

# Package ‘catmaply’

May 8, 2026

**Title** Heatmap for Categorical Data using 'plotly'

**Version** 0.9.5

**Description** Methods and plotting functions for displaying categorical data on an interactive heatmap using 'plotly'. Provides functionality for strictly categorical heatmaps, heatmaps illustrating categorized continuous data and annotated heatmaps. Also, there are various options to interact with the x-axis to prevent overlapping axis labels, e.g. via simple sliders or range sliders. Besides the viewer pane, resulting plots can be saved as a standalone HTML file, embedded in 'R Markdown' documents or in a 'Shiny' app.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.4.0)

**Imports** plotly, dplyr, magrittr, rlang, tidyr

**Suggests** testthat, viridis, lubridate, tidyverse, knitr, rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**URL** <https://github.com/VerkehrsbetriebeZuerich/catmaply>,  
<https://verkehrsbetriebezuerich.github.io/catmaply/>

**BugReports** <https://github.com/VerkehrsbetriebeZuerich/catmaply/issues>

**Language** en-US

**NeedsCompilation** no

**Author** Verkehrsbetriebe Zürich [cre, cph],  
Yves Mauron [aut],  
Christoph Baur [aut]

**Maintainer** Verkehrsbetriebe Zürich <github@vbz.ch>

**Repository** CRAN

**Date/Publication** 2025-06-05 07:50:14 UTC

## Contents

catmaply . . . . .	2
vbz . . . . .	7

<b>Index</b>	<b>8</b>
--------------	----------

---

catmaply	<i>Heatmap for categorical data using plotly</i>
----------	--

---

### Description

catmaply is used to easily plot categorical data on heatmaps using plotly. It can be used to plot heatmaps on categorical variables or, otherwise, plot continuous variables with categorical color range.

### Usage

```
catmaply(
  df,
  x,
  x_order,
  x_side = "top",
  x_tickangle = 90,
  x_range = 30,
  y,
  y_order,
  y_side = "left",
  y_tickangle = 0,
  z,
  xgap = 0,
  ygap = 0,
  text,
  text_color = "#444",
  text_size = 12,
  text_font_family = c("Open Sans", "verdana", "arial", "sans-serif"),
  hover_template,
  hover_hide = FALSE,
  color_palette = viridis::plasma,
  categorical_color_range = FALSE,
  categorical_col = NA,
  font_family = c("Open Sans", "verdana", "arial", "sans-serif"),
  font_size = 12,
  font_color = "#444",
  legend = TRUE,
  legend_col,
  legend_interactive = TRUE,
  tickformatstops = NULL,
```

```

  rang-slider = TRUE,
  slider = FALSE,
  slider_steps = list(slider_start = 1, slider_range = 15, slider_shift = 5,
    slider_step_name = "x"),
  slider_currentvalue_prefix = "",
  slider_step_visible = TRUE,
  slider_currentvalue_visible = TRUE,
  slider_tick_visible = TRUE,
  source = "catmaply"
)

```

### Arguments

<code>df</code>	data.frame or tibble holding the data.
<code>x</code>	column name holding the axis values for x.
<code>x_order</code>	column name holding the ordering axis values for x. if no order is specified, then x will be used for ordering x; (default:"x").
<code>x_side</code>	on which side the axis labels on the x axis should appear. options: c("top", "bottom"); (default:"top").
<code>x_tickangle</code>	the angle of the axis label on the x axis. options: range -180 until 180; (default:90).
<code>x_range</code>	the initial range that should be displayed on the x axis. Only works with non-time x-axis at the moment; (default: 30).
<code>y</code>	column name holding the axis values for y.
<code>y_order</code>	column name holding the ordering axis values for y. if no order is specified, then y will be used for ordering y; (default:"y").
<code>y_side</code>	on which side the axis labels on the y axis should appear. options: c("left", "right"); (default:"left").
<code>y_tickangle</code>	the angle of the axis label on the x axis. options: range -180 until 180; (default:0).
<code>z</code>	column name holding the values for the fields.
<code>xgap</code>	Sets the horizontal gap (in pixels) between bricks; (default: 0)
<code>ygap</code>	Sets the vertical gap (in pixels) between bricks; (default: 0)
<code>text</code>	optional column name holding the values that should be displayed in the fields. NA values will not be displayed.
<code>text_color</code>	font color to be used for text; (default: "#444").
<code>text_size</code>	font size to be used for text/annotation. Needs to be a number greater than or equal to 1; (default: 12).
<code>text_font_family</code>	the typeface that will be applied by the web browser for the text/annotation. The web browser will only be able to apply a font if it is available on the system which it operates. Provide multiple font families, separated by commas, to indicate the preference in which to apply fonts if they aren't available on the system; (default: c("Open Sans", "verdana", "arial", "sans-serif")).

`hover_template` template to be used to create the hover label; (default:missing).  
`hover_hide` boolean indicating if the hover label should be hidden or not; (default: FALSE).  
`color_palette` a color palette vector a function that is able to create one; (default: viridis::plasma).  
`categorical_color_range`  
if the resulting heatmap holds categorical field values or continuous values that belong to a category; (default: FALSE).  
`categorical_col`  
if `categorical_color_range` is TRUE, then this column is used to create categories; (default: NA).  
`font_family` the typeface that will be applied by the web browser. The web browser will only be able to apply a font if it is available on the system which it operates. Provide multiple font families, separated by commas, to indicate the preference in which to apply fonts if they aren't available on the system; (default: c("Open Sans", "verdana", "arial", "sans-serif")).  
`font_size` font size to be used for plot. needs to be a number greater than or equal to 1; (default: 12).  
`font_color` font color to be used for plot; (default: "#444").  
`legend` boolean indicating if legend should be displayed or not; (default: TRUE).  
`legend_col` column to be used for legend naming; (default: z/categorical\_col).  
`legend_interactive`  
whether the legend should be interactive or not; i.e. remove traces on click; (default: TRUE).  
`tickformatstops`  
used only if x axis is of type c("POSIXct", "POSIXt"). List of named list where each named list has one or more of the keys listed here: <https://plotly.com/r/reference/#heatmap-colorbar-tickformatstops>. Default is optimized for summarized data of level day 24 hours;  
if default value (NULL) is set, tickformatstops is set as follows: for column class "POSIXct" or "POSIXt"  

```
list(
list(dtickrange = list(NULL, 1000), value = "%H:%M:%S.%L ms"),
list(dtickrange = list(1000, 60000), value = "%H:%M:%S s"),
list(dtickrange = list(60000, 3600000), value = "%H:%M m"),
list(dtickrange = list(3600000, 86400000), value = "%H:%M h"),
list(dtickrange = list(86400000, 604800000), value = "%H:%M h"),
list(dtickrange = list(604800000, "M1"), value = "%H:%M h"),
list(dtickrange = list("M1", "M12"), value = "%H:%M h"),
list(dtickrange = list("M12", NULL), value = "%H:%M h")
)
)
for class equals Date: list(
list(dtickrange = list(NULL, 1000), value = "%H:%M:%S.%L ms"),
list(dtickrange = list(1000, 60000), value = "%H:%M:%S s"),
list(dtickrange = list(60000, 3600000), value = "%H:%M m"),
list(dtickrange = list(3600000, 86400000), value = "%H:%M h"),
list(dtickrange = list(86400000, 604800000), value = "%e. %b d"),
```

	<pre>list(dtickrange = list(604800000, "M1"), value = "%e. %b w"), list(dtickrange = list("M1", "M12"), value = "%b %y M"), list(dtickrange = list("M12", NULL), value = "%Y Y") ) ) (default: NULL)</pre>
rangeslider	boolean value indicating whether the rangeslider should be displayed or not; (default: TRUE).
slider	boolean value indicating whether to use slider or not; if specified, rangeslider will not be displayed; (default: FALSE).
slider_steps	<p>list holding the configuration of the steps to be created. There are two alternatives: auto and custom; whereas the auto mode creates the steps automatically and custom takes custom instructions on how to create the steps. For mode auto, a list with the following elements has to be submitted (values of the list element are just examples):</p> <pre>list( slider_start=1, slider_range=15, slider_shift=5, slider_step_name="x" )</pre> <p>This will create the steps automatically for you, essentially starting at position <code>slider_start</code>, shifting the window of size <code>slider_range</code> along the x axis with a stepsize of <code>slider_shift</code>. The stepnames are automatically selected with the x value of the left side of the <code>slider_range</code> (so for 1 it would take the first value of the x axis as name of the step).</p> <p>With custom, on the other hand, you can define the step configuration without any restrictions. The custom configuration needs to be defined in a list with the following elements.</p> <pre>list( list(name="Step_One", range=c(1, 50)), list(name="Step_Two", range=c(5, 55)), ... ).</pre> <p>(default: <code>list(slider_start=1, slider_range=15, slider_shift=5, slider_step_name="x")</code>).</p>
slider_currentvalue_prefix	prefix to be used for the slider title. Only used if <code>slider=TRUE</code> . (default: "").
slider_step_visible	boolean indicating if the step names should be displayed for the slider. (default: TRUE).
slider_currentvalue_visible	boolean indicating if the currently selected value should be displayed above the slider. (default: TRUE).

`slider_tick_visible` boolean indicating if the tickvalues should be displayed below the slider. (default: TRUE).

`source` a character string of length 1. Match the value of this string with the `source` argument in `event_data()` to retrieve the event data corresponding to a specific plot (shiny apps can have multiple plots).

### Value

plot\_ly object

### Examples

```
library(catmaply)

data("vbz")
df <- vbz[[3]]

# simple plot
catmaply(
  df,
  x=trip_seq,
  x_order = trip_seq,
  y = stop_name,
  y_order = stop_seq,
  z = occ_category
)

# categorical color range and template
catmaply(
  df,
  x = trip_seq,
  y = stop_name,
  y_order = stop_seq,
  z = occupancy,
  categorical_color_range=TRUE,
  categorical_col = occ_category,
  hover_template = paste(
    '<b>Trip</b>:', trip_seq,
    '<br><b>Stop</b>:', stop_seq,
    '<br><b>Occupancy</b>:', occ_category,
    '<extra></extra>'
  )
)
# for more examples, see vignette
```

---

vbz

*Sample files provided by VBZ*

---

**Description**

Sample data of three distinct routes.

**Usage**

vbz

**Format**

list with data.frame elements

**trip\_seq** Sequence order of trips.

**stop\_seq** Sequence order of stops.

**stop\_name** Name of the stop.

**trip\_id** Id of trip

**circulation\_name** Name of circulation.

**line\_name** Name of line.

**vehicle** Type of vehicle.

**occupancy** Occupancy.

**occ\_category** Category of occupancy.

**departure\_time** Time of departure.

**number\_of\_measurements** Number of measurements.

**occ\_cat\_name** Occupancy category name

**direction** Direction.

**Source**

vbz

# Index

\* **datasets**

vbz, [7](#)

catmaply, [2](#)

vbz, [7](#)