" or plot type = "scatter" (and 0 < alpha < 1).

Some functions additionally accept type-specific arguments (e.g., beside, horiz, and as_prop, for plot type = "bar", and cex for plot type = "scatter").

The type-specific functions usually generate some random data (scaled by a parameter n) that is being plotted. This data is returned (as an invisible R object) to enable a plot's reconstruction.

Value

The random data that was plotted (as an invisible R object).

See Also

seepal for plotting color palettes; usecol for using color palettes; shades_of to defining shades of a given color; ac for adjusting color transparency; pal_unikn for the default uni.kn color palette.

Other color functions: ac(), grepal(), newpal(), plot_cube(), seecol(), shades_of(), simcol(), usecol()

Examples

```
demopal(pal = pal_petrol, type = 1)

my_pal <- c(rev(pal_pinky), pal_seeblau)
# Selecting plot type:
demopal(my_pal, type = 2)          # by numeric index
demopal(my_pal, type = "polygon")  # by name

# Passing type-specific arguments:
demopal(type = "scatter", col_par = "black", n = 200, cex = c(2, 4, 6), seed = 101)
```

---

get_col_names *Get color names*

Description

get_col_names gets color names from color palettes.

Usage

```
get_col_names(col, custom_pals = all_pals)
```

Arguments

col	A vector of colors.
custom_pals	A vector of color palettes.

Value

A vector of color names.

Grau *uni.kn color Grau*

Description

Grau provides the preferred color of pal_grau (as an atomic HEX character value) and is defined as pal_grau[[3]].

Usage

```
Grau
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_grau for the corresponding color palette; pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Bordeaux, Karpfenblau, Peach, Petrol, Pinky, Seeblau, Seegruen, Signal

Examples

```
Grau  # HEX character "#9AA0A7" (as value)
all.equal(Grau, pal_grau[[3]])  # TRUE (same HEX values)

seecol(Grau)  # view color and details
```

grepal *Get a vector of colors whose names match a regular expression*

Description

grepal returns a vector of colors whose names match a regular expression (regex).

Usage

```
grepal(pattern, x = colors(), ignore_case = TRUE, plot = TRUE)
```

Arguments

pattern	A regular expression (specified as a string/character object).
x	A vector of R color names or a data frame of named colors (i.e., whose names can be searched). Default: x = colors().
ignore_case	Should the case of pattern be ignored (passed to ignore.case of the grep function)? Default: ignore_case = TRUE.
plot	Boolean: Plot the output (using seecol)? Default: plot = TRUE.

Details

By default, the **base** R vector of named colors (i.e., colors()) is searched for names matching a pattern (which can be a simple string or regular expression).

If x (i.e., the object to be searched) is provided, it is must be a vector of color names or a data frame of named color objects (i.e., a color palette).

If plot = TRUE, grepal also visualizes the detected colors (by passing its result to seecol, as a side-effect).

This function facilitates searching colors by name and yields (a vector of) colors of similar color hue (provided that the color's hue is expressed in a color's name). Its name grepal is an abbreviation of grep and "pal".

See Also

seecol for viewing and comparing color palettes; usecol for using color palettes; simcol for finding similar colors; newpal for defining new color palettes; shades_of to defining shades of a given color; ac for adjusting color transparency; pal_unikn for the default uni.kn color palette.

Other color functions: ac(), demopal(), newpal(), plot_cube(), seecol(), shades_of(), simcol(), usecol()

Examples

```
grepal("tan")

# With regular expressions:
some_grey  <- grepal("gr(a|e)y", plot = FALSE)
start_grey <- grepal("^gr(a|e)y", plot = FALSE)
only_grey  <- grepal("^gr(a|e)y$", plot = FALSE)

length(some_grey)
length(only_grey)

# With other color objects (df as x):
grepal("blau", x = pal_unikn)
grepal("SEE", x = pal_unikn_pref, ignore_case = FALSE)

# Applications:
seecol(grepal("white"), col_bg = "lightblue2", main = "See 'white' colors()")

olives  <- grepal("olive", plot = FALSE)
```

```
oranges <- grepal("orange", plot = FALSE)
seecol(list(olives, oranges),
       pal_names = c("olives", "oranges"),
       main = "Comparing olives and oranges")
```

---

heading *Plot a heading (as marked text elements)*

Description

heading plots 1 or more text strings (provided as a character vector labels) as a heading to an (existing or new) plot and places a colored box behind each label to mark it (i.e., highlighting the heading).

Usage

```
heading(
  labels,
  x = 0,
  y = 0.8,
  x_layout = NA,
  y_layout = "flush",
  col = "black",
  col_bg = "default",
  cex = 2,
  font = 2,
  new_plot = "slide"
)
```

Arguments

labels	A character vector specifying the text labels to be plotted.
x	A numeric vector of x-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: x = 0.
y	A numeric vector of y-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: y = .8.
x_layout	An optional numeric vector or character string to control the horizontal positions of labels. Numeric values are interpreted as increments to values of x and recycled (to enable stepwise or alternating patterns). 3 character string options are: "center" (i.e., center wrt. first label or plot center), "left" (i.e., left wrt. first label or plot center), "right" (i.e., right wrt. first label or plot center). Default: x_layout = NA (i.e., using values of x).

y_layout	A numeric value or character string to control the vertical positions of labels. Numeric values are interpreted as increments to values of y[1] and recycled (to enable stepwise or alternating patterns). 2 character string options are: "even" (i.e., even distribution of labels across available y-space) and "flush" (i.e., no space between adjacent labels, i.e., y_layout = 0). Default: y_layout = "flush".
col	The color(s) of the text label(s). Default: col_lbl = "black".
col_bg	The color(s) to highlight or fill the rectangle(s) with. Default: col_bg = "default" (to automatically select different shades of pal_seeblau).
cex	Numeric character expansion factor(s), multiplied by par("cex") to yield the character size(s). Default: cex = 2.
font	The font type(s) to be used. Default: font = 2 (i.e., bold).
new_plot	Boolean: Should a new plot be generated? Set to "blank" or "slide" to create a new plot, and to "none" to add to an existing plot. Default: new_plot = "slide" (i.e., create a new slide).

Details

Text formatting parameters (like col, col_bg, cex, font) are recycled to match length(labels).

heading uses the base graphics system graphics::.

See Also

slide and xbox to create simple plots (without text).

Examples

```
heading(labels = c("This is a headline", "containing two lines."))

# Note the warning:
heading(labels = c("Headlines", "with 3 or more lines",
                   "should not be arranged", "in such a step-wise fashion."))

# Avoiding the warning:
heading(labels = c("Headlines with", "3 or more lines should",
                     "not be arranged", "in a step-wise fashion."))

# Using non-default colors:
heading(labels = c("Ene,", "mene, miste,", "es rappelt", "in der Kiste."),
      cex = 1.6, col = "white", col_bg = usecol(c(Pinky, Seegruen, Bordeaux, Karpfenblau)))

# Using x_layout and y_layout:
heading(labels = c("Ene,", "mene, miste,", "es rappelt", "in der Kiste."),
        cex = 1.6, col = "white", col_bg = usecol(pal_pinky[2:5]),
        x = NA, y = .6, x_layout = "right", y_layout = "flush")

#' @family text functions
```

Karpfenblau *uni.kn color Karpfenblau*

Description

Karpfenblau provides the preferred color of pal_karpfenblau (as an atomic HEX character value) and is defined as pal_karpfenblau[[4]].

Usage

```
Karpfenblau
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_karpfenblau for the corresponding color palette; pal_unikn_pref for a uni.kn color palette with all preferred colors; pal_unikn for the default uni.kn color palette; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Bordeaux, Grau, Peach, Petrol, Pinky, Seeblau, Seegruen, Signal

Examples

```
Karpfenblau  # HEX character "#3E5496" (as value)
all.equal(Karpfenblau, pal_karpfenblau[[4]])  # TRUE (same HEX values)

seecol(Karpfenblau)  # view color and details
```

mark *Plot marked (or highlighted) text elements*

Description

mark plots 1 or more text strings (provided as a character vector labels) to an (existing or new) plot and places a colored box behind each label to mark it (i.e., highlight or make it stand out from the background).

Usage

```
mark(
  labels,
  x = 0,
  y = 0.55,
  x_layout = NA,
  y_layout = "even",
  col = "black",
  col_bg = Seeblau,
  cex = 2,
  font = 2,
  new_plot = "none"
)
```

Arguments

labels	A character vector specifying the text labels to be plotted.
x	A numeric vector of x-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: x = 0.
y	A numeric vector of y-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: y = .55.
x_layout	An optional numeric vector or character string to control the horizontal positions of labels. Numeric values are interpreted as increments to values of x and recycled (to enable stepwise or alternating patterns). 3 character string options are: "center" (i.e., center wrt. first label or plot center), "left" (i.e., left wrt. first label or plot center), "right" (i.e., right wrt. first label or plot center). Default: x_layout = NA (i.e., using values of x).
y_layout	A numeric value or character string to control the vertical positions of labels. Numeric values are interpreted as increments to values of y[1] and recycled (to enable stepwise or alternating patterns). 2 character string options are: "even" (i.e., even distribution of labels across available y-space) and "flush" (i.e., no space between adjacent labels, i.e., y_layout = 0). Default: y_layout = "even".
col	The color(s) of the text label(s). Default: col_lbl = "black".
col_bg	The color(s) to highlight or fill the rectangle(s) with. Default: col_bg = Seeblau.
cex	Numeric character expansion factor(s), multiplied by par("cex") to yield the character size(s). Default: cex = 2.
font	The font type(s) to be used. Default: font = 2 (i.e., bold).
new_plot	Should a new plot be generated? Set to "blank" or "slide" to create a new plot. Default: new_plot = "none" (i.e., add to an existing plot).

Details

The positions of the text elements in labels can be specified by providing their coordinates (as x and y arguments) or by providing an initial position and an y_layout (see below).

Text formatting parameters (like `col`, `col_bg`, `cex`, `font`) are recycled to match `length(labels)`.

`mark` uses the base graphics system `graphics::`.

See Also

`slide` and `xbox` to create simple plots (without text).

Other text functions: `post()`, `uline()`, `url_unikn()`

Examples

```
# Basics:
mark(labels = "This is a test.", new_plot = "blank")  # create a new blank plot
mark(labels = "More testing here...", y = .45, col_bg = pal_pinky[[2]])  # add to plot

# Example:
# (a) Mark text on an existing plot:
plot(x = 0, y = 0, type = "n", xlim = c(0, 1), ylim = c(0, 1), xlab = "", ylab = "")
mark(x = 0, y = .8, labels = "Mark (on an existing plot)")  # uses existing plot

# (b) Mark text on a new plot:
mark(x = 0, y = .8, labels = "Mark (and create a new plot)",
     new_plot = "slide")  # starts a new plot

# (c) More text and decorations:
mark(x = 0, y = c(.60, .50),
     labels = c("Highlighting text is simple", "and effective"),
     cex = 1.5, col_bg = c(pal_seeblau[[2]], pal_seeblau[[1]]))

mark(labels = c("It is also flexible", "but to be handled with care"),
     x = .4, y = .3, y_layout = "flush", cex = 1.2,
     col = c("white", "black"), col_bg = c(pal_seeblau[[5]], "gold"))

# Using x_layout and y_layout:

mark(labels = c("One, and", "two, and", "three and four is", "plenty and perhaps enough..."),
     cex = 1.4, font = 2, col = "white", col_bg = Bordeaux,
     x = .5, y = .6, x_layout = c(-.25, +.25), y_layout = 0, new_plot = "slide")
```

---

newpal *Define a new color palette*

Description

`newpal` allows defining new color palettes (as data frames or vectors).

Usage

```
newpal(col, names = NULL, pattern = NULL, replacement = NULL, as_df = FALSE)
```

Arguments

col	A required vector of colors (specified as R color names, HEX codes, or RGB values).
names	An optional character vector of color names. Default: names = NULL, using default color names. Setting names = NA removes all color names.
pattern	A pattern to be replaced in names (as REGEX). Default: pattern = NULL.
replacement	A replacement for pattern in names (as REGEX). Default: replacement = NULL.
as_df	Should the new color palette be returned as a data frame (rather than as a vector)? Default: as_df = FALSE.

Details

Specifying pattern and replacment allows modifying names by regular expressions (using gsub(..., perl = TRUE)).

By default, new palette is returned as a (named) vector. Setting as_df = TRUE returns new palette as a data frame.

Value

A (named) vector or data frame.

See Also

seecol for viewing and comparing color palettes; usecol for using color palettes; simcol for finding similar colors; grepal for finding named colors; shades_of to defining shades of a given color; ac for adjusting color transparency; pal_unikn for the default uni.kn color palette.

Other color functions: ac(), demopal(), grepal(), plot_cube(), seecol(), shades_of(), simcol(), usecol()

Examples

```
newpal(col = c("black", "white"), names = c("dark", "bright"))

# Example: 3 ways of defining a new color palette:

# (1) From R color names: -----

pal_flag_de <- newpal(col = c("black", "firebrick3", "gold"),
                      names = c("Schwarz", "Rot", "Gold"))
seecol(pal_flag_de, main = "Colors of the German flag")

# (2) From HEX values: -----

# (a) Google logo colors:
# Source: https://www.schemecolor.com/google-logo-colors.php
color_google <- c("#4285f4", "#34a853", "#fbbc05", "#ea4335")
names_google <- c("blueberry", "sea green", "selective   yellow", "cinnabar")
```

```
pal_google   <- newpal(color_google, names_google, pattern = "\\s+", replacement = "_")

seecol(pal_google, main = "Colors of the Google logo", col_brd = "white", lwd_brd = 10)

# (b) German flag (revised):
# Based on a different source at
# <https://www.schemecolor.com/germany-flag-colors.php>:
pal_flag_de_2 <- newpal(col = c("#000000", "#dd0000", "#ffce00"),
                        names = c("black", "red", "gold")
                        )
seecol(pal_flag_de_2, main = "Colors of the German flag (www.schemecolor.com)")

# (c) Mixing HEX and R color names:
pal_mpg <- newpal(col = c("#007367", "white", "#D0D3D4"),
                  names = c("MPG green", "white", "MPG grey"),
                pattern = "([A-Z])", replacement = "\\L\\1"  # replace upper by lowercase
                  )
seecol(pal_mpg, main = "The colors of the Max Planck Society", col_bg = "grey")

# (3) From RGB values: -----

# A barrier-free color palette
# Source: Okabe & Ito (2002): Color Universal Design (CUD):
#         Fig. 16 of <https://jfly.uni-koeln.de/color/>:

# (a) Vector of colors (as RGB values):
o_i_colors <- c(rgb(  0,   0,   0, maxColorValue = 255),  # black
                rgb(230, 159,   0, maxColorValue = 255),  # orange
                rgb( 86, 180, 233, maxColorValue = 255),  # skyblue
                rgb(  0, 158, 115, maxColorValue = 255),  # green
                rgb(240, 228,  66, maxColorValue = 255),  # yellow
                rgb(  0, 114, 178, maxColorValue = 255),  # blue
                rgb(213,  94,   0, maxColorValue = 255),  # vermillion
                rgb(204, 121, 167, maxColorValue = 255)   # purple
)

# (b) Vector of color names:
o_i_names <- c("black", "orange", "skyblue", "green", "yellow", "blue", "vermillion", "purple")

# (c) Use newpal() to combine colors and names:
pal_okabe_ito <- newpal(col = o_i_colors, names = o_i_names,
                        pattern = "(^[a-z])", replacement = "\\U\\1")  # capitalize initial

seecol(pal_okabe_ito,
       main = "Color-blind friendly color scale (Okabe & Ito, 2002)")

# (+) Compare custom color palettes: -----

my_pals <- list(pal_flag_de, pal_flag_de_2, pal_google, pal_mpg, pal_okabe_ito)
seecol(my_pals, col_brd = "white", lwd_brd = 4,
       main = "Comparing custom color palettes")
```

pal_bordeaux *uni.kn color palette bordeaux*

Description

pal_bordeaux provides an additional uni.kn color palette as a data frame containing 5 colors (shades of Bordeaux).

Usage

```
pal_bordeaux
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_peach and pal_pinky for alternative redish uni.kn color palettes; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_bordeaux
dim(pal_bordeaux)  # 1 5
pal_bordeaux[4]    # preferred (named) color "bordeaux4"
pal_bordeaux[[4]]  # preferred color "bordeaux4" OR "#8E2043"

# Plotting palette:
seecol(pal_bordeaux)
```

pal_grau *uni.kn color palette grau*

Description

pal_grau provides an additional uni.kn color palette as a data frame containing 5 colors (shades of Grau or grey).

Usage

```
pal_grau
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_grau
dim(pal_grau)  # 1 5
pal_grau[3]    # preferred (named) color "grau3"
pal_grau[[3]]  # preferred color "grau3" OR "#9AA0A7"

# Plotting palette:
seecol(pal_grau)
```

---

pal_karpfenblau *uni.kn color palette karpfenblau*

Description

pal_karpfenblau provides an additional uni.kn color palette as a data frame containing 5 colors (shades of Karpfenblau or blue carp).

Usage

```
pal_karpfenblau
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_seeblau for the default seeblau uni.kn color palette; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_karpfenblau
dim(pal_karpfenblau)  # 1 5
pal_karpfenblau[4]    # preferred (named) color "karpfenblau4"
pal_karpfenblau[[4]]  # preferred color "karpfenblau4" OR "#3E5496"

# Plotting palette:
seecol(pal_karpfenblau)
```

---

pal_peach *uni.kn color palette peach*

Description

pal_peach provides an additional uni.kn color palette as a data frame containing 5 colors (shades of Peach).

Usage

```
pal_peach
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_pinky and pal_bordeaux for alternative redish uni.kn color palettes; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_peach
dim(pal_peach)  # 1 5
pal_peach[4]    # preferred (named) color "peach4"
pal_peach[[4]]  # preferred color "peach4" OR "#FEA090"

# Plotting palette:
seecol(pal_peach)
```

`pal_petrol` *uni.kn color palette petrol*

Description

`pal_petrol` provides an additional uni.kn color palette as a data frame containing 5 colors (shades of `Petrol` or grue).

Usage

```
pal_petrol
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details, and https://en.wikipedia.org/wiki/New_riddle_of_induction for the portmanteau "grue".

See Also

`pal_unikn` for the unikn default color palette with all 5 colors of `pal_seeblau`; `pal_seegruen` for an alternative green/grue uni.kn color palette; `pal_unikn_pref` for a uni.kn color palette with all preferred colors; `seecol` for viewing and comparing color palettes; `usecol` for using color palettes.

Other color palettes: `pal_bordeaux`, `pal_grau`, `pal_karpfenblau`, `pal_peach`, `pal_pinky`, `pal_seeblau`, `pal_seegruen`, `pal_signal`, `pal_unikn`, `pal_unikn_dark`, `pal_unikn_light`, `pal_unikn_pair`, `pal_unikn_ppt`, `pal_unikn_pref`, `pal_unikn_web`

Examples

```
pal_petrol
dim(pal_petrol)  # 1 5
pal_petrol[4]    # preferred (named) color "petrol4"
pal_petrol[[4]]  # preferred color "petrol4" OR "#077187"

# Plotting palette:
seecol(pal_petrol)
```

pal_pinky *uni.kn color palette pinky*

Description

pal_pinky provides an additional uni.kn color palette as a data frame containing 5 colors (shades of Pinky or pink).

Usage

```
pal_pinky
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_peach and pal_bordeaux for alternative redish uni.kn color palettes; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_pinky
dim(pal_pinky)  # 1 5
pal_pinky[4]    # preferred (named) color "pinky4"
pal_pinky[[4]]  # preferred color "pinky4" OR "#E0607E"

# Plotting palette:
seecol(pal_pinky)
```

pal_seeblau *uni.kn color palette seeblau*

Description

pal_seeblau provides an additional uni.kn color palette as a data frame containing 5 colors (shades of Seeblau).

Usage

```
pal_seeblau
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_karpfenblau for an alternative blue uni.kn color palette; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_seeblau
dim(pal_seeblau)  # 1 5

# Preferred color:
pal_seeblau[3]    # preferred (named) color "seeblau3" (as df)
pal_seeblau[[3]]  # preferred color value "#59C7EB"

# Access by position:
pal_seeblau[3]    # named color "seeblau3" (as df)
pal_seeblau[[3]]  # color value "#59C7EB"

# Access by name:
pal_unikn["seeblau3"]    # color "seeblau3" (as df)
pal_unikn[["seeblau3"]]  # color value "#59C7EB"

# Plotting palette:
seecol(pal_seeblau)
```

---

pal_seegruen *uni.kn color palette seegruen*

Description

pal_seegruen provides an additional uni.kn color palette as a data frame containing 5 colors (shades of Seegruen).

Usage

```
pal_seegruen
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_petrol for an alternative green uni.kn color palette; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_seegruen
dim(pal_seegruen)  # 1 5
pal_seegruen[4]    # preferred (named) color "seegruen4"
pal_seegruen[[4]]  # preferred color "seegruen4" OR "#0A9086"

# Plotting palette:
seecol(pal_seegruen)
```

---

pal_signal *uni.kn color palette signal (Ampel colors)*

Description

pal_signal provides an additional uni.kn color palette as a data frame containing 3 colors (Ampel or traffic signal colors).

Usage

```
pal_signal
```

Details

The colors are arranged as in a traffic light ("Ampel"):

1. top: red or "bad"
2. mid: yellow or "alert"
3. bot: green or "good"

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_signal
dim(pal_signal)  # 1 3
pal_signal[2]    # (named) color "signal2"
pal_signal[[2]]  # color "signal2" OR "#EFDC60"

# Plotting palette:
seecol(pal_signal)
```

---

pal_unikn *uni.kn default color palette (11 colors)*

Description

pal_unikn combines the 5 shades of blue colors from color palette pal_seeblau with the 6 non-blue colors of pal_unikn_web to a divergent palette of 11 colors.

Usage

```
pal_unikn
```

Details

Adding seeblau5 (i.e., pal_seeblau[1]) to the default color palette pal_unikn also puts white at the central (middle) position of a color palette with 11 values:

pal_unikn[[6]] is white or "#FFFFFF".

A divergent palette is useful for creating color gradients.

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the default uni.kn color palette; pal_seeblau for the uni.kn seeblau color palette; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_unikn
dim(pal_unikn)  # 1 11

# Access by position:
pal_unikn[1]    # new color "seeblau5" (as df)
pal_unikn[[1]]  # new color value "#008ECE"

# Access by name:
pal_unikn["seeblau5"]   # new color "seeblau5" (as df)
pal_unikn[["seeblau5"]] # new color value "#008ECE"

# Viewing/using color palette:
seecol(pal_unikn)
demopal(pal_unikn, type = "curve", main = "Default colors of Konstanz University")

# Note:
pal_unikn[6] # "white" or "#FFFFFF" as central of 11 colors
```

---

pal_unikn_dark *uni.kn color palette of dark colors (10 colors)*

Description

pal_unikn_dark provides an additional uni.kn color palette that collects 2 dark colors of 5 color palettes as a data frame containing 10 colors (in 5 pairs).

Usage

```
pal_unikn_dark
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn_light for a lighter uni.kn color palette; pal_unikn_pair for a pairwise uni.kn color palette; pal_unikn for the default uni.kn color palette; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_unikn_dark
dim(pal_unikn_dark)  # 1 8
pal_unikn_dark[1]    # color "karpfenblau5" by position
pal_unikn_dark[[1]]  # color value by position: #324376"
pal_unikn_dark["karpfenblau5"]  # color value by name

# Viewing/using color palette:
seecol(pal_unikn_dark)
demopal(pal_unikn_dark, type = "points", main = "Dark colors of Konstanz University")
```

---

pal_unikn_light *uni.kn color palette of light colors (10 colors)*

Description

pal_unikn_light provides an additional uni.kn color palette that collects 2 light colors of 5 color palettes as a data frame containing 10 colors (in 5 pairs).

Usage

```
pal_unikn_light
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn_dark for a darker uni.kn color palette; pal_unikn_pair for a pairwise uni.kn color palette; pal_unikn for the default uni.kn color palette; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref, pal_unikn_web

Examples

```
pal_unikn_light
dim(pal_unikn_light)  # 1 10

# Access by position:
pal_unikn_light[1]    # color "seeblau3" (as df)
pal_unikn_light[[1]]  # color value "#59C7EB"

# Access by name:
```

```
pal_unikn_light["seeblau3"]   # color "seeblau3" (as df)
pal_unikn_light[["seeblau3"]] # color value "#59C7EB"

# Viewing/using color palette:
seecol(pal_unikn_light)
demopal(pal_unikn_light, type = "bar", main = "Light colors of Konstanz University")
```

`pal_unikn_pair` *uni.kn color palette of pairwise colors (16 colors)*

Description

`pal_unikn_pair` provides an additional uni.kn color palette that collects 16 paired colors of 8 color palettes as a data frame containing 16 colors (in 8 pairs).

Usage

```
pal_unikn_pair
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

`pal_unikn_light` for a lighter uni.kn color palette; `pal_unikn_dark` for a darker uni.kn color palette; `pal_unikn` for the default uni.kn color palette; `seecol` for viewing and comparing color palettes; `usecol` for using color palettes.

Other color palettes: `pal_bordeaux`, `pal_grau`, `pal_karpfenblau`, `pal_peach`, `pal_petrol`, `pal_pinky`, `pal_seeblau`, `pal_seegruen`, `pal_signal`, `pal_unikn`, `pal_unikn_dark`, `pal_unikn_light`, `pal_unikn_ppt`, `pal_unikn_pref`, `pal_unikn_web`

Examples

```
pal_unikn_pair
dim(pal_unikn_pair)  # 1 16
pal_unikn_pair[1]    # color "karpfenblau4" by position
pal_unikn_pair[[1]]  # color value by position: #3E5496"
pal_unikn_pair["karpfenblau4"]  # color value by name

# Viewing/using color palette:
seecol(pal_unikn_pair)
demopal(pal_unikn_pair, type = "polygon", main = "A pair-wise color palette")
```

pal_unikn_ppt *uni.kn secondary color palette (ppt version)*

Description

pal_unikn_ppt provides an alternative uni.kn color palette as a data frame containing 10 colors.

Usage

```
pal_unikn_ppt
```

Details

This is a secondary (ppt) variant with more muted colors.

See pal_unikn for the primary/default (web/sRGB) scale and https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_pref, pal_unikn_web

Examples

```
pal_unikn_ppt
dim(pal_unikn_ppt)  # 1 10

# Access by position:
pal_unikn_ppt[2]    # 2nd named color "seeblau3" (as df)
pal_unikn_ppt[[2]]  # 2nd color value "#59B6DC"

# Access by name:
pal_unikn_ppt["seeblau3"]    # color "seeblau3" (as df)
pal_unikn_ppt[["seeblau3"]]  # color value "#59B6DC"

# Plotting palette:
seecol(pal_unikn_ppt)
```

pal_unikn_pref *uni.kn color palette of preferred colors (9 colors)*

Description

pal_unikn_pref provides an additional uni.kn color palette that collects the preferred color of each palette as a data frame containing 9 (or 8 + 1) colors.

Usage

```
pal_unikn_pref
```

Details

The colors are arranged in a sequence that provides high contrasts between adjacent colors.

Note that the (alert) color Signal is not a preferred color according to the official color definition.

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the default uni.kn color palette; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_web

Examples

```
pal_unikn_pref
dim(pal_unikn_pref)  # 1 9

# Access by position:
pal_unikn_pref[1]    # color Seeblau (as df)
pal_unikn_pref[[1]]  # color value "#59C7EB"

# Access by name:
pal_unikn_pref["Seeblau"]   # color "seeblau3" (as df)
pal_unikn_pref[["Seeblau"]] # color value "#59C7EB"

# Viewing/using color palette:
seecol(pal_unikn_pref)
demopal(pal_unikn_pref, type = "mosaic", main = "Preferred colors of Konstanz University")
```

pal_unikn_web *uni.kn default color palette (web version)*

Description

pal_unikn_web provides the default uni.kn color palette as a data frame containing 10 colors.

Usage

```
pal_unikn_web
```

Details

This is the primary (web/sRGB) scale.

Note that pal_unikn provides a divergent color palette (of 11 colors).

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_ppt for an alternative (ppt) version; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other color palettes: pal_bordeaux, pal_grau, pal_karpfenblau, pal_peach, pal_petrol, pal_pinky, pal_seeblau, pal_seegruen, pal_signal, pal_unikn, pal_unikn_dark, pal_unikn_light, pal_unikn_pair, pal_unikn_ppt, pal_unikn_pref

Examples

```
pal_unikn_web
dim(pal_unikn_web)  # 1 10

# Access by position:
pal_unikn_web[2]    # 2nd named color "seeblau3" (as df)
pal_unikn_web[[2]]  # 2nd color value "#59C7EB"

# Access by name:
pal_unikn_web["seeblau3"]    # color "seeblau3" (as df)
pal_unikn_web[["seeblau3"]]  # color value "#59C7EB"

# Plotting palette:
seecol(pal_unikn_web)
```

Peach *uni.kn color Peach*

Description

Peach provides the preferred color of pal_peach (as an atomic HEX character value) and is defined as pal_peach[[4]].

Usage

```
Peach
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_peach for the corresponding color palette; pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Bordeaux, Grau, Karpfenblau, Petrol, Pinky, Seeblau, Seegruen, Signal

Examples

```
Peach  # HEX character "#FEA090" (as value)
all.equal(Peach, pal_peach[[4]])  # TRUE (same HEX values)

seecol(Peach)  # view color and details
```

Petrol *uni.kn color Petrol*

Description

Petrol provides the preferred color of pal_petrol (as an atomic HEX character value) and is defined as pal_petrol[[4]].

Usage

```
Petrol
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_petrol for the corresponding color palette; pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Bordeaux, Grau, Karpfenblau, Peach, Pinky, Seeblau, Seegruen, Signal

Examples

```
Petrol  # HEX character "#077187" (as value)
all.equal(Petrol, pal_petrol[[4]])  # TRUE (same HEX values)

seecol(Petrol)  # view color and details
```

Pinky *uni.kn color Pinky*

Description

Pinky provides the preferred color of pal_pinky (as an atomic HEX character value) and is defined as pal_pinky[[4]].

Usage

```
Pinky
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_pinky for the corresponding color palette; pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Bordeaux, Grau, Karpfenblau, Peach, Petrol, Seeblau, Seegruen, Signal

Examples

```
Pinky  # HEX character "#E0607E" (as value)
all.equal(Pinky, pal_pinky[[4]])  # TRUE (same HEX values)

seecol(Pinky)  # view color and details
```

---

plot_cube *Plot colors as points in a 3D RGB color cube*

Description

plot_cube plots one or more colors x as points in a 3-dimensional cube whose axes correspond to the red, green, and blue values of the RGB color model (ranging from 0 to 255).

Usage

```
plot_cube(
  x,
  type = "p",
  col_points = NA,
  cex_points = 2,
  pch = 21,
  col_brd = grey(0.2, 1),
  ref_lines = TRUE,
  col_line = NA,
  lwd_line = 3,
  line_connect = "all",
  col_fill = NA,
  fill_alpha = 0.8,
  fill_res = 20,
  shadows = "none",
  col_shadow = grey(0.7, 0.9),
  show_corners = TRUE,
  col_cube = grey(0.5, 1),
  grid = FALSE,
  grid_faces = FALSE,
  grid_inner = FALSE,
  grid_at = 2,
  col_grid = grey(0.6, 0.6),
  col_grid_inner = grey(0.8, 0.5),
  axis_labels = c("R", "G", "B"),
  axis_ticks = TRUE,
  n_ticks = 5,
  cex_axis = NA,
  labels = "names",
  theta = -30,
```

```
  phi = 15,
  main = "Colors in 3D RGB space",
  cex_lbl = 0.85,
  ...
)
```

Arguments

x	A (character) vector of colors to be plotted, specified as R color names (e.g., "deepskyblue"), HEX values (e.g., "#59C7EBFF"), or any mixture thereof. Color palettes (e.g., pal_unikn) are also accepted.
type	Type of geometric element to draw from the colors x: • "p": Plot colors as **points** (default); • "l": Connect colors by **lines** (a path through RGB space); • "b": Draw **both** points and lines; • "plane": Span a **plane** (filled polygon) through the colors (requires 3 or more colors; 3 colors yield a triangle). By default, connecting lines and plane fills are shown as color *gradients* (i.e., each location is shown in the color of the corresponding RGB mixture), to illustrate the additive mixing of colors in RGB space. In all cases, only the colors x *provided* (i.e., the points that define the line or plane) are labeled by default (see labels). Default: type = "p".
col_points	Color(s) used for the plotted points. Default: col_points = NA uses each point's own color (i.e., every color in x is plotted in that color). Providing a color (or color vector) overrides this, so that all points are shown in the specified color(s).
cex_points	Scaling factor for the size of points. Default: cex_points = 2.
pch	Plotting symbol for points (see points). Default: pch = 21 (a fillable circle).
col_brd	Border color of points (only used for fillable symbols, i.e., pch in 21:25). Default: col_brd = grey(.20, 1) (dark grey).
ref_lines	Boolean: Draw a vertical reference line from each point down to the base (R-by-G) plane of the cube (to ease judging a point's height on the B/blue axis)? Default: ref_lines = TRUE.
col_line	Color of connecting lines (for type = "l" or "b") and of the plane outline (for type = "plane"). Default: col_line = NA (i.e., show lines as a color gradient of the mixtures between consecutive colors).
lwd_line	Line width of connecting lines (for type = "l" or "b") and of plane outlines (for type = "plane"). Raise this value for thicker, more visible lines. Default: lwd_line = 3.
line_connect	How to connect 3 or more colors by lines (for type = "l" or "b"): • "all": Connect *all pairs* of colors (i.e., n * (n - 1) / 2 lines, a complete graph); • "pairs": Connect only *disjoint consecutive pairs* of colors, i.e., (1st, 2nd), (3rd, 4th), etc. (yielding floor(n / 2) lines; if n is odd, the final color remains unpaired, but is still shown as a point);

- "path": Connect only *consecutive* colors (i.e., n - 1 lines, tracing the palette in its given order).

For 2 colors, all 3 options yield the same single line. (The value "pairwise" is accepted as an alias for "all".) Default: line_connect = "all".

col_fill — Fill color for the plane (for type = "plane"). Default: col_fill = NA (i.e., fill the plane with a color gradient showing the gamut of all mixtures of the defining colors).

fill_alpha — Opacity of the plane fill (in range [0, 1]). Default: fill_alpha = .80.

fill_res — Resolution of color gradients (i.e., the number of line segments per connection, and the granularity of the plane tessellation). Higher values yield smoother gradients (but slower plotting). Default: fill_res = 20.

shadows — Project each point onto one or more faces of the cube (as "shadow" points, to aid judging its 3D location). Options (one or more of which may be combined) are:

- "floor": project onto the R-by-G plane (i.e., set B = 0);
- "back": project onto the R-by-B plane (i.e., set G = 0);
- "side": project onto the G-by-B plane (i.e., set R = 0);
- "all": project onto all 3 faces;
- "none": show no shadows.

Each shadow is a smaller marker on the corresponding face, linked to its point by a dotted connector. As a projection drops one color channel, a shadow also shows the color with that primary removed. Default: shadows = "none".

col_shadow — Color of the shadow markers. Default: col_shadow = grey(.70, .9) (a neutral grey).

Setting col_shadow = NA shows each shadow in its own *projected* color. Projecting a point onto a face sets the one color channel perpendicular to that face (i.e., the RGB axis *not* spanned by the face's plane) to zero: a "floor" shadow sets B = 0, a "back" shadow sets G = 0, and a "side" shadow sets R = 0 (see shadows). As one channel is dropped, a projected shadow color thus *systematically differs* from its source color: e.g., "violet" (238, 130, 238) casts an orange (238, 130, 0) floor shadow, a magenta (238, 0, 238) back shadow, and a blue (0, 130, 238) side shadow.

show_corners — Boolean option to mark the 8 corners of the cube by points in their corresponding colors (i.e., black, white, and the 6 primary/secondary colors RGB/CMY). Default: show_corners = TRUE.

col_cube — Color for the edges and axes of the cube. Default: col_cube = grey(.50, 1).

grid — Boolean convenience switch to enable a default reference-grid combination (akin to the grid of seecol). grid = TRUE turns on the inner/volume grid (grid_inner = TRUE), which in turn draws the face grid on all 6 faces (grid_faces = "all", so the interior lines end on the faces), at the default spacing (grid_at); grid = FALSE (default) shows no grid. For finer control, set grid_faces, grid_inner, and grid_at directly. Default: grid = FALSE.

grid_faces — Draw reference grid lines *on the cube's faces*, to ease judging a point's 3D location. Options are:

- FALSE: Show no face grid (default);
- "base": Grid on the 3 faces meeting at the black corner (floor + 2 near walls);
- "all": Grid on all 6 faces (a full box grid).

Line positions are set by grid_at. When grid_inner is enabled, grid_faces defaults to "all" (so the interior lines terminate on the faces), unless set explicitly. Default: grid_faces = FALSE.

grid_inner — Boolean to draw reference grid lines *across the cube's volume* (interior lines connecting opposite faces: front-to-back, top-to-bottom, and left-to-right), as a 3D orientation aid. Line positions are set by grid_at; best used sparsely (small grid_at), as each position adds many interior lines. (Enabling this also sets grid_faces = "all" by default; see above.) Default: grid_inner = FALSE.

grid_at — Positioning of the grid lines (for grid_faces and grid_inner) along each axis. Options are:

- an integer k: Dissect each axis into k equal sub-sections (e.g., 2 = halves/midpoint (default), 3 = thirds, 4 = quarters);
- "ticks": At the current axis-tick positions (see n_ticks);
- a numeric vector: Use custom positions (in 0:255).

(The endpoints 0 and 255 are always omitted, as they are the cube's edges.) Default: grid_at = 2 (sparse).

col_grid — Color of the *face* grid lines (grid_faces). Default: col_grid = grey(.60, .60) (a thin, transparent, darker grey).

col_grid_inner — Color of the *inner* (volume) grid lines (grid_inner). Default: col_grid_inner = grey(.80, .50) (a thin, transparent, lighter grey) — lighter than col_grid, so the face grid reads as the stronger reference.

axis_labels — Titles for the 3 cube axes (R, G, and B). Options are:

- a character vector of length\ 3: use these as axis titles;
- "short" (or "rgb"): use c("R", "G", "B") (default);
- "long": use c("Red", "Green", "Blue");
- NA (or "no"): show no axis titles.

Default: axis_labels = c("R", "G", "B").

axis_ticks — Draw tick marks and numeric value labels (in range 0 to 255) along the 3 cube axes? Either a Boolean value, or a numeric vector specifying custom tick positions (in 0:255). Default: axis_ticks = TRUE.

n_ticks — Number of (equally spaced) ticks per axis (only used when axis_ticks = TRUE). Default: n_ticks = 5 (i.e., ticks at 0, 64, 128, 191, 255).

cex_axis — Scaling factor for axis titles and tick values. Default: cex_axis = NA (i.e., use cex_lbl).

labels — Type(s) of labels for the points. One or more of the following label types may be specified (and are shown on separate lines when combined):

- "names": show the names of x (or the color values, if no names are available);
- "rgb": show each point's RGB triplet (as (R, G, B));

- "hex": show each point's HEX value (e.g., "#59C7EB").

Thus, labels = c("names", "rgb", "hex") shows all 3 representations of each color in the same cube. Setting labels = NA (or "no") shows no labels, and passing a character vector (of custom labels) shows these labels instead. Default: labels = "names".

theta	Horizontal rotation angle (azimuth, used by persp). Default: theta = -30.
phi	Vertical rotation angle (colatitude, used by persp). Default: phi = 15.
main	Main plot title. Default: main = "Colors in 3D RGB space".
cex_lbl	Scaling factor for text labels (axes, titles, point labels). Default: cex_lbl = .85.
...	Other (graphical) parameters passed to persp (e.g., d, expand, shade).

Details

Each color is converted into its RGB coordinates (via col2rgb) and shown as a point at the corresponding location in the color cube. Points are filled with the color they represent, so that the location of a point in the cube illustrates the additive composition of its color from the three primary colors (Red, Green, and Blue).

plot_cube uses base graphics (i.e., persp and trans3d) and requires no additional package dependencies.

Value

A projection matrix (of class returned by persp) that maps 3D coordinates to the 2D plotting plane (returned invisibly).

See Also

seecol for viewing and comparing color palettes; usecol for using color palettes; newpal for defining new color palettes; col2rgb for converting colors to RGB values.

Other color functions: ac(), demopal(), grepal(), newpal(), seecol(), shades_of(), simcol(), usecol()

Examples

```
# A. Basics: ------

plot_cube("tomato")                       # a single color
plot_cube(c("red", "green", "blue"))      # primary RGB colors
plot_cube(c("black", "grey50", "white"))  # grey diagonal

# Using unikn colors:
plot_cube(pal_unikn)       # a bi-polar color palette
plot_cube(pal_unikn_pref)  # a categorical color palette

# Reduced color options:
col_mix <- c("grey50", "deepskyblue", "forestgreen", "gold", "orchid", "tomato")
plot_cube(col_mix, show_corners = FALSE)    # hide corner colors
plot_cube(pal_unikn, col_points = "black")  # all points in 1 color
```

```
# Labeling points (by names, RGB or HEX values, or combinations):
col_4 <- c("red3", "green4", "gold", "violet")
plot_cube(col_4)                                    # names (default)
plot_cube(col_4, labels = "hex")                    # HEX values
plot_cube(col_4, labels = "rgb")                    # RGB values
plot_cube(col_4, labels = c("names", "hex", "rgb"))  # all 3 types
plot_cube(col_4, labels = "no")                     # no labels

# Connecting points by lines (type = "l" or "b"):
plot_cube(c("grey", "white"), type = "l")          # line (grey diagonal)
plot_cube(c("red", "blue"), type = "b")           # line with points (red-blue gradient)

# Changing the viewing angle:
plot_cube(pal_unikn, theta = 30, phi = 25)

# With >2 colors: all pairs (default), disjoint pairs, or a connecting path:
col_3 <- c("grey10", "orange", "steelblue")
plot_cube(col_3, type = "l")                              # all pairs
plot_cube(col_3, type = "b", lwd_line = 5)                # all pairs + points
plot_cube(pal_unikn, type = "b", line_connect = "path")  # path (palette order)

# Using line_connect = "pairs" to link disjoint color pairs, e.g., to show
# the 3 complementary RGB/CMY pairs (each line crosses the grey cube center):
plot_cube(c("red", "cyan", "green", "magenta", "blue", "yellow"), type = "b",
         line_connect = "pairs", lwd_line = 5, grid = TRUE, main = "Complementary colors")

# Spanning a plane through 3 colors (type = "plane"):
plot_cube(c("red", "green", "blue"), type = "plane")  # gamut of 3 primary RGB colors
plot_cube(c("cyan", "yellow", "magenta"),             # gamut of 3 opposite RGB colors
             type = "plane", labels = c("names", "rgb"))

# Shadow points (projections onto cube faces):
plot_cube(c("tomato", "deepskyblue"), shadows = "floor")  # on the floor (B = 0)
plot_cube(pal_seeblau, shadows = "all")                    # on all 3 faces
plot_cube(c("grey20", "grey50", "grey80"),             # show shadows in projected colors
          shadows = "all", col_shadow = NA)

# Axis options (titles and value ticks):
plot_cube(pal_unikn, axis_labels = "long")           # long axis titles
plot_cube(pal_unikn, n_ticks = 11)                   # finer axis scale (0, 25.5, ...)
plot_cube(pal_unikn, axis_ticks = c(0, 128, 255))    # custom tick positions
plot_cube(pal_unikn, axis_ticks = FALSE)             # no ticks (titles only)
plot_cube(pal_unikn, axis_labels = "no", axis_ticks = FALSE)  # bare cube

# Grid options (to aid judging 3D locations):
# Both reference grids improve orientation, but only when used sparsely.

# a. Default of grid = TRUE (splits cube into octants):
col_mix <- c("grey50", "deepskyblue", "forestgreen", "gold", "orchid", "tomato")
plot_cube(col_mix, grid = TRUE)

# b. Both grids at thirds (more structure, but quickly gets dense):
```

```
plot_cube(col_mix, grid = TRUE, grid_at = 3)

# c. Both grids at rare axis ticks (to align grid with tick labels):
plot_cube(col_mix, grid = TRUE, grid_at = "ticks", n_ticks = 4)

# d. Face grid only (often suffices, as inner grid may not be needed):
plot_cube(pal_unikn, grid_faces = "base", n_ticks = 5, grid_at = "ticks")


# B. Basic demonstrations of the 3D color cube (RGB/CMY): ------

# 1. The 8 color corners of the RGB cube (b/w + RGB + CMY):
col_8 <- c("black", "white", "red", "cyan", "blue", "yellow", "green", "magenta")
plot_cube(col_8, theta = -30, phi = 30, labels = c("names", "rgb", "hex"),
          main = "The 6 corners of the RGB cube")

# 2. The main black-grey-white diagonal of the cube (with all shades of grey):
plot_cube(c("black", "white"), type = "l", lwd_line = 5,
          main = "Main b/w-diagonal (with all shades of grey)")
greys <- paste("grey", seq(0, 100, length.out = 5))  # 5 shades of grey (white to black)
plot_cube(greys, type = "b", line_connect = "path",
          lwd_line = 5, labels = c("names", "hex", "rgb"),
          theta = -30, phi = 30,
          main = "5 shades of grey (on b/w diagonal)")

# 3. Complementarity of the RGB and CMY color systems:
#    Each CMY secondary (cyan, magenta, yellow) is the complement of an RGB
#    primary (red, green, blue): it occupies the opposite corner of the cube,
#    and a color and its complement mix to grey (at the cube's center).

# Show 6 color corners (RGB vs. CMY):
col_6 <- c("red", "cyan", "blue", "yellow", "green", "magenta")
plot_cube(col_6, labels = c("names", "rgb"),
          main = "RGB vs CMY: 6 complementary corners of the cube")

# Show all 4 cube diagonals:
plot_cube(col_8, type = "b", line_connect = "pairs", lwd_line = 4,
          grid = TRUE, main = "The 4 main diagonals")

# Show 2 triangular planes:
plot_cube(c("red", "blue", "green"), type = "plane", main = "The RGB plane")
plot_cube(c("cyan", "yellow", "magenta"), type = "plane", main = "The CYK plane")


# C. Illustrating colors and color palettes (unikn): ------

# 0. Collect main unikn colors (as a vector):
col_4 <- c(Seeblau, Pinky, Seegruen, Bordeaux)
col_8 <- c(col_4, Petrol, Karpfenblau, Peach, Grau)

# 1. Main unikn colors as (labeled) points:
plot_cube(col_8, grid = TRUE, main = "The main unikn colors in RGB space")
```

```
# 2. The default unikn palette (black - Seeblau shades - white) as a path:
plot_cube(pal_unikn, type = "b", line_connect = "path", lwd_line = 3,
          main = "A bi-polar palette (pal_unikn)")
plot_cube(pal_unikn_pref, type = "b", line_connect = "path", lwd_line = 3,
          main = "A categorical color palette (pal_unikn_pref)")

# 2. A color palette's shade ramp as a connecting path:
plot_cube(pal_pinky, type = "b", line_connect = "path", lwd_line = 5,
          theta = -40, phi = 20, shadows = "all", main = "The path of pal_pinky")
plot_cube(usecol(c(rev(pal_seegruen), "white", pal_bordeaux)), type = "b",
          line_connect = "path", lwd_line = 5, main = "A custom color gradient")

# 4. Pairwise relations among main colors:
plot_cube(col_4, type = "b", lwd_line = 2, theta = -40, phi = 20,
          main = "Pairwise relations of 4 key colors")

# 5. The gamut (plane) of mixtures spanned by 3 main colors:
plot_cube(c(Seeblau, Pinky, Seegruen), type = "plane",
          theta = 10, phi = 20, main = "Gamut of 3 key colors")
```

`post` *Post text (in an xbox)*

Description

`post` plots 1 or more text strings (provided as a character vector `labels`) to an (existing or new) `xbox`.

Usage

```
post(
  labels,
  x = 0.03,
  y = 0.55,
  x_layout = NA,
  y_layout = "even",
  col = "white",
  col_bg = Seeblau,
  cex = 1,
  font = 1,
  new_plot = "none"
)
```

Arguments

`labels`	A character vector specifying the text labels to be plotted.

x	A numeric vector of x-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: x = .03.
y	A numeric vector of y-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: y = .55.
x_layout	An optional numeric vector or character string to control the horizontal positions of labels. Numeric values are interpreted as increments to values of x and recycled (to enable stepwise or alternating patterns). 3 character string options are: "center" (i.e., center wrt. first label or plot center), "left" (i.e., left wrt. first label or plot center), "right" (i.e., right wrt. first label or plot center). Default: x_layout = NA (i.e., using values of x).
y_layout	A numeric value or character string to control the vertical positions of labels. Numeric values are interpreted as increments to values of y[1] and recycled (to enable stepwise or alternating patterns). 2 character string options are: "even" (i.e., even distribution of labels across available y-space) and "flush" (i.e., no space between adjacent labels, i.e., y_layout = 0). Default: y_layout = "even".
col	The color(s) of the text label(s). Default: col_lbl = "white".
col_bg	The background color(s) of the xbox. Default: col_bg = Seeblau.
cex	Numeric character expansion factor(s), multiplied by par("cex") to yield the character size(s). Default: cex = 1.0.
font	The font type(s) to be used. Default: font = 1 (i.e., plain text).
new_plot	Should a new plot be generated? Set to "xbox" to plot to a basic xbox (with square dimensions, i.e., dim = c(1, 1)). Default: new_plot = "none" (i.e., assumes a pre-existing xbox).

Details

The positions of the text elements in labels can be specified by providing their coordinates (as x and y arguments) or by providing an initial position and an y_layout (see below).

Text formatting parameters (like col, col_bg, cex, font) are recycled to match length(labels).

post uses the base graphics system graphics::.

See Also

xbox to create a new xbox (without text).

Other text functions: mark(), uline(), url_unikn()

Examples

```
# Create a new xbox:
post(labels = "This is a test.", new_plot = "xbox",
     cex = 1.2, font = 2, col_bg = pal_seeblau[[5]])

# Add text to an existing xbox:
```

```
post(labels = c("More text follows here,",
                "yet another line here,",
                "and even more here."),
                y = .4, y_layout = .04,
                new_plot = "none")

# Using x_layout and y_layout:
post(labels = c("Ene,", "mene, miste,", "es rappelt", "in der Kiste."),
     cex = 1.4, font = 2, col = "white", col_bg = Pinky,
     x = .1, y = .5, x_layout = "left", y_layout = .05, new_plot = "xbox")

post(labels = c("Hello world!", "Does this work?", "That's good!", "Please carry on..."),
     cex = 1.4, font = 2, col = "white", col_bg = Karpfenblau,
     x = .01, y = .6, x_layout = .10, y_layout = .05, new_plot = "xbox")
```

`Seeblau` *uni.kn color Seeblau*

Description

`Seeblau` provides the preferred color of `pal_seeblau` (as an atomic HEX character value) and is defined as `pal_seeblau[[3]]`.

Usage

```
Seeblau
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

`pal_seeblau` for the corresponding color palette; `pal_unikn` for the unikn default color palette with all 5 colors of `pal_seeblau`; `pal_unikn_pref` for a uni.kn color palette with all preferred colors; `seecol` for viewing and comparing color palettes; `usecol` for using color palettes.

Other preferred colors: `Bordeaux`, `Grau`, `Karpfenblau`, `Peach`, `Petrol`, `Pinky`, `Seegruen`, `Signal`

Examples

```
Seeblau  # HEX character "#59C7EB" (as value)
all.equal(Seeblau, pal_seeblau[[3]])  # TRUE (same HEX values)

seecol(Seeblau)  # view color and details
```

`seecol` *Plot color palettes (to see their colors)*

Description

`seecol` provides an interface to plotting (or "seeing") the colors of a palette or comparing multiple color palettes.

Usage

```
seecol(
  pal = "unikn_all",
  n = "all",
  alpha = NA,
  hex = NULL,
  rgb = NULL,
  col_bg = NULL,
  col_brd = NULL,
  lwd_brd = NULL,
  grid = TRUE,
  scale_x = FALSE,
  main = NA,
  sub = NULL,
  title = NULL,
  mar_note = NA,
  pal_names = NA,
  cube = FALSE,
  ...
)
```

Arguments

`pal` A single color palette (as a vector of colors), multiple color palettes (as a list), or a recognized keyword (as a character string). Default: `pal = "unikn_all"` (i.e., plot all color palettes provided by the **unikn** package).

Recognized keywords are:

1. `"all"`: All color palettes of the **unikn** package.
2. `"all_unikn"` or `"unikn_all"`: All uni.kn color palettes (of the University of Konstanz).
3. `"unikn_basic"`: All basic uni.kn palettes.
4. `"grad_all"`: All uni.kn palettes with color gradients.
5. `"pair_all"`: All uni.kn palettes with pairwise colors.
6. `"pref_all"`: All preferred uni.kn colors and their gradients.
7. `"add"`: Additional/contributed color palettes (deprecated, as additional color palettes were migrated to the **unicol** R package).

seecol does also recognize keywords (e.g., "all_unikn") or keywords without "unikn" (e.g., "basic").

n — Number of colors to show or use. If n is lower or higher than the length of the current color palette pal, the color palette is reduced or extrapolated (using grDevices::colorRampPalette). Default: n = "all" (i.e., show all colors in palette).

alpha — A factor modifying the opacity alpha (as alpha.f in adjustcolor) to a value in [0, 1]. Default: alpha = NA (i.e., no modification of opacity).

hex — Should HEX color values be shown? Default: hex = NULL (i.e., show HEX color values when there is sufficient space to print them).

rgb — Should RGB color values be shown? Default: rgb = NULL (i.e., show RGB color values when there is sufficient space to print them).

col_bg — Color of plot background. Default: col_bg = NULL.

col_brd — Color of shape borders (if shown). Default: col_brd = NULL.

lwd_brd — Line width of shape borders (if shown). Default: lwd_brd = NULL.

grid — Show a reference grid in the color plot? In the standard (swatch) view, this shows a background grid; in the cube view (see cube), it is passed to plot_cube as its grid switch (so grid = TRUE shows both the face and inner grids at their mid-points). Default: grid = TRUE.

scale_x — Scale color shapes (when comparing multiple palettes) to a fixed total width? Default: scale_x = FALSE.

main — Main plot title (as a character string). Default: main = NA creates a default title.

sub — Optional subtitle (as a character string). Default: sub = NULL (i.e., no subtitle).

title — Deprecated plot title. Use main instead.

mar_note — Optional margin note (on bottom right). Default: mar_note = NA (i.e., no margin note).

pal_names — Names of color palettes or colors (as a character vector). Default: pal_names = NA (for default names).

cube — Render a *single* color palette as points in a 3D RGB color cube (via plot_cube)? Either a logical value/Boolean, or a list of arguments passed to plot_cube:

- cube = FALSE: Show the standard (swatch) view (default);
- cube = TRUE: Show the color cube, using main as its title and deriving point labels from the current hex and rgb settings (color names are always shown; hex and/or rgb values are added when set to TRUE);
- cube = list(...): show the color cube, passing the named list entries on to plot_cube (overriding the derived defaults, e.g., cube = list(type = "b", shadows = "all")).

The cube also inherits the current grid setting (passed to plot_cube), unless the cube list specifies a grid (via grid, grid_faces, or grid_inner). When multiple palettes are provided, the cube view is skipped (with a message) and the standard comparison view is shown. Default: cube = FALSE.

... — Other graphical parameters (passed to plot).

Details

seecol has two main modes, based on the contents of its pal argument:

1. if pal is set to a *specific* color palette (or a vector of multiple colors or color palettes):
   Plot the current color palette and optional details on its colors.
2. if pal = "unikn_all" or a list of *multiple* color palettes:
   Plot visual vectors of all current color palettes for comparing them.

Specifying distinct = TRUE removes visual duplicate colors (based on HEX values, ignoring transparency), but only when showing an individual color palette pal.

Various title options (i.e., main, sub, and mar_note) and a pal_names argument add control over plotted text labels. However, the length of a character vector provided to pal_names must correspond to the number of (custom) color palettes or colors.

See Also

usecol for using color palettes; simcol for finding similar colors; newpal for defining new color palettes; grepal for finding named colors; shades_of to defining shades of a given color; ac for adjusting color transparency; plot_cube for viewing colors in a 3D RGB color cube; pal_unikn for the default uni.kn color palette.

Other color functions: ac(), demopal(), grepal(), newpal(), plot_cube(), shades_of(), simcol(), usecol()

Examples

```
# See multiple color palettes:
seecol()  # default: seecol(pal = "all")

# See details of one color palette:
seecol(pal_unikn)  # see a specific color palette

# Combining colors or color palettes:
seecol(c(rev(pal_seeblau), pal_seegruen))        # combine color palettes
seecol(c(rev(pal_seeblau), "white", pal_pinky))  # combine color palettes and color names
seecol(c("black", "firebrick", "gold"))          # combine color names

# Scale a set of color palettes to a fixed width:
seecol(scale_x = TRUE)

# Using n to reduce or extend color palettes:
seecol(n =  3)  # viewing reduced ranges of all palettes
seecol(n = 12)  # viewing extended ranges of all palettes

seecol(pal_unikn, n = 5,
       main = "Reduced version of pal_unikn (n = 5)")  # reducing pal_unikn
seecol(pal_seeblau, n = 8,
       main = "Extended version of pal_seeblau (n = 8)")  # extending pal_seeblau

# Combining and extending color palettes:
seecol(c(rev(pal_seeblau), "white", pal_bordeaux), n = 17,
```

```
       main = "Diverging custom color palette (with 17 colors)")

# Defining custom color palettes:
pal_mpg <- c("#007367", "white", "#D0D3D4")  # mixing hex values and color names
names(pal_mpg) <- c("mpg green", "mpg white", "mpg grey")  # color names

pal_bdg <- usecol(c(Bordeaux, "gold"), n = 5)  # using usecol

# Viewing extended color palette:
seecol(pal_mpg, n = 9, main = "Custom color palette of the Max Planck Society")

# Comparing (and labeling) custom color palettes:
seecol(list(pal_mpg, pal_bdg, pal_unikn), scale_x = TRUE,
       pal_names = c("Max Planck", "Bordeaux-Gold", "Uni Konstanz"),
       main = "Comparing and labeling custom color palettes")

# Viewing a single palette in a 3D RGB color cube (using plot_cube()):
seecol(pal_unikn, cube = TRUE)                    # default cube view (with color names)
seecol(pal_unikn_pref, cube = TRUE, hex = TRUE)   # add HEX labels to points
seecol(pal_pinky, grid = FALSE,                   # pass arguments to plot_cube()
       cube = list(type = "b", theta = -35, line_connect = "path", shadows = "all"))

## Viewing palettes from other color packages:
# library(RColorBrewer)
# seecol(brewer.pal(name = "RdBu", n = 9))               # view color palette
# seecol(brewer.pal(name = "RdBu", n = 9), cube = TRUE)  # view in 3D color cube
# seecol(brewer.pal(name = "RdBu", n = 9), n = 15)       # extend to 15 colors
```

`Seegruen` *uni.kn color Seegruen*

Description

`Seegruen` provides the preferred color of `pal_seegruen` (as an atomic HEX character value) and is defined as `pal_seegruen[[4]]`.

Usage

```
Seegruen
```

Details

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_seegruen for the corresponding color palette; pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Bordeaux, Grau, Karpfenblau, Peach, Petrol, Pinky, Seeblau, Signal

Examples

```
Seegruen  # HEX character "#0A9086" (as value)
all.equal(Seegruen, pal_seegruen[[4]])  # TRUE (same HEX values)

seecol(Seegruen)  # view color and details
```

---

shades_of *Get n shades of a color*

Description

shades_of returns a vector of n colors that are shades of a color gradient ranging from an initial color col_1 to a final color col_n.

Usage

```
shades_of(n = 5, col_1 = "black", col_n = "white", alpha = NA)
```

Arguments

n	Number of desired colors. Default: n = 5.
col_1	Initial color. Default: col_1 = "black".
col_n	Final (n-th) color. Default: col_n = "white".
alpha	A factor modifying the opacity alpha (as alpha.f in adjustcolor) to a value in [0, 1]. Default: alpha = NA (i.e., no modification of opacity).

Details

By default, the color gradient returned contains n = 5 colors that range from the initial color col_1 = "black" to the final color col_n = "white". Specifying different values for n and the initial or final colors yields different color ranges.

shades_of is mostly a wrapper for a special usecol command. However, usecol allows defining more complex color gradients (e.g., by specifying more than two colors).

See Also

seecol for viewing and comparing color palettes; usecol for using color palettes; simcol for finding similar colors; newpal for defining new color palettes; grepal for finding named colors; ac for adjusting color transparency.

Other color functions: ac(), demopal(), grepal(), newpal(), plot_cube(), seecol(), simcol(), usecol()

Examples

```
grey50 <- shades_of(50, col_1 = 'grey1')
seecol(grey50, main = "50 shades of grey1")

blue_black <- shades_of(5, Seeblau, col_n = "black")
seecol(blue_black, main = "5 shades from Seeblau to black")

wine_white <- shades_of(6, Bordeaux, alpha = 1/2)
seecol(wine_white, col_brd = "black", lwd_brd = .5,
       main = "Shades of semi-transparent Bordeaux")
```

Signal *uni.kn color Signal or alert*

Description

Signal provides the alert color of pal_signal (as an atomic HEX character value) and is defined as pal_signal[2].

Usage

```
Signal
```

Details

The official specification of pal_signal does not identify a preferred color. We provide Signal as a dedicated color as it is suited for creating color gradients (see usecol).

See https://www.uni-konstanz.de/en/university/news-and-media/create-online-and-print-media/corporate-design/ for details.

See Also

pal_signal for the corresponding color palette; pal_unikn for the unikn default color palette with all 5 colors of pal_seeblau; pal_unikn_pref for a uni.kn color palette with all preferred colors; seecol for viewing and comparing color palettes; usecol for using color palettes.

Other preferred colors: Bordeaux, Grau, Karpfenblau, Peach, Petrol, Pinky, Seeblau, Seegruen

Examples

```
Signal  # HEX character "#EFDC60" (as value)
all.equal(Signal, pal_signal[[2]])  # TRUE (same HEX values)

seecol(Signal)  # view color and details
```

`simcol` *Find similar colors*

Description

`simcol` finds and shows colors from a palette of color candidates `col_candidates` that are similar to some target color `col_target`.

Usage

```
simcol(
  col_target,
  col_candidates = colors(),
  tol = c(25, 50, 75),
  distinct = TRUE,
  plot = TRUE
)
```

Arguments

`col_target`	A (required) target color.
`col_candidates`	A palette of color candidates to be considered. Default: `col_candidates = colors()`.
`tol`	Numeric tolerance value(s) (either 1 or 3 numeric values, in the RGB range from 0 to 255). Values are considered in the order of the RGB value rank in `col_target`. Default: `tol = c(25, 50, 75)`.
`distinct`	Boolean: Return only visually distinct colors? Default: `distinct = TRUE` (i.e., remove visual duplicates).
`plot`	Boolean: Plot the output (using `seecol`)? Default: `plot = TRUE`.

Details

`simcol` returns a vector of the (named) colors or color values in `col_candidates` (set to `colors()` of **grDevices** per default) that are similar to the specified target color `col_target`.

If `plot = TRUE`, `simcol` also visualizes the detected colors (by passing its result to `seecol`, as a side-effect).

Color similarity is defined in terms of the distance between colors' RGB values, which must be within the numeric tolerance threshold(s) specified by `tol` (with `0 <= tol <= 255`). Higher `tol` values correspond to more permissive similarity judgments.

If `tol` is a scalar, the values of all three RGB dimensions of `col_candidates` must be within the corresponding values of `col_target` to be judged as 'similar'. If `tol` contains three values, the three RGB dimension are compared in order of the dimensions' rank in `col_target` (i.e., the primary dimension must be within `tol[1]`, etc.). Thus, providing three `tol` values allows for more fine-grained similarity matching.

Value

A named vector of colors or color values.

See Also

`seecol` for plotting/seeing color palettes; `usecol` for using color palettes; `newpal` for defining new color palettes; `grepal` for finding named colors; `shades_of` to defining shades of a given color; `ac` for adjusting color transparency.

Other color functions: `ac()`, `demopal()`, `grepal()`, `newpal()`, `plot_cube()`, `seecol()`, `shades_of()`, `usecol()`

Examples

```
# Basic uses:
simcol(col_target = "red")
simcol("tan", tol = 15)
simcol(Seeblau, tol = c(20, 30, 40))
simcol("blue", col_candidates = pal_unikn_pref, tol = 120)

# Fine-tuning the range of color matching:
simcol(Seeblau, tol = 30)  # = simcol(Seeblau, tol = c(30, 30, 30))
simcol(Seeblau, tol = c(20, 20, 80))

# Increasing tolerance widens range:
simcol("grey", c("white", "grey", "black"), tol = 255, distinct = FALSE, plot = FALSE)
```

---

slide *Plot a slide (or frame)*

Description

`slide` plots an empty slide (or frame) as a colored rectangle.

Usage

```
slide(col = NA, dim = c(4/3, 1), border = grey(0.33, 1), lwd = 1.5)
```

Arguments

col	The color to fill the slide with (i.e., its background color). Default: col = NA (i.e., system default for transparency).
dim	The x- and y-dimensions of the slide. Default: dim = c(4/3, 1) (i.e., unit height, 4/3 wider than high).
border	The color of the slide's border. Setting border = NA hides border. Default: border = grey(.33, 1).
lwd	The line width of the slide's border. Setting lwd = 0 or lwd = NA removes border. Default: lwd = 1.5.

See Also

heading, line, or mark to add text to a slide; xbox to plot a box.

Other plot functions: theme_bwkn(), theme_grau(), theme_minikn(), theme_unikn(), xbox()

Examples

```
slide()  # default slide (or frame)
slide(lwd = NA)  # borderless slide

# Dimensions:
slide(dim = c(18, 9))   # larger and 2:1 dimensions
slide(dim = c(1/3, 1))  # smaller and 1:3 dimensions

# Formatting:
slide(col = pal_seeblau[[1]], border = pal_seeblau[[5]], lwd = 2)
```

---

theme_bwkn *An alternative bw theme (for ggplot2)*

Description

theme_bwkn provides an alternative bw theme for use in **ggplot2** and **unikn** contexts.

Usage

```
theme_bwkn(
  col_title = grey(0.1, 1),
  col_line = grey(0.4, 1),
  base_size = 12,
  base_family = "sans",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis_ticks_mm = 2
)
```

Arguments

col_title	Color of text elements used for plot title, axis titles, and legend (optional). Default: col_title = grey(.10, 1) (i.e., dark grey). Consider using col_title = unikn::pal_seeblau[[4]].
col_line	Color of line elements used for axis lines, text and ticks, plot caption and subtitle, and panel border. Default: col_line = grey(.40, 1) (i.e., medium grey).
base_size	Base font size (optional, numeric). Default: base_size = 12.
base_family	Base font family (optional, character). Default: base_family = "sans". Options include "mono", "sans" (default), and "serif".
base_line_size	Base line size (optional, numeric). Default: base_line_size = base_size/22.
base_rect_size	Base rectangle size (optional, numeric). Default: base_rect_size = base_size/22.
axis_ticks_mm	Length of axis ticks (in mm). Default: axis_ticks_mm = 2.

Details

theme_bwkn is light and no-nonsense, and based on theme_bw.

This theme works well for most visualizations that require grid and plot panel boundaries.

See Also

theme_unikn for default theme; theme_grau for a grey theme; theme_minikn for a minimal theme.

Other plot functions: slide(), theme_grau(), theme_minikn(), theme_unikn(), xbox()

Examples

```
# Plotting iris dataset (using ggplot2, theme_grau, and unikn colors):

library('ggplot2')  # theme_bwkn requires ggplot2

ggplot(datasets::iris) +
 geom_jitter(aes(x = Sepal.Length, y = Sepal.Width, color = Species), size = 3, alpha = 2/3) +
  facet_wrap(~Species) +
  scale_color_manual(values = usecol(pal = c(Pinky, Seeblau, Seegruen))) +
  labs(tag = "C",
       title = "Iris sepals by species (using theme_bwkn)",
       caption = "Data from datasets::iris") +
  coord_fixed(ratio = 3/2) +
  theme_bwkn()
```

theme_grau *An alternative grey theme (for ggplot2)*

Description

theme_grau provides an alternative grey theme for use in **ggplot2** and **unikn** contexts.

Usage

```
theme_grau(
  col_title = grey(0, 1),
  col_line = pal_grau[[3]],
  base_size = 11,
  base_family = "sans",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis_ticks_mm = 2
)
```

Arguments

col_title	Color of title elements (optional). Default: col_title = grey(0, 1) (i.e., "black"). Consider using col_title = unikn::pal_seeblau[[4]].
col_line	Color of line elements used for axis ticks (optional). Default: col_line = pal_grau[[3]] (i.e., medium grau).
base_size	Base font size (optional, numeric). Default: base_size = 11.
base_family	Base font family (optional, character). Default: base_family = "sans". Options include "mono", "sans" (default), and "serif".
base_line_size	Base line size (optional, numeric). Default: base_line_size = base_size/22.
base_rect_size	Base rectangle size (optional, numeric). Default: base_rect_size = base_size/22.
axis_ticks_mm	Length of axis ticks (in mm). Default: axis_ticks_mm = 2.

Details

theme_grau is no-nonsense, but fills panel backgrounds in "grau" (specifically, pal_seeggrau[[1]]).

This theme works well for dark colors and bright color accents, but is of limited use with transparent colors.

See Also

theme_unikn for default theme; theme_bwkn for a bw theme; theme_minikn for a minimal theme.

Other plot functions: slide(), theme_bwkn(), theme_minikn(), theme_unikn(), xbox()

Examples

```
# Plotting iris dataset (using ggplot2, theme_grau, and unikn colors):

library('ggplot2')  # theme_grau requires ggplot2

ggplot(datasets::iris) +
 geom_jitter(aes(x = Sepal.Length, y = Sepal.Width, color = Species), size = 3, alpha = 2/3) +
  facet_wrap(~Species) +
  scale_color_manual(values = usecol(pal = c(Pinky, Seeblau, Seegruen))) +
  labs(tag = "B",
       title = "Iris sepals by species (using theme_grau)",
       caption = "Data from datasets::iris") +
  coord_fixed(ratio = 3/2) +
  theme_grau()
```

---

theme_minikn *An alternative minimal theme (for ggplot2)*

Description

theme_minikn provides an alternative minimal theme for use in **ggplot2** and **unikn** contexts.

Usage

```
theme_minikn(
  col_title = grey(0.1, 1),
  col_line = grey(0.4, 1),
  base_size = 12,
  base_family = "sans",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis_ticks_mm = 1
)
```

Arguments

col_title	Color of text elements used for plot, axis and legend titles, as well as for strip titles. Default: col_title = grey(.10, 1) (i.e., dark grey). Consider using col_title = unikn::pal_seeblau[[4]].
col_line	Color of line elements used for axis lines, text and ticks, and for plot caption and subtitle. Default: col_line = grey(.40, 1) (i.e., medium grey).
base_size	Base font size (optional, numeric). Default: base_size = 12.

base_family	Base font family (optional, character). Default: base_family = "sans". Options include "mono", "sans" (default), and "serif".
base_line_size	Base line size (optional, numeric). Default: base_line_size = base_size/22.
base_rect_size	Base rectangle size (optional, numeric). Default: base_rect_size = base_size/22.
axis_ticks_mm	Length of axis ticks (in mm). Default: axis_ticks_mm = 1.

Details

theme_minikn is a minimal and no-nonsense, and based on theme_minimal.

This theme works well for most visualizations that do not require grid lines or panel boundaries.

See Also

theme_unikn for default theme; theme_grau for a grey theme; theme_bwkn for a bw theme.

Other plot functions: slide(), theme_bwkn(), theme_grau(), theme_unikn(), xbox()

Examples

```
# Plotting iris dataset (using ggplot2, theme_grau, and unikn colors):

library('ggplot2')  # theme_minikn requires ggplot2

ggplot(datasets::iris) +
 geom_jitter(aes(x = Sepal.Length, y = Sepal.Width, color = Species), size = 3, alpha = 2/3) +
  facet_wrap(~Species) +
  scale_color_manual(values = usecol(pal = c(Pinky, Seeblau, Seegruen))) +
  labs(tag = "D",
       title = "Iris sepals by species (using theme_minikn)",
       caption = "Data from datasets::iris") +
  coord_fixed(ratio = 3/2) +
  theme_minikn()
```

---

theme_unikn *Basic unikn theme (for ggplot2)*

Description

theme_unikn provides a basic **unikn** theme to use in **ggplot2** commands.

Usage

```
theme_unikn(
  col_title = pal_seeblau[[4]],
  col_line = grey(0.05, 1),
  base_size = 11,
  base_family = "sans",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis_ticks_mm = 2
)
```

Arguments

col_title	Color of the plot title (optional). Default: col_title = pal_seeblau[[4]]. Consider using col_title = "black" when data does not use Seeblau colors.
col_line	Color of line elements used for axis lines and ticks, strip and panel border. Default: col_line = grey(.05, 1) (i.e., dark grey).
base_size	Base font size (optional, numeric). Default: base_size = 11.
base_family	Base font family (optional, character). Default: base_family = "sans". Options include "mono", "sans" (default), and "serif".
base_line_size	Base line size (optional, numeric). Default: base_line_size = base_size/22.
base_rect_size	Base rectangle size (optional, numeric). Default: base_rect_size = base_size/22.
axis_ticks_mm	Length of axis ticks (in mm). Default: axis_ticks_mm = 2.

Details

The theme is lightweight and no-nonsense, but somewhat opinionated (e.g., in using mostly grey scales to allow emphasizing data points with color accents).

See Also

theme_grau for a grey theme; theme_bwkn for a bw theme; theme_minikn for a minimal theme.

Other plot functions: slide(), theme_bwkn(), theme_grau(), theme_minikn(), xbox()

Examples

```
# Plotting iris dataset (using ggplot2, theme_unikn, and unikn colors):

library('ggplot2')  # theme_unikn requires ggplot2

ggplot(datasets::iris) +
 geom_jitter(aes(x = Petal.Length, y = Petal.Width, color = Species), size = 3, alpha = 2/3) +
  facet_wrap(~Species) +
  scale_color_manual(values = usecol(pal = c(Pinky, Seeblau, Seegruen))) +
  labs(tag = "A", title = "Iris petals by species (using theme_unikn)",
       caption = "Data from datasets::iris") +
  theme_unikn()
```

uline *Plot underlined text elements*

Description

uline plots 1 or more text strings (provided as a character vector labels) to an (existing or new) plot and places a colored line underneath each label (to underline it).

Usage

```
uline(
  labels,
  x = 0,
  y = 0.55,
  x_layout = NA,
  y_layout = "even",
  col = "black",
  col_bg = Seeblau,
  cex = 1.5,
  font = 1,
  new_plot = "none"
)
```

Arguments

labels	A character vector specifying the text labels to be plotted.
x	A numeric vector of x-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: x = 0.
y	A numeric vector of y-coordinates at which the text labels in labels should be written. If the lengths of x and y differ, the shorter one is recycled. Default: y = .55.
x_layout	An optional numeric vector or character string to control the horizontal positions of labels. Numeric values are interpreted as increments to values of x and recycled (to enable stepwise or alternating patterns). 3 character string options are: "center" (i.e., center wrt. first label or plot center), "left" (i.e., left wrt. first label or plot center), "right" (i.e., right wrt. first label or plot center). Default: x_layout = NA (i.e., using values of x).
y_layout	A numeric value or character string to control the vertical positions of labels. Numeric values are interpreted as increments to values of y[1] and recycled (to enable stepwise or alternating patterns). 2 character string options are: "even" (i.e., even distribution of labels across available y-space) and "flush" (i.e., no space between adjacent labels, i.e., y_layout = 0). Default: y_layout = "even".

col	The color(s) of the text label(s). Default: col_lbl = "black".
col_bg	The color(s) of the line (under the text labels of labels). Default: col_bg = Seeblau.
cex	Numeric character expansion factor(s), multiplied by par("cex") to yield the character size(s). Default: cex = 1.5.
font	The font type(s) to be used. Default: font = 1 (i.e., plain text).
new_plot	Boolean: Should a new plot be generated? Set to "blank" or "slide" to create a new plot. Default: new_plot = "none" (i.e., add to an existing plot).

Details

The positions of the text elements in labels can be specified by providing their coordinates (as x and y arguments) or by providing an initial position and an y_layout (see below).

Text formatting parameters (like col, col_bg, cex, font) are recycled to match length(labels).

uline uses the base graphics system graphics::.

See Also

slide and xbox to create simple plots (without text).

Other text functions: mark(), post(), url_unikn()

Examples

```
uline(labels = "This is a test.", new_plot = "blank")  # create a new blank plot
uline(labels = "More testing here...", y = .33, col_bg = pal_pinky[[2]])  # add to plot

# 2 basic cases:
# (a) Underline text on an existing plot:
plot(x = 0, y = 0, type = "n", xlim = c(0, 1), ylim = c(0, 1), xlab = "", ylab = "")
uline(x = 0, y = .8, labels = "Underline text (on an existing plot)")  # add to plot

# (b) Underline text on a new plot:
uline(x = .02, y = .80, labels = "Underline text (on a new plot)",
     new_plot = "slide")  # create a new plot

# Example:
lbl_line <- c("This is neat, true, and terribly important.")
uline(labels = lbl_line, new_plot = "blank")  # create a new plot
uline(labels = "(which is why we underline it).", y = .40, cex = 1.2)  # add to plot
```

unikn.guide *Open the unikn package guides*

Description

Open the unikn package guides

Usage

```
unikn.guide()
```

url_unikn url_unikn *formats an URL the uni.kn way*

Description

url_unikn removes various patterns (e.g., "http", "https", "://", "www.") from the front of a given URL and returns the remaining character string with an n-dash (Unicode \u2013) prefix, rather than the former figure dash (\u2012) prefix (as the latter created issues on Fedora Linux systems).

Usage

```
url_unikn(url = "https://www.uni-konstanz.de/")
```

Arguments

url	The url to be written (as copied from a web browser).

See Also

xbox to create a new xbox (without text).

Other text functions: mark(), post(), uline()

Examples

```
url_unikn("https://www.uni-konstanz.de/")
```

usecol *Use a color or color palette*

Description

usecol allows using a color or color palette pal (e.g., for plotting).

Usage

```
usecol(
  pal = pal_unikn,
  n = "all",
  alpha = NA,
  distinct = FALSE,
  use_names = FALSE,
  use_col_ramp = FALSE
)
```

Arguments

pal	A color palette (as a vector of colors or color palettes). Default: pal = pal_unikn.
n	An integer value specifying the desired number of colors from the palette. Default: n = "all" (i.e., use all colors of a color palette). For the palettes defined by **unikn**, n is set to a pre-defined selection of colors if the desired number of colors is smaller than the available number. For all other palettes and values of n larger than length(pal), n compresses or extends the palette using colorRampPalette.
alpha	A factor modifying the opacity alpha (as alpha.f in adjustcolor) to a value in [0, 1]. Default: alpha = NA (i.e., no modification of opacity).
distinct	Boolean: Return only visually distinct colors? Default: distinct = FALSE (i.e., include duplicate colors).
use_names	A logical value indicating whether colors should be returned as a named vector. Default: use_names = FALSE, for compatibility with ggplot.
use_col_ramp	A logical value specifying whether the default of using pre-selected colors should be overridden and colorRampPalette should be used to process n. Default: use_col_ramp = FALSE.

Details

usecol also allows modifying and combining color palettes in various ways.

Value

A (named) vector of colors (of type character).

See Also

seecol for viewing and comparing color palettes; simcol for finding similar colors; newpal for defining new color palettes; grepal for finding named colors; shades_of to defining shades of a given color; ac for adjusting color transparency; pal_unikn for the default uni.kn color palette.

Other color functions: ac(), demopal(), grepal(), newpal(), plot_cube(), seecol(), shades_of(), simcol()

Examples

```
usecol(pal = pal_unikn, n = "all")  # default color palette
usecol(pal = pal_unikn, n =  4)     # selecting n dedicated colors
usecol(pal = pal_unikn, n = 20)     # extending color palette

# Mixing a new color palette:
pal_1 <- usecol(pal = c(rev(pal_seeblau), "white", pal_pinky))
seecol(pal_1)

# Mixing and extending a color palette:
pal_2 <- usecol(pal = c(rev(pal_seegruen), "white", pal_bordeaux), n = 20)
seecol(pal_2)

# Defining and using a custom color palette:
pal_princeton_1 <- c("#E77500", "white", "black")
names(pal_princeton_1) <- c("orange_w", "white", "black")

pal_3 <- usecol(pal_princeton_1, n = 7)
seecol(pal_3)

# Removing visual duplicates:
usecol(c("black", "#000000", "gray", "grey", "red", "red1"), distinct = TRUE)
seecol(usecol(c(pal_unikn, pal_seeblau), distinct = TRUE), title = "Using distinct colors")
```

---

xbox *Plot a box (with x)*

Description

xbox plots a box with a cross (x) in its top-right corner.

Usage

```
xbox(col = Seeblau, dim = c(1, 1), use_x = TRUE)
```

Arguments

col	The color to fill the box with (i.e., its background color). Default: col = Seeblau.
dim	The x- and y-dimensions of the box (as numeric). Default: dim = c(1, 1) (i.e., a unit square).
use_x	Plot a cross in upper right corner (as logical)? Default: use_x = TRUE.

Details

The cross (x) appears rectangular when viewing the plot at the correct aspect ratio (as defined by `dim`).

See Also

`post` to add text to an xbox; `slide` to plot a new slide (or frame).

Other plot functions: `slide()`, `theme_bwkn()`, `theme_grau()`, `theme_minikn()`, `theme_unikn()`

Examples

```
xbox()  # default box

# Options:
xbox(col = Bordeaux)
xbox(dim = c(2, 1)) # 2:1 dimension (wider than high)
```

Index

∗ **color functions**
ac, 3
demopal, 5
grepal, 7
newpal, 13
plot_cube, 32
seecol, 42
shades_of, 46
simcol, 48
usecol, 59
∗ **color palettes**
pal_bordeaux, 16
pal_grau, 16
pal_karpfenblau, 17
pal_peach, 18
pal_petrol, 19
pal_pinky, 20
pal_seeblau, 20
pal_seegruen, 21
pal_signal, 22
pal_unikn, 23
pal_unikn_dark, 24
pal_unikn_light, 25
pal_unikn_pair, 26
pal_unikn_ppt, 27
pal_unikn_pref, 28
pal_unikn_web, 29
∗ **plot functions**
slide, 49
theme_bwkn, 50
theme_grau, 52
theme_minikn, 53
theme_unikn, 54
xbox, 60
∗ **preferred colors**
Bordeaux, 4
Grau, 7
Karpfenblau, 11
Peach, 30
Petrol, 30
Pinky, 31
Seeblau, 41
Seegruen, 45
Signal, 47
∗ **text functions**
mark, 11
post, 39
uline, 56
url_unikn, 58
∗ **utility functions**
get_col_names, 6

ac, 3, *6*, *8*, *14*, *44*, *47*, *49*, *60*
ac(), *6*, *8*, *14*, *36*, *44*, *47*, *49*, *60*
adjustcolor, *3*, *43*, *46*, *59*

Bordeaux, 4, *7*, *11*, *16*, *30, 31*, *41*, *46, 47*

col2rgb, *36*
colorRampPalette, *59*

defcol *(*newpal*)*, 13
defpal *(*newpal*)*, 13
democol *(*demopal*)*, 5
demofun *(*demopal*)*, 5
demopal, 5
demopal(), *3*, *8*, *14*, *36*, *44*, *47*, *49*, *60*
demoplot *(*demopal*)*, 5

get_col_names, 6
Grau, *4*, 7, *11*, *16*, *30, 31*, *41*, *46, 47*
grepal, *3*, 7, *14*, *44*, *47*, *49*, *60*
grepal(), *3*, *6*, *14*, *36*, *44*, *47*, *49*, *60*

heading, 9, *50*

Karpfenblau, *4*, *7*, 11, *17*, *30, 31*, *41*, *46, 47*

line, *50*

mark, 11, *50*

mark(), *40*, *57, 58*

newpal, *3*, *8*, 13, *36*, *44*, *47*, *49*, *60*
newpal(), *3*, *6*, *8*, *36*, *44*, *47*, *49*, *60*

pal_bordeaux, *4*, 16, *17–29*
pal_grau, *7*, *16*, 16, *18–29*
pal_karpfenblau, *11*, *16, 17*, 17, *18–29*
pal_peach, *16–18*, 18, *19–30*
pal_petrol, *16–18*, 19, *20–31*
pal_pinky, *16–19*, 20, *21–29*, *31*
pal_seeblau, *4*, *7*, *10*, *16–20*, 20, *21–31*, *41*, *46, 47*
pal_seegruen, *16–21*, 21, *23–29*, *45, 46*
pal_signal, *16–22*, 22, *23–29*, *47*
pal_unikn, *4*, *6–8*, *11*, *14*, *16–23*, 23, *24–31*, *41*, *44*, *46, 47*, *59, 60*
pal_unikn_dark, *16–23*, 24, *25–29*
pal_unikn_light, *16–24*, 25, *26–29*
pal_unikn_pair, *16–25*, 26, *27–29*
pal_unikn_ppt, *16–26*, 27, *28, 29*
pal_unikn_pref, *4*, *7*, *11*, *16–27*, 28, *29–31*, *41*, *46, 47*
pal_unikn_web, *16–28*, 29
Peach, *4*, *7*, *11*, *18*, 30, *31*, *41*, *46, 47*
persp, *36*
Petrol, *4*, *7*, *11*, *19*, *30*, 30, *31*, *41*, *46, 47*
Pinky, *4*, *7*, *11*, *20*, *30, 31*, 31, *41*, *46, 47*
plot_cube, 32, *43, 44*
plot_cube(), *3*, *6*, *8*, *14*, *44*, *47*, *49*, *60*
points, *33*
post, 39, *61*
post(), *13*, *57, 58*

Seeblau, *4*, *7*, *11*, *20*, *30, 31*, 41, *46, 47*
seecol, *3, 4*, *7, 8*, *11*, *14*, *16–31*, *34*, *36*, *41*, 42, *46–49*, *60*
seecol(), *3*, *6*, *8*, *14*, *36*, *47*, *49*, *60*
Seegruen, *4*, *7*, *11*, *21*, *30, 31*, *41*, 45, *47*
seepal, *6*
seepal *(*seecol*)*, 42
shades_of, *6*, *8*, *14*, *44*, 46, *49*, *60*
shades_of(), *3*, *6*, *8*, *14*, *36*, *44*, *49*, *60*
Signal, *4*, *7*, *11*, *28*, *30, 31*, *41*, *46*, 47
simcol, *3*, *8*, *14*, *44*, *47*, 48, *60*
simcol(), *3*, *6*, *8*, *14*, *36*, *44*, *47*, *60*
slide, *10*, *13*, 49, *57*, *61*
slide(), *51, 52*, *54, 55*, *61*

theme_bwkn, 50, *52*, *54, 55*
theme_bwkn(), *50*, *52*, *54, 55*, *61*
theme_grau, *51*, 52, *54, 55*
theme_grau(), *50, 51*, *54, 55*, *61*
theme_minikn, *51, 52*, 53, *55*
theme_minikn(), *50–52*, *55*, *61*
theme_unikn, *51, 52*, *54*, 54
theme_unikn(), *50–52*, *54*, *61*
trans3d, *36*

uline, 56
uline(), *13*, *40*, *58*
unikn.guide, 58
url_unikn, 58
url_unikn(), *13*, *40*, *57*
usecol, *3, 4*, *6–8*, *11*, *14*, *16–31*, *36*, *41*, *44*, *46, 47*, *49*, 59
usecol(), *3*, *6*, *8*, *14*, *36*, *44*, *47*, *49*

xbox, *10*, *13*, *39, 40*, *50*, *57, 58*, 60
xbox(), *50–52*, *54, 55*