

# Package ‘ggtaichi’

September 30, 2026

**Type** Package

**Title** Taichi-Diagram Visualization for Two Data Sources

**Version** 0.3.0

**Description** A data visualization design that compares two (usually on a par with each other) data sources on one grid of taichi (yin-yang) diagrams, where the two interlocking fish of every symbol are filled by the two sources, while inheriting 'ggplot2' features.

**License** GPL (>= 3)

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**URL** <https://pursuitofdatascience.github.io/ggtaichi/>,  
<https://github.com/PursuitOfDataScience/ggtaichi>

**BugReports** <https://github.com/PursuitOfDataScience/ggtaichi/issues>

**Depends** R (>= 3.5.0)

**Imports** rlang, grid, grDevices, stats, utils, farver, ggplot2 (>= 3.4.0), ggnewscale (>= 0.4.5)

**RoxygenNote** 7.3.2

**Suggests** rmarkdown, knitr, testthat (>= 3.0.0), vdiffr, gganimate, gifski, ggiraph, colorspace

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Youzhi Yu [aut, cre]

**Maintainer** Youzhi Yu <yuyouzhi666@icloud.com>

**Repository** CRAN

**Date/Publication** 2026-09-30 18:40:09 UTC

Contents

|                                |           |
|--------------------------------|-----------|
| cafes_tg . . . . .             | 2         |
| draw_key_taichi . . . . .      | 3         |
| geom_taichi . . . . .          | 4         |
| geom_taichi_diff . . . . .     | 12        |
| geom_yin_fish . . . . .        | 13        |
| pitts_emojis . . . . .         | 15        |
| pitts_tg . . . . .             | 16        |
| remove_padding . . . . .       | 17        |
| scale_taichi . . . . .         | 18        |
| states_tg . . . . .            | 21        |
| taichi_check_palette . . . . . | 22        |
| taichi_palette . . . . .       | 23        |
| taichi_palette_pair . . . . .  | 25        |
| taichi_summary . . . . .       | 26        |
| theme_taichi . . . . .         | 27        |
| <b>Index</b>                   | <b>29</b> |

---

|          |                                                   |
|----------|---------------------------------------------------|
| cafes_tg | <i>Synthetic café orders: espresso vs. matcha</i> |
|----------|---------------------------------------------------|

---

Description

A small, deliberately *synthetic* two-source dataset for demos and vignettes: weekly orders (per 100 customers) of espresso and matcha drinks across eight fictional neighbourhoods over a 12-week season. It provides an evergreen alternative to the COVID-era `pitts_tg` / `states_tg` data, and because both columns share the same units it is the natural demo for `shared_limits` / `shared_legend` in `geom_taichi()`. The values are simulated with a fixed seed (espresso cools off over the season while matcha picks up, at neighbourhood-specific rates, plus noise); the generating script ships in `data-raw/cafes_tg.R` in the source repository.

Usage

`cafes_tg`

Format

- A data frame with 96 rows and 4 columns:
- week** Week of the season, 1 to 12.
  - neighbourhood** One of eight fictional neighbourhoods (factor).
  - espresso** Weekly espresso orders per 100 customers.
  - matcha** Weekly matcha orders per 100 customers.

**Source**

Simulated by the package author; see `data-raw/cafes_tg.R`.

**Examples**

```
library(ggplot2)
ggplot(cafes_tg, aes(x = week, y = neighbourhood)) +
  geom_taichi(yin = matcha, yang = espresso, shared_legend = TRUE) +
  theme_taichi()
```

---

|                 |                                   |
|-----------------|-----------------------------------|
| draw_key_taichi | <i>A taichi-shaped legend key</i> |
|-----------------|-----------------------------------|

---

**Description**

The legend key glyph used by `geom_yin_fish()` and `geom_yang_fish()`, and therefore by `geom_taichi()`: a small taichi symbol whose relevant fish is filled with the key's colour while the other is left as an outline, so the key looks like the mark it describes and says which half of the glyph the scale governs. Under `geom_taichi(shared_legend = TRUE)`, where the one legend governs both fish, its keys fill both halves. Pass it to a layer's `key_glyph` argument to use it elsewhere, or use `key_glyph = "rect"` for plain ggplot2 rectangles.

**Usage**

```
draw_key_taichi(data, params, size, fish = "both")
```

**Arguments**

|                     |                                                                                                                                                                                                                                                               |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>   | A one-row data frame of the key's aesthetics, supplied by ggplot2.                                                                                                                                                                                            |
| <code>params</code> | The layer's parameters, supplied by ggplot2. <code>eyes = TRUE</code> is honoured, so a plot drawn with eyes gets keys with eyes. For a single-fish key the eye takes the layer's own eye colour and (constant) eye size, so it matches the eyes on the plot. |
| <code>size</code>   | The key size in mm, supplied by ggplot2. Unused: the glyph is drawn in a square viewport that fills the key, so it stays round whatever the key's aspect ratio.                                                                                               |
| <code>fish</code>   | Which fish carries <code>data\$fill</code> : "yin", "yang", or "both". The last fills the yin fish with the key colour and the yang fish with a pale version of it, for a decorative complete symbol with the classic white and black eyes.                   |

**Details**

Keys only appear for discrete fills. A continuous fill is drawn by `ggplot2::guide_colourbar()`, which is a gradient bar rather than a set of keys, so this glyph has no effect there.

**Value**

A grob.

## Examples

```
library(ggplot2)
d <- data.frame(x = c(1, 2), y = 1,
                grp = factor(c("a", "b")), value = c(1, 2))

# the default for the fish geoms
ggplot(d, aes(x, y)) +
  geom_yin_fish(aes(fill = grp))

# a full symbol, or the old rectangles
ggplot(d, aes(x, y)) +
  geom_yin_fish(aes(fill = grp), key_glyph = draw_key_taichi)
ggplot(d, aes(x, y)) +
  geom_yin_fish(aes(fill = grp), key_glyph = "rect")
```

---

geom\_taichi

*Taichi*

---

## Description

The taichi geom turns each cell of a heatmap-like grid into a taichi (yin-yang) diagram. The two interlocking "fish" of the diagram use luminance to show the values from two data sources on the same plot, so four dimensions of data can be expressed at once: the x and y position of every taichi symbol plus the yin and yang values that fill its two halves. With the optional eyes enabled and mapped to data (see `eyes`, `yin_eye_size`, `yang_eye_size`), a single glyph can carry up to six dimensions.

## Usage

```
geom_taichi(
  yin,
  yang,
  yin_name = NULL,
  yang_name = NULL,
  yin_colors = c("gray100", "gray85", "gray50", "gray35", "gray0"),
  yang_colors = c("#FED7D8", "#FE8C91", "#F5636B", "#E72D3F", "#C20824"),
  palette = NULL,
  yin_scale = NULL,
  yang_scale = NULL,
  angle = NULL,
  eyes = FALSE,
  yin_eye_size = 0.15,
  yang_eye_size = 0.15,
  yin_eye_colour = NULL,
  yang_eye_colour = NULL,
  explicit = c("none", "difference", "ratio", "log_ratio", "z"),
  explicit_channel = c("eye_size", "angle", "border", "radius"),
  explicit_range = NULL,
```

```

radius_exponent = 0.57,
interactive = FALSE,
tooltip = NULL,
data_id = NULL,
onclick = NULL,
data_id_by = c("cell", "fish", "source"),
shared_limits = FALSE,
shared_legend = FALSE,
width = NULL,
height = NULL,
alpha = NA,
na.rm = FALSE,
colour = NA,
linewidth = 0.1,
linetype = 1,
show.legend = NA,
key_glyph = NULL,
...
)

```

### Arguments

|             |                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| yin         | The unquoted column name (or a literal string naming a column) for the yin (dark) fish of the taichi symbol. To pass a name held in a variable, use <code>.data[[nm]]</code> or <code>!!rlang::sym(nm)</code> ; a bare variable would be mapped as a constant fill, exactly as it would be inside <code>aes()</code> .                                                                                                 |
| yang        | The unquoted column name (or a literal string naming a column) for the yang (light) fish of the taichi symbol, as yin.                                                                                                                                                                                                                                                                                                 |
| yin_name    | The label name (in quotes) for the legend of the yin rendering. Default is NULL (uses the column name). It titles a supplied yin_scale too, unless that scale object names itself.                                                                                                                                                                                                                                     |
| yang_name   | The label name (in quotes) for the legend of the yang rendering. Default is NULL (uses the column name). It titles a supplied yang_scale too, unless that scale object names itself.                                                                                                                                                                                                                                   |
| yin_colors  | A color vector, usually as hex codes, for the yin fish fill. Used as a gradient for continuous data and as a discrete palette for factor/character data. Ignored if yin_scale is provided.                                                                                                                                                                                                                             |
| yang_colors | A color vector, usually as hex codes, for the yang fish fill. Used as a gradient for continuous data and as a discrete palette for factor/character data. Ignored if yang_scale is provided.                                                                                                                                                                                                                           |
| palette     | A matched pair of ramps to use instead of yin_colors / yang_colors: the name of a <code>taichi_palette()</code> preset ("balanced", "diverging", "viridis_pair", "brewer_pair", "greyscale_safe", "default") or a list with yin and yang colour vectors, such as the result of <code>taichi_palette_pair()</code> . Default NULL, which keeps the package's historical grey / seal-red pair. See the Palettes section. |
| yin_scale   | An optional fill scale for the yin fish: either a ready scale object or a scale constructor function (e.g. <code>ggplot2::scale_fill_viridis_d</code> ). Overrides auto-                                                                                                                                                                                                                                               |

|                                              |                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                              | detection. It must govern a <i>fill</i> aesthetic; a scale for another aesthetic (say <code>scale_colour_viridis_c</code> ) is rejected with an error rather than quietly leaving the fish on the default gradient.                                                                                                                                                                                          |
| <code>yang_scale</code>                      | An optional fill scale for the yang fish, as <code>yin_scale</code> .                                                                                                                                                                                                                                                                                                                                        |
| <code>angle</code>                           | Rotation of each glyph in degrees, counter-clockwise: either a single number or an unquoted column name (one angle per cell). A mapped column must be numeric.                                                                                                                                                                                                                                               |
| <code>eyes</code>                            | Logical. If TRUE, draws the classic taichi eyes (dots), each centred in its fish's head. Default FALSE, preserving the plain v0.1.0 look.                                                                                                                                                                                                                                                                    |
| <code>yin_eye_size, yang_eye_size</code>     | Size of each eye as a proportion of the glyph radius: a constant (default 0.15) or an unquoted data column to encode a variable (see the Eyes section for the rescaling rule).                                                                                                                                                                                                                               |
| <code>yin_eye_colour, yang_eye_colour</code> | Colour of each eye dot: a constant, an unquoted data column containing colour strings, or NULL (the default) to take the colour from the theme: the yin eye from the theme's paper and the yang eye from its ink, which is white and black on every light theme and swaps on a dark one. On ggplot2 before 4.0.0, where themes cannot set geom defaults, NULL falls back to the literal "white" and "black". |
| <code>explicit</code>                        | Which relationship between the two sources to compute and show as a third channel: "none" (the default), "difference", "ratio", "log_ratio" or "z". See the Explicit encoding section, and <a href="#">taichi_summary()</a> for the definitions.                                                                                                                                                             |
| <code>explicit_channel</code>                | Where the computed statistic goes: "eye_size" (the default), "angle", "border" or "radius". Ignored when <code>explicit = "none"</code> . The chosen channel cannot also be set by hand: <code>explicit_channel = "angle"</code> together with an angle argument, for example, is an error rather than a silent override.                                                                                    |
| <code>explicit_range</code>                  | Two numbers giving the output range of <code>explicit_channel</code> , or NULL (the default) for that channel's own sensible range.                                                                                                                                                                                                                                                                          |
| <code>radius_exponent</code>                 | Only used by <code>explicit_channel = "radius"</code> : the exponent relating the statistic to the glyph radius. 0.5 is strict area scaling; the default 0.57 is the cartographic apparent-magnitude (Flannery) compensation, which makes larger symbols slightly larger than the geometry alone would give because readers systematically underestimate the area ratio between big and small circles.       |
| <code>interactive</code>                     | If TRUE, draw the fish (and their eyes) as <b>ggiraph</b> grobs carrying tooltip, <code>data_id</code> and <code>onclick</code> , so that <code>ggiraph::girafe()</code> turns the plot into a widget. Needs the <b>ggiraph</b> package. Default FALSE, which renders exactly as before and does not touch <b>ggiraph</b> . See the Interactivity section.                                                   |
| <code>tooltip, data_id, onclick</code>       | Optional unquoted data columns overriding the interactive attributes. Used only when <code>interactive = TRUE</code> ; by default tooltip is built from the two values, their difference and the cell's coordinates, <code>data_id</code> from <code>data_id_by</code> , and <code>onclick</code> is empty.                                                                                                  |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data_id_by    | Scope of the default data_id, i.e. what one hover highlights: "cell" (both fish of that glyph; the default), "fish" (one fish), or "source" (every fish of that source, in every cell). Ignored when data_id is supplied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| shared_limits | If TRUE and both sources are of the same type (both continuous, or both discrete), the two auto-built fill scales share common limits (the union range, or the union of levels, of yin and yang), so equal values read as equal ink. Explicit limits passed through ... take precedence. As of 0.3.0 the shared limits are also pushed into a custom yin_scale / yang_scale that does not set limits of its own, so a supplied binned scale shares breaks too. Default FALSE.                                                                                                                                                                                                                                                                                                                |
| shared_legend | If TRUE, treats the two sources as directly comparable: implies shared_limits = TRUE, paints both fish with yin_colors, and shows a single legend (the yang guide is dropped). Unless yin_name is supplied, the legend is titled "yin / yang". Default FALSE. This is the perceptually correct choice whenever the two sources really are directly comparable: one ramp means equal values are equal ink by construction, with no palette pairing to get wrong (see the Palettes section). The cost is that the sources are then told apart only by their position inside the glyph: yin is the top bulb, yang the bottom. When a custom yang_scale is supplied it is used as given, so making the two palettes agree is then your business; the duplicate yang guide is dropped either way. |
| width, height | Width and height of each cell. Typically omitted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| alpha         | Alpha transparency for the fish fills. A single value for the whole layer (see the Styling section).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| na.rm         | If TRUE, silently removes rows with missing positions (a missing fill is drawn in the scale's na.value; see the Missing values section).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| colour        | Outline colour of the fish. A single value for the whole layer (see the Styling section).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| linewidth     | Outline width of the fish (in mm). Replaces the deprecated size aesthetic of ggtaichi 0.1.0. A single value for the whole layer (see the Styling section).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| linetype      | Outline linetype of the fish. A single value for the whole layer (see the Styling section).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| show.legend   | Logical. Should the layer be included in the legend?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| key_glyph     | The legend key glyph, passed on to <a href="#">layer()</a> . Both fish default to a small taichi with their own half filled (see <a href="#">draw_key_taichi()</a> ), and the one legend shared_legend keeps fills both halves; pass "rect" for the plain ggplot2 rectangles of earlier versions. Keys only appear for discrete fills; a continuous fill gets a colourbar.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ...           | Additional arguments passed to <i>both</i> auto-built fill scales (e.g., shared limits or na.value). Because they go to both, an argument that suits only one kind of scale will be rejected by the other when yin and yang are of different types: a numeric limits, for instance, draws ggplot2's "Continuous limits supplied to discrete scale" warning from the discrete fish. For per-fish scale options, supply yin_scale / yang_scale instead. The scale arguments geom_taichi() fills in itself (name, values and colors / colours) are not accepted here; use yin_name / yang_name and yin_colors / yang_colors.                                                                                                                                                                    |

## Details

A seventh channel, `angle`, rotates the glyph, and `explicit` adds an eighth that is *computed* rather than mapped: the relationship between the two sources, shown as a third channel of the same mark. See the Explicit encoding section.

## Value

A `ggtaichi_plot` object: the two fish layers plus the fill scales they need, ready to be added to a `ggplot` with `+`. It is not a plot on its own.

## Discrete and continuous fills

`geom_taichi()` inspects the plot data at `+` time. A numeric `yin / yang` column gets a continuous `scale_fill_gradientn` built from `yin_colors / yang_colors`; a factor, character, or logical column (including computed expressions such as `factor(week)`) gets a discrete `scale_fill_manual` whose palette is interpolated from the same color vectors. With the default vectors, or a palette, the discrete palette samples the ramp and skips its palest end so that no category is invisible on a white panel; an explicitly supplied color vector is used as-is, one colour per level in order. Supply `yin_scale / yang_scale` to override the automatic choice entirely. Each fish has its own fill scale, so a fill scale added to the plot afterwards (`+ scale_fill_viridis_c()`) replaces the `yang` one only; pass it as `yang_scale` instead.

The automatic discrete palette samples a *sequential* ramp, which implies that the levels are ordered. That suits an ordered factor and overstates an unordered one: if your categories have no natural order, supply a qualitative palette through `yin_colors / yang_colors` or a scale through `yin_scale / yang_scale`, so the fill does not assert a ranking the data does not have.

The glyphs are drawn from the plot's own data, the data the scales are chosen from, so `geom_taichi()` has no data argument; to draw one fish from other data, use `geom_yin_fish()` / `geom_yang_fish()`, which take one.

Because the choice is made when the layer is added, replacing the plot's data afterwards keeps the scales picked for the original data. Swapping in data of the same types is fine; if the new `yin / yang` columns are of the *other* kind, `ggplot2` reports a "Discrete value supplied to a continuous scale" (or the reverse) at draw time. Rebuild the plot rather than substituting its data.

## Eyes

`eyes = TRUE` draws the classic taichi dots, each sitting in its own fish's head: the yin eye in the top bulb, the yang eye in the bottom bulb. The size and colour arguments accept either a constant or an (unquoted) data column, so the eyes can encode up to two further variables. A mapped eye-size column is rescaled to radii between 0.05 and 0.3 of the glyph radius, unless all its non-zero values already lie in  $(0, 0.5]$ , in which case they are used directly as radius proportions. Cells whose eye size is `NA` or `0` are drawn without an eye, so a column may mix proportions with zeros to suppress individual eyes. A column whose values are all equal, and not already proportions, gets the midpoint radius, 0.175.

## Styling

`alpha`, `colour`, `linewidth` and `linetype` are layer-wide constants here. Each has a concrete default, so `geom_taichi()` always passes it to both fish layers as a parameter, and a parameter takes

precedence over an inherited mapping: a plot-level `aes(linewidth = ...)` (or `alpha`, `colour`, `linetype`) has no effect on the glyphs. To drive one of those from a column, build the layers yourself with `geom_yin_fish()` / `geom_yang_fish()`, which take all four as ordinary aesthetics.

`width` and `height` behave differently, because they default to `NULL` and are forwarded only when you actually supply them: a plot-level `aes(width = ...)` *does* size the cells per row. So the data-driven channels of `geom_taichi()` are `yin`, `yang`, `angle`, the two eyes, and `width` / `height` via `aes()`.

### Missing values

A fish whose fill value is `NA` is painted in the scale's `na.value` colour (pass e.g. `na.value = "transparent"` through ... to change it), while `na.rm = TRUE` silently drops rows with missing positions.

### Time on an axis

Putting time on `x` makes each row of the grid a time series drawn as a row of discrete glyphs, and the series is then encoded in *fill* rather than in position, so slope is not encoded at all and a reader infers a trend by comparing the shade of neighbouring cells. That is a poor substitute for a line. Use a taichi grid for the question *which series differ from each other, and where*; put a line chart or a horizon plot beside it for *what is the trend*.

### Explicit encoding

Two fish sharing one position is a *superposition* comparison. It is very good at "are these similar?" and "which is bigger here?", and it cannot answer "by how much?", because that needs the relationship itself to be computed and drawn. `explicit` does exactly that, turning one of "difference" (`yin - yang`), "ratio", "log\_ratio" or "z" into a third channel of the glyph. The statistics are the ones `taichi_summary()` tabulates, including its rule that a ratio of a non-positive value is `NA` rather than `Inf`.

Every channel shows how far the statistic is from agreement, which is 0 for every statistic except "ratio", whose agreement is 1. A ratio is measured by its plain distance from 1, so a ratio of 2 reads as a wider gap than one of 0.5; use "log\_ratio" when a doubling and a halving should look alike.

`explicit_channel` chooses where it goes:

"eye\_size" The default, and the tidiest: the eyes already exist and are visually subordinate to the fills, so the two fish keep carrying the two sources while eye size carries the gap between them. A big eye reads as "look here", which is what a big gap means. Cells where the two sources agree exactly get no eye at all, so a plain glyph means agreement. Implies `eyes = TRUE`.

"angle" The most *accurate* option. Direction and angle are read far more precisely than shading, so encoding the gap as tilt makes it legible to a precision the fills can never reach: an upright glyph means the two sources agree and the lean shows which way and how far. The cost is the symbol's upright orientation, which is why it is a choice rather than the default.

"border" Outline width. Unobtrusive, and it composes with everything else, but the least precise of the four. Because the default colour is `NA` (no outline at all), this channel gives the outline a visible colour unless you set colour yourself.

"radius" Glyph size, scaled by area rather than by diameter so that the eye's area-based reading is the correct one: the radius follows the statistic raised to `radius_exponent`, a touch above the square root that strict area scaling would use. Cells where the sources agree shrink; use it when the interesting thing is *where* they disagree.

`explicit_range` sets the channel's output range; each channel has a sensible default (`c(0, 0.3)` of the glyph radius for eyes, `c(-45, 45)` degrees for angle, `c(0, 1)` mm for the border, `c(0.4, 1)` of the cell for radius). The statistic is rescaled across the whole layer, so the mapping is comparable between facets.

`geom_taichi_diff()` draws the same statistic as a diverging heatmap when the glyph is not the right chart for the question, and `taichi_summary()` returns it as a table.

## Palettes

The two ramps are compared against each other, so they need to be matched: if one spans a wider luminance range than the other then equal values do not read as equal ink and one fish looks heavier wherever the data says the sources are level. The default grey-and-red pair is *not* matched (run `taichi_check_palette()` with no arguments to see the numbers) and is kept only for continuity. `palette` selects a matched pair instead ("balanced" is the recommended one), and it also accepts the output of `taichi_palette_pair()`. It sets `yin_colors` and `yang_colors` together, so passing both is an error, and as a ramp: a discrete fill samples it evenly, palest end skipped, rather than taking its first colours.

## Interactivity

Fill is the least accurate channel there is, which is why an interactive version of a taichi grid is not a gimmick: hovering supplies the exact values without giving up the encoding. With `interactive = TRUE` the layers emit **ggiraph** grobs, and the plot becomes a widget when it is passed to `ggiraph::girafe()`:

```
p <- ggplot(cafes_tg, aes(week, neighbourhood)) +
  geom_taichi(yin = matcha, yang = espresso, interactive = TRUE)
ggiraph::girafe(ggobj = p)
```

The default tooltip carries both values, their difference, and the cell's coordinates. `data_id_by` decides what a hover highlights: "cell" (the default) lights up both fish of one glyph, "fish" one fish at a time, and "source" every fish of one source at once. That last one turns the superposition display into a single-source display for as long as the pointer rests there, letting a reader decompose the comparison instead of doing it in their head. `tooltip`, `data_id` and `onclick` take a data column to override any of it. The default ids name a cell by its x and y values, so in a faceted plot the same cell lights up in every panel at once. To keep the panels apart, give `data_id` an expression that includes the facet variable, such as `paste(state, week, category)`.

The static rendering is unchanged: with `interactive = FALSE` the package does not touch **ggiraph** at all, and with it `TRUE` the same geometry is drawn, only in grobs that carry the extra attributes. **plotly** is not and will not be supported: `ggplotly()` cannot translate custom grobs, which is exactly what this package draws.

**Examples**

```
library(ggplot2)

# taichi with numeric fills

data <- data.frame(x = rep(c(1, 2, 3), 3),
                  y = rep(c(1, 2, 3), each = 3),
                  yin_values = 1:9,
                  yang_values = 9:1)

ggplot(data, aes(x, y)) +
  geom_taichi(yin = yin_values,
             yang = yang_values)

# categorical (discrete) fills are detected automatically

data$yin_class <- rep(c("low", "mid", "high"), 3)

ggplot(data, aes(x, y)) +
  geom_taichi(yin = yin_class,
             yang = yang_values)

# classic eyes, rotation, and data-driven eye sizes

ggplot(data, aes(x, y)) +
  geom_taichi(yin = yin_values,
             yang = yang_values,
             eyes = TRUE,
             yin_eye_size = yang_values,
             angle = 45)

# a matched palette pair, and the gap between the sources as eye size

ggplot(data, aes(x, y)) +
  geom_taichi(yin = yin_values,
             yang = yang_values,
             palette = "balanced",
             shared_limits = TRUE,
             explicit = "difference")

# the same gap as tilt: the most accurate channel available

ggplot(data, aes(x, y)) +
  geom_taichi(yin = yin_values,
             yang = yang_values,
             explicit = "difference",
             explicit_channel = "angle")

# tooltips carry the exact values; needs ggiraph to view

if (requireNamespace("ggiraph", quietly = TRUE)) {
  p <- ggplot(data, aes(x, y)) +
```

```

geom_taichi(yin = yin_values, yang = yang_values,
            interactive = TRUE, data_id_by = "source")
# ggiraph::girafe(ggobj = p)
}

```

---

geom\_taichi\_diff

*The difference between the two sources, as a heatmap*


---

## Description

Sometimes the right chart for "how much bigger?" is not a glyph at all but a diverging heatmap, and a package that offers one next to its signature mark is more useful than one that insists on the mark. `geom_taichi_diff()` computes one of `taichi_summary()`'s statistics per cell and draws it with `ggplot2::geom_tile()` on a diverging scale centred on "the two sources agree".

## Usage

```

geom_taichi_diff(
  yin,
  yang,
  method = c("difference", "ratio", "log_ratio", "z"),
  palette = "diverging",
  name = NULL,
  midpoint = NULL,
  symmetric = TRUE,
  na.value = "grey90",
  ...
)

```

## Arguments

|           |                                                                                                                                                                                                                                                                                                                               |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| yin, yang | Unquoted column names (or strings naming columns) for the two sources, as in <code>geom_taichi()</code> .                                                                                                                                                                                                                     |
| method    | Which statistic to draw: "difference" (yin - yang, the default), "ratio", "log_ratio" or "z". See <code>taichi_summary()</code> for the definitions; "log_ratio" is the better choice than "ratio" whenever the two sources span orders of magnitude, because it is symmetric around agreement.                               |
| palette   | The diverging colours: the name of a <code>taichi_palette()</code> preset (the yang ramp's dark end becomes the low colour, the yin ramp's dark end the high colour, so the tile agrees with the glyph), a list with yin and yang elements, or a character vector of exactly three colours giving low, mid and high directly. |
| name      | Legend title. Defaults to a label naming the statistic and the two columns, e.g. "matcha - espresso".                                                                                                                                                                                                                         |
| midpoint  | The value that counts as "the sources agree" and is painted in the mid colour. Defaults to 1 for "ratio" and 0 for every other method.                                                                                                                                                                                        |

|                        |                                                                                                                                                                                                                                     |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>symmetric</code> | If TRUE (the default) the fill limits are made symmetric about midpoint, so the mid colour really does sit at the centre of the legend and the two directions are coloured comparably. Set it to FALSE to use the plain data range. |
| <code>na.value</code>  | Colour for cells whose statistic is missing, which includes every non-positive cell under "ratio" and "log_ratio".                                                                                                                  |
| <code>...</code>       | Further arguments passed to <code>ggplot2::geom_tile()</code> , for example width, height, colour or linewidth. A data given here is also what the symmetric limits are computed from.                                              |

### Details

It is the explicit-encoding companion to `geom_taichi()`: same data, same grid, same statistics, but the relationship itself is on the page instead of being left to the reader's eye. Use it beside a taichi grid, not instead of one: the glyph shows the levels, this shows the gap.

### Value

An object that, added to a `ggplot2::ggplot()` with `+`, draws the difference tiles and their diverging fill scale. It is not a plot on its own.

### See Also

`taichi_summary()` for the same numbers as a table, and `geom_taichi()`'s explicit argument for the in-glyph version.

### Examples

```
library(ggplot2)

ggplot(cafes_tg, aes(week, neighbourhood)) +
  geom_taichi_diff(yin = matcha, yang = espresso) +
  theme_taichi()

# log ratio, symmetric about "the two agree"
ggplot(cafes_tg, aes(week, neighbourhood)) +
  geom_taichi_diff(yin = matcha, yang = espresso, method = "log_ratio") +
  theme_taichi()
```

---

geom\_yin\_fish

*The individual taichi fish layers*

---

### Description

`geom_yin_fish()` and `geom_yang_fish()` each draw one of the two interlocking fish of a taichi symbol per (x, y) cell. They are the building blocks that `geom_taichi()` assembles (together with two fill scales and a `ggnewscale::new_scale_fill()` break); use them directly when you want full control: to bring your own fill scale for a single fish, to stack scales differently, or to draw only one source.

**Usage**

```
geom_yin_fish(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  width = NULL,
  height = NULL,
  eyes = FALSE,
  interactive = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  key_glyph = NULL,
  ...
)
```

```
geom_yang_fish(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  width = NULL,
  height = NULL,
  eyes = FALSE,
  interactive = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  key_glyph = NULL,
  ...
)
```

**Arguments**

|                                            |                                                                                                                                                                                                    |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| mapping, data, stat, position, inherit.aes | See <a href="#">ggplot2::layer()</a> .                                                                                                                                                             |
| width, height                              | Cell size; defaults to the resolution of the data.                                                                                                                                                 |
| eyes                                       | Logical. Draw the classic eye dot inside this fish's head?                                                                                                                                         |
| interactive                                | Logical. Draw <b>ggiraph</b> grobs carrying the tooltip, data_id and onclick aesthetics, so that <code>ggiraph::girafe()</code> can turn the plot into a widget. Needs the <b>ggiraph</b> package. |
| na.rm                                      | If TRUE, silently removes rows with missing positions; a missing fill is drawn in the scale's na.value instead.                                                                                    |
| show.legend                                | Logical. Should this layer be included in the legends?                                                                                                                                             |
| key_glyph                                  | Legend key glyph; defaults to this fish's half of a taichi symbol (see <a href="#">draw_key_taichi()</a> ). Passed to <a href="#">ggplot2::layer()</a> .                                           |

... Other arguments passed to `ggplot2::layer()`: either aesthetics used as constant parameters (e.g. `eye_size = 0.2`) or geom parameters.

## Details

Both geoms understand the aesthetics `x`, `y`, `fill`, `colour`, `linewidth`, `linetype`, `alpha`, `width`, `height`, `angle` (degrees, counter-clockwise), `radius` (a proportion of the cell's own radius, so 0.5 draws a half-size glyph in the same cell), `border` (a per-cell outline width in mm, overriding `linewidth`), `eye_size`, and `eye_colour` (the latter two only matter when `eyes = TRUE`), plus `tooltip`, `data_id` and `onclick`, which are only read when `interactive = TRUE`. At `angle = 0` the yin fish is the left half of the circle plus the top bulb (its head); the yang fish is the right half plus the bottom bulb.

## Value

A `ggplot2` layer drawing one fish per cell.

## Examples

```
library(ggplot2)
d <- data.frame(x = 1:3, y = 1, value = 1:3)

# a yin-only plot with an ordinary fill scale
ggplot(d, aes(x, y)) +
  geom_yin_fish(aes(fill = value)) +
  scale_fill_viridis_c()

# both fish, manually stacked with ggnewscale
ggplot(d, aes(x, y)) +
  geom_yin_fish(aes(fill = value)) +
  scale_fill_viridis_c(name = "yin") +
  ggnewscale::new_scale_fill() +
  geom_yang_fish(aes(fill = rev(value))) +
  scale_fill_viridis_c(name = "yang", option = "magma")
```

---

pitts\_emojis

*Popular Emojis*

---

## Description

The most popular emoji of a given week in a given category from the Meltwater Tweet sample, as HTML `<img>` tags. The vector is aligned row-for-row with `pitts_tg`, so `pitts_emojis[i]` is the emoji for the week / category combination in row `i` of that data set. The tags are meant to be drawn as rich text, e.g. with `ggtext::geom_richtext()` or `annotate("richtext", ...)`.

## Usage

```
pitts_emojis
```

### Format

A character vector of 270 HTML <img> tags (90 distinct emoji), one per row of `pitts_tg`.

### Note on the image URLs

Each tag points at a remotely hosted PNG on the Emojipedia asset host used when the data was collected in 2020. Those objects are no longer served publicly, so the tags no longer render as pictures on their own. The emoji each tag refers to is still readable from its file name, which ends in the Unicode code point (for example `thinking-face_1f914.png` is U+1F914); substitute your own image paths or the literal emoji characters to draw them.

### Source

The most frequent emoji per week and category in the Meltwater Twitter sample described in `pitts_tg`, processed by the package author.

---

pitts\_tg

*Pittsburgh COVID-related Google & Twitter incidence rates*

---

### Description

A data set containing the 30-week incidence rates of COVID-related categories in the Pittsburgh Metropolitan Statistical Area (MSA), from week 1 beginning June 1, 2020 to week 30, which ended on the last Sunday of the year. The data columns are introduced below. One quick note about the columns of the data set: `week_start` is present for illustration purposes, as a reminder of what the week column counts. In other words, it does not participate in any visualization.

### Usage

`pitts_tg`

### Format

A data frame with 270 rows and 6 columns:

**msa** Metropolitan statistical area (Pittsburgh only).

**week** week 1 to week 30.

**week\_start** The date of the Monday the week starts on.

**category** One of 9 COVID-related categories: Covid, General Virus, Masks, Sanitizing, Social Distancing, Symptoms, Tests, Treatment, Working.

**Twitter** weekly tweets percentage (%) in the MSA falling into each category.

**Google** weekly Google search percentage (%) in the MSA falling into each category.

### Source

Just like `states_tg`, Google is processed from Google Health API, and Twitter from Meltwater, a Twitter vendor. Both data sources are processed by the author of the package.

---

|                |                                       |
|----------------|---------------------------------------|
| remove_padding | <i>Remove ggplot2 default padding</i> |
|----------------|---------------------------------------|

---

## Description

ggplot2 pads both continuous and discrete axes with a little expansion, which can make a taichi grid look like it is floating. `remove_padding()` trims that space. Called with no arguments it inspects the plot it is added to and figures out for itself whether each axis is continuous or discrete; pass "c" (continuous) or "d" (discrete) explicitly to override the detection, e.g. when the axis mapping is a computed expression the plot data cannot answer for.

## Usage

```
remove_padding(x = NULL, y = NULL, ...)
```

## Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x, y</code> | NULL (the default) to auto-detect the scale type of that axis from the plot's data and mapping, "c" for a continuous axis, or "d" for a discrete one. Auto-detection reads the <i>plot's</i> mapping, so name the type explicitly when <code>x / y</code> are mapped in a layer rather than in <code>ggplot()</code> , or when the mapping is a computed expression the plot data cannot answer for.                                                                                                                                                                                     |
| <code>...</code>  | Additional arguments passed on to the underlying <code>ggplot2::scale_x_continuous()</code> / <code>ggplot2::scale_x_discrete()</code> (and <code>y</code> ) calls. They go to <i>both</i> scales, so when the two axes are of different types only arguments that continuous and discrete scales share (name, breaks, labels, guide, ...) can be used here: a continuous-only argument such as <code>n.breaks</code> would be rejected by the discrete scale with an "unused argument" error. For per-axis options, add your own <code>scale_x_*(expand = c(0, 0))</code> call instead. |

## Details

A continuous axis holding dates, date-times or hms times gets the matching scale (`ggplot2::scale_x_date()`, `ggplot2::scale_x_datetime()`, `ggplot2::scale_x_time()`) rather than `ggplot2::scale_x_continuous()`, so its labels stay dates instead of turning into day or second counts.

## Value

An object that, added to a ggplot, replaces both position scales with padding-free ones.

## Examples

```
library(ggplot2)
d <- data.frame(x = 1:3, y = c("a", "b", "c"), yin = 1:3, yang = 3:1)

# auto-detects x as continuous and y as discrete
ggplot(d, aes(x, y)) +
  geom_taichi(yin = yin, yang = yang) +
```

```
remove_padding()

# explicit override, identical result here
ggplot(d, aes(x, y)) +
  geom_taichi(yin = yin, yang = yang) +
  remove_padding(x = "c", y = "d")
```

---

**scale\_taichi***Fill scales for the taichi fish*

---

## Description

Ready-made fill scales carrying ggtaichi's palette conventions, for the places where `geom_taichi()`'s automatic scale choice is not what you want: pass one to its `yin_scale` / `yang_scale` argument, or use it directly with `geom_yin_fish()` / `geom_yang_fish()` when you are stacking scales by hand.

## Usage

```
scale_taichi_yin_c(
  name = ggplot2::waiver(),
  palette = "default",
  colors = NULL,
  colours = NULL,
  ...
)
```

```
scale_taichi_yang_c(
  name = ggplot2::waiver(),
  palette = "default",
  colors = NULL,
  colours = NULL,
  ...
)
```

```
scale_taichi_yin_d(
  name = ggplot2::waiver(),
  palette = "default",
  colors = NULL,
  colours = NULL,
  n = NULL,
  ...
)
```

```
scale_taichi_yang_d(
  name = ggplot2::waiver(),
  palette = "default",
  colors = NULL,
```

```

    colours = NULL,
    n = NULL,
    ...
)

scale_taichi_yin_binned(
  name = ggplot2::waiver(),
  palette = "default",
  colors = NULL,
  colours = NULL,
  ...
)

scale_taichi_yang_binned(
  name = ggplot2::waiver(),
  palette = "default",
  colors = NULL,
  colours = NULL,
  ...
)

scale_taichi_yin_viridis_c(name = ggplot2::waiver(), ...)

scale_taichi_yang_viridis_c(name = ggplot2::waiver(), ...)

scale_taichi_yin_viridis_d(name = ggplot2::waiver(), ...)

scale_taichi_yang_viridis_d(name = ggplot2::waiver(), ...)

```

## Arguments

|                 |                                                                                                                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name            | Legend title. Defaults to the aesthetic's label, as elsewhere in ggplot2.                                                                                                                                                                                                                                                    |
| palette         | The palette pair the ramp is taken from: the name of a <a href="#">taichi_palette()</a> preset, or a list with yin and yang colour vectors, for example the output of <a href="#">taichi_palette_pair()</a> .                                                                                                                |
| colors, colours | An explicit colour vector, used instead of palette. The discrete scales use it as given, one colour per level in order, and interpolate only when there are more levels than colours.                                                                                                                                        |
| ...             | Passed on to the underlying ggplot2 scale ( <a href="#">ggplot2::scale_fill_gradientn()</a> , <a href="#">ggplot2::discrete_scale()</a> , <a href="#">ggplot2::scale_fill_stepsn()</a> or <a href="#">ggplot2::scale_fill_viridis</a> , so limits, breaks, labels, guide, na.value, n.breaks and the rest all work as usual. |
| n               | For the discrete scales, how many colours to take from a preset palette before interpolating; defaults to 5, the length of the built-in ramps. Not used with a list palette or with colors.                                                                                                                                  |

## Details

Each function comes in a yin and a yang form, which differ only in which half of the palette pair they take.

`scale_taichi_yin_c()`, `scale_taichi_yang_c()` Continuous gradients, the same construction `geom_taichi()` builds automatically for numeric sources.

`scale_taichi_yin_d()`, `scale_taichi_yang_d()` Discrete scales that sample the palette's ramp for however many levels the data has, skipping its palest end so no category is invisible on a white panel: the rule `geom_taichi()` applies to its default colours and to a palette for factor, character and logical sources. An explicit colors vector is used as given instead, again as in `geom_taichi()`, so a qualitative palette keeps its own colours.

`scale_taichi_yin_binned()`, `scale_taichi_yang_binned()` Binned scales: the fill is matched to one of a handful of discrete steps instead of to a position on a continuous luminance ramp.

`scale_taichi_yin_viridis_c()` **and friends** The Mako and Rocket viridis-family ramps, which are close to luminance matched and stay ordered under colour-vision deficiency.

## Value

A ggplot2 fill scale.

## Why binned scales are the cheapest accuracy win

Reading a value off a continuous luminance ramp is the least accurate perceptual task there is, and it gets worse as a grid grows. Matching a patch to one of five labelled bins is much closer to a categorical lookup, and the legend then tells the reader exactly which values share a colour. On any grid too dense to compare cell by cell (roughly, once the glyphs are smaller than a few millimetres), binning both fish is the single cheapest thing you can do for readability:

```
geom_taichi(yin = matcha, yang = espresso,
            yin_scale = scale_taichi_yin_binned(n.breaks = 5),
            yang_scale = scale_taichi_yang_binned(n.breaks = 5),
            shared_limits = TRUE)
```

`shared_limits` and `shared_legend` compose with all of these: the limits `ggtaichi` computes are pushed into the scale you supply, so the two fish end up with the same breaks and equal values land in the same bin.

## See Also

`taichi_palette()` and `taichi_palette_pair()` for the palettes themselves, `taichi_check_palette()` to check a pair is fair.

## Examples

```
library(ggplot2)
d <- data.frame(x = rep(1:4, 4), y = rep(1:4, each = 4),
               yin = 1:16, yang = 16:1)
```

```
# binned fills, matched limits: the cheapest readability win there is
ggplot(d, aes(x, y)) +
  geom_taichi(yin = yin, yang = yang,
             yin_scale = scale_taichi_yin_binned(n.breaks = 4),
             yang_scale = scale_taichi_yang_binned(n.breaks = 4),
             shared_limits = TRUE)

# a viridis-family pair
ggplot(d, aes(x, y)) +
  geom_taichi(yin = yin, yang = yang,
             yin_scale = scale_taichi_yin_viridis_c,
             yang_scale = scale_taichi_yang_viridis_c)
```

states\_tg

*States' COVID-related Google & Twitter incidence rates*

## Description

A data set containing the 31-week incidence rates of COVID-related categories in 4 states (Florida, Missouri, New York, and Texas), from week 1 beginning June 1, 2020 to week 31, which begins December 28, 2020 and so runs a few days past the end of the year. The data columns are introduced below. One quick note about the columns of the data set: `week_start` is present for illustration purposes, as a reminder of what the week column counts. In other words, it does not participate in any visualization.

## Usage

```
states_tg
```

## Format

A data frame with 1116 rows and 6 columns:

**state** One of the four states: Florida, Missouri, New York, Texas.

**week** week 1 to week 31.

**week\_start** The date of the Monday the week starts on.

**category** One of 9 COVID-related categories: Covid, General Virus, Masks, Sanitizing, Social Distancing, Symptoms, Tests, Treatment, Working.

**Twitter** weekly tweets percentage (%) in state falling into each category.

**Google** weekly Google search percentage (%) in state falling into each category.

## Source

Just like `pitts_tg`, Google is processed from Google Health API, and Twitter from Meltwater, a Twitter vendor. Both data sources are processed by the author of the package.

---

taichi\_check\_palette    *Measure whether two fill ramps are a fair pair*

---

## Description

The taichi's whole purpose is to compare two sources inside one mark, and the fills are what carries the comparison. If the two ramps do not span the same luminance range then equal values do not produce equal visual weight, and one fish appears to dominate wherever the data says the two sources are level. `taichi_check_palette()` measures that, so the question can be settled with numbers instead of taste.

## Usage

```
taichi_check_palette(
  yin_colors = NULL,
  yang_colors = NULL,
  n = 9,
  palette = NULL,
  tolerance = 5
)
```

## Arguments

|                                      |                                                                                                                                                                                       |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>yin_colors, yang_colors</code> | The two colour vectors to compare. Defaults to the package's built-in ramps (or, if <code>palette</code> is given, to that preset's ramps).                                           |
| <code>n</code>                       | Number of steps at which to sample each ramp. Nine gives a readable table and catches mid-ramp problems that the control points alone would hide.                                     |
| <code>palette</code>                 | Optionally the name of a preset (see <a href="#">taichi_palette()</a> ) or a list with yin and yang elements, checked instead of <code>yin_colors</code> / <code>yang_colors</code> . |
| <code>tolerance</code>               | Largest luminance difference, in $L^*$ units, still counted as a pass: a single non-negative number.                                                                                  |

## Details

Called with no arguments it measures the package's own defaults, which is worth doing once: the default grey yin ramp spans the full luminance range (100 down to 0) while the default red yang ramp spans roughly 89 to 41, a mismatch of about 41  $L^*$  units at the dark end. The defaults are kept for continuity, but `palette = "balanced"` (or any pair from [taichi\\_palette\\_pair\(\)](#)) is the more defensible choice for a published comparison.

## Value

An object of class `taichi_palette_check`, with a `print()` method that lays the measurements out as a table. It is a list with elements `steps` (a data frame of per-step colours, luminance and chroma), `max_luminance_diff`, `max_chroma_diff`, `monotone` (a logical pair), `verdict` ("pass",

"warning" or "fail"), `cvd` (a data frame of median step-wise colour distances for normal vision and each simulation, or NULL when **colorspace** is not installed), `tolerance`, and `space`, naming the colour space every number was measured in (they are not comparable with figures computed in another space).

### How the verdict is decided

Both ramps are resampled to `n` steps in Lab space (the space `ggplot2`'s gradient scales interpolate in), and each step's CIE  $L^*$  (luminance) and chroma are recorded. The reported mismatch is the largest absolute luminance difference between corresponding steps. The verdict is "pass" below tolerance  $L^*$  units (default 5, around the point where a difference between two large patches becomes noticeable), "warning" up to three times that, and "fail" above it. A ramp whose luminance is not monotone is reported separately: a sequential fill that lightens and then darkens has no readable order, whatever its match.

When **colorspace** is installed, the two ramps are also simulated under deuteranopia, protanopia and tritanopia, and the median CIE2000 distance between corresponding steps is reported for each, alongside the same figure for normal vision. Read it as a comparison: a simulation much below the normal-vision row means the deficiency is costing those readers the distinction, and anything below about 10 means the two fish are not tellable apart at all. The median rather than the minimum is deliberate: two luminance-matched ramps necessarily converge at their pale end, where both are near white, and a minimum would report that as a fault of every well-matched pair.

### See Also

`taichi_palette_pair()` to build a matched pair, `taichi_palette()` for the presets.

### Examples

```
# the package defaults, measured honestly
taichi_check_palette()

# a matched pair
taichi_check_palette(palette = "balanced")
```

---

|                |                          |
|----------------|--------------------------|
| taichi_palette | Ready-made palette pairs |
|----------------|--------------------------|

---

### Description

The named palette pairs that `geom_taichi()`'s `palette` argument accepts, available on their own so they can be inspected, checked with `taichi_check_palette()`, or used with the individual fish geoms.

### Usage

```
taichi_palette(name = "balanced", n = 5)
```

**Arguments**

|      |                                                                                                                               |
|------|-------------------------------------------------------------------------------------------------------------------------------|
| name | Name of the preset: one of "default", "balanced", "diverging", "viridis_pair", "brewer_pair", "greyscale_safe".               |
| n    | Number of colours per ramp; ramps of a fixed length are interpolated (in Lab space) when n differs from their natural length. |

**Value**

A list of two character vectors of hex colours, yin and yang.

**The presets**

"default" The package's own grey yin / seal-red yang ramps, the look of every ggtaichi release so far. It is *not* luminance matched (the grey ramp spans the full range, the red one does not); run `taichi_check_palette()` to see by how much. Kept as the default for continuity, not because it is the most defensible choice.

"balanced" Two HCL ramps from `taichi_palette_pair()`, matched in luminance and chroma and separated only by hue (blue yin, brick-red yang). The recommended choice when the two sources are directly comparable.

"diverging" As "balanced", but both ramps reach a shared near-white light end, so the two fish read as the two arms of one diverging scale. Use it with `shared_limits = TRUE`.

"viridis\_pair" The Mako and Rocket ramps, reversed to run light to dark. They come from the same generator and are close to luminance matched, and each ramp on its own stays ordered under colour-vision deficiency. The two are harder to tell apart from *each other* under red-green deficiency than "balanced" is, though, so run `taichi_check_palette(palette = "viridis_pair")` and look at the protan row before choosing it.

"brewer\_pair" ColorBrewer's sequential Blues and Oranges. Familiar and print-friendly; less exactly matched than "balanced".

"greyscale\_safe" A grey yin ramp and a hued yang ramp on the *same* luminance trajectory. In colour the two fish are told apart by hue; in greyscale they collapse to the same ink, so equal values still read as equal, and the two sources are then distinguished by their position in the glyph (yin is the top bulb, yang the bottom). The right choice for a journal figure that may be printed in black and white. Note the promise precisely: it is about **greyscale**, not about the CMYK gamut. A hued ramp can still shift when converted for offset printing; if that matters, soft-proof the figure.

**See Also**

`taichi_palette_pair()`, `taichi_check_palette()`

**Examples**

```
taichi_palette("balanced")
taichi_palette("greyscale_safe", n = 3)

# every preset, measured
for (p in c("default", "balanced", "diverging", "viridis_pair",
```

```

    "brewer_pair", "greyscale_safe")) {
  cat(p, ": max |dL| = ",
      round(taichi_check_palette(palette = p)$max_luminance_diff, 1),
      "\n", sep = "")
}

```

---

taichi\_palette\_pair      *Build a luminance-matched pair of sequential palettes*

---

## Description

The two fish of a taichi symbol are compared *against each other*, so the two fill ramps have to be matched: if one ramp spans a wider luminance range than the other, equal values do not produce equal visual weight and one fish systematically appears to dominate. That makes palette pairing a correctness problem rather than a matter of taste: see [taichi\\_check\\_palette\(\)](#) for the measurement and vignette("ggtaichi") for the discussion.

## Usage

```

taichi_palette_pair(
  n = 5,
  hues = c(250, 20),
  luminance = c(30, 90),
  chroma = 60
)

```

## Arguments

|           |                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Number of colours per ramp. The default, 5, matches the length of the built-in <code>yin_colors</code> / <code>yang_colors</code> vectors.                                                                                                                                                                                                                                              |
| hues      | Two hues in degrees, <code>c(yin, yang)</code> , on the HCL colour wheel (0 red, 120 green, 250 blue). Pick hues far apart; opposite hues (a difference near 180) are the most distinguishable.                                                                                                                                                                                         |
| luminance | The two ends of the shared luminance trajectory as <code>c(dark, light)</code> , on the CIE $L^*$ scale from 0 (black) to 100 (white). Both ramps run from the light end to the dark end, so the first colour belongs to the lowest value, the same convention as <code>yin_colors</code> / <code>yang_colors</code> .                                                                  |
| chroma    | Chroma (colourfulness) at the dark end of each ramp. Chroma tapers towards the light end, because a very light colour cannot also be saturated; both ramps taper identically. Values above about 80 will be clipped to the sRGB gamut, and clipping is hue-dependent, which is exactly what breaks the match. Keep it moderate and verify with <a href="#">taichi_check_palette()</a> . |

## Details

`taichi_palette_pair()` constructs two sequential ramps in the perceptual HCL space that differ **only** in hue: they share one luminance trajectory and one chroma trajectory, so step *i* of the yin ramp and step *i* of the yang ramp carry the same visual weight.

**Value**

A list of two character vectors of hex colours, `yin` and `yang`, each of length `n`. Pass them straight to `geom_taichi()` as `yin_colors` / `yang_colors`, or hand the whole list to its `palette` argument.

**See Also**

`taichi_check_palette()` to measure any pair, `taichi_palette()` for the ready-made presets.

**Examples**

```
pair <- taichi_palette_pair()
pair

# the pair is matched by construction; the package defaults are not
taichi_check_palette(pair$yin, pair$yang)

library(ggplot2)
d <- data.frame(x = rep(1:3, 3), y = rep(1:3, each = 3),
                yin = 1:9, yang = 9:1)
ggplot(d, aes(x, y)) +
  geom_taichi(yin = yin, yang = yang, palette = pair, shared_limits = TRUE)
```

---

taichi\_summary

---

*Summarise the two sources cell by cell*


---

**Description**

A taichi grid is a *superposition* comparison: the two sources share one position, which makes "are these similar?" and "which is bigger here?" easy to see and "by how much?" impossible. `taichi_summary()` is the tidy answer to the last question: the numbers behind the glyph, one row per input row, for the reader who needs a table rather than a picture.

**Usage**

```
taichi_summary(data, yin, yang, x = NULL, y = NULL)
```

**Arguments**

|                        |                                                                                                                                                                                            |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>      | A data frame, the same one passed to <code>ggplot2::ggplot()</code> .                                                                                                                      |
| <code>yin, yang</code> | Unquoted column names (or strings naming columns) for the two sources, exactly as in <code>geom_taichi()</code> .                                                                          |
| <code>x, y</code>      | Optional unquoted column names identifying each cell. When given they are carried through to the output as the first columns, which makes the result joinable back onto the plotting data. |

**Value**

A data frame with one row per row of data: the optional `x` and `y` cell identifiers, `yin`, `yang`, `difference`, `ratio`, `log_ratio`, `z`, `dominant` and `rank`.

### The statistics

`difference` `yin - yang`. Positive means the yin fish (the top bulb) carries the larger value.

`ratio` `yin / yang`, and NA wherever either value is not strictly positive (a ratio of a negative or zero quantity is not a ratio), so Inf is never returned.

`log_ratio` `log2(yin / yang)`, so a value of 1 means yin is twice yang and -1 means half. Symmetric around zero, which `ratio` is not, and the right choice when the two sources span orders of magnitude.

`z` The difference of the two standardised sources: each column is centred and scaled across the whole grid, then subtracted. This is the statistic to use when the two sources are *not* in the same units, because it asks "which source is unusually high here, relative to its own spread?" rather than "which number is bigger?".

`dominant` Which source is larger in that cell, named after the columns supplied, or "tie".

`rank` The cell's rank by size of `abs(difference)`, 1 being the widest gap. Cells with a missing difference get NA.

### A caveat on rank

A grid of 96 cells is 96 implicit comparisons, and the most extreme cell in a grid that size is frequently the most extreme *noise* rather than the largest real effect. Treat `rank` as a list of places to look, not as a list of findings: taking the top-ranked cell and reporting it is the multiple-comparisons problem in miniature. With one observation per cell there is no per-cell test to correct with, so the honest check is on the field as a whole.

### See Also

`geom_taichi()`'s explicit argument, which shows one of these statistics inside the glyph, and `geom_taichi_diff()`, which draws it as a diverging heatmap.

### Examples

```
summ <- taichi_summary(cafes_tg, yin = matcha, yang = espresso,
                       x = week, y = neighbourhood)
head(summ)

# the five widest gaps: places to look, not findings (see the caveat above)
head(summ[order(summ$rank), ], 5)
```

---

theme\_taichi

Plot Themes

---

### Description

A light theme tuned for the taichi grid: it bottoms the legends, drops the panel grid and axis ticks, and gives the canvas a soft off-white background reminiscent of rice paper.

### Usage

```
theme_taichi(  
  base_size = 11,  
  base_family = "",  
  base_line_size = base_size/22,  
  base_rect_size = base_size/22  
)
```

### Arguments

|                |                                                             |
|----------------|-------------------------------------------------------------|
| base_size      | base font size; every text size in the theme scales with it |
| base_family    | base font family                                            |
| base_line_size | base size for line elements                                 |
| base_rect_size | base size for rect elements                                 |

### Value

A [theme](#) object that can be added to any ggplot, in the same way as [theme\\_bw\(\)](#).

### Opinionated choices

Two of the theme's settings surprise people often enough to be worth spelling out. The y axis *title* is blanked, on the assumption that the y axis of a taichi grid is a list of category names that already reads as a label, so `labs(y = "...")` has no visible effect under this theme. Legend text is rotated 90 degrees, which keeps a wide continuous legend from running off the bottom of the plot. Both are ordinary theme elements, so add a [theme\(\)](#) call afterwards to put them back:

```
+ theme_taichi() + theme(axis.title.y = element_text(),  
                          legend.text = element_text(angle = 0))
```

### Examples

```
library(ggplot2)  
d <- data.frame(x = 1:3, y = 1:3, yin = 1:3, yang = 3:1)  
  
ggplot(d, aes(x, y)) +  
  geom_taichi(yin = yin, yang = yang) +  
  theme_taichi()
```

# Index

## \* datasets

- cafes\_tg, [2](#)
- pitts\_emojis, [15](#)
- pitts\_tg, [16](#)
- states\_tg, [21](#)

aes, [5](#), [9](#)

cafes\_tg, [2](#)

draw\_key\_taichi, [3](#), [7](#)

draw\_key\_taichi(), [14](#)

geom\_taichi, [4](#)

geom\_taichi(), [3](#), [12](#), [13](#), [18](#), [20](#), [23](#), [26](#), [27](#)

geom\_taichi\_diff, [10](#), [12](#)

geom\_taichi\_diff(), [27](#)

geom\_yang\_fish, [8](#), [9](#)

geom\_yang\_fish(geom\_yin\_fish), [13](#)

geom\_yang\_fish(), [3](#), [18](#)

geom\_yin\_fish, [8](#), [9](#), [13](#)

geom\_yin\_fish(), [3](#), [18](#)

ggnewscale::new\_scale\_fill(), [13](#)

ggplot, [8](#)

ggplot2::discrete\_scale(), [19](#)

ggplot2::geom\_tile(), [12](#), [13](#)

ggplot2::ggplot(), [13](#), [26](#)

ggplot2::guide\_colourbar(), [3](#)

ggplot2::layer(), [14](#), [15](#)

ggplot2::scale\_fill\_gradientn(), [19](#)

ggplot2::scale\_fill\_stepsn(), [19](#)

ggplot2::scale\_fill\_viridis\_c(), [19](#)

ggplot2::scale\_x\_continuous(), [17](#)

ggplot2::scale\_x\_date(), [17](#)

ggplot2::scale\_x\_datetime(), [17](#)

ggplot2::scale\_x\_discrete(), [17](#)

ggplot2::scale\_x\_time(), [17](#)

layer, [7](#)

pitts\_emojis, [15](#)

pitts\_tg, [15](#), [16](#), [16](#)

remove\_padding, [17](#)

scale\_fill\_gradientn, [8](#)

scale\_fill\_manual, [8](#)

scale\_taichi, [18](#)

scale\_taichi\_yang\_binned

(scale\_taichi), [18](#)

scale\_taichi\_yang\_c(scale\_taichi), [18](#)

scale\_taichi\_yang\_d(scale\_taichi), [18](#)

scale\_taichi\_yang\_viridis\_c

(scale\_taichi), [18](#)

scale\_taichi\_yang\_viridis\_d

(scale\_taichi), [18](#)

scale\_taichi\_yin\_binned(scale\_taichi),  
[18](#)

scale\_taichi\_yin\_c(scale\_taichi), [18](#)

scale\_taichi\_yin\_d(scale\_taichi), [18](#)

scale\_taichi\_yin\_viridis\_c

(scale\_taichi), [18](#)

scale\_taichi\_yin\_viridis\_d

(scale\_taichi), [18](#)

states\_tg, [21](#)

taichi\_check\_palette, [10](#), [22](#)

taichi\_check\_palette(), [20](#), [23–26](#)

taichi\_palette, [5](#), [23](#)

taichi\_palette(), [12](#), [19](#), [20](#), [22](#), [23](#), [26](#)

taichi\_palette\_pair, [5](#), [10](#), [25](#)

taichi\_palette\_pair(), [19](#), [20](#), [22–24](#)

taichi\_summary, [6](#), [9](#), [10](#), [26](#)

taichi\_summary(), [12](#), [13](#)

theme, [28](#)

theme\_bw, [28](#)

theme\_taichi, [27](#)