# Package: profiplots (via r-universe)

October 13, 2024

**Title** Profinit Plotting Theme

**Version** 0.2.3

**Description** Help unify visual output of R analyses in the Profinit EU company. So far, there are color and fill scales for 'ggplot2', plotting theme for 'ggplot2', color palettes and utils to make the tools default choices.

**License** MIT + file LICENSE

**URL** https://profinit.eu

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Imports** ggplot2 (>= 3.2.0)

**Suggests** tidyverse, qpdf, scales, purrr, forcats, dplyr, knitr, rmarkdown, ggalluvial, ggrepel, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Dominik Matula [aut, cre], Profinit EU, Ltd. [cph, fnd]

**Maintainer** Dominik Matula <dominik.matula@profinit.eu>

**Date/Publication** 2023-11-16 11:20:07 UTC

**Repository** https://matulad.r-universe.dev

**RemoteUrl** https://github.com/cran/profiplots

**RemoteRef** HEAD

**RemoteSha** 8756788e21b0274102b0cfdc7806dd0f83c18d2a

## Contents

---

`profinit_cols` *Profinit colors*

---

## Description

This function provides an access to standardized Profinit colors hex codes.

## Usage

```
profinit_cols(..., named = FALSE)
```

## Arguments

| | |
|---|---|
| `...` | Either character name(s) or order of Profinit colors. Leave empty for the full color set. |
| `named` | Flag whether to produce named vector of profinit colours. Defautls to `FALSE`. |

## Value

Named character vector the same length as input params.

## Examples

```
# to get Profinit red hexcode
profinit_cols("red")

# to get Profinit red and gray hexcodes
profinit_cols("red", "grey")

# to get the first Profinit color
profinit_cols(1)

# to get all Profinit colors
profinit_cols()
scales::show_col(profinit_cols())
```

profinit_pal *Profinit color palettes*

## Description

Returns function that interpolates (if `exact=FALSE`) chosen Profinit color palette. That is, the resulting function is able to give you desired number of colors (hexes) within given Profinit color palette.

## Usage

```
profinit_pal(pal_name = "blue-red", reverse = FALSE, exact = NULL, ...)
```

## Arguments

| | |
|---|---|
| `pal_name` | Character name of chosen Profinit palette. See `profinit_pal.pals()` for available palettes. |
| `reverse` | Boolean indicating whether the palette should be reversed |
| `exact` | Indicates whether the color scale is supposed to be followed exactly. Be ware you may run out of colors. Defaults to `TRUE` for discrete palette names, `FALSE` otherwise. |
| `...` | Additional arguments to pass to `grDevices::colorRampPalette()` |

## Details

See the examples for more details about its usecase.

## Value

A function providing desired number of colors (hex codes) from the specified palette.

## Examples

```
# Example 1 - get colors from the 'discrete' Profinit palette
discrete_cols <- profinit_pal("discrete")
discrete_cols(3)
scales::show_col(discrete_cols(3))

# .. The number of colors is limited in this palette. Once you reach the
# limit, Profinit's grey is used to fill the missings:
scales::show_col(discrete_cols(10))

# .. You can bypass this via either enabling the interpolation (defaults
# to `FALSE` for `discrete`-like color palettes and `TRUE` for other palettes)
# or try your luck with full set of Profinit colors.
discrete_cols_int <- profinit_pal("discrete", exact = FALSE)
scales::show_col(discrete_cols_int(10))
```

```
# Example 2 - get colors from other Profinit palette
profinit_reds_cols <- profinit_pal("reds")
profinit_reds_cols(3)
scales::show_col(profinit_reds_cols(3))
# Again, we can interpolate
profinit_reds_cols(15)
scales::show_col(profinit_reds_cols(15))

# Example 3 - using palette in baseR plots
plot(mtcars$mpg, mtcars$qsec, col=profinit_pal("discrete")(5), pch=16)
```

---

## profinit_pal.pals *List of available Profinit palettes*

---

### Description

Note: Function name has been chosen to match the `grDevices::palette.pals()`.

### Usage

```
profinit_pal.pals()
```

### Value

A character vector of all available palettes in this package.

### Examples

```
profinit_pal.pals()

# Now you are able to use this information, e.g.:
reds_palette <- profinit_pal("reds")
reds_palette(4)
scales::show_col(reds_palette(4))
```

---

## scale_color_profinit *Profinit color scale constructor*

---

### Description

Profinit color scale constructor

`scale_color_profinit_d` - Discrete Profinit color scale (similar to `scale_color_viridis_d`). It can't be used with continuous variables. There is BrE equivalent, too: `scale_colour_profinit_d`.

`scale_color_profinit_c` - Continuous Profinit color scale (similar to `scale_color_viridis_c`). It can't be used with discrete variables. There is BrE equivalent, too: `scale_colour_profinit_c`.

## Usage

```
scale_color_profinit(
  palette = "blue-red",
  discrete = TRUE,
  reverse = FALSE,
  exact = NULL,
  na.value = NULL,
  ...
)

scale_colour_profinit(
  palette = "blue-red",
  discrete = TRUE,
  reverse = FALSE,
  exact = NULL,
  na.value = NULL,
  ...
)

scale_color_profinit_d(palette = "blue-red", ...)

scale_colour_profinit_d(palette = "blue-red", ...)

scale_color_profinit_c(palette = "blue-red", ...)

scale_colour_profinit_c(palette = "blue-red", ...)
```

## Arguments

| | |
|---|---|
| palette | Character name of palette in profinit_palettes. Use profinit_pal.pals() to list available palettes. |
| discrete | Boolean indicating whether color aesthetic is discrete or not. Defaults to TRUE. |
| reverse | Boolean indicating whether the palette should be reversed. Defaults to FALSE. |
| exact | Indicates whether the color scale is supposed to be followed exactly. Be ware you may run out of colors. Defaults to TRUE for discrete palette names, FALSE otherwise. |
| na.value | What value is going to be used for missings. Defaults to profinit's grey with some transparency added. |
| ... | Additional arguments passed to ggplot2::discrete_scale() or ggplot2::scale_color_gradientn( used respectively when discrete is set to TRUE or FALSE |

## Value

Ggplot2 color scale constructor based on Profinit color palette.

## Examples

```
library(ggplot2)

iris_plt <- ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  geom_point() +
  theme_profinit()

iris_plt + scale_color_profinit()

# Now, let's use another Profinit palette:
# (see `profinit_pal.pals()` for all the options)
iris_plt + scale_color_profinit(palette = "reds-dark")
```

---

`scale_fill_profinit` *Fill scale constructor for Profinit colors.*

---

## Description

Fill scale constructor for Profinit colors.

`scale_fill_profinit_d` - Discrete Profinit fill scale (similar to `scale_fill_viridis_d`). It can't be used with continuous variables.

`scale_fill_profinit_c` - Continuous Profinit fill scale (similar to `scale_fill_viridis_c`). It can't be used with discrete variables.

## Usage

```
scale_fill_profinit(
  palette = "blue-red",
  discrete = TRUE,
  reverse = FALSE,
  exact = NULL,
  ...
)

scale_fill_profinit_d(palette = "blue-red", ...)

scale_fill_profinit_c(palette = "blue-red", ...)
```

## Arguments

| | |
|---|---|
| `palette` | Character name of palette in `profinit_palettes` |
| `discrete` | Boolean, indicating whether color aesthetic is discrete or not |
| `reverse` | Boolean, indicating whether the palette should be reversed |

| | |
|---|---|
| exact | Indicates whether the color scale is supposed to be followed exactly. Be ware, you may run out of colors. Defaults to TRUE for discrete palette names, FALSE otherwise. |
| ... | Additional arguments passed to discrete_scale() or scale_fill_gradientn(), used respectively when discrete is TRUE or FALSE |

## Value

Ggplot2 fill scale constructor based on Profinit color palette.

## Examples

```
library(ggplot2)
plt <- ggplot(ggplot2::diamonds, ggplot2::aes(x = clarity, y = price, fill = clarity)) +
  geom_boxplot() +
  theme_profinit()

plt + scale_fill_profinit()

# Discrete scale, follow exact color codes (default for the `discrete` palette).
# ! You may run out of colors - see the example below.
# Use either exact = FALSE (next example) or `discrete-full` palette.
plt + scale_fill_profinit("discrete")

# Now, the colors are approximated
plt + scale_fill_profinit("discrete", exact = FALSE)
```

---

# set_theme *Change default plotting behavior*

---

## Description

This function alternates the default graphics behavior (both, base R graphics and ggplot2 graphics) to follow Profinit visual guide. Be were, the recent setup is not stored anywhere before changing the values.

The function sets to defaults color and fill scales in ggplot2, R4 color palette (base graphics) and greyish color theme in ggplot2.

## Usage

```
set_theme(pal_name = "blue-red", pal_name_discrete = "discrete", exact = NULL)

unset_theme()
```

## Arguments

| | |
|---|---|
| pal_name | Profinit palette name to be used. Defaults to blue-red. |
| pal_name_discrete | (Optional). Palette name for discrete color scales to be used by ggplot. Defaults to discrete. |
| exact | Indicates whether discrete values will be treated exactly as present in the palette OR whether and interpolation can take a place. |

## Value

The function returns nothing, it edits default behavior. Recent default configs are stored in a backup option.

The function returns nothing, it just changes default behaviour.

## Functions

- set_theme(): Sets default plotting theme to Profinit.
- unset_theme(): Resets default plotting theme.

## Examples

```
# Example 1 - BaseR

# As a starting point, there is a plot i base R graphic:
sample_df <- data.frame(x=1:8, y=1:8, category=as.character(1:8), col_cont = 1:8)
barplot(sample_df$y, col=sample_df$category)

# Now, by applying hte set_theme() we can change the default behavior:
profiplots::set_theme("blue-red", "blue-red")
barplot(sample_df$y, col=sample_df$category)

# To turn the theming off, just call:
profiplots::unset_theme()

# Example 2 - GGplot
library("ggplot2")
plot_gg <- ggplot(sample_df, aes(x=x, y=y, fill=category)) + stat_identity(geom="bar")
plot_gg

# Now, let's trun it into Profinit graphics.
profiplots::set_theme("blue-red", "blue-red")
plot_gg

# To remove the Profinit theme from defaults, just call:
profiplots::unset_theme()
```

`theme_profinit` *Profinit ggplot theme*

## Description

The current version of the theme is based on `ggplot2::theme_minimal()` not tweaking any of its settings. Later on, the theme will be customised to better fill our needs and preferences.

## Usage

```
theme_profinit()
```

## Value

Ggplot2's theme object adjusted according to Profinit's preferences.

## See Also

scale_fill_profinit and scale_color_profinit for Profinit color scales.

## Examples

```
# default behavior
library(ggplot2)
iris_plt <- ggplot(iris, aes(x=Sepal.Length, fill=Species)) +
  geom_density(alpha=.5) +
  scale_fill_profinit()
iris_plt

# With profinit theme
iris_plt + theme_profinit()
```

# Index