Package 'ggCheysson'

November 7, 2026

Title Graphic Styles of Emile Cheysson for 'ggplot2'

Version 1.2.0

Date 2026-11-07

Description Implements for 'ggplot2' the stylistic elements (fonts, hatched patterns, color palettes) used by 'Emile Cheysson' in the 'Albums de Statistique Graphique', sometimes called the pinnacle of the Golden Age of Statistical Graphics.

Imports ggplot2, rlang (>= 1.0.0)

Suggests ggpattern, ggthemes, gridpattern, systemfonts, showtext, sysfonts, ragg, knitr, rmarkdown, Guerry, sf, tidyr, dplyr, testthat (>= 3.0.0), callr

License GPL (>= 3)

Language en-US

Encoding UTF-8

URL https://github.com/friendly/ggCheysson, https://friendly.github.io/ggCheysson/

BugReports https://github.com/friendly/ggCheysson/issues

Depends R (>= 3.5)

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

NeedsCompilation no

Author Michael Friendly [aut, cre] (ORCID: <https://orcid.org/0000-0002-3237-0941>), RJ Andrews [ctb], Tom Shanley [ctb], Kenneth Fields [ctb]

Maintainer Michael Friendly <friendly@yorku.ca>

Repository CRAN

Date/Publication 2026-11-07 12:50:02 UTC

Contents

albumImages . . . 2
cheysson_font . . . 4
cheysson_fonts . . . 4
cheysson_fonts_available . . . 5
cheysson_labels . . . 6
cheysson_name . . . 8
cheysson_pal . . . 9
cheysson_palettes . . . 10
cheysson_pattern . . . 12
cheysson_patterns . . . 13
cheysson_pattern_params . . . 14
cheysson_swatches . . . 15
list_cheysson_pals . . . 17
list_cheysson_patterns . . . 18
load_cheysson_fonts . . . 19
scale_cheysson . . . 20
scale_color_cheysson . . . 22
scale_fill_cheysson_pattern . . . 23
scale_pattern_cheysson . . . 24
show_palette . . . 26
show_palettes . . . 27
show_swatches . . . 28
theme_cheysson . . . 28
theme_cheysson_map . . . 30
theme_cheysson_minimal . . . 31

Index **33**

`albumImages` *Album Images Metadata*

Description

Metadata linking the 25 color palettes from the Albums de Statistique Graphique to their corresponding album years, plate numbers, and David Rumsey collection reference numbers.

Usage

```
albumImages
```

Format

A data frame with 25 rows and 6 variables:

adventDay Advent calendar day number (1-25) from original digitization

RumseyListNo David Rumsey Map Collection list number

Album Year of the album (1880-1906)

Qty Number of colors/pattern elements in the palette - not a plate identifier (see cheysson_palettes for the actual unique plate number, derived from `RumseyListNo`'s decimal suffix)

Type Palette type: "Sequential", "Diverging", "Grouped", or "Category"

link URL to the digitized image in the David Rumsey Map Collection

Details

This dataset provides the mapping between the original SVG pattern files (named by Advent calendar days by RJ Andrews) and the actual album years and plate numbers from the *Albums de Statistique Graphique*.

The naming convention "adventDay" comes from RJ Andrews' original digitization project where he released one palette per day during December as an Advent calendar. The package uses the Album year and plate number for more intuitive palette naming (e.g., "1880_07" instead of "dec06"). The plate number is derived from `RumseyListNo`'s decimal suffix (e.g. "12511.007" -> 7), which is the real unique per-plate identifier - not `Qty`, which is a count of colors/pattern elements in the palette and can repeat across genuinely different plates.

Source

- David Rumsey Map Collection: https://www.davidrumsey.com/
- RJ Andrews Album Colors: https://github.com/infowetrust/albumcolors
- Tom Shanley Observable: https://web.archive.org/web/20210130125506/https://observablehq.com/@tomshanley/cheysson-color-palettes

See Also

cheysson_palettes, cheysson_patterns

Examples

```
# View the dataset
head(albumImages)

# Find information about a specific album
subset(albumImages, Album == 1880)

# Count palettes by type
table(albumImages$Type)

# Get the Rumsey link for a specific palette
albumImages[albumImages$Album == 1881 & albumImages$Qty == 3, "link"]
```

cheysson_font *Get Cheysson font family name*

Description

Returns the appropriate Cheysson font family name for a given purpose.

Usage

```
cheysson_font(type = c("regular", "italic", "sans", "outline", "title"))
```

Arguments

type	Font type: "regular", "italic", "sans", "outline", or "title"

Value

Character string with font family name

Examples

```
cheysson_font("title")  # Returns "CheyssonTitle"
cheysson_font("regular")  # Returns "Cheysson"
```

cheysson_fonts *Cheysson Font Families*

Description

Metadata about the five Cheysson font families included in the package, describing their characteristics and recommended uses.

Usage

```
cheysson_fonts
```

Format

A data frame with 5 rows and 4 variables:

family Font family name (e.g., "Cheysson", "CheyssonTitle")

description Brief description of the font style

use Recommended use cases for the font

file TrueType font filename in the inst/fonts/ directory

Details

The package includes five hand-drawn font families created by Kenneth Fields (ESRI) to match the lettering style of the original Albums de Statistique Graphique:

- **Cheysson**: Regular serif font suitable for body text, axis labels, and legends
- **CheyssonItalic**: Italic variant for emphasis and annotations
- **CheyssonSansCaps**: Sans-serif capitals for axis labels and category names
- **CheyssonOutlineCaps**: Outlined capitals for decorative plot titles and headings
- **CheyssonTitle**: Decorative font for main plot titles

These fonts must be loaded before use with `load_cheysson_fonts()`. The Cheysson themes (`theme_cheysson()`, etc.) automatically select appropriate fonts for different plot elements.

Source

Font families created by Kenneth Fields (ESRI) based on the lettering style of Émile Cheysson's Albums de Statistique Graphique.

See Also

`load_cheysson_fonts()`, `cheysson_fonts_available()`, `theme_cheysson()`

Examples

```
# View font metadata
cheysson_fonts

# Get recommended uses
cheysson_fonts[, c("family", "use")]

# Find the title font
subset(cheysson_fonts, grepl("title", use, ignore.case = TRUE))
```

---

cheysson_fonts_available

Check if Cheysson fonts are loaded

Description

Checks whether Cheysson fonts have been registered and are available for use.

Usage

```
cheysson_fonts_available(method = NULL)
```

Arguments

method	Check for "systemfonts" or "showtext". If NULL (default), checks both methods.

Value

Logical indicating whether fonts are available

Examples

```
if (cheysson_fonts_available()) {
  message("Cheysson fonts are ready to use!")
}
```

---

cheysson_labels *Cheysson Palette Naming Crosswalk*

Description

A one-row-per-plate lookup table tying together every naming scheme used for these 25 palettes: the package's own cheysson_palettes/ cheysson_patterns names, RJ Andrews' original Advent calendar labels, and Tom Shanley's Observable notebook IDs. Useful whenever you have a palette identified in someone else's terms - an Advent day, a David Rumsey catalog number, or a label copied from Andrews' or Shanley's own work - and need the package name to actually use it in `scale_color_cheysson()` and friends, or vice versa (see `cheysson_name()` for a lookup function that does this translation for you).

Usage

```
cheysson_labels
```

Format

A data frame with 25 rows and 10 variables:

name The package's current palette name (e.g. "1881_22") - pass this to `scale_color_cheysson()`, `cheysson_pal()`, `cheysson_pattern()`, etc. Unique; matches `names(cheysson_palettes)` and `names(cheysson_patterns)` exactly

advent_day Advent calendar day (1-25) from RJ Andrews' original digitization project, where he released one palette per day during December

album_year Year of the album (1880-1906)

plate The plate number within the album, derived from `rumsey_no`'s decimal suffix (e.g. "12511.007" -> 7) - the real unique per-plate identifier that `name` is built from

type Palette type: "Sequential", "Diverging", "Grouped", or "Category"

qty Number of colors/pattern elements in the palette - despite the name, not a plate identifier (see `plate`)

rumsey_no David Rumsey Map Collection list number

old_name The name this palette would have received under the package's original (Album + `qty`) naming scheme, before a 2026-09-16 fix - kept for historical reference. Not unique: 4 pairs of genuinely different plates shared an `old_name`, silently overwriting one another during data extraction until the fix

andrews_label RJ Andrews' own label for this palette, in the form "Dec.DD-YYYY.PP" (Advent day - album year . plate)

shanley_id Tom Shanley's Observable notebook ID for this palette: `type` followed directly by `rumsey_no`'s digits with the dot removed (e.g. "diverging12514021")

Details

This table exists because the package's own `name` (Album + plate) is a convenient but package-internal convention - it isn't what you'll see if you're reading Andrews' or Shanley's original write-ups, or working from a Rumsey catalog reference. `andrews_label` and `shanley_id` are derived programmatically from the verified rule linking all three schemes (album year + plate, taken from `rumsey_no`'s decimal suffix), not scraped from either source - both were cross-checked against every legible label/ID in the reference images below and matched for all 24/24 checked. (Two of Shanley's own ID *strings* have isolated typos - a digit transposition and a missing trailing digit; his prose descriptions don't have these typos, and this table's `shanley_id` gives the corrected form, not the typo'd one.)

`old_name` also documents a real bug this table helped uncover: before 2026-09-16, palette names were built from `qty` (a color count) instead of `plate` (the real per-plate identifier), causing 4 collisions between genuinely different plates and silently dropping 5 of the original 25 palettes from the package. Comparing `name` to `old_name` shows exactly what changed for each plate.

Source

- David Rumsey Map Collection: https://www.davidrumsey.com/
- RJ Andrews Album Colors: https://github.com/infowetrust/albumcolors - see also `man/figures/RJ-Andrews-color-palettes.jpg` in the package sources for his own reference grid of all 25 labeled palettes
- Tom Shanley Observable: https://web.archive.org/web/20210130125506/https://observablehq.com/@tomshanley/cheysson-color-palettes - see also `man/figures/shanley-palettes.png` in the package sources for his own reference grid of all 25 labeled palettes
- `data-raw/observable/albumColors.csv` (this package) - the original Advent day / Rumsey number / album / type mapping everything else here is derived from

See Also

cheysson_palettes, cheysson_patterns, albumImages, `cheysson_name()`

Examples

```
# View the crosswalk
head(cheysson_labels)

# Find the package name for a palette described in Andrews' terms
cheysson_labels[cheysson_labels$andrews_label == "Dec.01-1883.21", "name"]

# Find the package name for a palette described in Shanley's terms
cheysson_labels[cheysson_labels$shanley_id == "grouped12511007", "name"]

# See what a palette used to be called before the 2026-09-16 naming fix
subset(cheysson_labels, name != old_name)[, c("name", "old_name")]
```

`cheysson_name` *Translate a palette label from another naming scheme to the current name*

Description

Looks up a palette by whichever label you have it in - the package's own (possibly outdated) name, RJ Andrews' Advent-calendar label, Tom Shanley's Observable notebook ID, a David Rumsey catalog number, or an advent day - and returns the current name to pass to `scale_color_cheysson()`, `cheysson_pal()`, `cheysson_pattern()`, and friends.

Usage

```
cheysson_name(
  label,
  from = c("auto", "name", "andrews", "shanley", "rumsey_no", "old_name"),
  advent_day = NULL
)
```

Arguments

`label`	Character string to look up.
`from`	Which naming scheme `label` is in: • `"auto"` (default): try, in order, the current `name`, `andrews_label`, `shanley_id`, `rumsey_no`, and finally `old_name` (see Details for why `old_name` is tried last and can be ambiguous) • `"name"`: the package's current palette name • `"andrews"`: RJ Andrews' label, e.g. "Dec.01-1883.21" • `"shanley"`: Tom Shanley's Observable ID, e.g. "diverging12514021" • `"rumsey_no"`: David Rumsey catalog number, e.g. "12514.021" • `"old_name"`: the name this palette had before the 1.1.0 naming fix - see Details, this can be ambiguous

advent_day — Optional advent day (1-25) to disambiguate when label alone matches more than one palette (only possible for from = "old_name"/"auto"), or to assert which palette you mean even when label is already unambiguous. An advent_day that contradicts what label resolves to is an error.

Details

Before ggCheysson 1.1.0, 4 different old_name values were each used by 2-3 genuinely different palettes (a naming collision bug - see cheysson_labels), so translating from old_name alone can be ambiguous. When it is, cheysson_name() returns the palette that actually shipped under that name in 1.0.1 (the one with the highest advent_day among the candidates - this is what the pre-fix extraction logic itself picked) and issues a warning() listing every candidate it did *not* return. Pass advent_day explicitly to get a different one without a warning.

Value

A character string: the current palette name.

See Also

cheysson_labels

Examples

```
# RJ Andrews' label -> current name
cheysson_name("Dec.01-1883.21")

# Tom Shanley's ID -> current name
cheysson_name("diverging12514021")

# An old (pre-1.1.0) name that's unambiguous
cheysson_name("1883_04", from = "old_name")

# An old name that collided across 2 palettes: returns the one that
# shipped under this name in 1.0.1, with a warning
cheysson_name("1880_07", from = "old_name")

# Disambiguate explicitly - no warning
cheysson_name("1880_07", from = "old_name", advent_day = 6)
```

---

cheysson_pal *Get a Cheysson color palette*

Description

Returns colors from a specified Cheysson palette. Palettes can be referenced by name (e.g., "1880_21") or by selecting a palette of a particular type.

Usage

```
cheysson_pal(palette = "1880_21", n = NULL, type = 1)
```

Arguments

palette	Name of palette (e.g., "1880_21"), or palette type ("sequential", "diverging", "grouped", "category"). If a type is specified, the first palette of that type is returned.
n	Number of colors to return. If NULL, returns all colors in the palette. If n is smaller than the palette, sequential and diverging palettes return colors spread over the whole palette, keeping both ends; other types return the first n. If n is greater than the number of colors in the palette, colors will be interpolated.
type	If palette is a type name, optionally specify which palette of that type to use (default is 1 for the first).

Value

A character vector of hex color codes

Examples

```
# Get all colors from a specific palette
cheysson_pal("1880_21")

# Get 5 colors from a palette
cheysson_pal("1880_21", n = 5)

# Get colors from first sequential palette
cheysson_pal("sequential")

# Get colors from second category palette
cheysson_pal("category", type = 2)
```

---

cheysson_palettes *Cheysson Color Palettes*

Description

Color palettes extracted from the Albums de Statistique Graphique produced under the direction of Émile Cheysson. These palettes are organized by album year and plate number.

Usage

```
cheysson_palettes
```

Format

A list of 25 color palettes, each containing:

colors Character vector of hex color codes

type Palette type: "sequential", "diverging", "grouped", or "category"

album Year of the album

plate Plate number within the album

rumsey_no David Rumsey collection reference number

dec_day Advent calendar day from original source

Details

The palettes are named using the convention `YYYY_PP` where YYYY is the album year and PP is the zero-padded plate number. For example, "1880_07" refers to plate 7 from the 1880 album.

Palette types:

- **Sequential** (7 palettes): Ordered colors for quantitative data
- **Diverging** (2 palettes): Two contrasting colors with neutral midpoint
- **Grouped** (10 palettes): Related colors for comparing groups
- **Category** (6 palettes): Distinct colors for categorical data

Palette order: sequential palettes are stored from low to high (light to dark), and diverging palettes from one extreme through the neutral middle to the other, so `reverse = TRUE` in the scales flips the direction the same way for every palette. Category and grouped palettes keep the order of RJ Andrews' swatches. (`1886_26`, typed "sequential" in the source, has two hues and no single light-to-dark order.) Some sequential palettes, such as `1881_12`, have a single color: their steps are in the hatching, see cheysson_patterns.

Source

Color patterns digitized by RJ Andrews from the David Rumsey Map Collection https://github.com/infowetrust/albumcolors

Observable implementation by Tom Shanley https://web.archive.org/web/20210130125506/https://observablehq.com/@tomshanley/cheysson-color-palettes

See Also

`cheysson_pal()`, `scale_color_cheysson()`

Examples

```
# List available palettes
names(cheysson_palettes)

# Get colors from a specific palette
cheysson_palettes$`1880_07`$colors
```

```
# Find palettes by type
sequential_pals <- Filter(function(x) x$type == "sequential", cheysson_palettes)
names(sequential_pals)
```

`cheysson_pattern` *Get Cheysson pattern specifications*

Description

Returns pattern specifications from a Cheysson palette for use with ggpattern.

Usage

```
cheysson_pattern(palette = "1881_12", n = NULL, type = 1)
```

Arguments

`palette`	Name of palette (e.g., "1881_12") or palette type ("sequential", "diverging", "grouped", "category").
`n`	Number of patterns to return. If NULL, returns all patterns. If `n` is smaller than the palette, sequential and diverging palettes return patterns spread over the whole palette, keeping both ends; other types return the first `n`. If `n` is larger, patterns are recycled.
`type`	If palette is a type name, which palette of that type to use (default 1).

Value

A list of pattern specifications suitable for ggpattern

Examples

```
# Get all patterns from a palette
cheysson_pattern("1881_12")

# Get first 3 patterns
cheysson_pattern("1881_12", n = 3)

# Get patterns from a sequential palette
cheysson_pattern("sequential")
```

`cheysson_patterns` *Cheysson Pattern Data*

Description

Pattern specifications (fills and hatching) extracted from the Albums de Statistique Graphique. These patterns combine solid colors with line hatching (stripes and crosshatching) as used in Cheysson's maps.

Usage

```
cheysson_patterns
```

Format

A list of 25 pattern palettes, each containing:

patterns List of pattern specifications with fill colors and hatching parameters

type Palette type: "sequential", "diverging", "grouped", or "category"

album Year of the album

plate Plate number within the album

rumsey_no David Rumsey collection reference number

dec_day Advent calendar day from original source

n_patterns Number of patterns in the palette

Details

Each pattern specification includes:

- **type**: "solid", "stripe", or "crosshatch"
- **fill**: Base fill color
- **pattern_fill**: Color for pattern lines
- **pattern_angle**: Angle of stripes (in degrees)
- **pattern_density**: Density of pattern lines (0-1)
- **pattern_spacing**: Spacing between pattern lines
- **pattern_linewidth**: Width of pattern lines

Patterns are stored in the same order as cheysson_palettes: sequential palettes from low to high (lightest hatching to solid), diverging palettes end to end with the solid fills at the extremes.

Source

Pattern specifications digitized from the David Rumsey Map Collection

See Also

cheysson_pattern(), scale_pattern_fill_cheysson()

Examples

```
# List available pattern palettes
names(cheysson_patterns)

# Get patterns from a specific palette
cheysson_patterns$`1881_12`
```

---

cheysson_pattern_params

Create ggpattern-compatible pattern parameters

Description

Converts Cheysson pattern specifications to parameters suitable for ggpattern geoms.

Usage

```
cheysson_pattern_params(patterns, param = "fill")
```

Arguments

patterns	List of pattern specifications from cheysson_pattern()
param	Which parameter to extract: "type", "fill", "pattern_fill", "pattern_fill2", "pattern_angle", "pattern_density", "pattern_spacing", or "pattern_type". "pattern_fill2" is the color of a crosshatch's second set of lines; it equals "pattern_fill" except for the two-color crosshatches in 1883_30 and 1886_17.

Value

Vector of parameter values

Examples

```
patterns <- cheysson_pattern("1881_12")
cheysson_pattern_params(patterns, "fill")
cheysson_pattern_params(patterns, "pattern_angle")
```

cheysson_swatches *Cheysson swatches as ggpattern patterns (experimental)*

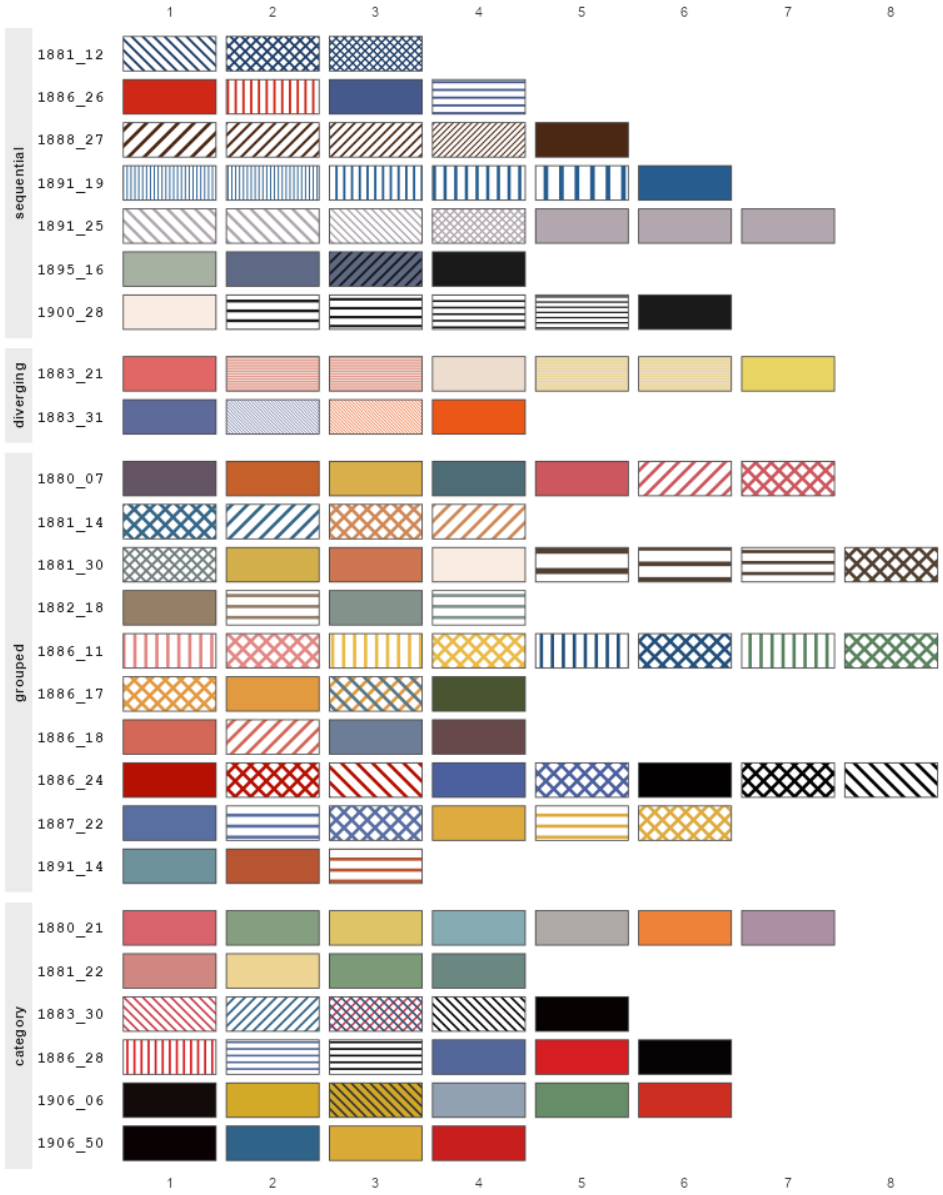
Description

Each element of each palette in cheysson_patterns is registered with 'gridpattern' as a pattern of its own, a *swatch*, which draws its paper or fill color, hatch type, line colors and angle together. A swatch is named "<palette>-<element>", e.g. "1886_28-1", and is used through the ordinary pattern aesthetic of 'ggpattern', with no other pattern aesthetics or geom arguments needed:

Details

```
ggplot(df, aes(x, y, pattern = group)) +
  geom_col_pattern() +
  scale_pattern_manual(values = paste0("1886_28-", 1:6))
```

All the swatches are shown below, one row per palette, so the swatch in row 1886_24, column 2 is "1886_24-2". Draw this atlas, or the part of it for chosen palettes or palette types, with show_swatches().



Other pattern aesthetics such as `pattern_fill` or `pattern_angle` are ignored by a swatch, apart from `pattern_spacing`, `pattern_density` and (see below) `pattern_units`.

The swatches are registered when 'gridpattern' is loaded (it is in Suggests, so ggCheysson doesn't load it), using `gridpattern::register_pattern()` when available (gridpattern >= 1.5.0), else the `ggpattern_geometry_funcs` option.

Spacing and density

The swatches of a palette keep the relative line spacings Cheysson used, e.g. the hatching of sequential palette `"1888_27"` gets denser at each step. Each swatch's spacing is a *base spacing*, by default

0.025, times its spacing relative to the palette's typical (median) hatched swatch, as measured on the plates (limited to between half and twice the base, since some measurements are unreliable). All swatches draw their lines with a density of 0.3.

`pattern_spacing` sets the base spacing, so the palette keeps its proportions, and `pattern_density` sets the density. What setting them means depends on the installed version of 'ggpattern':

- **ggpattern >= 1.4.0**: they replace the default base spacing and density, in `pattern_units`.
- **Earlier versions** (which always supply their own defaults of 0.05 and 0.2): they adjust the defaults relative to those values, e.g. `pattern_spacing = 0.1` doubles the spacing.

Either way, with `pattern_units` other than `"snpc"` (e.g. `"cm"`), the default base spacing is 0.3 cm, converted to those units.

Set them only when needed, since code that sets them may draw differently when 'ggpattern' is updated.

Legend keys

'ggpattern' zooms the pattern in a legend key by a fixed factor, which doesn't depend on the spacing or density. So with wide, thick lines (large `pattern_spacing` and `pattern_density`), a key can look almost solid while the bars are clearly hatched. Two fixes: set `pattern_key_scale_factor` in the geom below its default of 1, which draws the lines in the keys larger (e.g. `geom_col_pattern(pattern_key_scale_fact = 0.5)`), or make the keys bigger, e.g. `theme(legend.key.size = unit(1.5, "cm"))`. With the default spacing and density, the keys need neither.

See Also

cheysson_patterns, `scale_cheysson()`, `show_swatches()`

`list_cheysson_pals` *List available Cheysson palettes*

Description

Returns information about available Cheysson color palettes, optionally filtered by type.

Usage

```
list_cheysson_pals(type = NULL)
```

Arguments

`type`	Optional palette type to filter by: "sequential", "diverging", "grouped", or "category". If NULL (default), returns all palettes.

Value

A data frame with columns: name, type, album, plate, n_colors

Examples

```
# List all palettes
list_cheysson_pals()

# List only sequential palettes
list_cheysson_pals("sequential")

# List only category palettes
list_cheysson_pals("category")
```

`list_cheysson_patterns` *List available Cheysson pattern palettes*

Description

Returns information about available Cheysson pattern palettes.

Usage

```
list_cheysson_patterns(type = NULL)
```

Arguments

`type`	Optional palette type to filter by: "sequential", "diverging", "grouped", or "category". If NULL (default), returns all palettes.

Value

A data frame with columns: name, type, album, plate, n_patterns

Examples

```
# List all pattern palettes
list_cheysson_patterns()

# List only sequential palettes
list_cheysson_patterns("sequential")
```

`load_cheysson_fonts` *Load Cheysson fonts*

Description

Registers the Cheysson font families for use in plots. This function should be called before using Cheysson fonts in ggplot2.

Usage

```
load_cheysson_fonts(method = c("systemfonts", "showtext"))
```

Arguments

`method`	Font loading method: "systemfonts" (default) or "showtext". systemfonts is recommended for most uses. showtext is useful for saving plots to files.

Details

The package includes five Cheysson font families:

- **Cheysson**: Regular serif font for body text
- **CheyssonItalic**: Italic variant
- **CheyssonSansCaps**: Sans-serif capitals
- **CheyssonOutlineCaps**: Outlined capitals for titles
- **CheyssonTitle**: Decorative font for titles

When using showtext, you must call `showtext::showtext_auto()` before creating plots, and `showtext::showtext_auto(FALSE)` when done.

Windows users: The systemfonts method works for saved plots (with ragg) but custom fonts won't appear in the on-screen plot window. For on-screen preview with fonts:

- Use `method = "showtext"` instead, or
- In RStudio: Tools > Global Options > General > Graphics > Backend: "AGG"

For saving plots with systemfonts, use `ggsave(..., device = ragg::agg_png)`.

Value

Invisibly returns a character vector of loaded font family names

Examples

```
# Load fonts (default method)
load_cheysson_fonts()

# Use in a plot
library(ggplot2)
p <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  labs(title = "Using Cheysson Fonts") +
  theme(
    text = element_text(family = "Cheysson"),
    plot.title = element_text(family = "CheyssonTitle")
  )

# Save to temporary file
tmp <- tempfile(fileext = ".png")
if (requireNamespace("ragg", quietly = TRUE)) {
  ggsave(tmp, p, device = ragg::agg_png)
} else {
  ggsave(tmp, p)
}
unlink(tmp)
```

`scale_cheysson` *Apply a Cheysson color-and-pattern palette in one step*

Description

Experimental. The interface of `scale_cheysson()` may change in a future version.

Usage

```
scale_cheysson(
  palette = "1881_12",
  reverse = FALSE,
  na.value = "none",
  aesthetics = NULL,
  ...
)

aes_cheysson(x, ...)
```

Arguments

`palette`	Name of palette (e.g., "1883_30") or palette type ("sequential", "diverging", "grouped", "category"). See cheysson_patterns.

reverse	Whether to reverse the palette order. Default is FALSE.
na.value	Pattern for missing values. The default, "none", draws the geom's own fill, which no swatch shows, so e.g. geom_col_pattern(fill = "grey80") sets the color for missing values.
aesthetics	Deprecated and ignored: scale_cheysson() used to return one scale per aesthetic. To use only some of a palette's properties, use the individual scales (see scale_pattern_cheysson).
...	Additional arguments passed to ggplot2::discrete_scale(), e.g. name, labels or guide. For aes_cheysson(): other aesthetic mappings to include, e.g. x and y.
x	The variable to map, as in ggplot2::aes().

Details

Cheysson's palettes combine colors with hatching: each element of a palette has a paper or fill color, a hatch type (solid, stripe or crosshatch), line colors and an angle. ggCheysson registers each element as a 'ggpattern' pattern of its own, a *swatch* (see cheysson_swatches), and scale_cheysson() is a discrete scale for the pattern aesthetic that gives each level of the mapped variable one swatch. So a palette needs only aes(pattern = x) and this one scale.

For a sequential or diverging palette with more elements than the data has levels, the scale uses elements spread over the whole palette (keeping both ends), not the first ones. See cheysson_patterns for the stored order.

The swatches set their own colors, angles, line spacing and density, so no other pattern aesthetics need to be mapped or set, and those that are (e.g. pattern_fill) are ignored, apart from pattern_spacing, pattern_density and pattern_units. What these mean depends on the installed version of 'ggpattern'; see cheysson_swatches.

aes_cheysson(x) is deprecated: it is now the same as aes(pattern = x), since the other aesthetics it used to map aren't needed.

Value

scale_cheysson() returns a ggplot2 discrete scale for the pattern aesthetic.

aes_cheysson() returns an aesthetic mapping of x to pattern, plus any mappings in ....

See Also

cheysson_swatches for the swatches; scale_pattern_cheysson for the individual scales, one per pattern aesthetic; cheysson_pattern() for the underlying pattern specifications.

Examples

```
if (requireNamespace("ggpattern", quietly = TRUE)) {
  library(ggplot2)
  library(ggpattern)

  data <- data.frame(
    category = LETTERS[1:5],
```

```
    value = c(15, 23, 18, 20, 12)
  )

  ggplot(data, aes(category, value, pattern = category)) +
    geom_col_pattern(colour = "black") +
    scale_cheysson("1883_30") +
    theme_minimal()
}
```

`scale_color_cheysson` *Cheysson color scales for ggplot2*

Description

Color and fill scales using Cheysson palettes from the Albums de Statistique Graphique.

Usage

```
scale_color_cheysson(
  palette = "1880_21",
  discrete = TRUE,
  reverse = FALSE,
  ...
)

scale_colour_cheysson(
  palette = "1880_21",
  discrete = TRUE,
  reverse = FALSE,
  ...
)

scale_fill_cheysson(palette = "1880_21", discrete = TRUE, reverse = FALSE, ...)
```

Arguments

`palette`	Name of palette (e.g., "1880_21") or palette type ("sequential", "diverging", "grouped", "category"). Default is "1880_21".
`discrete`	Whether to use a discrete (TRUE) or continuous (FALSE) scale. Default is TRUE.
`reverse`	Whether to reverse the palette colors. Default is FALSE.
`...`	Additional arguments passed to ggplot2 scale functions

Value

A ggplot2 scale object that can be added to a plot. For discrete scales, returns a discrete_scale object. For continuous scales, returns a continuous scale object (gradient).

Examples

```
library(ggplot2)

# Discrete color scale
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() +
  scale_color_cheysson()

# Use a specific palette
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() +
  scale_color_cheysson(palette = "1881_22")

# Use a sequential palette for continuous data
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Petal.Length)) +
  geom_point() +
  scale_color_cheysson(palette = "sequential", discrete = FALSE)

# Fill scale with category colors
ggplot(iris, aes(Species, Sepal.Width, fill = Species)) +
  geom_boxplot() +
  scale_fill_cheysson(palette = "category")
```

---

scale_fill_cheysson_pattern

Apply Cheysson patterns to fill aesthetic

Description

Convenience function that applies the base fill color from Cheysson patterns. Use in combination with pattern_* scales for full pattern effect.

Usage

```
scale_fill_cheysson_pattern(palette = "1881_12", reverse = FALSE, ...)
```

Arguments

palette	Name of palette (e.g., "1881_12") or palette type ("sequential", "diverging", "grouped", "category"). Default is "1881_12".
reverse	Whether to reverse the pattern order. Default is FALSE.
...	Additional arguments passed to ggplot2 scale functions

Value

A ggplot2 discrete scale object for the fill aesthetic. Applies the base fill colors from Cheysson patterns.

Examples

```
# Requires ggpattern package
if (requireNamespace("ggpattern", quietly = TRUE)) {
  library(ggplot2)
  library(ggpattern)

  data <- data.frame(
    category = LETTERS[1:4],
    value = c(15, 23, 18, 20)
  )

  ggplot(data, aes(category, value, fill = category)) +
    geom_col_pattern(aes(pattern = category)) +
    scale_fill_cheysson_pattern("category") +
    scale_pattern_type_cheysson("category") +
    theme_minimal()
}
```

---

scale_pattern_cheysson

Cheysson pattern scales for ggpattern

Description

Pattern fill scales using Cheysson patterns from the Albums de Statistique Graphique. These scales work with ggpattern geoms to apply both colors and hatching patterns.

Usage

```
scale_pattern_fill_cheysson(palette = "1881_12", reverse = FALSE, ...)

scale_pattern_fill2_cheysson(palette = "1881_12", reverse = FALSE, ...)

scale_pattern_type_cheysson(palette = "1881_12", reverse = FALSE, ...)

scale_pattern_angle_cheysson(palette = "1881_12", reverse = FALSE, ...)

scale_pattern_density_cheysson(palette = "1881_12", reverse = FALSE, ...)
```

Arguments

palette	Name of palette (e.g., "1881_12") or palette type ("sequential", "diverging", "grouped", "category"). Default is "1881_12".
reverse	Whether to reverse the pattern order. Default is FALSE.
...	Additional arguments passed to ggplot2 scale functions

Details

These scales require the ggpattern package. Use with ggpattern geoms like `geom_col_pattern()`, `geom_bar_pattern()`, etc.

The scales apply multiple pattern aesthetics simultaneously:

- `fill`: Base fill color
- `pattern`: Type of pattern (none, stripe, crosshatch) - set via `scale_pattern_type_cheysson()`, which targets ggpattern's `pattern` aesthetic
- `pattern_fill`: Color of pattern lines
- `pattern_fill2`: Color of a crosshatch's second set of lines (differs from `pattern_fill` only in the two-color crosshatches of `1883_30` and `1886_17`)
- `pattern_angle`: Angle of stripes
- `pattern_density`: Density of pattern lines

For a sequential or diverging palette with more patterns than the data has levels, the scales use patterns spread over the whole palette (keeping both ends), not the first ones. See cheysson_patterns for the stored order.

Value

A ggplot2 discrete scale object for the specified pattern aesthetic (pattern_fill, pattern_fill2, pattern, pattern_angle, or pattern_density). These scales apply the historically accurate Cheysson patterns to ggpattern geoms.

Examples

```
# Requires ggpattern package
if (requireNamespace("ggpattern", quietly = TRUE)) {
  library(ggplot2)
  library(ggpattern)

  # Basic bar chart with patterns
  data <- data.frame(
    category = LETTERS[1:4],
    value = c(15, 23, 18, 20)
  )

  ggplot(data, aes(category, value, fill = category)) +
    geom_col_pattern(
      aes(
        pattern = category,
```

```
      pattern_fill = category,
      pattern_angle = category
    ),
    pattern_density = 0.3,
    color = "black"
  ) +
  scale_pattern_fill_cheysson("category") +
  scale_pattern_type_cheysson("category") +
  scale_pattern_angle_cheysson("category") +
  theme_minimal()
}
```

`show_palette` *Display a Cheysson palette with color swatches and hex codes*

Description

Creates a visual display of a color palette showing color swatches along with their hex codes. This is useful for documentation, presentations, and exploring the available palettes.

Usage

```
show_palette(palette = "1880_21", n = NULL, show_info = TRUE, cex = 1)
```

Arguments

`palette`	Name of palette (e.g., "1880_21"), or palette type ("sequential", "diverging", "grouped", "category").
`n`	Number of colors to display. If NULL (default), shows all colors in the palette. If specified, will interpolate if n > palette size.
`show_info`	Logical; if TRUE (default), displays palette metadata (type, album, plate) above the swatches.
`cex`	Text size multiplier for hex codes (default 1).

Value

Invisibly returns a character vector of the displayed hex codes. The function is called primarily for its side effect of creating a plot.

Examples

```
# Display a specific palette
show_palette("1880_21")

# Display palette without metadata
show_palette("1881_30", show_info = FALSE)
```

```
# Display 10 interpolated colors
show_palette("1895_16", n = 10)

# Display first sequential palette
show_palette("sequential")
```

`show_palettes` *Display multiple Cheysson palettes*

Description

Creates a visual display of multiple color palettes, useful for comparing palettes or showing all palettes of a certain type.

Usage

```
show_palettes(palettes = NULL, ncol = 1, cex = 0.8)
```

Arguments

`palettes`	Character vector of palette names. If NULL (default), shows all palettes. Can also be a palette type ("sequential", "diverging", "grouped", "category") to show all palettes of that type.
`ncol`	Number of columns for layout (default 1).
`cex`	Text size multiplier (default 0.8).

Value

Invisibly returns NULL. The function is called for its side effect of creating a plot.

Examples

```
# Show all sequential palettes (use ncol > 1 to keep panels legible with
# several palettes; a tall single-column layout can also fail to render
# when captured for the pkgdown reference site)
show_palettes("sequential", ncol = 2)

# Show specific palettes
show_palettes(c("1880_21", "1883_21", "1895_16"))
```

show_swatches *Display the Cheysson swatches*

Description

Draws an atlas of the swatches (see cheysson_swatches): one row per palette, one column per element, with the rows grouped into panels by palette type. Each cell is drawn by the registered swatch itself, so the swatch for a cell is named by its row and column, e.g. row `1886_28`, column 3 is `"1886_28-3"`.

Usage

```
show_swatches(palettes = NULL)
```

Arguments

`palettes`	Character vector of palette names. If NULL (default), shows all palettes. Can also be one or more palette types ("sequential", "diverging", "grouped", "category") to show all palettes of those types.

Value

A ggplot object, which draws the atlas when printed. Requires the 'ggpattern' package.

See Also

cheysson_swatches, `scale_cheysson()`, `show_palettes()`

Examples

```
if (requireNamespace("ggpattern", quietly = TRUE)) {
  show_swatches("sequential")
}
```

theme_cheysson *Cheysson theme for ggplot2*

Description

A ggplot2 theme inspired by the visual style of the *Albums de Statistique Graphique*, using Cheysson fonts and appropriate styling.

Usage

```
theme_cheysson(
  base_size = 11,
  base_family = "auto",
  title_family = "auto",
  axis_title_family = "auto",
  load_fonts = TRUE
)
```

Arguments

base_size	Base font size (default: 11)
base_family	Base font family. If "auto" (default), uses Cheysson if available, otherwise falls back to sans-serif
title_family	Font family for titles (default: "auto")
axis_title_family	Font family for axis titles (default: "auto")
load_fonts	Automatically load Cheysson fonts if not already loaded (default: TRUE)

Details

This theme applies the following styling:

- Cheysson fonts for all text elements
- Minimal grid lines
- Classic axis styling
- Subtle colors matching historical aesthetics

Font selection:

- Plot title: CheyssonTitle (decorative)
- Axis titles: CheyssonSansCaps (capitals)
- Body text: Cheysson (regular)

Value

A ggplot2 theme object

Examples

```
## Not run:
# Not run automatically: rendering text with a Cheysson font registered via
# systemfonts can crash grid with "invalid font type" on some graphics
# devices (reproduced under R CMD check --as-cran); works fine interactively.
library(ggplot2)

# Load fonts first (required for proper rendering)
load_cheysson_fonts()
```

```
# Basic usage
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  labs(title = "Automobile Statistics") +
  theme_cheysson()

# With Cheysson color palette
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point(size = 3) +
  scale_color_cheysson("1881_22") +
  theme_cheysson()

## End(Not run)
```

`theme_cheysson_map` *Cheysson map theme*

Description

A theme specifically designed for maps, with no grid and minimal elements, in the style of Cheysson's cartographic works.

Usage

```
theme_cheysson_map(
  base_size = 11,
  base_family = "auto",
  title_family = "auto",
  load_fonts = TRUE
)
```

Arguments

`base_size`	Base font size (default: 11)
`base_family`	Base font family. If "auto" (default), uses Cheysson if available, otherwise falls back to sans-serif
`title_family`	Font family for titles (default: "auto")
`load_fonts`	Automatically load Cheysson fonts if not already loaded (default: TRUE)

Value

A ggplot2 theme object that can be added to a plot with +.

Examples

```
## Not run:
# Not run automatically: see theme_cheysson() for why (same font/grid crash)
# once a plot with this theme is actually rendered/printed.
# For use with spatial data/maps
library(ggplot2)

# Load fonts first
load_cheysson_fonts()

# Example with spatial data (requires sf package)
if (requireNamespace("sf", quietly = TRUE)) {
  # ggplot(map_data) + geom_sf() + theme_cheysson_map()
}

## End(Not run)
```

`theme_cheysson_minimal` *Minimal Cheysson theme*

Description

A more minimal version of theme_cheysson with fewer grid lines, suitable for maps and diagrams.

Usage

```
theme_cheysson_minimal(
  base_size = 11,
  base_family = "auto",
  title_family = "auto",
  axis_title_family = "auto",
  load_fonts = TRUE
)
```

Arguments

`base_size`	Base font size (default: 11)
`base_family`	Base font family. If "auto" (default), uses Cheysson if available, otherwise falls back to sans-serif
`title_family`	Font family for titles (default: "auto")
`axis_title_family`	Font family for axis titles (default: "auto")
`load_fonts`	Automatically load Cheysson fonts if not already loaded (default: TRUE)

Value

A ggplot2 theme object that can be added to a plot with +.

Examples

```
## Not run:
# Not run automatically: see theme_cheysson() for why (same font/grid crash).
library(ggplot2)

# Load fonts first
load_cheysson_fonts()

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_cheysson_minimal()

## End(Not run)
```

Index

∗ **datasets**
albumImages, 2
cheysson_fonts, 4
cheysson_labels, 6
cheysson_palettes, 10
cheysson_patterns, 13

aes_cheysson *(*scale_cheysson*)*, 20
albumImages, 2, *7*

cheysson-swatches *(*cheysson_swatches*)*, 15
cheysson_font, 4
cheysson_fonts, 4
cheysson_fonts_available, 5
cheysson_fonts_available(), *5*
cheysson_labels, 6, *9*
cheysson_name, 8
cheysson_name(), *6, 7*
cheysson_pal, 9
cheysson_pal(), *8*, *11*
cheysson_palettes, *3*, *6, 7*, 10, *13*
cheysson_pattern, 12
cheysson_pattern(), *8*, *14*, *21*
cheysson_pattern_params, 14
cheysson_patterns, *3*, *6, 7*, *11*, 13, *15*, *17*, *20, 21*, *25*
cheysson_swatches, 15, *21*, *28*

ggplot2::aes(), *21*
ggplot2::discrete_scale(), *21*

list_cheysson_pals, 17
list_cheysson_patterns, 18
load_cheysson_fonts, 19
load_cheysson_fonts(), *5*

scale_cheysson, 20
scale_cheysson(), *17*, *28*
scale_color_cheysson, 22
scale_color_cheysson(), *6*, *8*, *11*
scale_colour_cheysson *(*scale_color_cheysson*)*, 22
scale_fill_cheysson *(*scale_color_cheysson*)*, 22
scale_fill_cheysson_pattern, 23
scale_pattern_angle_cheysson *(*scale_pattern_cheysson*)*, 24
scale_pattern_cheysson, *21*, 24
scale_pattern_density_cheysson *(*scale_pattern_cheysson*)*, 24
scale_pattern_fill2_cheysson *(*scale_pattern_cheysson*)*, 24
scale_pattern_fill_cheysson *(*scale_pattern_cheysson*)*, 24
scale_pattern_fill_cheysson(), *14*
scale_pattern_type_cheysson *(*scale_pattern_cheysson*)*, 24
show_palette, 26
show_palettes, 27
show_palettes(), *28*
show_swatches, 28
show_swatches(), *15*, *17*

theme_cheysson, 28
theme_cheysson(), *5*
theme_cheysson_map, 30
theme_cheysson_minimal, 31