Package 'colorist'

November 23, 2020

Title Coloring Wildlife Distributions in Space-Time

Version 0.1.2

Description Color and visualize wildlife distributions in space-time using raster data. In addition to enabling display of sequential change in distributions through the use of small multiples, 'colorist' provides functions for extracting several features of interest from a sequence of distributions and for visualizing those features using HCL (hue-chroma-luminance) color palettes. Resulting maps allow for ``fair'' visual comparison of intensity values (e.g., occurrence, abundance, or density) across space and time and can be used to address questions about where, when, and how consistently a species, group, or individual is likely to be found.

License GPL-3

URL https://github.com/mstrimas/colorist

BugReports https://github.com/mstrimas/colorist/issues

Depends R (>= 3.2.0)

Imports colorspace, ggplot2, grDevices, magrittr, raster, scales, stats, tidyr

Suggests knitr, RColorBrewer, rmarkdown, sf, rnaturalearth, tigris

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Justin Schuetz [aut] (<https://orcid.org/0000-0002-6163-538X>), Matthew Strimas-Mackey [aut, cre] (<https://orcid.org/0000-0001-8929-7776>), Tom Auer [aut] (<https://orcid.org/0000-0001-8619-7147>)

Maintainer Matthew Strimas-Mackey <mes335@cornell.edu>

Repository CRAN

Date/Publication 2020-11-23 20:10:07 UTC

R topics documented:

`elephant_ud` *African Elephant utilization distributions*

Description

A RasterStack of utilization distributions for two individual African Elephants in Etosha National Park in 2011. Cell values represent the probability density that an elephant was found at a given location within the year and the two layers contain data for the two individual elephants. Utilization distributions were generated using the `adehabitatHR` package from GPS tracking data. W. Kilian, W.M. Getz, R. Zidon, and M. Tsalyuk graciously provided permission to use their data for visualization purposes.

Usage

```
elephant_ud
```

Format

An object of class `RasterStack` of dimension 208 x 193 x 2.

Source

`https://www.datarepository.movebank.org/handle/10255/move.812`

References

Tsalyuk, M., W. Kilian, B. Reineking, W. Marcus. 2018. Temporal variation in resource selection of African elephants follows long term variability in resource availability. Ecological Monographs. https://doi.org/10.1002/ecm.1348

Kilian, W., W.M. Getz, R. Zidon, M. Tsalyuk. 2018. Data from: Temporal variation in resource selection of African elephants follows long term variability in resource availability. Movebank Data Repository. https://doi.org/10.5441/001/1.3nj3qj45

`fiespa_occ` *eBird Status & Trends Field Sparrow occurrence probability*

Description

A RasterStack of the expected probability of occurrence of Field Sparrow from the eBird Status & Trends project. Each of the 12 layers in the stack represent the estimated occurrence for a given month of the year over a regular grid of points covering the full range of the species. To reduce file size, these data have been aggregated from the native 2.96 km spatial resolution and weekly temporal resolution to monthly, 14.8 km resolution.

Usage

```
fiespa_occ
```

Format

An object of class `RasterStack` of dimension 193 x 225 x 12.

Details

For further details on these data, and to access the data for more species, consult the documentation for the `ebirdst` package.

Source

https://ebird.org/science/status-and-trends

References

Fink, D., T. Auer, A. Johnston, M. Strimas-Mackey, O. Robinson, S. Ligocki, B. Petersen, C. Wood, I. Davies, B. Sullivan, M. Iliff, S. Kelling. 2020. eBird Status and Trends, Data Version: 2018; Released: 2020. Cornell Lab of Ornithology, Ithaca, New York. https://doi.org/10.2173/ebirdst.2018

`fisher_ud` *Fisher utilization distributions*

Description

A RasterStack of utilization distributions for a single Fisher in New York state over the course of nine nights in April 2011. Cell values represent the probability density that the individual was found at a given location between sunset and sunrise and the nine layers represent nine nights of data. Utilization distributions were generated using the `adehabitatHR` package from GPS tracking data.

Usage

```
fisher_ud
```

Format

An object of class `RasterStack` of dimension 176 x 177 x 9.

Source

`https://www.datarepository.movebank.org/handle/10255/move.330`

References

LaPoint, S., P. Gallery, M. Wikelski, R. Kays. 2013. Animal behavior, cost-based corridor models, and real corridors. Landscape Ecology 28(8): 1615–1630. `https://doi.org/10.1007/s10980-013-9910-0`

LaPoint, S., P. Gallery, M. Wikelski, R. Kays. 2013. Data from: Animal behavior, cost-based corridor models, and real corridors. Movebank Data Repository. `https://doi.org/10.5441/001/1.2tp2j43g`

`legend_set` *Make an HCL legend for an unordered set of distributions*

Description

This function creates a legend to accompany a map describing an unordered set of distributions.

Usage

```
legend_set(
  palette,
  specificity = TRUE,
  group_labels = NULL,
  label_i = "Maximum\nintensity",
  label_s = "Specificity",
  axis_i = c("low", "high"),
  axis_s = c("low", "high"),
  return_df = FALSE
)
```

Arguments

`palette`	data frame containing a color palette generated by palette_set.
`specificity`	logical indicating whether to visualize intensity and layer information for the full range of potential specificity values (i.e., 0-100) or for a single specificity value (i.e., 100). Typically, a single specificity value is appropriate for map_multiples visualizations.
`group_labels`	(axis_l) character vector with labels for each distribution.
`label_i`	character vector with a single element describing the meaning of specificity.
`label_s`	character vector with a single element describing the meaning of intensity values.
`axis_i`	character vector with two elements describing the meaning of low and high intensity values.
`axis_s`	character vector with two elements describing the meaning of low and high specificity values.
`return_df`	logical indicating whether to return the legend as a `ggplot2` object or return a data frame containing the necessary data to build the legend.

Value

A `ggplot2` plot object of the legend. Alternatively, `return_df = TRUE` will return a data frame containing a data frame containing the data needed to build the legend. The data frame columns are:

- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `layer_id`: integer identifying the layer containing the maximum intensity value; mapped to hue.
- `color`: the hexadecimal color associated with the given layer and specificity values.
- `intensity`: maximum cell value across layers divided by the maximum value across all layers and cells; mapped to alpha level.

See Also

legend_timecycle for cyclical sequences of distributions and legend_timeline for linear sequences of distributions.

Other legend: `legend_timecycle()`, `legend_timeline()`

Examples

```
# load elephant data
data(elephant_ud)

# generate hcl palette
pal <- palette_set(elephant_ud)

# create legend for palettes
legend_set(pal)
```

`legend_timecycle` *Make an HCL legend for a cyclical sequence of distributions*

Description

This function creates a legend to accompany a map describing a cyclical sequence of distributions.

Usage

```
legend_timecycle(
  palette,
  specificity = TRUE,
  origin_label = NULL,
  label_i = "Maximum\nintensity",
  label_l = "Layer",
  label_s = c("Low specificity", "Moderate specificity", "High specificity"),
  return_df = FALSE
)
```

Arguments

`palette`	data frame containing a color palette generated by palette_timecycle.
`specificity`	logical indicating whether to visualize intensity and layer information for three specificity values (i.e., 0, 50, 100) or for a single specificity value (i.e., 100). Typically, a single specificity value is appropriate for map_multiples visualizations.
`origin_label`	character vector with a single element to be used as the label at the 12 o'clock position of the legend wheel.
`label_i`	character vector with a single element describing the meaning of intensity values.

`label_l`	character vector with a single element describing the meaning of layer values.
`label_s`	character vector with three elements describing differences in the meaning of three specificity values (i.e., 0, 50, 100).
`return_df`	logical indicating whether to return the legend as a `ggplot2` object or return a data frame containing the necessary data to build the legend.

Value

A `ggplot2` plot object of the legend. Alternatively, `return_df = TRUE` will return a data frame containing the data needed to build the legend. The data frame columns are:

- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `layer_id`: integer identifying the layer containing the maximum intensity value; mapped to hue.
- `color`: the hexadecimal color associated with the given layer and specificity values.
- `intensity`: maximum cell value across layers divided by the maximum value across all layers and cells; mapped to alpha level.

See Also

legend_timeline for linear sequences of distributions and legend_set for distributions of distinct groups.

Other legend: `legend_set()`, `legend_timeline()`

Examples

```
# load field sparrow data
data(fiespa_occ)

# generate hcl palette
pal <- palette_timecycle(fiespa_occ)

# create legend for palette
legend_timecycle(pal)
```

`legend_timeline` *Make an HCL legend for a linear sequence of distributions*

Description

This function creates a legend to accompany a map describing a linear sequence of distributions.

Usage

```
legend_timeline(
  palette,
  specificity = TRUE,
  time_labels = NULL,
  label_i = "Maximum\nintensity",
  label_l = "Layer",
  label_s = c("Low specificity", "Moderate specificity", "High specificity"),
  axis_i = c("low", "high"),
  return_df = FALSE
)
```

Arguments

`palette`	data frame containing a color palette generated by palette_timeline.
`specificity`	logical indicating whether to visualize intensity and layer information for three specificity values (i.e., 0, 50, 100) or for a single specificity value (i.e., 100). Typically, a single specificity value is appropriate for map_multiples visualizations.
`time_labels`	character vector with two elements to be used as labels for the start and end points of the time axis (i.e. x-axis) in the legend.
`label_i`	character vector with a single element describing the meaning of intensity values.
`label_l`	character vector with a single element describing the meaning of layer values.
`label_s`	character vector with three elements describing differences in the meaning of specificity across three legend wheels.
`axis_i`	character vector with two elements describing the meaning of low and high intensity values.
`return_df`	logical indicating whether to return the legend as a `ggplot2` object or return a data frame containing the necessary data to build the legend.

Value

A `ggplot2` plot object of the legend. Alternatively, `return_df = TRUE` will return a data frame containing a data frame containing the data needed to build the legend. The data frame columns are:

- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `layer_id`: integer identifying the layer containing the maximum intensity value; mapped to hue.
- `color`: the hexadecimal color associated with the given layer and specificity values.
- `intensity`: maximum cell value across layers divided by the maximum value across all layers and cells; mapped to alpha level.

See Also

legend_timecycle for cyclical sequences of distributions and legend_set for distributions of distinct groups.

Other legend: `legend_set()`, `legend_timecycle()`

Examples

```
# load fisher data
data(fisher_ud)

# generate hcl palette
pal <- palette_timeline(fisher_ud)

# create legend for palette
legend_timeline(pal)
```

`map_multiples` *Visualize multiple distributions in a series of maps*

Description

This function enables visualization of distributional information in a series of small multiples by combining distribution metrics and an HCL color palette.

Usage

```
map_multiples(
  x,
  palette,
  ncol,
  lambda_i = 0,
  labels = NULL,
  return_type = c("plot", "df")
)
```

Arguments

`x`	RasterStack of distributions processed by `metrics_pull()`.
`palette`	data frame containing an HCL color palette generated using `palette_timecycle()`, `palette_timeline()`, or `palette_set()`.
`ncol`	integer specifying the number of columns in the grid of plots.
`lambda_i`	number that allows visual tuning of intensity values via the `scales::modulus_trans()` function (see Details). Negative numbers increase the opacity of cells with low intensity values. Positive numbers decrease the opacity of cells with low intensity values.
`labels`	character vector of layer labels for each plot. The default is to not show labels.
`return_type`	character specifying whether the function should return a `ggplot2` plot object ("plot") or data frame ("df"). The default is to return a `ggplot2` object.

Details

The `lambda_i` parameter allows for visual tuning of intensity values with unusual distributions. For example, distributions often contain highly skewed intensity values because individuals spend a vast majority of their time within a relatively small area or because populations are relatively dense during some seasons and relatively dispersed during others. This can make visualizing distributions a challenge. The `lambda_i` parameter transforms intensity values via the `scales::modulus_trans()` function, allowing users to adjust the relative visual weight of high and low intensity values.

Value

By default, or when `return_type = "plot"`, the function returns a map that is a `ggplot2` plot object.

When `return_type = "df"`, the function returns a data frame containing eight columns:

- `x,y`: coordinates of raster cell centers.
- `cell_number`: integer indicating the cell number.
- `layer_cell`: a unique ID for the cell within the layer in the format `"layer-cell_number"`.
- `intensity`: maximum cell value across layers divided by the maximum value across all layers and cells; mapped to alpha level.
- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `layer_id`: the identity of the raster layer from which an intensity value was pulled; mapped to hue.
- `color`: the hexadecimal color associated with the given layer and specificity values.

See Also

Other map: `map_single()`

Examples

```
# load fisher data
data("fisher_ud")

# prepare data
r <- metrics_pull(fisher_ud)

# generate palette
pal <- palette_timeline(fisher_ud)

# produce maps, adjusting lambda_i to make areas that were used less
# intensively more conspicuous
map_multiples(r, pal, lambda_i = -5, labels = paste("night", 1:9))
```

map_single *Visualize distributions in a single map*

Description

This function enables visualization of distributional information in a single map by combining distribution metrics and an HCL color palette.

Usage

```
map_single(
  x,
  palette,
  layer,
  lambda_i = 0,
  lambda_s = 0,
  return_type = c("plot", "stack", "df")
)
```

Arguments

`x`	RasterStack of distributions processed by `metrics_pull()` or `metrics_distill()`.
`palette`	data frame containing an HCL color palette generated using `palette_timecycle()`, `palette_timeline()`, or `palette_set()`.
`layer`	integer (or character) corresponding to the layer ID (or name) of layer. A single distribution from within `x` is mapped when the `layer` argument is specified. The `layer` argument is ignored if `metrics_distill()` was used to generate `x`.
`lambda_i`	number that allows visual tuning of intensity values via the `scales::modulus_trans()` function (see Details). Negative numbers increase the opacity of cells with low intensity values. Positive numbers decrease the opacity of cells with low intensity values.
`lambda_s`	number that allows visual tuning of specificity values via the `scales::modulus_trans()` function (see Details). Negative numbers increase the chroma of cells with low specificity values. Positive numbers decrease the chroma of cells with low specificity values.
`return_type`	character specifying whether the function should return a `ggplot2` plot object ("plot"), `RasterStack` ("stack"), or data frame ("df"). The default is to return a `ggplot2` object.

Details

The `lambda_i` parameter allows for visual tuning of intensity values with unusual distributions. For example, distributions often contain highly skewed intensity values because individuals spend a vast majority of their time within a relatively small area or because populations are relatively dense during some seasons and relatively dispersed during others. This can make visualizing distributions a challenge. The `lambda_i` parameter transforms intensity values via the `scales::modulus_trans()` function, allowing users to adjust the relative visual weight of high and low intensity values.

The `lambda_s` parameter allows for visual tuning of specificity values via the `scales::modulus_trans()` function. Adjustment of `lambda_s` affects the distribution of chroma values across areas of relatively low and high specificity, thus modifying information available to viewers. USE WITH CAUTION!

Value

By default, or when `return_type = "plot"`, the function returns a map that is a `ggplot2` plot object.

When `return_type = "stack"`, the function returns a `RasterStack` containing five layers that enable RGBa visualization of a map using other R packages or external GIS software:

- `R`: red, integer values (0-255).
- `G`: green, integer values (0-255).
- `B`: blue, integer values (0-255).
- `alpha`: opacity, numeric values (0-255).
- `n_layers`: number of layers in `x` with non-NA values.

When `return_type = "df"`, the function returns a data frame containing seven columns:

- `x,y`: coordinates of raster cell centers.
- `cell_number`: integer indicating the cell number within the raster.
- `intensity`: maximum cell value across layers divided by the maximum value across all layers and cells; mapped to alpha level.
- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `layer_id`: integer identifying the layer containing the maximum intensity value; mapped to hue.
- `color`: the hexadecimal color associated with the given layer and specificity values.

See Also

Other map: `map_multiples()`

Examples

```
# load elephant data
data("elephant_ud")

# prepare metrics
r <- metrics_distill(elephant_ud)

# generate palette
pal <- palette_set(elephant_ud)

# produce map, adjusting lambda_i to make areas that were used less
# intensively more conspicuous
map_single(r, pal, lambda_i = -5)

# return RasterStack containing RGBa values
m <- map_single(r, pal, lambda_i = -5, return_type = "stack")

# visualize RGBa values
library(raster)
plotRGB(m, 1, 2, 3, alpha = as.vector(m[[4]]))
```

`metrics_distill` *Distill a raster stack into a set of distribution metrics*

Description

This function is used to summarize several distributional features of interest across a series of distributions. Distributional information in the original raster stack is "distilled" for subsequent visualization.

Usage

```
metrics_distill(x)
```

Arguments

x	RasterStack of distributions. Layers typically contain information about the distribution of a single individual or species at multiple points in time. Alternatively, layers may contain information about the distributions of multiple individuals or species within a single time period. Other conceptualizations are possible.

Details

Specificity values range from 0 to 100. Values of 0 indicate intensity values are identical in all layers. Values of 100 indicate intensity values are restricted to a single layer. Interpretation of specificity values depends on the layers provided. If layers describe the distribution of a species at different times of the year, specificity can be interpreted as a measure of seasonality (i.e., 0 = stable year-round occurrence in a cell, 100 = highly seasonal occurrence). If layers describe space use by multiple individuals, specificity can be interpreted as a measure of exclusivity (i.e., 0 = equal use of a cell by all individuals, 100 = exclusive use by one individual).

The number of layers with non-NA values is recorded to aid interpretation of distributions. Ideally, n_layers values are identical in every cell, indicating that users have knowledge of distributions over the same area in every layer of their raster stack. When n_layers values are unequal, it indicates that users have unequal knowledge of distributions in their raster stack. Distributions are more likely to be misrepresented and misinterpreted if cells do not contain intensity values in every layer.

Value

A RasterStack with four layers:

- `intensity`: the maximum intensity value across all layers.
- `layer_id`: an integer identifying layer containing the maximum intensity value.
- `specificity`: the degree to which intensity values are unevenly distributed across layers (see Details).
- `n_layers`: the number of layers with non-NA values (see Details).

The maximum cell value in the stack is stored as the `"maximum"` attribute. The link between the `layer_id` and the layer names from the underlying raster is stored as a data frame in the `layer_names` attribute.

See Also

Other metrics: `metrics_pull()`

Examples

```
# load elephant data
data("elephant_ud")

# distill
r <- metrics_distill(elephant_ud)
print(r)

# maximum value across all layers stored as an attribute
attr(r, "maximum")
# link between layer id and name stored as an attribute
attr(r, "layer_names")
```

`metrics_pull` *Transform raster stack values to intensity values*

Description

This function transforms raster stack values that describe individual distributions or species distributions into standardized intensity values. All the distributional information in the original raster stack is preserved for visualization.

Usage

```
metrics_pull(x)
```

Arguments

`x`	RasterStack of distributions. Layers typically contain information about the distribution of a single individual or species at multiple points in time. Alternatively, layers may contain information about the distributions of multiple individuals or species within a single time period. Other conceptualizations are possible.

Value

A RasterStack containing intensity values. Intensity values are calculated by dividing cell values in every layer by the maximum cell value in the entire stack, thus ensuring intensities are comparable across layers.

The maximum cell value in the stack is stored as the `"maximum"` attribute.

See Also

Other metrics: `metrics_distill()`

Examples

```
# load elephant data
data("elephant_ud")
r <- metrics_pull(elephant_ud)
print(r)
# maximum value for the stack stored as an attribute
attr(r, "maximum")
```

`palette_set` *Make an HCL palette for visualizing an unordered set of distributions*

Description

This function generates an HCL palette for visualizing a small set of distributions (i.e., eight or fewer) that are not ordered in a linear or cyclical sequence (e.g., a set of utilization distributions describing space use by five separate individuals in the same population or a set of four species distributions that depend on similar food resources).

Usage

```
palette_set(x, custom_hues)
```

Arguments

`x`	RasterStack or integer describing the number of layers for which colors need to be generated.
`custom_hues`	vector of integers between -360 and 360 representing hues in the color wheel. For further details, consult the documentation for colorspace::rainbow_hcl. The length of the vector must equal the number of layers described by `x`. Hues are assigned to layers in order.

Value

A data frame with three columns:

- `layer_id`: integer identifying the layer containing the maximum intensity value; mapped to hue.
- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `color`: the hexadecimal color associated with the given layer and specificity values.

See Also

palette_timecycle for cyclical sequences of distributions and palette_timeline for linear sequences of distributions.

Other palette: `palette_timecycle()`, `palette_timeline()`

Examples

```
# load elephant data
data(elephant_ud)

# generate hcl color palette
pal <- palette_set(elephant_ud)
head(pal)

# visualize the palette in HCL space with colorspace::hclplot
library(colorspace)
hclplot(pal[pal$specificity == 100, ]$color)
```

`palette_timecycle` *Make an HCL palette for visualizing a cyclical sequence of distributions*

Description

This function generates an HCL palette for visualizing a cyclical sequence of distributions (e.g., a series of distributions describing species occurrence in each of 52 weeks of the annual cycle or a series of utilization distributions describing typical space use by an individual animal in each hour of a 24-hour daily cycle).

Usage

```
palette_timecycle(x, start_hue = 240, clockwise = TRUE)
```

Arguments

`x`	RasterStack or integer describing the number of layers for which colors need to be generated.
`start_hue`	integer between -360 and 360 representing the starting hue in an HCL color wheel. For further details, consult the documentation for colorspace::rainbow_hcl. The default value of 240 will start the palette at "blue".
`clockwise`	logical indicating which direction to move around color wheel. The default `clockwise = TRUE` will yield a "blue-green-yellow-pink-blue" palette when `start_hue = 240`, while `clockwise = FALSE` will yield a "blue-pink-yellow-green-blue" palette.

Value

A data frame with three columns:

- `layer_id`: integer identifying the layer containing the maximum intensity value; mapped to hue.
- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `color`: the hexadecimal color associated with the given layer and specificity values.

See Also

palette_timeline for linear sequences of distributions and palette_set for unordered sets of distributions.

Other palette: `palette_set()`, `palette_timeline()`

Examples

```
# load field sparrow data
data(fiespa_occ)

# generate hcl color palette
pal <- palette_timecycle(fiespa_occ)
head(pal)

# visualize the palette in HCL space with colorspace::hclplot
library(colorspace)
hclplot(pal[pal$specificity == 100, ]$color)
```

`palette_timeline` *Make an HCL palette for visualizing a linear sequence of distributions*

Description

This function generates an HCL palette for visualizing a linear sequence of distributions (e.g., a series of utilization distributions describing space use by an individual animal across each of 20 consecutive days or a series of species distributions describing projected responses to global warming in 0.5 C increments).

Usage

```
palette_timeline(x, start_hue = -130, clockwise = FALSE)
```

Arguments

`x`	RasterStack or integer describing the number of layers for which colors need to be generated.
`start_hue`	integer between -360 and 360 representing the starting hue in the color wheel. For further details, consult the documentation for colorspace::rainbow_hcl. Recommended values are -130 ("blue-pink-yellow" palette) and 50 ("yellow-green-blue" palette).
`clockwise`	logical indicating which direction to move around an HCL color wheel. When `clockwise = FALSE` the ending hue will be `start_hue + 180`. When `clockwise = TRUE` the ending hue will be `start_hue -180`. The default value `clockwise = FALSE` will yield a "blue-pink-yellow" palette when `start_hue = -130`, while `clockwise = TRUE` will yield a "blue-green-yellow" palette.

Value

A data frame with three columns:

- `layer_id`: integer identifying the layer containing the maximum intensity value; mapped to hue.
- `specificity`: the degree to which intensity values are unevenly distributed across layers; mapped to chroma.
- `color`: the hexadecimal color associated with the given layer and specificity values.

See Also

palette_timecycle for cyclical sequences of distributions and palette_set for unordered sets of distributions.

Other palette: `palette_set()`, `palette_timecycle()`

Examples

```
# load fisher data
data(fisher_ud)

# generate hcl color palette
pal_a <- palette_timeline(fisher_ud)
head(pal_a)

# use a clockwise palette
pal_b <- palette_timeline(fisher_ud, clockwise = TRUE)

# try a different starting hue
pal_c <- palette_timeline(fisher_ud, start = 50)

# visualize the palette in HCL space  with colorspace::hclplot
```

```
library(colorspace)
hclplot(pal_a[pal_a$specificity == 100, ]$color)
hclplot(pal_b[pal_b$specificity == 100, ]$color)
hclplot(pal_c[pal_c$specificity == 100, ]$color)
```

Index