Package: hueR (via r-universe)

September 16, 2024

Title Create Grouped Palettes from Hue values

Version 0.0.0.9000

Description .

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports dplyr, ggplot2, colorspace

Suggests scales, ggfittext, gapminder, devtools

Repository https://david-barnett.r-universe.dev

RemoteUrl https://github.com/david-barnett/hueR

RemoteRef HEAD

RemoteSha dbb1c3c4b3108d3150e7d750c1edaf15b3fff993

Contents

`hueGroupPal` *Make HCL palette for groups with multiple levels*

Description

Makes a palette for dataframe where levels within groups defined by the `group` variable share the same hue but different shades, levels within group based on the `shade` variable.

Usage

```
hueGroupPal(
  df,
  group,
  shade,
  maxShades = 5,
  hues = hueSet(),
  huePalFun = huePal(),
  manual = c(Other = "lightgrey")
)
```

Arguments

`df`	dataframe with at least two variables (treated as categories)
`group`	variable name used to assign hues
`shade`	variable name used to assign colour shades of same hue
`maxShades`	maximum allowed number of shades per hue
`hues`	hues available to use for unique levels of `group` variable
`huePalFun`	function used to create single hue palette for levels of `shade` variable
`manual`	NULL or manual additions or replacements for returned palette in the style of `c(name = value, ....)`

Value

named character vector of colours

Examples

```
library(dplyr)
library(ggplot2)

# sort countries, within continents, by average population
sortedSummary <- gapminder::gapminder %>%
  group_by(continent, country) %>%
  summarise(AvPop = mean(pop, na.rm = TRUE), .groups = "keep") %>%
  group_by(continent) %>%
  arrange(.by_group = TRUE, desc(AvPop))
```

```
# create palettes
countryPal6 <- sortedSummary %>%
  hueGroupPal(group = "continent", shade = "country", maxShades = 6)


# plot population per year
gapminder::gapminder %>%
  group_by(year) %>%
  ggplot(aes(
    x = factor(year), y = pop,
    # setting as factor with levels in correct order ensures ordering of bars
    fill = factor(country, levels = names(countryPal6))
  )) +
  geom_col() +
  guides(fill = "none") +
  # setting manual scale of course sets correct colours
  scale_fill_manual(values = countryPal6) +
  ggfittext::geom_fit_text(
    aes(ymin = 0, ymax = pop, label = country),
    position = "stack", colour = "white"
  ) +
  theme_classic() +
  coord_cartesian(expand = FALSE)


# plot population per year as share of world total that year
gapminder::gapminder %>%
  group_by(year) %>%
  mutate(popPerc = pop/sum(pop, na.rm = TRUE)) %>%
  ggplot(aes(
    x = factor(year), y = popPerc,
    # setting as factor with levels in correct order ensures ordering of bars
    fill = factor(country, levels = names(countryPal6))
  )) +
  geom_col() +
  guides(fill = "none") +
  # setting manual scale of course sets correct colours
  scale_fill_manual(values = countryPal6) +
  ggfittext::geom_fit_text(
    aes(ymin = 0, ymax = popPerc, label = country),
    position = "stack", colour = "white"
  ) +
  theme_classic() +
  coord_cartesian(expand = FALSE)


# plot with modified palette
countryPal6alt <- sortedSummary %>%
  hueGroupPal(group = "continent", shade = "country", maxShades = 6,
              hues = hueSet(start = 0))

gapminder::gapminder %>%
  group_by(year) %>%
  mutate(popPerc = pop/sum(pop, na.rm = TRUE)) %>%
  # dplyr::filter(year > 1970) %>%
  ggplot(aes(
    x = factor(year), y = popPerc,
    # setting as factor with levels in correct order ensures ordering of bars
    fill = factor(country, levels = names(countryPal6alt))
  )) +
  geom_col() +
  guides(fill = "none") +
  # setting manual scale of course sets correct colours
  scale_fill_manual(values = countryPal6alt) +
  ggfittext::geom_fit_text(grow = TRUE,
                           aes(ymin = 0, ymax = popPerc, label = country),
                           position = "stack", colour = "white"
  ) +
  theme_classic() +
  coord_cartesian(expand = FALSE)
```

huePal *Create (named) single-hue gradient palette*

Description

Function to create create HCL palette of n colours based on fixed hue. Luminance monotonically increases, whilst chroma increases and then decreases.

If names provided: return named palette of same length as unique(names), with n distinct colours (if n is left null, all colours are unique)

If the hue value is left as NULL, a function will be returned, which can generate a palette when given a hue values and n and/or names

Usage

```
huePal(
  hue = NULL,
  names = NULL,
  n = NULL,
  minChroma = 40,
  maxChroma = 150,
  minLum = 10,
  maxLum = 98,
  power = 0.8
)
```

Arguments

hue	numeric hue value, or NULL to return palette function
names	names for palette, or NULL for unnamed palette of length n

`n`	number of unique shades in palette
`minChroma`	minimum Chroma for palette
`maxChroma`	maximum Chroma for palette
`minLum`	minimum luminance for palette
`maxLum`	maximum luminance for palette
`power`	power used by `colorspace::sequential_hcl()`

Details

Uses `colorspace::sequential_hcl()` with a fixed hue to generate palettes.

The palette generated will be roughly centered around the midpoint of the luminance range, and at approximately the maximum chroma.

Note that edges of generated palette are cut off so min and max luminances are never returned. This is because they are, by default, too dark/light to be distinguishable across hues.

Value

vector of colours, possibly named, or a function

Examples

```
pal <- huePal(hue = 120, n = 9)
scales::show_col(pal, borders = NA)
huePal(hue = 120, names = letters[1:9]) == huePal(hue = 120, n = 9)

# more names than shades --> repeats last shade
extendedPal <- huePal(hue = 120, names = letters[1:16], n = 9)
scales::show_col(extendedPal, borders = NA)
```

`hueSet` *Get a set of equally spaced hues values*

Description

Rotates around the 360 degrees of hue on the HCL colour wheel,

Starts at `start` and rotates `cycles` times around the wheel to obtain `n` colours at equal intervals

Usage

```
hueSet(n = 10, start = 180, cycles = 2)
```

Arguments

`n`	number of hues to return
`start`	starting hue value, in degrees from 0 to 359
`cycles`	number of cycles rotating around the

Value

vector

Examples

```
hueSet()

hueSet(cycles = 1)

hueSet(start = 0, cycles = 1)

hueSet(start = 0, n = 9, cycles = 3)
```

---

mergeAfterN *Merge values of vector after first n unique values*

Description

Merge values of vector after first n unique values

Usage

```
mergeAfterN(x, n, other = "other")
```

Arguments

x	vector
n	number of unique values/levels to keep
other	name of new value/level to replace excess values with

Value

vector

Examples

```
library(dplyr)
letters %>% mergeAfterN(15)
LETTERS %>% mergeAfterN(10, other = "?")

## Real data example ##
# works for factors too
gapminder::gapminder %>%
  dplyr::filter(year < 1970) %>%
  dplyr::pull(country) %>%
  mergeAfterN(10) %>%
  head(50)
```

`rep_last` *Repeat last value in vector to create longer vector*

Description

Helper function

Usage

```
rep_last(x, length.out)
```

Arguments

`x`	vector
`length.out`	desired length of vector

Examples

```
rep_last(letters[1:10], length.out = 15)
```

Index