Package 'epiviz'

March 5, 2026

Title Data Visualisation Functions for Epidemiological Data Science
Products

Version 0.1.2

Description Tools for making epidemiological reporting easier with consistent
static and dynamic charts and maps. Builds on 'ggplot2' for static
visualizations as described in Wickham (2016) <doi:10.1007/978-3-319-24277-4>
and 'plotly' for interactive visualizations as described in
Sievert (2020) <doi:10.1201/9780429447273>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

LazyDataCompression xz

Suggests testthat (>= 3.0.0), mockery, withr, yarrr, knitr, rmarkdown,
shiny

Config/testthat/edition 3

Imports grDevices, assertthat, classInt, dplyr, ggplot2, RColorBrewer,
plotly, sf, leaflet, htmltools, stringr, scales, tidyr,
lubridate, ISOweek, forcats, slider, rlang, ellmer, jsonlite,
lifecycle

Depends R (>= 4.1.0)

VignetteBuilder knitr

URL https://github.com/ukhsa-collaboration/epiviz

BugReports https://github.com/ukhsa-collaboration/epiviz/issues

NeedsCompilation no

Author Harshana Liyanage [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9738-6349>),
James Duke [aut],
Alex Bhattacharya [aut] (ORCID:
<https://orcid.org/0000-0003-3000-2771>),

Liam Fitzpatrick [aut],
James Leatherland [aut],
Jordan Worth [aut],
Emma Parker [aut],
Jade Talbot [ctb] (ORCID: <https://orcid.org/0000-0001-6460-5170>),
Karen Crawford [ctb],
Frederick Sloots [ctb],
Charlie Turner [ctb]

Contents

---

age_sex_pyramid   *Age-Sex Pyramid*

Description

This function creates an age-sex pyramid visualization, either as a static ggplot or an interactive plotly chart. The function can take either a line list (ungrouped data) or already grouped data as input. When using a line list, the function processes the data, groups it by age and sex, and then generates the pyramid. If grouped data is provided, it directly creates the pyramid.

Usage

```
age_sex_pyramid(
  dynamic = FALSE,
  base = NULL,
  params = list(df, var_map = list(age_var = "age", dob_var = "date_of_birth", sex_var =
    "sex", age_group_var = "age_group", value_var = "value", ci_lower = "ci_lower",
    ci_upper = "ci_upper"), mf_colours = c("#440154", "#2196F3"), x_breaks = 10,
   x_axis_title = "Number of cases", y_axis_title = "Age group (years)", text_size = 12,
    ci = NULL, ci_colour = "red", age_breakpoints = c(0, 5, 19, 65, Inf),
   age_calc_refdate = Sys.Date(), grouped = FALSE, legend_pos = "top", chart_title = "")
)
```

Arguments

dynamic	Logical. If TRUE, the function returns an interactive plotly chart. If FALSE, a static ggplot chart is returned.
base	An optional base plot to add the pyramid to. Default is NULL.
params	A list of parameters including:

df Data frame containing the data to be used.

var_map A list mapping variable names in the data frame to the expected names used in the function.

age_var Name of the variable in df containing age values. Default is 'age'.

dob_var Name of the variable in df containing date of birth values. Default is 'date_of_birth'.

sex_var Name of the variable in df containing sex values. Default is 'sex'.

age_group_var Name of the variable in df containing pre-grouped age groups (if grouped = TRUE).

value_var Name of the column containing the value counts (if grouped = TRUE).

ci_lower Name of the column containing lower confidence limits (if ci = TRUE).

ci_upper Name of the column containing upper confidence limits (if ci = TRUE).

mf_colours A vector of 2 colours used to fill the male and female bars in the plot. The first colour will be used for males, and the second for females. Default is c("#440154", "#2196F3").

x_breaks Number of breaks on the x-axis. Default is 10.

y_axis_title Title of the y-axis. Default is "Individual count".

x_axis_title Title of the x-axis. Default is "Number of cases".

text_size Size of the text in the plot. Default is 12.

ci Confidence interval. If ci = "errorbar" then confidence intervals will be be plotted with each bar as errorbars. If ci = "errorbar" and grouped = FALSE, then default confidence intervals are applied using the normal approximation to the Poisson distribution, with bounds set at $\pm 1.96 \times \sqrt{n}$.

ci_colour Colour of the plotted errorbars if ci = TRUE. Default is "red".

age_breakpoints A numeric vector specifying the breakpoints for age groups. Default is c(0, 5, 19, 65, Inf).

age_calc_refdate Reference date for calculating age from date of birth. Default is Sys.Date().

grouped Logical. If TRUE, assumes the data is pre-grouped by age and sex. If FALSE, the function processes the line list data. Default is FALSE.

legend_pos Position of the legend. Default is "top".

chart_title Text to use as the chart title.

Details

When grouped = FALSE, the function processes a line list by grouping the data by age and sex, calculating age based on either the provided age column or date of birth, and then generating the age-sex pyramid. When grouped = TRUE, it assumes the data is already grouped and uses the provided values directly to generate the pyramid.

Value

A ggplot or plotly object representing the age-sex pyramid, depending on the value of dynamic.

Examples

```
# Example using a line list
df <- epiviz::lab_data
result <- age_sex_pyramid(
  dynamic = FALSE,
  params = list(
    df = df,
    var_map = list(dob_var = 'date_of_birth', sex_var = 'sex'),
    grouped = FALSE
  )
)
```

---

col_chart   *Column chart*

Description

A function for producing either a static (ggplot) or dynamic (plotly) column chart.

Usage

```
col_chart(
  dynamic = FALSE,
  params = list(df = NULL, x = NULL, y = NULL, x_time_series = FALSE,
    x_time_series_bar_labels = FALSE, time_period = "day", group_var = NULL,
    group_var_barmode = "stack", fill_colours = "lightblue", bar_border_colour =
   "transparent", bar_labels = NULL, bar_labels_pos = "bar_above", bar_labels_font_size
    = 8, bar_labels_font_colour = "black", bar_labels_angle = 0, bar_labels_percent =
   FALSE, case_boxes = FALSE, case_boxes_colour = "white", axis_flip = FALSE, ci = NULL,
    ci_upper = NULL, ci_lower = NULL, ci_legend = TRUE,
      ci_legend_title =
   "Confidence interval", ci_colours = "red", errorbar_width = NULL, chart_title = NULL,
    chart_title_size = 13, chart_title_colour = "black", chart_footer = NULL,
    chart_footer_size = 12, chart_footer_colour = "black", x_axis_title = NULL,
   y_axis_title = NULL, x_axis_title_font_size = 11, y_axis_title_font_size = 11,
```

```
    x_axis_label_angle = NULL, y_axis_label_angle = NULL, x_axis_label_font_size = 9,
  y_axis_label_font_size = 9, x_limit_min = NULL, x_limit_max = NULL, y_limit_min =
    NULL,
     y_limit_max = NULL, x_axis_break_labels = NULL, y_axis_break_labels =
   NULL, x_axis_n_breaks = NULL, y_axis_n_breaks = NULL, x_axis_reverse = FALSE,
  y_percent = FALSE, show_gridlines = TRUE, show_axislines = TRUE, legend_title = "",
  legend_pos = "right", legend_font_size = 8, legend_title_font_size = 8, hline = NULL,
  hline_colour = "black", hline_width = 0.5, hline_type = "dashed", hline_label = NULL,
    hline_label_colour = "black", hover_labels = NULL)
)
```

Arguments

dynamic   Logical indicating whether to produce a dynamic (plotly) output. Default is `FALSE`, which will return a static ggplot output.

params   A named list containing arguements used to create the plot.

        df A data frame containing data used to create the column chart.

        x character, Name of the variable in `df` containing the values used to populate the x-axis.

        y character, Name of the variable in `df` containing the values used to populate the y-axis.)

        x_time_series If `x_time_series = TRUE` then x-axis data will be aggregated into a time series. `x` must be a date variable when `x_time_series = TRUE`. The granularity of the time series can be set using the `time_period` parameter. If `y` is not provided, then `x_time_series` will assume that each row corresponds to a single observation and aggregate accordingly.

        x_time_series_bar_labels If `x_time_series_bar_labels = TRUE` then labels equalling y-axis values will be added to each bar.

        time_period The time period to be used along the x-axis. Options include `c("day","year","month","quarter","year_month","year_quarter", "iso_year","iso_week","start_iso_year_week","iso_year_week")`. Default = `"day"`

        group_var Name of the variable in `df` used to define separate groups within each bar, e.g. 'species' or 'region'.

        group_var_barmode Indicates how grouped bar data should be plotted. Options include `c("group","stack")`. Default = `"stack"`.

        fill_colours Colours used to fill bars on chart. If `group_var` has not been provided, then `fill_colours` must be a character containing a single colour (default = `"lightblue"`). If `group_var` has been provided, then `fill_colours` must be a character vector of colours with a number of elements equal to the number of unique groups in `group_var`. If a named character vector is provided where the names are values within `group_var`, then each colour will be mapped to it's corresponding value in `group_var` on the output chart and legend (e.g. `c("KLEBSIELLA PNEUMONIAE" = "#007C91", "STAPHYLOCOCCUS AUREUS" = "#8A1B61", "PSEUDOMONAS AERUGINOSA" = "#FF7F32")` or `setNames(c("#007C91", "#8A1B61","#FF7F32"), c("KLEBSIELLA PNEUMONIAE","STAPHYLOCOCCUS AUREUS","PSEUDOMONAS AERUGINOSA")))`

bar_border_colour character, Colour of the border around each bar. Default = `"transparent"`, meaning that no border colour is drawn as default.

bar_labels character, Name of the variable in df containing the labels to be used for each bar. To set bar lables when x_time_series = TRUE, set `x_time_series_bar_labels` = TRUE and labels will be added to each bar equal to y-axis values.

bar_labels_pos character, The position on the bars that labels will be plotted, permitted values are `c('bar_above','bar_base','bar_centre','above_errorbar')`. Default = `'bar_above'`.

bar_labels_font_size numeric, Font size for the bar labels. Default = 8.

bar_labels_font_colour character, Font colour for the bar labels. Default = `'black'`.

bar_labels_angle numeric, Font angle for the bar labels.

bar_labels_percent boolean, If bar_labels_percent = TRUE then the values in `bar_labels` will be converted into a percentage before plotting.

case_boxes boolean, If case_boxes = TRUE then a boundary box will be drawn around each case within each bar. Defaults to case_boxes = FALSE.

case_boxes_colour The colour of the border around each case box if case_boxes = TRUE. Default = `"white"`.

axis_flip boolean, If set to TRUE then x and y axes will be flipped and bars will be drawn hozizontally rather than vertically.

ci Confidence interval. If ci = `"errorbar"` then confidence intervals be be plotted with each bar as errorbars. If ci is provided, then `ci_upper` and `ci_lower` must also be provided.

ci_upper character, Name of the variable in df used as the upper confidence limit for each bar. Mandatory when ci is provided.

ci_lower character, Name of the variable in df used as the lower confidence limit for each bar. Mandatory when ci is provided.

ci_legend Logical indicating whether a separate legend should be included in the chart for confidence interval parameters. Only applies when group_var is provided. Defaults to FALSE.

ci_legend_title Text to use as title for separate legend when ci_legend = TRUE. Default = `"Confidence interval"`.

ci_colours Colour(s) used for plotting errorbars when ci = `"errorbar"`. If group_var has been provided, then fill_colours must be a character vector of colours with a number of elements equal to the number of unique groups in group_var. If a named character vector is provided where the names are values within group_var, then each colour will be mapped to it's corresponding value in group_var on the output chart and legend (e.g. `c("KLEBSIELLA PNEUMONIAE" = "#007C91"`, `"STAPHYLOCOCCUS AUREUS"` = `"#8A1B61"`,`"PSEUDOMONAS AERUGINOSA"` = `"#FF7F32")` or `setNames(c("#007C91","#8A1B61","` `c("KLEBSIELLA PNEUMONIAE","STAPHYLOCOCCUS AUREUS","PSEUDOMONAS AERUGINOSA")))`

errorbar_width Horizontal width of the plotted error bars when ci = `"errorbar"`.

chart_title Text to use as the chart title.

chart_title_size Font size of chart title. Default = 13.

chart_title_colour Font colour of chart title. Default = `"black"`.

chart_footer Text to use as chart footer.

chart_footer_size Font size of chart footer. Default = 12.

chart_footer_colour Font colour of chart footer. Default = "black".

x_axis_title Text used for x-axis title. Defaults to name of x-variable if not stated.

y_axis_title Text used for y-axis title. Defaults to name of y-variable if not stated.

x_axis_title_font_size Font size of the x-axis title. Default = 11.

y_axis_title_font_size Font size of the y-axis title. Default = 11.

x_axis_label_angle Angle for x-axis label text.

y_axis_label_angle Angle for y-axis label text.

x_axis_label_font_size Font size for the x-axis tick labels. Default = 9.

y_axis_label_font_size Font size for the y-axis tick labels. Default = 9.

x_limit_min Lower limit for the x-axis. Default used if not provided.

x_limit_max Upper limit for the x-axis. Default used if not provided.

y_limit_min Lower limit for the y-axis. Default used if not provided.

y_limit_max Upper limit for the y-axis. Default used if not provided.

x_axis_break_labels Vector of values to use for x-axis breaks. Defaults used if not provided. If x_time_series = TRUE then Values provided must match the formatting of time_period.

y_axis_break_labels Vector of values to use for y-axis breaks. Defaults used if not provided.

x_axis_n_breaks Scales x-axis with approximately n breaks. Cannot be provided if x_axis_break_labels has also been provided.

y_axis_n_breaks Scales y-axis with approximately n breaks. Cannot be used if y_axis_break_labels has also been provided.

x_axis_reverse Reverses x-axis scale if x_axis_reverse = TRUE.

y_percent Converts y-axis to percentage scale if y_percent = TRUE.

show_gridlines Logical to show chart gridlines. Default = TRUE.

show_axislines Logical to show chart axis lines. Default = TRUE.

legend_title Text used for legend title.

legend_pos Position of the legend. Permitted values = c("top","bottom","right","left")

legend_font_size Font size used in the legend. Default = 8.

legend_title_font_size Font size used for the legend title. Default = 8.

hline Adds horizontal line across the chart at the corresponding y-value. Multiple values may be provided as a vector to add multiple horizontal lines.

hline_colour Colour of the horizontal lines if hline is provided. A vector of colours can be provided to colour individual hlines if multiple hlines have been provided. Default = "black".

hline_width Numerical width of the horizontal lines if hline is provided. A vector of numerical widths can be provided for individual hlines if multiple hlines have been provided. Default = 0.5.

hline_type Line style of the horizontal lines if `hline` is provided. A vector of line styles can be provided to style hlines if multiple hlines have been provided. Permitted values = c("solid", "dotted", "dashed", "longdash", "dotdash"). Default = ″`dashed`″.

hline_label Text to label the horizontal lines if `hline` is provided. A vector of text strings can be provided to label individual hlines if multiple hlines have been provided.

hline_label_colour Colour of the horizontal line labels if `hline_labels` is provided. A vector of colours can be provided to colour individual hline_labels if multiple hline_labels have been provided. Default = "black".

hover_labels string, Text to be used in the hover-over labels in a dynamic chart. Accepts html, use `'%{x}'` to reference corresponding x-axis values (i.e. date intervals) and `'%{y}'` to reference y-axis values, e.g. `hover_labels =` ″`<b>Date:</b> %{x}<br><b>Count:</b> %{y}`″.

Value

A ggplot or plotly object.

Examples

```
# Basic column chart
library(epiviz)
library(dplyr)

detections_by_region <- lab_data |>
 filter(specimen_date >= as.Date("2023-01-01") & specimen_date <= as.Date("2023-12-31")) |>
 group_by(region) |>
 summarise(detections = n()) |>
 ungroup()

my_chart <- col_chart(
  params = list(
    df = detections_by_region,
    x = "region",
    y = "detections",
    fill_colours = "#007C91",
    chart_title = "Laboratory Detections by Region 2023",
    x_axis_title = "Region",
    y_axis_title = "Number of detections"
  )
)
```

---

epi_curve	*Epi Curve*

Description

A function for producing either a static (ggplot) or dynamic (plotly) epidemic curve.

Usage

```
epi_curve(
 dynamic = FALSE,
 params = list(df = NULL, y = NULL, date_var = NULL, date_start = NULL, date_end = NULL,
   time_period = "day", group_var = NULL, group_var_barmode = "stack", fill_colours =
   "lightblue", bar_border_colour = "transparent", case_boxes = FALSE, case_boxes_colour
    = "white", rolling_average_line = FALSE, rolling_average_line_lookback = 7,
    rolling_average_line_colour = "red", rolling_average_line_width = 1,
   rolling_average_line_legend_label = "Rolling average", cumulative_sum_line = FALSE,
   cumulative_sum_line_colour = "darkblue",
     cumulative_sum_line_width = 1,
   cumulative_sum_line_legend_label = "Cumulative sum", cumulative_sum_line_axis_title =
   "Cumulative Sum", chart_title = NULL, chart_title_size = 13, chart_title_colour =
   "black", chart_footer = NULL, chart_footer_size = 12, chart_footer_colour = "black",
    x_axis_title = NULL, y_axis_title = NULL, x_axis_title_font_size = 11,
   y_axis_title_font_size = 11, x_axis_label_angle = NULL, y_axis_label_angle = NULL,
    x_axis_label_font_size = 9, y_axis_label_font_size = 9, y_limit_min = NULL,

    y_limit_max = NULL, x_axis_break_labels = NULL, y_axis_break_labels = NULL,
   y_axis_n_breaks = NULL, show_gridlines = FALSE, show_axislines = TRUE, legend_title =
   "", legend_pos = "right", legend_font_size = 8, legend_title_font_size = 8, hline =
   NULL, hline_colour = "black", hline_width = 0.5, hline_type = "dashed", hline_label =
    NULL, hline_label_colour = "black", hover_labels = NULL)
 )
```

Arguments

dynamic	Logical indicating whether to produce a dynamic (plotly) output. Default is FALSE, which will return a static ggplot output.
params	A named list containing arguments used to create the plot.
	df A data frame containing data used to create the epi curve.
	date_var character, Name of the variable in df containing the dates used to populate the x-axis.
	y If data is pre-aggregated, the name of the variable in df containing the aggregated values (i.e. the values used to populate the y-axis.)
	date_start A date that will determine the minimum value along the x-axis. Any rows with date_var < date_start will be excluded from aggregates.

date_end A date that will determine the maximum value along the x-axis. Any rows with `date_var` > `date_end` will be excluded from aggregates.

time_period The time period to be used along the x-axis. Options include `c("day","year","month","quarter","year_month","year_quarter", "iso_year","iso_week","start_iso_year_week","iso_year_week")`. Default = `"day"`

group_var Name of the variable in df used to define separate groups within each bar, e.g. species or region.

group_var_barmode Indicates how grouped bar data should be plotted. Options include `c("group","stack")`. Default = `"stack"`.

fill_colours Colours used to fill bars on chart. If `group_var` has not been provided, then `fill_colours` must be a character containing a single colour (default = `"lightblue"`). If `group_var` has been provided, then `fill_colours` must be a character vector of colours with a number of elements equal to the number of unique groups in `group_var`. If a named character vector is provided where the names are values within `group_var`, then each colour will be mapped to it's corresponding value in `group_var` on the output chart and legend (e.g. `c("KLEBSIELLA PNEUMONIAE" = "#007C91", "STAPHYLOCOCCUS AUREUS" = "#8A1B61", "PSEUDOMONAS AERUGINOSA" = "#FF7F32")` or `setNames(c("#007C91", "#8A1B61","#FF7F32"), c("KLEBSIELLA PNEUMONIAE","STAPHYLOCOCCUS AUREUS","PSEUDOMONAS AERUGINOSA")))`

bar_border_colour Colour of the border around each bar. No border colour is drawn as default.

case_boxes boolean, If `case_boxes = TRUE` then a boundary box will be drawn around each case within each bar. Defaults to `case_boxes = FALSE`.

case_boxes_colour The colour of the border around each case box if if `case_boxes = TRUE`. Default = `"white"`.

rolling_average_line boolean, If `rolling_average_line = TRUE`, then a line showing the rolling mean will be added to the plot. Default = `FALSE`.

rolling_average_line_lookback Integer denoting the lookback window across which the rolling mean will be calculated (including the current time interval). Each integer denotes a division within `time_period`, e.g. if `time_period = "year_month"` and `rolling_average_line_lookback = 3` then the rolling mean will be calculated using values from the current month and the previous 2 months, and if `time_period = "day"` and `rolling_average_line_lookback = 7` then the rolling mean will be calculated using values from the previous 7 days including the current day. If there are less values within the lookback window than `rolling_average_line_lookback`, then the mean will be calculated from an incomplete window using the values available (i.e. if `rolling_average_line_lookback = 7` and `time_period = "day"` but there are only 4 values within the previous 7 days, then the rolling mean will be calculated from the 4 available values.)

rolling_average_line_colour character Colour of the rolling average line. Default = `"red"`.

rolling_average_line_width numeric Width of the rolling average line. Default = 1.

rolling_average_line_legend_label character Label to be used for the rolling average line in the chart legend.

cumulative_sum_line boolean, If `cumulative_sum_line_line` = `TRUE`, then a line showing the cumulative sum will be added to the plot. Default = `FALSE`. Values for the cumulative sum will be plotted on the secondary y-axis.

cumulative_sum_line_colour character Colour of the cumulative line. Default = `"darkblue"`.

cumulative_sum_line_width numeric Width of the cumulative sum line. Default = 1.

cumulative_sum_line_legend_label character Label to be used for the cumulative sum line in the chart legend.

cumulative_sum_line_axis_title character Axis title for the cumulative sum line secondary axis.

chart_title Text to use as chart title.

chart_title_size Font size of chart title.

chart_title_colour Font colour of chart title.

chart_footer Text to use as chart footer.

chart_footer_size Font size of chart footer.

chart_footer_colour Font colour of chart footer.

x_axis_title Text used for x-axis title. Defaults to name of x-variable if not stated.

y_axis_title Text used for y-axis title. Defaults to name of y-variable if not stated.

x_axis_title_font_size Font size of the x-axis title.

y_axis_title_font_size Font size of the y-axis title.

x_axis_label_angle Angle for x-axis label text.

y_axis_label_angle Angle for y-axis label text.

x_axis_label_font_size Font size for the x-axis tick labels.

y_axis_label_font_size Font size for the y-axis tick labels.

y_limit_min Lower limit for the y-axis. Default used if not provided.

y_limit_max Upper limit for the y-axis. Default used if not provided.

x_axis_break_labels Vector of values to use for x-axis breaks. Defaults used if not provided. Values provided must match the formatting of `time_period`.

y_axis_break_labels Vector of values to use for y-axis breaks. Defaults used if not provided.

y_axis_n_breaks Scales y-axis with approximately n breaks. Cannot be used if `y_axis_break_labels` is also provided.

show_gridlines Logical to show chart gridlines. Default = `FALSE`.

show_axislines Logical to show chart axis lines. Default = `TRUE`.

legend_title Text used for legend title.

legend_pos Position of the legend. Permitted values = c("top","bottom","right","left")

legend_font_size Font size used in the legend.

legend_title_font_size Font size used for the legend title.

hline Adds horizontal line across the chart at the corresponding y-value. Multiple values may be provided as a vector to add multiple horizontal lines.

hline_colour Colour of the horizontal lines if `hline` is provided. A vector of colours can be provided to colour individual hlines if multiple hlines have been provided.

hline_width Numerical width of the horizontal lines if `hline` is provided. A vector of numerical widths can be provided for individual hlines if multiple hlines have been provided.

hline_type Line style of the horizontal lines if `hline` is provided. A vector of line styles can be provided to style hlines if multiple hlines have been provided. Permitted values = c("solid", "dotted", "dashed", "longdash", "dotdash").

hline_label Text to label the horizontal lines if `hline` is provided. A vector of text strings can be provided to label individual hlines if multiple hlines have been provided.

hline_label_colour Colour of the horizontal line labels if `hline_labels` is provided. A vector of colours can be provided to colour individual hline_labels if multiple hline_labels have been provided.

hover_labels string, Text to be used in the hover-over labels in a dynamic chart. Accepts html, use `'%{x}'` to reference corresponding x-axis values (i.e. date intervals) and `'%{y}'` to reference y-axis values, e.g. hover_labels = "Date: %{x}
Count: %{y}".

Value

A ggplot or plotly object.

Examples

```
# Basic epi curve example
library(epiviz)

basic_epi_curve <- epi_curve(
  params = list(
    df = lab_data,
    date_var = "specimen_date",
    date_start = "2020-01-01",
    date_end = "2023-12-31",
    time_period = "year_month",
    fill_colours = "#007C91",
    chart_title = "Laboratory Detections per Month",
    x_axis_title = "Year - Month",
    y_axis_title = "Number of detections"
  )
)
```

epi_map	*Epi Map*

Description

A function for producing either static (ggplot) or dynamic (leaflet) choropleth maps.

Usage

```
epi_map(
  dynamic = FALSE,
 params = list(df = NULL, value_col = NULL, data_areacode = NULL, inc_shp = TRUE,
   shp_name = NULL, shp_areacode = NULL, fill_palette = "Blues", fill_opacity = 1,
   break_intervals = NULL, break_labels = NULL, force_cat = TRUE, n_breaks = NULL,
   labels = NULL, map_title = "", map_title_size = 13, map_title_colour = "black",
   map_footer = "", map_footer_size = 12, map_footer_colour = "black", area_labels =
   FALSE, area_labels_topn = NULL, legend_title = "", legend_pos = "topright", map_zoom
    = NULL, border_shape_name = NULL,
     border_code_col = NULL, border_areaname =
    NULL)
)
```

Arguments

dynamic	Logical indicating whether to produce a dynamic (leaflet) output. Default is FALSE, which will return a static ggplot output.
params	A named list containing arguements used in map.

- **df** Data frame containing values used to fill areas on the output map. Can include pre-merged shapefile data if inc_shp = TRUE.
- **value_col** Name of the variable in df used to fill map areas.
- **data_areacode** Name of the variable in df containing the name or code of the map areas to be plotted. (Mandatory if shp_name argument passed).
- **inc_shp** boolean parameter to indicate whether df already includes shapefile data.
- **shp_name** Data frame name or filepath of the shapefile containing the spatial information for the resultant map output. This will not be used if inc_shp = TRUE.
- **shp_areacode** Name of the variable in shp_name containing the name or code of the map areas to be plotted. (Mandatory if shp_name argument passed).
- **fill_palette** Colour palette used to fill the map areas. Can be provided as either the name of an RColorBrewer palette (e.g. `fill_palette = "YlOrRd"`), a character containing a single rgb colour code, hexcode, or colour name that will be used to generate a colour range (e.g. `fill_palette = "#007C91"`), or a character vector containing multiple rgb codes, hexcodes, or colour names that will be used to generate a colour range (e.g. `c("#007C91","purple","red")`). Defaults to the RColorBrewer "Blues" palette.)

fill_opacity numeric value between 0 and 1 to determine map fill-color opacity.

break_intervals numeric vector of interval points for legend (Mandatory if break_labels argument is passed, break_intervals and break_labels must be of equal length).

break_labels vector of labels to include in the legend. (Mandatory if break_labels argument is passed, break_intervals and break_labels must be of equal length).

force_cat boolean parameter to determine whether all arguments passed in break_labels are used in the legend, even if there are no values present in the data.

n_breaks Number of break intervals. This argument is an alternative to supplying defined breaks via break_labels, and will provide a number of evenly distributed breaks as specified (default = 5). If break_labels argument is passed, n_breaks will be ignored.

labels name of string variable in df containing labels for each map area. If dynamic = FALSE, these labels will be positioned in the centre of each map area. If dynamic = TRUE, then these labels will appear as hover-over labels. If dynamic = TRUE, labels can include HTML.

map_title string to determine map title.

map_title_size font size of map title.

map_title_colour string to determine map title colour.

map_footer_size font size of map footer.

map_footer_colour string to determine map title colour.

area_labels boolean parameter to add data_areacode as static area labels to the map areas. If dynamic = FALSE and a labels parameter has alredy been supplied, then area_labels will be ignored.

area_labels_topn numeric value to display only area_labels for areas with the top n values of value_col (e.g. if area_labels_topn = 5, only area_labels for map areas with the top 5 values of value_col will be displayed).

legend_title string to determine legend title.

legend_pos string to determine legend position. When dynamic = TRUE, both ggplot and leaflet permissable legend positions can be provided. When dynamic = FALSE, only leaflet permissable legend positions can be provided (i.e."topright", "bottomright", "bottomleft", or "topleft").

map_zoom A single row data frame with variables of 'LAT', 'LONG', and 'zoom' that allows the map to be zoomed in on a specific region (e.g. data.frame(LONG = -2.547855, LAT = 53.00366, zoom = 6)). LAT = numerical latitude coordinate for the centre point of the zoom, LONG = numerical longitude coordinate for the centre point of the zoom, zoom = numerical value to represent the depth of zoom.

border_shape_name Optional filepath for a shapefile containing additional borders to include in the output map. This should be a higher geography than the base map (e.g. if creating a map displaying UTLAs, a shapefile containing regional boundaries or higher should be used). Only boundaries contained within border_shape_name will be used, areas will be unfilled.

border_code_col Variable name of the area code / name within border_shape_name. Required if a specific area within the border shapefile is required.

border_areaname Character vector containing the name of specific areas within border_code_col to be plotted. If supplied, only the boundaries included in border_areaname will be plotted. If not supplied, the boundaries of all areas within border_shape_name will be plotted.

Value

A ggplot or leaflet object.

Examples

```
# Basic epi_map example
library(epiviz)
library(dplyr)

# Prepare data
London_staph_detections <- lab_data |>
  filter(region == "London", organism_species_name == "STAPHYLOCOCCUS AUREUS") |>
  group_by(local_authority_name) |>
  summarise(detections = n())

# Create static map
my_map <- epi_map(
  params = list(
    df = London_staph_detections,
    value_col = "detections",
    data_areacode = "local_authority_name",
    inc_shp = FALSE,
    shp_name = London_LA_boundaries_2023,
    shp_areacode = "LAD23NM",
    map_title = "Staphylococcus Aureus detections in London",
    legend_title = "Detections"
  )
)
```

---

lab_data   *Synthetic Lab Data for EpiViz functions*

Description

A dataset containing synthetic lab data for epidemiological visualisation purposes.

Usage

```
data(lab_data)
```

Format

A data frame with the following columns:

date_of_birth Date of birth of the patients.

sex Gender of the patients (Factor with levels: "Female", "Male").

organism_species_name Organism species name (Factor with levels: "KLEBSIELLA PNEUMO-
    NIAE").

specimen_date Date of specimen collection.

lab_code Laboratory codes (Factor with unique levels).

local_authority_name Name of the local authority.

local_authority_code Code of the local authority.

region Name of UKHSA regions.

Examples

```
data(lab_data)
head(lab_data)
```

---

line_chart   *Line Chart*

Description

This function generates a line chart from a data frame using specified x and y variables. Optionally,
the plot can be rendered as an interactive Plotly object. The function also allows grouping of data
based on a specified grouping variable.

Usage

```
line_chart(
  dynamic = FALSE,
  base = NULL,
  params = list(df, x, y, ci = NULL, lower = NULL, upper = NULL, error_colour =
  c("#f2c75c"), group_var, line_colour = c("blue"), line_type = "solid", width = 1,
  title = NULL, x_label = NULL, x_label_angle = NULL, y_label = NULL, y_label_angle =
  NULL, y_percent = FALSE, st_theme = NULL, add_points = FALSE, show_gridlines = FALSE,
  show_axislines = TRUE, legend_title = NULL, legend_position = NULL, hline = NULL,
    hline_colour = "red", hline_label = NULL),
  ...
)
```

Arguments

dynamic	A logical value. If TRUE, the line chart will be rendered as a Plotly object for interactivity. If FALSE, a static ggplot2 object will be returned.
base	A base plotly or ggplot2 object to add the line chart to. Default is NULL.
params	A list containing the following elements:

  - df A data frame containing the data to be plotted.
  - x A character string specifying the name of the column in df to be used for the x-axis.
  - y A character string specifying the name of the column in df to be used for the y-axis.
  - group_var A character string specifying the name of the column in df to be used for grouping the data.
  - ci Optional. A character string specifying the column in df for confidence intervals.
  - lower Optional. A character string specifying the column in df for lower bounds of confidence intervals.
  - upper Optional. A character string specifying the column in df for upper bounds of confidence intervals.
  - error_colour The color for error bars. Default is #f2c75c.
  - line_colour List of colours for lines. Default is blue.
  - line_type Line type for single graph, or list of line types Permissable values: "solid", "dotted", "dashed", "longdash", "dotdash"
  - width A numeric value specifying the width of the lines.
  - title Optional. A character string specifying the title of the plot.
  - x_label Optional. A character string specifying the label for the x-axis.
  - x_label_angle Optional. A numeric value specifying the rotation angle for the x-axis labels.
  - y_label Optional. A character string specifying the label for the y-axis.
  - y_label_angle Optional. A numeric value specifying the rotation angle for the y-axis labels.
  - y_percent Optional. A logical value. If TRUE, the y-axis will be scaled to percentages.
  - st_theme Optional. A ggplot2 theme object to customize the style of the plot.
  - add_points Optional. A logical value. If TRUE, points will be added to the line chart.

...	Additional arguments passed to geom_line for static (ggplot2) plots or to plot_ly/add_trace for dynamic (Plotly) plots, allowing custom styling of the lines (e.g., alpha, size, marker, etc.).

Value

A plotly or ggplot2 object representing the line chart.

Examples

```
library(dplyr)
library(epiviz)

# Prepare data
test_df <- epiviz::lab_data
test_df$specimen_date <- as.Date(test_df$specimen_date)

summarised_df <- test_df |>
  filter(specimen_date >= as.Date("2023-01-01") &
         specimen_date <= as.Date("2023-12-31")) |>
  group_by(organism_species_name, specimen_date) |>
  summarize(count = n(), .groups = 'drop')

# Create params list
params <- list(
  df = as.data.frame(summarised_df),
  x = "specimen_date",
  y = "count",
  group_var = "organism_species_name",
  line_colour = c("blue", "green", "orange")
)

# Generate static line chart
result <- line_chart(params = params, dynamic = FALSE)
```

---

llm_interpret   *Interpret Epidemiological Data or Visualisations using LLMs*

Description

[Experimental]

This function interprets a given data frame or ggplot visualisation by sending it to a language model API via the ellmer package. It supports multiple LLM providers, allowing users to specify the desired provider and model through environment variables.

Usage

```
llm_interpret(input, word_limit = 100, prompt_extension = NULL)
```

Arguments

input   An input object, either a data frame or a ggplot object, representing the data or
                   visualisation to be interpreted.

word_limit   Integer. The desired word length for the response. Defaults to 100.

prompt_extension

                   Character. Optional additional instructions to extend the standard prompt. De-
                   faults to NULL.

Details

Supported LLM Providers and Models:

- **OpenAI**: Utilises OpenAI's models via chat_openai(). Requires setting the OPENAI_API_KEY environment variable. Applicable models include:

    - "gpt-4.1-nano"

- **Google Gemini**: Utilises Google's Gemini models via chat_gemini(). Requires setting the GOOGLE_API_KEY environment variable. Applicable models include:

    - "gemini-2.5-flash-lite"

- **Anthropic Claude**: Utilises Anthropic's Claude models via chat_anthropic(). Requires setting the CLAUDE_API_KEY environment variable. Applicable models include:

    - "claude-sonnet-4-20250514"

Environment Variables:

- LLM_PROVIDER: Specifies the LLM provider ("openai", "gemini", "anthropic").

- LLM_API_KEY: The API key corresponding to the chosen provider.

- LLM_MODEL: The model identifier to use.

Note: Ensure that the appropriate environment variables are set before invoking this function. The function will throw an error if the specified provider is unsupported or if required environment variables are missing.

Value

A character string containing the narrative or interpretation of the input object as generated by the LLM.

Tested Models

As of October 2025, this function has been tested and verified to work with the following models:

- OpenAI: gpt-4.1-nano

- Anthropic: claude-sonnet-4-20250514

- Google Gemini: gemini-2.5-flash-lite

Additional models may be tested in the future. Users can provide custom instructions through the prompt_extension parameter for specialised analysis requirements.

---

point_chart   *Point Chart*

Description

A function for producing either static (ggplot) or dynamic (plotly) point charts.

Usage

```
point_chart(
  dynamic = FALSE,
  base = NULL,
 params = list(df = NULL, x = NULL, y = NULL, point_shape = "triangle", point_size =
    1.5, point_colours = "blue", point_labels = NULL, point_labels_size = 5,
    point_labels_hjust = 0, point_labels_vjust = 0, point_labels_nudge_x = 0,
  point_labels_nudge_y = 0, group_var = NULL, ci = NULL, ci_upper = NULL, ci_lower =
  NULL, ci_legend = TRUE, ci_legend_title = "Confidence interval", ci_colours = "red",
    errorbar_width = NULL, y_sec_axis = FALSE, y_sec_axis_no_shift = TRUE,
    y_sec_axis_percent_full = FALSE, chart_title = NULL,
     chart_title_size = 13,
    chart_title_colour = "black", chart_footer = NULL, chart_footer_size = 12,
    chart_footer_colour = "black", x_axis_title = NULL, y_axis_title = NULL,
  x_axis_title_font_size = 11, y_axis_title_font_size = 11, x_axis_label_angle = NULL,
  y_axis_label_angle = NULL, x_axis_label_font_size = 9, y_axis_label_font_size = 9,
  x_axis_reverse = FALSE, y_percent = FALSE, x_limit_min = NULL, x_limit_max = NULL,
    y_limit_min = NULL, y_limit_max = NULL, x_axis_break_labels = NULL,
    y_axis_break_labels = NULL,
     x_axis_n_breaks = NULL, y_axis_n_breaks = NULL,
  x_axis_date_breaks = NULL, st_theme = NULL, show_gridlines = TRUE, show_axislines =
    TRUE, legend_title = "", legend_pos = "right", legend_font_size = 8,
  legend_title_font_size = 8, point_size_legend = FALSE, point_size_legend_title = "",
  hline = NULL, hline_colour = "black", hline_width = 0.5, hline_type = "dashed",
    hline_label = NULL, hline_label_colour = "black")
)
```

Arguments

dynamic	Logical indicating whether to produce a dynamic (plotly) output. Default is FALSE, which will return a static ggplot output.
base	A base ggplot or plotly object that the output will be applied to. If dynamic = TRUE then base must be a plotly object, and if dynamic = FALSE then base must be a ggplot object.
params	A named list containing arguements used to create the plot. '

     df A data frame containing values used to create the point chart.

     x Name of the variable in df used to populate the x-axis.

     y Name of the variable in df used to populate the y-axis.

point_shape Shape of the plotted points. Permitted values of `c('circle',` `'triangle','square','plus','square cross','asterisk','diamond')`. When `group_var` is provided, point shapes will be automatically assigned based on group.

point_size Size of the plotted point symbols. If supplied as a number, all points will be plotted with this size. If supplied as the name of a numeric variable within `df`, then the size of each point will be relative to the value of that numeric variable in the manner of a bubble chart.

point_colours Colour of the points to be plotted (default = "blue"). When `group_var` is provided, `point_colours` can be set as a character vector to define colours for each group.

point_labels Name of a variable in df containing text labels to plot against each point on the chart. If not provided the no labels will be applied. If `dynamic = TRUE` then `point_labels` will be applied as hover-labels, and `point_labels` will accept html to format the output labels.

point_labels_size Font size of `point_labels` on output chart when `dynamic = FALSE`.

point_labels_hjust Horizontal justification of `point_labels` on output chart when `dynamic = FALSE`. Permitted values = c(0, 0.5, 1) for left, centre, and right justified respectively.

point_labels_vjust Vertical justification of `point_labels` on output chart when `dynamic = FALSE`. Permitted values = c(0, 0.5, 1) for bottom, middle, and top justified respectively.

point_labels_nudge_x Horizontal adjustment to nudge `point_labels` by when `dynamic = FALSE`. Useful for offsetting text from points.

point_labels_nudge_y Vertical adjustment to nudge `point_labels` by when `dynamic = FALSE`. Useful for offsetting text from points.

group_var Name of the variable in df used to define separate groups of points in the chart.

ci Confidence interval. If `ci = "errorbar"` then confidence intervals be be plotted with each point as errorbars, and if `ci = "ribbon"` then confidence intervals will be added to the chart as a ribbon plot for each group. If `ci` is provided, then `ci_upper` and `ci_lower` must also be provided.

ci_upper Name of the variable in df used as the upper confidence limit for each point. Mandatory when `ci` is provided.

ci_lower Name of the variable in df used as the lower confidence limit for each point. Mandatory when `ci` is provided.

ci_legend Logical indicating whether a separate legend should be included in the chart for confidence interval parameters. Only applies when `group_var` is provided. Defaults to `FALSE`.

ci_legend_title Text to use as title for separate legend when `ci_legend = TRUE`.

ci_colours Colour(s) used for plotting confidence intervals. When `ci = "errorbar"` this will determine the colour of the plotted errorbars, when `ci = "ribbon"` this will determine the colour of the plotted ribbons.

errorbar_width Horizontal width of the plotted error bars when `ci = "errorbar"`.

y_sec_axis Logical to indicate whether data should be plotted on the secondary (right) y-axis. Default = `FALSE`.

y_sec_axis_no_shift Forces the secondary y-axis scale to begin at 0. Default = `TRUE`.

y_sec_axis_percent_full Forces the secondary y-axis scale to range from 0-100% when `y_percent = TRUE`

chart_title Text to use as chart title.

chart_title_size Font size of chart title.

chart_title_colour Font colour of chart title.

chart_footer Text to use as chart footer.

chart_footer_size Font size of chart footer.

chart_footer_colour Font colour of chart footer.

x_axis_title Text used for x-axis title. Defaults to name of x-variable if not stated.

y_axis_title Text used for y-axis title. Defaults to name of y-variable if not stated.

x_axis_title_font_size Font size of the x-axis title.

y_axis_title_font_size Font size of the y-axis title.

x_axis_label_angle Angle for x-axis label text.

y_axis_label_angle Angle for y-axis label text.

x_axis_label_font_size Font size for the x-axis tick labels.

y_axis_label_font_size Font size for the y-axis tick labels.

x_axis_reverse Reverses x-axis scale if `x_axis_reverse = TRUE`.

y_percent Converts y-axis to percentage scale if `y_percent = TRUE`.

x_limit_min Lower limit for the x-axis. Default used if not provided.

x_limit_max Upper limit for the x-axis. Default used if not provided.

y_limit_min Lower limit for the y-axis. Default used if not provided.

y_limit_max Upper limit for the y-axis. Default used if not provided.

x_axis_break_labels Vector of values to use for x-axis breaks. Defaults used if not provided.

y_axis_break_labels Vector of values to use for y-axis breaks. Defaults used if not provided.

x_axis_n_breaks Scales x-axis with approximately n breaks. Cannot be provided if `x_axis_break_labels` has also been provided.

y_axis_n_breaks Scales y-axis with approximately n breaks. Cannot be used if `y_axis_break_labels` has also been provided.

x_axis_date_breaks A string giving the distance between breaks like "2 weeks", or "10 years". Valid specifications are 'sec', 'min', 'hour', 'day', 'week', 'month' or 'year', optionally followed by 's'. Matches ggplot scale_date() conventions (see https://ggplot2.tidyverse.org/reference/scale_date.html). Cannot be used if `y_axis_break_labels` is also provided.

st_theme Name of a ggplot theme to be applied to a static plot. Can only be provided when `dynamic = FALSE`

show_gridlines Logical to show chart gridlines. Default = `TRUE`.

show_axislines Logical to show chart axis lines. Default = `TRUE`.

legend_title Text used for legend title.

legend_pos Position of the legend. Permitted values = c("top","bootom","right","left")

legend_font_size Font size used in the legend.

legend_title_font_size Font size used for the legend title.

point_size_legend Include a legend for point_size. Default = FALSE

point_size_legend_title Text used for point legend title.

hline Adds horizontal line across the chart at the corresponding y-value. Multiple values may be provided as a vector to add multiple horizontal lines.

hline_colour Colour of the horizontal lines if hline is provided. A vector of colours can be provided to colour individual hlines if multiple hlines have been provided.

hline_width Numerical width of the horizontal lines if hline is provided. A vector of numerical widths can be provided for individual hlines if multiple hlines have been provided.

hline_type Line style of the horizontal lines if hline is provided. A vector of line styles can be provided to style hlines if multiple hlines have been provided. Permitted values = c("solid", "dotted", "dashed", "longdash", "dotdash").

hline_label Text to label the horizontal lines if hline is provided. A vector of text strings can be provided to label individual hlines if multiple hlines have been provided.

hline_label_colour Colour of the horizontal line labels if hline_labels is provided. A vector of colours can be provided to colour individual hline_labels if multiple hline_labels have been provided.

Value

A ggplot or plotly object.

Examples

```
# Basic point chart example
library(epiviz)
library(dplyr)

detections_per_month <- epiviz::lab_data |>
  group_by(specimen_month = lubridate::floor_date(specimen_date, 'month')) |>
  summarise(detections = n()) |>
  ungroup()

# Create static point chart
my_chart <- point_chart(
  params = list(
    df = detections_per_month,
    x = "specimen_month",
    y = "detections",
    point_colours = "#007C91",
    point_size = 3,
    chart_title = "Detections per Month",
```

```
      x_axis_title = "Month",
      y_axis_title = "Number of detections"
   )
)
```

Index