Package: certeplot2 (via r-universe)

September 11, 2024

Title A Certe R Package for Convenient Plotting

Version 2.0.1

Description A Certe R Package for fast and convenient plotting based on 'plot2', by providing wrappers around 'tidyverse' packages such as 'ggplot2', while also providing plotting in the Certe organisational style. This package is part of the 'certedata' universe.

URL https://certe-medical-epidemiology.github.io/certeplot2, https://github.com/certe-medical-epidemiology/certeplot2

Depends R (>= 4.1.0), plot2 (>= 1.25.0)

Imports certestyle, AMR (>= 2.0.0), dplyr (>= 1.0.0), ggplot2 (>= 3.5.1), tidyr (>= 1.0.0), yaml (>= 2.2.0)

Suggests certegis, certestats, sf (>= 1.0.0), testthat (>= 2.0.0)

License GPL-2

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 2

Repository https://certe-medical-epidemiology.r-universe.dev

RemoteUrl https://github.com/certe-medical-epidemiology/certeplot2

RemoteRef HEAD

RemoteSha a3f5217cf83566af7fc8394726d729bd7d7d81b4

Contents

plot2-extensions *Methods for* plot2()

Description

These are the implemented methods for different S3 classes to be used in plot2(). Since they have an extensive list of arguments, they are placed here on a separate manual page.

Usage

```
## S3 method for class 'bug_drug_combinations'
plot2(
  .data,
  x = ab,
  y = value,
  category = name,
  facet = mo,
  type = "column",
  x.title = FALSE,
  y.title = FALSE,
  category.title = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  tag = NULL,
  title.linelength = 60,
  title.colour = getOption("plot2.colour_font_primary", "black"),
  subtitle.linelength = 60,
  subtitle.colour = getOption("plot2.colour_font_secondary", "grey35"),
  na.replace = "",
  na.rm = FALSE,
  facet.position = "top",
  facet.fill = NULL,
  facet.bold = TRUE,
  facet.italic = FALSE,
  facet.size = 10,
  facet.margin = 8,
  facet.repeat_lbls_x = TRUE,
  facet.repeat_lbls_y = TRUE,
  facet.fixed_y = NULL,
  facet.fixed_x = TRUE,
  facet.drop = FALSE,
  facet.nrow = NULL,
  facet.relative = FALSE,
  x.date_breaks = NULL,
  x.date_labels = NULL,
  x.date_remove_years = NULL,
```

```
  category.focus = NULL,
  colour = get_colour("certe_sir2", 7),
  colour_fill = NULL,
  colour_opacity = 0,
  x.lbl_angle = ifelse(horizontal, 0, 90),
  x.lbl_align = NULL,
  x.lbl_italic = FALSE,
  x.lbl_taxonomy = TRUE,
  x.remove = FALSE,
  x.position = "bottom",
  x.max_items = Inf,
  x.max_txt = "(rest, x%n)",
  category.max_items = Inf,
  category.max_txt = "(rest, x%n)",
  facet.max_items = Inf,
  facet.max_txt = "(rest, x%n)",
  x.breaks = NULL,
  x.n_breaks = NULL,
  x.transform = "identity",
  x.expand = NULL,
  x.limits = NULL,
  x.labels = NULL,
  x.character = NULL,
  x.drop = FALSE,
  x.mic = FALSE,
  x.zoom = FALSE,
  y.remove = FALSE,
  y.24h = FALSE,
  y.age = FALSE,
  y.scientific = NULL,
  y.percent = FALSE,
  y.percent_break = 0.1,
  y.breaks = NULL,
  y.n_breaks = NULL,
  y.limits = NULL,
  y.labels = NULL,
  y.expand = NULL,
  y.transform = "identity",
  y.position = "left",
  y.zoom = FALSE,
  y_secondary = NULL,
  y_secondary.type = type,
  y_secondary.title = TRUE,
  y_secondary.colour = "certeroze",
  y_secondary.colour_fill = "certeroze6",
  y_secondary.scientific = NULL,
  y_secondary.percent = FALSE,
  y_secondary.labels = NULL,
```

```
  category.labels = NULL,
  category.percent = FALSE,
  category.breaks = NULL,
  category.limits = NULL,
  category.expand = 0,
  category.midpoint = NULL,
  category.transform = "identity",
  category.date_breaks = NULL,
  category.date_labels = NULL,
  category.character = NULL,
  x.sort = NULL,
  category.sort = FALSE,
  facet.sort = TRUE,
  x.complete = NULL,
  category.complete = NULL,
  facet.complete = NULL,
  datalabels = TRUE,
  datalabels.round = ifelse(y.percent, 2, 1),
  datalabels.format = "%n",
  datalabels.colour = "grey25",
  datalabels.colour_fill = NULL,
  datalabels.size = (3 * text_factor),
  datalabels.angle = 0,
  datalabels.lineheight = 1,
  decimal.mark = dec_mark(),
  big.mark = big_mark(),
  summarise_function = base::sum,
  stacked = FALSE,
  stackedpercent = TRUE,
  horizontal = TRUE,
  reverse = TRUE,
  smooth = NULL,
  smooth.method = NULL,
  smooth.formula = NULL,
  smooth.se = TRUE,
  smooth.level = 0.95,
  smooth.alpha = 0.25,
  smooth.linewidth = 0.75,
  smooth.linetype = 3,
  smooth.colour = NULL,
  size = NULL,
  linetype = 1,
  linewidth = NULL,
  binwidth = NULL,
  width = NULL,
  jitter_seed = NA,
  violin_scale = "count",
  legend.position = NULL,
```

```
  legend.title = NULL,
  legend.reverse = TRUE,
  legend.barheight = 6,
  legend.barwidth = 1.5,
  legend.nbin = 300,
  legend.italic = FALSE,
  sankey.node_width = 0.15,
  sankey.node_whitespace = 0.03,
  sankey.alpha = 0.5,
  sankey.remove_axes = NULL,
  zoom = FALSE,
  sep = " / ",
  print = FALSE,
  text_factor = 1,
  font = getOption("plot2.font"),
  theme = getOption("plot2.theme", "theme_minimal2"),
  background = getOption("plot2.colour_background", "white"),
  markdown = TRUE,
  data = NULL,
  minimum = 30,
  remove_intrinsic_resistant = TRUE,
  language = "nl",
  ...
)

## S3 method for class 'antibiogram'
plot2(
  .data,
  x = NULL,
  y = NULL,
  category = NULL,
  facet = NULL,
  type = NULL,
  x.title = NULL,
  y.title = NULL,
  category.title = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  tag = NULL,
  title.linelength = 60,
  title.colour = getOption("plot2.colour_font_primary", "black"),
  subtitle.linelength = 60,
  subtitle.colour = getOption("plot2.colour_font_secondary", "grey35"),
  na.replace = "",
  na.rm = FALSE,
  facet.position = "top",
  facet.fill = NULL,
```

```
facet.bold = TRUE,
facet.italic = FALSE,
facet.size = 10,
facet.margin = 8,
facet.repeat_lbls_x = TRUE,
facet.repeat_lbls_y = TRUE,
facet.fixed_y = NULL,
facet.fixed_x = TRUE,
facet.drop = FALSE,
facet.nrow = NULL,
facet.relative = FALSE,
x.date_breaks = NULL,
x.date_labels = NULL,
x.date_remove_years = NULL,
category.focus = NULL,
colour = getOption("plot2.colour", "ggplot2"),
colour_fill = NULL,
colour_opacity = 0,
x.lbl_angle = 0,
x.lbl_align = NULL,
x.lbl_italic = FALSE,
x.lbl_taxonomy = FALSE,
x.remove = FALSE,
x.position = "bottom",
x.max_items = Inf,
x.max_txt = "(rest, x%n)",
category.max_items = Inf,
category.max_txt = "(rest, x%n)",
facet.max_items = Inf,
facet.max_txt = "(rest, x%n)",
x.breaks = NULL,
x.n_breaks = NULL,
x.transform = "identity",
x.expand = NULL,
x.limits = NULL,
x.labels = NULL,
x.character = NULL,
x.drop = FALSE,
x.mic = FALSE,
x.zoom = FALSE,
y.remove = FALSE,
y.24h = FALSE,
y.age = FALSE,
y.scientific = NULL,
y.percent = FALSE,
y.percent_break = 0.1,
y.breaks = NULL,
y.n_breaks = NULL,
```

```
  y.limits = NULL,
  y.labels = NULL,
  y.expand = NULL,
  y.transform = "identity",
  y.position = "left",
  y.zoom = FALSE,
  y_secondary = NULL,
  y_secondary.type = type,
  y_secondary.title = TRUE,
  y_secondary.colour = "certeroze",
  y_secondary.colour_fill = "certeroze6",
  y_secondary.scientific = NULL,
  y_secondary.percent = FALSE,
  y_secondary.labels = NULL,
  category.labels = NULL,
  category.percent = FALSE,
  category.breaks = NULL,
  category.limits = NULL,
  category.expand = 0,
  category.midpoint = NULL,
  category.transform = "identity",
  category.date_breaks = NULL,
  category.date_labels = NULL,
  category.character = NULL,
  x.sort = NULL,
  category.sort = TRUE,
  facet.sort = TRUE,
  x.complete = NULL,
  category.complete = NULL,
  facet.complete = NULL,
  datalabels = TRUE,
  datalabels.round = ifelse(y.percent, 2, 1),
  datalabels.format = "%n",
  datalabels.colour = "grey25",
  datalabels.colour_fill = NULL,
  datalabels.size = (3 * text_factor),
  datalabels.angle = 0,
  datalabels.lineheight = 1,
  decimal.mark = dec_mark(),
  big.mark = big_mark(),
  summarise_function = base::sum,
  stacked = FALSE,
  stackedpercent = FALSE,
  horizontal = FALSE,
  reverse = horizontal,
  smooth = NULL,
  smooth.method = NULL,
  smooth.formula = NULL,
```

```
  smooth.se = TRUE,
  smooth.level = 0.95,
  smooth.alpha = 0.25,
  smooth.linewidth = 0.75,
  smooth.linetype = 3,
  smooth.colour = NULL,
  size = NULL,
  linetype = 1,
  linewidth = NULL,
  binwidth = NULL,
  width = NULL,
  jitter_seed = NA,
  violin_scale = "count",
  legend.position = NULL,
  legend.title = NULL,
  legend.reverse = FALSE,
  legend.barheight = 6,
  legend.barwidth = 1.5,
  legend.nbin = 300,
  legend.italic = FALSE,
  sankey.node_width = 0.15,
  sankey.node_whitespace = 0.03,
  sankey.alpha = 0.5,
  sankey.remove_axes = NULL,
  zoom = FALSE,
  sep = " / ",
  print = FALSE,
  text_factor = 1,
  font = getOption("plot2.font"),
  theme = getOption("plot2.theme", "theme_minimal2"),
  background = getOption("plot2.colour_background", "white"),
  markdown = TRUE,
  data = NULL,
  ...
)

## S3 method for class 'sir_df'
plot2(
  .data,
  x = NULL,
  y = isolates,
  category = interpretation,
  facet = antibiotic,
  type = "column",
  x.title = TRUE,
  y.title = FALSE,
  category.title = NULL,
  title = FALSE,
```

```
  subtitle = NULL,
  caption = NULL,
  tag = NULL,
  title.linelength = 60,
  title.colour = getOption("plot2.colour_font_primary", "black"),
  subtitle.linelength = 60,
  subtitle.colour = getOption("plot2.colour_font_secondary", "grey35"),
  na.replace = "",
  na.rm = FALSE,
  facet.position = "top",
  facet.fill = NULL,
  facet.bold = TRUE,
  facet.italic = FALSE,
  facet.size = 10,
  facet.margin = 8,
  facet.repeat_lbls_x = TRUE,
  facet.repeat_lbls_y = TRUE,
  facet.fixed_y = NULL,
  facet.fixed_x = TRUE,
  facet.drop = FALSE,
  facet.nrow = NULL,
  facet.relative = FALSE,
  x.date_breaks = NULL,
  x.date_labels = NULL,
  x.date_remove_years = NULL,
  category.focus = NULL,
  colour = get_colour("certe_sir2", 5),
  colour_fill = NULL,
  colour_opacity = 0,
  x.lbl_angle = 0,
  x.lbl_align = NULL,
  x.lbl_italic = FALSE,
  x.lbl_taxonomy = TRUE,
  x.remove = FALSE,
  x.position = "bottom",
  x.max_items = Inf,
  x.max_txt = "(rest, x%n)",
  category.max_items = Inf,
  category.max_txt = "(rest, x%n)",
  facet.max_items = Inf,
  facet.max_txt = "(rest, x%n)",
  x.breaks = NULL,
  x.n_breaks = NULL,
  x.transform = "identity",
  x.expand = NULL,
  x.limits = NULL,
  x.labels = NULL,
  x.character = NULL,
```

```
x.drop = FALSE,
x.mic = FALSE,
x.zoom = FALSE,
y.remove = FALSE,
y.24h = FALSE,
y.age = FALSE,
y.scientific = NULL,
y.percent = FALSE,
y.percent_break = 0.1,
y.breaks = NULL,
y.n_breaks = NULL,
y.limits = NULL,
y.labels = NULL,
y.expand = NULL,
y.transform = "identity",
y.position = "left",
y.zoom = FALSE,
y_secondary = NULL,
y_secondary.type = type,
y_secondary.title = TRUE,
y_secondary.colour = "certeroze",
y_secondary.colour_fill = "certeroze6",
y_secondary.scientific = NULL,
y_secondary.percent = FALSE,
y_secondary.labels = NULL,
category.labels = NULL,
category.percent = FALSE,
category.breaks = NULL,
category.limits = NULL,
category.expand = 0,
category.midpoint = NULL,
category.transform = "identity",
category.date_breaks = NULL,
category.date_labels = NULL,
category.character = NULL,
x.sort = NULL,
category.sort = FALSE,
facet.sort = TRUE,
x.complete = NULL,
category.complete = NULL,
facet.complete = NULL,
datalabels = TRUE,
datalabels.round = ifelse(y.percent, 2, 1),
datalabels.format = "%n",
datalabels.colour = "grey25",
datalabels.colour_fill = NULL,
datalabels.size = (3 * text_factor),
datalabels.angle = 0,
```

```
  datalabels.lineheight = 1,
  decimal.mark = dec_mark(),
  big.mark = big_mark(),
  summarise_function = base::sum,
  stacked = FALSE,
  stackedpercent = TRUE,
  horizontal = FALSE,
  reverse = TRUE,
  smooth = NULL,
  smooth.method = NULL,
  smooth.formula = NULL,
  smooth.se = TRUE,
  smooth.level = 0.95,
  smooth.alpha = 0.25,
  smooth.linewidth = 0.75,
  smooth.linetype = 3,
  smooth.colour = NULL,
  size = NULL,
  linetype = 1,
  linewidth = NULL,
  binwidth = NULL,
  width = NULL,
  jitter_seed = NA,
  violin_scale = "count",
  legend.position = NULL,
  legend.title = NULL,
  legend.reverse = FALSE,
  legend.barheight = 6,
  legend.barwidth = 1.5,
  legend.nbin = 300,
  legend.italic = FALSE,
  sankey.node_width = 0.15,
  sankey.node_whitespace = 0.03,
  sankey.alpha = 0.5,
  sankey.remove_axes = NULL,
  zoom = FALSE,
  sep = " / ",
  print = FALSE,
  text_factor = 1,
  font = getOption("plot2.font"),
  theme = getOption("plot2.theme", "theme_minimal2"),
  background = getOption("plot2.colour_background", "white"),
  markdown = TRUE,
  data = NULL,
  ...
)

## S3 method for class 'qc_test'
```

```
plot2(
  .data,
  x = x,
  y = y,
  category = rule,
  facet = NULL,
  type = "point",
  x.title = "Index",
  y.title = "Value",
  category.title = NULL,
  title = paste0("QC Chart (", attributes(.data)$guideline, ")"),
  subtitle = NULL,
  caption = NULL,
  tag = NULL,
  title.linelength = 60,
  title.colour = getOption("plot2.colour_font_primary", "black"),
  subtitle.linelength = 60,
  subtitle.colour = getOption("plot2.colour_font_secondary", "grey35"),
  na.replace = "",
  na.rm = FALSE,
  facet.position = "top",
  facet.fill = NULL,
  facet.bold = TRUE,
  facet.italic = FALSE,
  facet.size = 10,
  facet.margin = 8,
  facet.repeat_lbls_x = TRUE,
  facet.repeat_lbls_y = TRUE,
  facet.fixed_y = NULL,
  facet.fixed_x = TRUE,
  facet.drop = FALSE,
  facet.nrow = NULL,
  facet.relative = FALSE,
  x.date_breaks = NULL,
  x.date_labels = NULL,
  x.date_remove_years = NULL,
  category.focus = NULL,
 colour = get_colour(c(Observation = "grey75", `Rule 1` = "certeblauw", `Rule 2` =
  "certegroen", `Rule 3` = "certeroze", `Rule 4` = "certegeel", `Rule 5` = "certelila",
  `Rule 6` = "certebruin", `Rule 7` = "certeblauw2", `Rule 8` = "certegroen2")),
  colour_fill = NULL,
  colour_opacity = 0,
  x.lbl_angle = 0,
  x.lbl_align = NULL,
  x.lbl_italic = FALSE,
  x.lbl_taxonomy = FALSE,
  x.remove = FALSE,
  x.position = "bottom",
```

```
x.max_items = Inf,
x.max_txt = "(rest, x%n)",
category.max_items = Inf,
category.max_txt = "(rest, x%n)",
facet.max_items = Inf,
facet.max_txt = "(rest, x%n)",
x.breaks = NULL,
x.n_breaks = NULL,
x.transform = "identity",
x.expand = NULL,
x.limits = NULL,
x.labels = NULL,
x.character = NULL,
x.drop = FALSE,
x.mic = FALSE,
x.zoom = TRUE,
y.remove = FALSE,
y.24h = FALSE,
y.age = FALSE,
y.scientific = NULL,
y.percent = FALSE,
y.percent_break = 0.1,
y.breaks = NULL,
y.n_breaks = NULL,
y.limits = NULL,
y.labels = NULL,
y.expand = NULL,
y.transform = "identity",
y.position = "left",
y.zoom = TRUE,
y_secondary = NULL,
y_secondary.type = type,
y_secondary.title = TRUE,
y_secondary.colour = "certeroze",
y_secondary.colour_fill = "certeroze6",
y_secondary.scientific = NULL,
y_secondary.percent = FALSE,
y_secondary.labels = NULL,
category.labels = NULL,
category.percent = FALSE,
category.breaks = NULL,
category.limits = NULL,
category.expand = 0,
category.midpoint = NULL,
category.transform = "identity",
category.date_breaks = NULL,
category.date_labels = NULL,
category.character = NULL,
```

```
x.sort = NULL,
category.sort = TRUE,
facet.sort = TRUE,
x.complete = NULL,
category.complete = NULL,
facet.complete = NULL,
datalabels = TRUE,
datalabels.round = ifelse(y.percent, 2, 1),
datalabels.format = "%n",
datalabels.colour = "grey25",
datalabels.colour_fill = NULL,
datalabels.size = (3 * text_factor),
datalabels.angle = 0,
datalabels.lineheight = 1,
decimal.mark = dec_mark(),
big.mark = big_mark(),
summarise_function = base::sum,
stacked = FALSE,
stackedpercent = FALSE,
horizontal = FALSE,
reverse = horizontal,
smooth = NULL,
smooth.method = NULL,
smooth.formula = NULL,
smooth.se = TRUE,
smooth.level = 0.95,
smooth.alpha = 0.25,
smooth.linewidth = 0.75,
smooth.linetype = 3,
smooth.colour = NULL,
size = 2,
linetype = 1,
linewidth = NULL,
binwidth = NULL,
width = NULL,
jitter_seed = NA,
violin_scale = "count",
legend.position = "right",
legend.title = NULL,
legend.reverse = FALSE,
legend.barheight = 6,
legend.barwidth = 1.5,
legend.nbin = 300,
legend.italic = FALSE,
sankey.node_width = 0.15,
sankey.node_whitespace = 0.03,
sankey.alpha = 0.5,
sankey.remove_axes = NULL,
```

```
  zoom = TRUE,
  sep = " / ",
  print = FALSE,
  text_factor = 1,
  font = getOption("plot2.font"),
  theme = getOption("plot2.theme", "theme_minimal2"),
  background = getOption("plot2.colour_background", "white"),
  markdown = TRUE,
  data = NULL,
  ...
)

## S3 method for class 'early_warning_cluster'
plot2(
  .data,
  x = NULL,
  y = NULL,
  category = NULL,
  facet = NULL,
  type = "line",
 x.title = ifelse(attributes(.data)$period_length_months == 12, "Maand in periode",
   "Week in periode"),
  y.title = paste0("Cases (", attributes(.data)$moving_average_days,
   "-daags zwevend gemiddelde)"),
  category.title = "Periode",
  title = paste0(n_distinct(.data$clusters$cluster), " cluster",
   ifelse(n_distinct(.data$clusters$cluster) != 1, "s", "")),
  subtitle = NULL,
  caption = paste0("O.b.v. uitbijter-vrije geschiedenis (coeff = ",
   format2(attributes(.data)$remove_outliers_coefficient), ") met pct = ",
   format2(attributes(.data)$threshold_percentile)),
  tag = NULL,
  title.linelength = 60,
  title.colour = getOption("plot2.colour_font_primary", "black"),
  subtitle.linelength = 60,
  subtitle.colour = getOption("plot2.colour_font_secondary", "grey35"),
  na.replace = "",
  na.rm = FALSE,
  facet.position = "top",
  facet.fill = NULL,
  facet.bold = TRUE,
  facet.italic = FALSE,
  facet.size = 10,
  facet.margin = 8,
  facet.repeat_lbls_x = TRUE,
  facet.repeat_lbls_y = TRUE,
  facet.fixed_y = NULL,
  facet.fixed_x = TRUE,
```

```
facet.drop = FALSE,
facet.nrow = NULL,
facet.relative = FALSE,
x.date_breaks = "1 month",
x.date_labels = "mmm",
x.date_remove_years = FALSE,
category.focus = NULL,
colour = getOption("plot2.colour", "ggplot2"),
colour_fill = NULL,
colour_opacity = 0,
x.lbl_angle = 0,
x.lbl_align = NULL,
x.lbl_italic = FALSE,
x.lbl_taxonomy = FALSE,
x.remove = FALSE,
x.position = "bottom",
x.max_items = Inf,
x.max_txt = "(rest, x%n)",
category.max_items = Inf,
category.max_txt = "(rest, x%n)",
facet.max_items = Inf,
facet.max_txt = "(rest, x%n)",
x.breaks = seq(0, 9999, 14),
x.n_breaks = NULL,
x.transform = "identity",
x.expand = NULL,
x.limits = NULL,
x.labels = function(x) x/7,
x.character = NULL,
x.drop = FALSE,
x.mic = FALSE,
x.zoom = FALSE,
y.remove = FALSE,
y.24h = FALSE,
y.age = FALSE,
y.scientific = NULL,
y.percent = FALSE,
y.percent_break = 0.1,
y.breaks = NULL,
y.n_breaks = NULL,
y.limits = NULL,
y.labels = NULL,
y.expand = NULL,
y.transform = "identity",
y.position = "left",
y.zoom = FALSE,
y_secondary = NULL,
y_secondary.type = type,
```

```
y_secondary.title = TRUE,
y_secondary.colour = "certeroze",
y_secondary.colour_fill = "certeroze6",
y_secondary.scientific = NULL,
y_secondary.percent = FALSE,
y_secondary.labels = NULL,
category.labels = md_to_expression,
category.percent = FALSE,
category.breaks = NULL,
category.limits = NULL,
category.expand = 0,
category.midpoint = NULL,
category.transform = "identity",
category.date_breaks = NULL,
category.date_labels = NULL,
category.character = TRUE,
x.sort = NULL,
category.sort = "asc",
facet.sort = TRUE,
x.complete = NULL,
category.complete = NULL,
facet.complete = NULL,
datalabels = TRUE,
datalabels.round = ifelse(y.percent, 2, 1),
datalabels.format = "%n",
datalabels.colour = "grey25",
datalabels.colour_fill = NULL,
datalabels.size = (2.5 * text_factor),
datalabels.angle = 0,
datalabels.lineheight = 1,
decimal.mark = dec_mark(),
big.mark = big_mark(),
summarise_function = base::sum,
stacked = FALSE,
stackedpercent = FALSE,
horizontal = FALSE,
reverse = horizontal,
smooth = NULL,
smooth.method = NULL,
smooth.formula = NULL,
smooth.se = TRUE,
smooth.level = 0.95,
smooth.alpha = 0.25,
smooth.linewidth = 0.75,
smooth.linetype = 3,
smooth.colour = NULL,
size = NULL,
linetype = 1,
```

```
  linewidth = NULL,
  binwidth = NULL,
  width = NULL,
  jitter_seed = NA,
  violin_scale = "count",
  legend.position = "right",
  legend.title = NULL,
  legend.reverse = FALSE,
  legend.barheight = 6,
  legend.barwidth = 1.5,
  legend.nbin = 300,
  legend.italic = FALSE,
  sankey.node_width = 0.15,
  sankey.node_whitespace = 0.03,
  sankey.alpha = 0.5,
  sankey.remove_axes = NULL,
  zoom = FALSE,
  sep = " / ",
  print = FALSE,
  text_factor = 1,
  font = getOption("plot2.font"),
  theme = getOption("plot2.theme", "theme_minimal2"),
  background = getOption("plot2.colour_background", "white"),
  markdown = TRUE,
  data = NULL,
  ...
)
```

Arguments

`.data`, `data`	data to plot
`x`	plotting 'direction' for the x axis. This can be: • A single variable from `.data`, such as `x = column1` • A function to calculate over one or more variables from `.data`, such as `x = format(column1, "%Y")`, or `x = ifelse(column1 == "A", "Group A", "Other")` • Multiple variables from `.data`, such as `x = c(column1, column2, column2)`, or using selection helpers such as `x = where(is.character)` or `x = starts_with("var_")` *(only allowed and required for Sankey plots using* `type = "sankey"`*)*
`y`	values to use for plotting along the y axis. This can be: • A single variable from `.data`, such as `y = column1` • Multiple variables from `.data`, such as `y = c(column1, column2)` or `y = c(name1 = column1, "name 2" = column2)`, or using selection helpers such as `y = where(is.double)` or `y = starts_with("var_")` *(multiple variables only allowed if* `category` *is not set)* • A function to calculate over `.data` returning a single value, such as `y = n()` for the row count, or based on other variables such as `y = n_distinct(person_id)`, `y = max(column1)`, or `y = median(column2) / column3`

- A function to calculate over .data returning multiple values, such as y = quantile(column1, c(0.25, 0.75)) or y = range(age) *(multiple values only allowed if* category *is not set)*

category, facet
: plotting 'direction' (category is called 'fill' and 'colour' in ggplot2). This can be:

  - A single variable from .data, such as category = column1
  - A function to calculate over one or more variables from .data, such as category = median(column2) / column3, or facet = ifelse(column1 == "A", "Group A", "Other")
  - Multiple variables from .data, such as facet = c(column1, column2) (use sep to control the separator character)
  - One or more variables from .data using selection helpers, such as category = where(is.double) or facet = starts_with("var_")

  The category can also be a date or date/time (class Date or POSIXt).

type, y_secondary.type
: type of visualisation to use. This can be:

  - A ggplot2 geom name or their abbreviation such as "col" and "point". All geoms are supported (including geom_blank()).
    Full function names can be used (e.g., "geom_histogram"), but they can also be abbreviated (e.g., "h", "hist"). The following geoms can be abbreviated by their first character: area ("a"), boxplot ("b"), column ("c"), histogram ("h"), jitter ("j"), line ("l"), point ("p"), ribbon ("r"), and violin ("v").
    Please note: in ggplot2, 'bars' and 'columns' are equal, while it is common to many people that 'bars' are oriented horizontally and 'columns' are oriented vertically since Microsoft Excel has been using these terms this way for many years. For this reason, type = "bar" will set type = "col" and horizontal = TRUE.
  - One of these additional types:
    - "barpercent" (short: "bp"), which is effectively a shortcut to set type = "col" and horizontal = TRUE and x.max_items = 10 and x.sort = "freq-desc" and datalabels.format = "%n (%p)".
    - "linedot" (short: "ld"), which sets type = "line" and adds two point geoms using add_point(); one with large white dots and one with smaller dots using the colours set in colour. This is essentially equal to base R plot(..., type = "b") but with closed shapes.
    - "dumbbell" (short: "d"), which sets type = "point" and horizontal = TRUE, and adds a line between the points (using geom_segment()). The line colour cannot be changed. This plot type is only possible when the category has two distinct values.
    - "sankey" (short: "s") creates a Sankey plots using category for the flows and requires x to contain multiple variables from .data. At default, it also sets x.expand = c(0.05, 0.05) and y.limits = c(NA, NA) and y.expand = c(0.01, 0.01). The so-called 'nodes' (the 'blocks' with text) are considered the datalabels, so you can set the text size and colour of the nodes using datalabels.size, datalabels.colour,

and datalabels.colour_fill. The transparency of the flows can be set using sankey.alpha, and the width of the nodes can be set using sankey.node_width. Sankey plots can also be flipped using horizontal = TRUE.

- Left blank. In this case, the type will be determined automatically: "boxplot" if there is no x axis or if the length of unique values per x axis item is at least 3, "point" if both the y and x axes are numeric, and the option "plot2.default_type" otherwise (which defaults to "col"). Use type = "blank" or type = "geom_blank" to *not* add a geom.

title, subtitle, caption, tag, x.title, y.title, category.title, legend.title, y_secondary.title
: a title to use. This can be:

  - A character, which supports markdown by using md_to_expression() internally if markdown = TRUE (which is the default)
  - A function to calculate over .data, such as title = paste("Based on n =", n_distinct(person_id), "individuals") or subtitle = paste("Total rows:", n()), see *Examples*
  - An expression, e.g. using parse(text = "...")

  The category.title defaults to TRUE if the legend items are numeric.

title.linelength
: maximum number of characters per line in the title, before a linebreak occurs

title.colour
: text colour of the title

subtitle.linelength
: maximum number of characters per line in the subtitle, before a linebreak occurs

subtitle.colour
: text colour of the subtitle

na.replace
: character to put in place of NA values if na.rm = FALSE

na.rm
: remove NA values from showing in the plot

facet.position, facet.fill, facet.bold, facet.italic, facet.size, facet.margin, facet.repeat_lbls_x, facet.repeat_lbls_y, facet.drop, facet.nrow, facet.relative
: additional settings for the plotting direction facet

facet.fixed_y
: a logical to indicate whether all y scales should have the same limits. Defaults to TRUE only if the coefficient of variation (standard deviation divided by mean) of the maximum values of y is less than 25%.

facet.fixed_x
: a logical to indicate whether all x scales should have the same breaks. This acts like the inverse of x.drop.

x.date_breaks
: breaks to use when the x axis contains dates, will be determined automatically if left blank. This accepts values such as "1 day" and "2 years".

x.date_labels
: labels to use when the x axis contains dates, will be determined automatically if left blank. This accepts 'Excel' date-language such as "d mmmm yyyy".

x.date_remove_years
: a logical to indicate whether the years of all x values must be unified. This will set the years of all x values to 1970 if the data does not contain a leap year, and

to 1972 otherwise. This allows to plot years on the `category` while maintaining a date range on `x`. The default is `FALSE`, unless `category` contains all years present in `x`.

`category.focus` a value of `category` that should be highlighted, meaning that all other values in `category` will be greyed out. This can also be a numeric value between 1 and the length of unique values of `category`, e.g. `category.focus = 2` to focus on the second legend item.

`colour` get_colour(s) to set, will be evaluated with `get_colour()` if set. This can also be one of the viridis colours with automatic implementation for any plot: `"viridis"`, `"magma"`, `"inferno"`, `"plasma"`, `"cividis"`, `"rocket"`, `"mako"` or `"turbo"`. Also, this can also be a named vector to match values of `category`, see *Examples*. Using a named vector can also be used to manually sort the values of `category`.

`colour_fill` get_colour(s) to be used for filling, will be determined automatically if left blank and will be evaluated with `get_colour()`

`colour_opacity` amount of opacity for `colour`/`colour_fill` (0 = solid, 1 = transparent)

`x.lbl_angle` angle to use for the x axis in a counter-clockwise direction (i.e., a value of `90` will orient the axis labels from bottom to top, a value of `270` will orient the axis labels from top to bottom)

`x.lbl_align` alignment for the x axis between `0` (left aligned) and `1` (right aligned)

`x.lbl_italic` logical to indicate whether the x labels should in in *italics*

`x.lbl_taxonomy` a logical to transform all words of the x labels into italics that are in the microorganisms data set of the `AMR` package. This uses `md_to_expression()` internally and will set `x.labels` to parse expressions.

`x.remove`, `y.remove`
a logical to indicate whether the axis labels and title should be removed

`x.position`, `y.position`
position of the axis

`x.max_items`, `category.max_items`, `facet.max_items`
number of maximum items to use, defaults to infinite. All other values will be grouped and summarised using the `summarise_function` function. **Please note:** the sorting will be applied first, allowing to e.g. plot the top *n* most frequent values of the x axis by combining `x.sort = "freq-desc"` with `x.max_items = n`.

`x.max_txt`, `category.max_txt`, `facet.max_txt`
the text to use of values not included number of `*.max_items`. The placeholder `%n` will be replaced with the outcome of the `summarise_function` function, the placeholder `%p` will be replaced with the percentage.

`x.breaks`, `y.breaks`
a breaks function or numeric vector to use for the axis

`x.n_breaks`, `y.n_breaks`
number of breaks, only useful if `x.breaks` cq. `y.breaks` is `NULL`

`x.transform`, `y.transform`, `category.transform`
a transformation function to use, e.g. `"log2"`. This can be: `"asinh"`, `"asn"`, `"atanh"`, `"boxcox"`, `"compose"`, `"date"`, `"exp"`, `"hms"`, `"identity"`, `"log"`, `"log10"`, `"log1p"`, `"log2"`, `"logit"`, `"modulus"`, `"probability"`, `"probit"`, `"pseudo_log"`, `"reciprocal"`, `"reverse"`, `"sqrt"`, `"time"`, `"timespan"`, `"yj"`.

`x.expand`, `y.expand`
: expansion to use for the axis, can be length 1 or 2. `x.expand` defaults to 0.5 and `y.expand` defaults to `0.25`, except for sf objects (then both default to 0).

`x.limits`, `y.limits`
: limits to use for the axis, can be length 1 or 2. Use `NA` for the highest or lowest value in the data, e.g. `y.limits = c(0, NA)` to have the y scale start at zero.

`x.labels`, `y.labels`, `y_secondary.labels`
: a labels function or character vector to use for the axis

`x.character`
: a logical to indicate whether the values of the x axis should be forced to character. The default is `FALSE`, except for years (values between 2000 and 2050) and months (values from 1 to 12).

`x.drop`
: logical to indicate whether factor levels should be dropped

`x.mic`
: logical to indicate whether the x axis should be formatted as MIC values, by dropping all factor levels and adding missing factors of 2

`x.zoom`, `y.zoom`
: a logical to indicate if the axis should be zoomed on the data, by setting `x.limits = c(NA, NA)` and `x.expand = 0` for the x axis, or `y.limits = c(NA, NA)` and `y.expand = 0` for the y axis

`y.24h`
: a logical to indicate whether the y labels and breaks should be formatted as 24-hour sequences

`y.age`
: a logical to indicate whether the y labels and breaks should be formatted as ages in years

`y.scientific`, `y_secondary.scientific`
: a logical to indicate whether the y labels should be formatted in scientific notation. Defaults to `TRUE` only if the range of the y values spans more than `10e5`.

`y.percent`, `y_secondary.percent`
: a logical to indicate whether the y labels should be formatted as percentages

`y.percent_break`
: a value on which the y axis should have breaks

`y_secondary`
: values to use for plotting along the secondary y axis. This functionality is poorly supported by `ggplot2` and might give unexpected results. Setting the secondary y axis will set the colour to the axis titles.

`y_secondary.colour`, `y_secondary.colour_fill`
: colours to set for the secondary y axis, will be evaluated with `get_colour()`

`category.labels`, `category.percent`, `category.breaks`, `category.expand`, `category.midpoint`
: settings for the plotting direction `category`.

`category.limits`
: limits to use for a numeric category, can be length 1 or 2. Use `NA` for the highest or lowest value in the data, e.g. `category.limits = c(0, NA)` to have the scale start at zero.

`category.date_breaks`
: breaks to use when the category contains dates, will be determined automatically if left blank. This will be passed on to `seq.Date(by = ...)` and thus can be: a number, taken to be in days, or a character string containing one of "day", "week", "month", "quarter" or "year" (optionally preceded by an integer and a space, and/or followed by "s").

`category.date_labels`
: labels to use when the category contains dates, will be determined automatically if left blank. This accepts 'Excel' date-language such as `"d mmmm yyyy"`.

`category.character`
: a logical to indicate whether the values of the category should be forced to character. The default is `FALSE`, except for years (values between 2000 and 2050) and months (values from 1 to 12).

`x.sort`, `category.sort`, `facet.sort`
: sorting of the plotting direction, defaults to `TRUE`, except for continuous values on the x axis (such as dates and numbers). Applying one of the sorting methods will transform the values to an ordered factor, which `ggplot2` uses to orient the data. Valid values are:

  - A manual vector of values
  - `TRUE`: sort factors on their levels, otherwise sort ascending on alphabet, while maintaining numbers in the text (*numeric* sort)
  - `FALSE`: sort according to the order in the data
  - `NULL`: do not sort/transform at all
  - `"asc"` or `"alpha"`: sort as `TRUE`
  - `"desc"`: sort factors on their reversed levels, otherwise sort descending on alphabet, while maintaining numbers in the text (*numeric* sort)
  - `"order"` or `"inorder"`: sort as `FALSE`
  - `"freq"` or `"freq-desc"`: sort descending according to the frequencies of `y` computed by `summarise_function` (highest value first)
  - `"freq-asc"`: sort ascending according to the frequencies of `y` computed by `summarise_function` (lowest value first)

`x.complete`, `category.complete`, `facet.complete`
: a value to complete the data. This makes use of `tidyr::full_seq()` and `tidyr::complete()`. For example, using `x.complete = 0` will apply `data |> complete(full_seq(x, ...), fill = list(x = 0))`. Using value `TRUE` (e.g., `x.complete = TRUE`) is identical to using value `0`.

`datalabels`
: values to show as datalabels, see also `datalabels.format`. This can be:

  - Left blank. This will default to the values of `y` in column-type plots, or when plotting spatial 'sf' data, the values of the first column. It will print a maximum of 25 labels unless `datalabels = TRUE`.
  - `TRUE` or `FALSE` to force or remove datalabels
  - A function to calculate over `.data`, such as `datalabels = paste(round(column1), "\n", column2)`

`datalabels.round`
: number of digits to round the datalabels, applies to both `"%n"` and `"%p"` for replacement (see `datalabels.format`)

`datalabels.format`
: format to use for datalabels. This can be a function (such as `euros()`) or a text. For the text, `"%n"` will be replaced by the count number, and `"%p"` will be replaced by the percentage of the total count. Use `datalabels.format = NULL` to *not* transform the datalabels.

`datalabels.colour`, `datalabels.colour_fill`, `datalabels.size`, `datalabels.angle`, `datalabels.lineheight`
: settings for the datalabels

`decimal.mark`
: decimal mark, defaults to `dec_mark()`

`big.mark`
: thousands separator, defaults to `big_mark()`

`summarise_function`
: a function to use if the data has to be summarised, see *Examples*. This can also be `NULL`, which will be converted to `function(x) x`.

`stacked`
: a logical to indicate that values must be stacked

`stackedpercent`
: a logical to indicate that values must be 100% stacked

`horizontal`
: a logical to turn the plot 90 degrees using `coord_flip()`. This option also updates some theme options, so that e.g., `x.lbl_italic` will still apply to the original x axis.

`reverse`
: a logical to reverse the *values* of `category`. Use `legend.reverse` to reverse the *legend* of `category`.

`smooth`
: a logical to add a smooth. In histograms, this will add the density count as an overlaying line (default: `TRUE`). In all other cases, a smooth will be added using `geom_smooth()` (default: `FALSE`).

`smooth.method`, `smooth.formula`, `smooth.se`, `smooth.level`, `smooth.alpha`, `smooth.linewidth`, `smooth.linetype`, `smooth.colour`
: settings for `smooth`

`size`
: size of the geom. Defaults to `2` for geoms point and jitter, `5` for a dumbbell plots (using `type = "dumbbell"`), and to `0.75` otherwise.

`linetype`
: linetype of the geom, only suitable for geoms that draw lines. Defaults to 1.

`linewidth`
: linewidth of the geom, only suitable for geoms that draw lines. Defaults to:
  - `0.5` for geoms that have no area (such as line), and for geoms boxplot/violin
  - `0.1` for sf
  - `0.25` for geoms that are continous and have fills (such as area)
  - `1.0` for dumbbell plots (using `type = "dumbbell"`)
  - `0.5` otherwise (such as histogram and area)

`binwidth`
: width of bins (only useful for `geom = "histogram"`), can be specified as a numeric value or as a function that calculates width from `x`, see `geom_histogram()`. It defaults to approx. `diff(range(x))` divided by 12 to 22 based on the data.

`width`
: width of the geom. Defaults to `0.75` for geoms boxplot, violin and jitter, and to `0.5` otherwise.

`jitter_seed`
: seed (randomisation factor) to be set when using `type = "jitter"`

`violin_scale`
: scale to be set when using `type = "violin"`, can also be set to `"area"`

`legend.position`
: position of the legend, must be `"top"`, `"right"`, `"bottom"`, `"left"` or `"none"` (or `NA` or `NULL`), can be abbreviated. Defaults to `"right"` for numeric `category` values and 'sf' plots, and `"top"` otherwise.

`legend.reverse`, `legend.barheight`, `legend.barwidth`, `legend.nbin`, `legend.italic`
: other settings for the legend

`sankey.node_width`
: width of the vertical nodes in a Sankey plot (i.e., when `type = "sankey"`)

`sankey.node_whitespace`
: whitespace between the nodes

`sankey.alpha`
: alpha of the flows in a Sankey plot (i.e., when `type = "sankey"`)

`sankey.remove_axes`
: logical to indicate whether all axes must be removed in a Sankey plot (i.e., when `type = "sankey"`)

`zoom`
: a logical to indicate if the plot should be scaled to the data, i.e., not having the x and y axes to start at 0. This will set `x.zoom = TRUE` and `y.zoom = TRUE`.

`sep`
: separator character to use if multiple columns are given to either of the three directions: `x`, `category` and `facet`, e.g. `facet = c(column1, column2)`

`print`
: a logical to indicate if the result should be printed instead of just returned

`text_factor`
: text factor to use, which will apply to all texts shown in the plot

`font`
: font (family) to use, can be set with `options(plot2.font = "...")`. Can be any installed system font or any of the > 1400 font names from Google Fonts. When using custom fonts in R Markdown, be sure to set the chunk option `fig.showtext = TRUE`, otherwise an informative error will be generated.

`theme`
: a valid `ggplot2` theme to apply, or `NULL` to use the default `theme_grey()`. This argument accepts themes (e.g., `theme_bw()`), functions (e.g., `theme_bw`) and characters themes (e.g., `"theme_bw"`). The default is `theme_minimal2()`, but can be set with `options(plot2.theme = "...")`.

`background`
: the background colour of the entire plot, can also be `NA` to remove it. Will be evaluated with `get_colour()`. Only applies when `theme` is not `NULL`.

`markdown`
: a logical to turn all labels and titles into plotmath expressions, by converting common markdown language using the `md_to_expression()` function (defaults to `TRUE`)

`minimum`
: minimum number of results, defaults to `30`

`remove_intrinsic_resistant`
: a logical to indicate that rows with 100% resistance must be removed from the data set before plotting

`language`
: language to be used for antibiotic names

`...`
: any argument to give to the geom. This will override automatically-set settings for the geom.

Details

For antimicrobial resistance (AMR) data analysis, use the `bug_drug_combinations()` or the `sir_df()` function from the `AMR` package on a data set with antibiograms. The result can be used as input for `plot2()`.

The QC-test can be acquired with `certestats::qc_test()`. It applies the Nelson QC rules for a vector of values.

The detection of disease clusters can be done using `certestats::early_warning_cluster()`. Use `size` to alter the size of the triangles that indicate clusters.

Examples

```
# AMR DATA ANALYSIS ----------------------------------------------------
if (require("AMR")) {
  example_isolates[, c("mo", "AMX", "AMC", "ward")] |>
    antibiogram(mo_transform = "gramstain",
                language = "nl") |>
    plot2()
}

if (require("AMR")) {
  example_isolates[, c("mo", "AMX", "AMC", "ward")] |>
    antibiogram(mo_transform = "gramstain",
                language = "nl",
                syndromic_group = "ward") |>
    plot2()
}

# DISEASE CLUSTERS -----------------------------------------------------
cases <- data.frame(date = sample(seq(as.Date("2015-01-01"),
                                      as.Date("2022-12-31"),
                                      "1 day"),
                                  size = 300),
                    patient = sample(LETTERS, size = 300, replace = TRUE))
check <- certestats::early_warning_cluster(cases,
                                           minimum_cases = 1,
                                           threshold_percentile = 0.75)

check |> plot2()
```

`scale_certe` *Certe Scales for Aesthetics*

Description

These scales apply the colours of Certe, using the 'certestyle' package.

Usage

```
scale_colour_certe_c(
  ...,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour"
)

scale_color_certe_c(
  ...,
```

```
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour"
)

scale_fill_certe_c(
  ...,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "fill"
)

scale_colour_certe_d(colour = "certe")

scale_color_certe_d(colour = "certe")

scale_fill_certe_d(colour = "certe")
```

Arguments

`...` Arguments passed on to `continuous_scale`

`scale_name` **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

`palette` A palette function that when called with a numeric vector with values between 0 and 1 returns the corresponding output values (e.g., `scales::pal_area()`).

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks computed by the transformation object
- A numeric vector of positions
- A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (one minor break between each major break)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major breaks.

`n.breaks` An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only

have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

`labels` One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`rescaler` A function used to scale the input values to the range [0, 1]. This is always `scales::rescale()`, except for diverging and n colour gradients (i.e., `scale_colour_gradient2()`, `scale_colour_gradientn()`). The `rescaler` is ignored by position scales, which always use `scales::rescale()`. Also accepts rlang lambda function notation.

`oob` One of:

- Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation.
- The default (`scales::censor()`) replaces out of bounds values with `NA`.
- `scales::squish()` for squishing out of bounds values into range.
- `scales::squish_infinite()` for squishing infinite values into range.

`trans` **[Deprecated]** Deprecated in favour of `transform`.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

`values` if colours should not be evenly positioned along the gradient this vector gives the position (between 0 and 1) for each colour in the `colours` vector. See `rescale()` for a convenience function to map an arbitrary range to between 0 and 1.

`space` colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.

`na.value` Colour to use for missing values

`guide` Type of legend. Use `"colourbar"` for continuous colour bar, or `"legend"` for discrete colour legend.

aesthetics	Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the colour and fill aesthetics at the same time, via aesthetics = c("colour", "fill").
colour	a Certe colour set: "certe", "certe2", "certe3", etc. Will be evaluated with get_colour().

Index