Package 'chronochrt'

November 12, 2022

Title Creating Chronological Charts

Version 0.1.2

Date 2022-11-12

Description Easy way to draw chronological charts from tables, aiming to include an intuitive environment for anyone new to R. Includes 'ggplot2' geoms and theme for chronological charts.

Language en-GB

License GPL-3

URL https://gitlab.com/archaeothommy/chronochrt

BugReports https://gitlab.com/archaeothommy/chronochrt/-/issues

Imports dplyr (>= 1.0.0), ggplot2 (>= 3.4.0), grid, magick, magrittr, readr, tibble, tidyr, rlang, tidyselect

Suggests readxl, testthat (>= 3.0.0), knitr, rmarkdown, covr, vdiffr

Encoding UTF-8

RoxygenNote 7.2.2

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Thomas Rose [aut, cre] (<https://orcid.org/0000-0002-8186-3566>), Chiara Girotto [aut] (<https://orcid.org/0000-0001-6412-342X>)

Maintainer Thomas Rose <thomas.rose@daad-alumni.de>

Repository CRAN

Date/Publication 2022-11-12 17:20:02 UTC

R topics documented:

add_chron 2
add_label_image 4
add_label_text 6
arrange_regions 7

chronochrt 8
convert_to_chron 9
geom_chronochRt 10
geom_chronochRtImage 13
import_chron 15
plot_chronochrt 17
theme_chronochrt 19

Index **21**

`add_chron` *Create or add chronological data*

Description

This function either creates a data set with chronological data which can be directly used for plotting or it adds chronological data to such a data set.

Usage

```
add_chron(
  data,
  region,
  name,
  start,
  end,
  level = 1,
  add = FALSE,
  new_table = FALSE,
  ...
)
```

Arguments

`data`	A data set with chronological data. Must not be provided if `new_table = FALSE`.
`region`	A character string or character vector with the title(s) of the section(s).
`name`	A character string or character vector with the name(s) of the chronological unit(s).
`start`	A number or a vector with the start date(s) of the chronological unit(s). Use negative values for BCE dates. See Details how to handle insecure start dates.
`end`	A number or a vector with the end date(s) of the chronological unit(s). Use negative values for BCE dates. See Details how to handle insecure end dates.
`level`	A whole number or numeric vector of whole numbers (i.e. 1, 2, 3, ...) with the level(s) of the chronological unit(s). The default is 1, i.e. the top unit.
`add`	A logical value (`TRUE` or `FALSE`) or a logical vector signalling whether the chronological units within a geographical area should be drawn separately (`TRUE`) or not (`FLASE`, the default).

new_table	Logical operator. If TRUE, a new data set will be created. If FALSE, the default, the input will be added to an existing data set.
...	Further arguments or columns to include in or additional arguments passed to tibble or add_row.

Details

If the input is in the same order as the arguments, the arguments do not need to be explicitly named. Values can be provided as one number or one character string, if they are the same for all other data. If not, they must be provided as vectors with equal lengths.

start and end of neighbouring chronological units as well as respective oldest sub-units must be the same to achieve good plotting results. Dates in BCE must provided as negative data. Currently, only years can be handled (i.e. 2020 but not 20.10.2020).The package can handle the year 0.

If start and end dates are not certain or the change between chronological units is regarded a period, dates must be given as character string in the format "1000/2000". Consistency is required for matching start and end dates to avoid unclean or chaotic border orientation in the plot.

The level indicates the position of the chronological unit. level = 1 denotes a top chronological unit (e.g. Ha), a sub-unit (e.g. Ha B) is level = 2, a sub-sub-unit (e.g. Ha B1) level = 3 etc. The parameter add indicates whether the respective chronological unit(s) should be plotted in the same or an additional column. This might be useful if competing chronologies in one region exist (e.g. short and long chronologies). See the vignette for a detailed explanation how the parameters level and add work.

Additional columns might be useful to e.g. specify the x and y position of the names of the chronological units to place them at an arbitrary spot.

Value

A tibble with chronological data ready-to-use for plotting with plot_chronochrt.

Examples

```
# Create new data set

chrons <- add_chron(region = c("A", "B"),
                    name = c("a", "a"),
                    start = -100,
                    end = c(200, 150),
                    level = c(1, 1),
                    add = FALSE,
                    new_table = TRUE)

# Add chrons to an existing data set
chrons2 <- add_chron(data = chrons,
                     region = "A",
                     name = c("1", "2"),
                     start = c(-100, 100),
                     end = c(100, 200),
                     level = 2,
                     add = FALSE,
```

```
                     new_table = FALSE)

# Include chrons with unclear start/end data
chrons <- add_chron(data = chrons,
                    region = "B",
                    name = c("1", "2"),
                    start = c(-100, "0/50"),
                    end = c("0/50", 150),
                    level = 2,
                    add = FALSE,
                    new_table = FALSE)

# They can be linked using the pipe operator \code{%>%}:
library(magrittr)

chrons <- add_chron(region = c("A", "B"),
                    name = c("a", "a"),
                    start = -100,
                    end = c(200, 150),
                    level = c(1, 1),
                    add = FALSE,
                    new_table = TRUE) %>%
          add_chron(region = "B",
                    name = c("1", "2"),
                    start = c(-100, "0/50"),
                    end = c("0/50", 150),
                    level = 2,
                    add = FALSE)
```

`add_label_image` *Provide image labels for a chronological chart*

Description

The function creates a tibble with the paths of the image labels to be plotted in a chronological chart or adds them to an already existing tibble.

Usage

```
add_label_image(
  data,
  region,
  year,
  position = 0.75,
  image_path,
  new = FALSE,
  ...
)
```

Arguments

data	An object to which labels should be added. Must not be provided if new = FALSE.
region	A character string or character vector with the titles of the sections the label(s) should be placed in.
year	A number or a numeric vector with the year(s) at which the label should be placed (i.e. its vertical position).
position	A number or a numeric vector with the horizontal position(s) of the label. See Details for explanation.
image_path	A character string or character vector with the file path(s) or URL(s) to the image files.
new	Logical operator. If TRUE, a new data set will be created. If FALSE, the default, the input will be added to an existing data set.
...	Further columns to include or additional arguments passed to tibble or add_row.

Details

If the input is in the same order as the arguments, the arguments do not need to be explicitly named. Values can be provided as a number or character string, if they are the same for all other data. If not, they must be provided as vectors with equal lengths.

Value

A tibble with image labels ready-to-use for plotting with plot_chronochrt.

Examples

```
# Create new label data set
labels <- add_label_image(region = "A",
                          year = -50,
                          position = 0.5,
                          image_path = "https://www.r-project.org/logo/Rlogo.png",
                          new = TRUE)

# Add labels to existing data set
labels <- add_label_image(data = labels,
                          region = "B",
                          year = 50,
                          position = 0.9,
                          image_path = "https://www.r-project.org/logo/Rlogo.png",
                          new = FALSE)

# They can be linked using the pipe operator \code{%>%}:
library(magrittr)

labels <- add_label_image(region = "A",
                          year = -50,
                          position = 0.5,
                          image_path = "https://www.r-project.org/logo/Rlogo.png",
                          new = TRUE) %>%
```

```
add_label_image(region = "B",
                year = 50,
                position = 0.9,
                image_path = "https://www.r-project.org/logo/Rlogo.png")
```

`add_label_text` *Provide text labels for a chronological chart*

Description

The function creates a tibble with text labels to be plotted in a chronological chart or adds them to an an already existing tibble.

Usage

```
add_label_text(data, region, year, position = 0.9, label, new = FALSE, ...)
```

Arguments

`data`	An object to which labels should be added. Must not be provided if `new = FALSE`.
`region`	A character string or character vector with the the titles of the sections the label(s) should be placed in.
`year`	A number or a numeric vector with the year(s) at which the label should be placed (i.e. its vertical position).
`position`	A number or a numeric vector with the horizontal position(s) of the label. See Details for explanation.
`label`	A character string or character vector with the text of the label(s).
`new`	Logical operator. If `TRUE`, a new data set will be created. If `FALSE`, the default, the input will be added to an existing data set.
`...`	Further columns to include, or additional arguments passed to `tibble` or `add_row`.

Details

If the input is in the same order as the arguments, the arguments do not need to be explicitly named. Values can be provided as one number or one character string, if they are the same for all other data. If not, they must be provided as vectors with equal lengths. It is assumed that most of the labels will be located on the right side of each column. The `position` of a label defines it right most end to prevent it from running outside the plotting area. Vertically, it will be placed centred on the `year` given. Text in labels can be wrapped by inserting `\n` (without blanks around it).

Value

A tibble with text labels ready-to-use in `plot_chronochrt`.

Examples

```
# Create new label data set
labels <- add_label_text(region = "A",
                         year = -50,
                         position = 0.5,
                         label = "Flood",
                         new = TRUE)

# Add labels to existing data set
labels <- add_label_text(data = labels,
                         region = "B",
                         year = 50,
                         position = 0.9,
                         label = "Earthquake",
                         new = FALSE)

# They can be linked using the pipe operator \code{%>%}:
library(magrittr)

labels <- add_label_text(region = "A",
                         year = -50,
                         position = 0.5,
                         label = "Flood",
                         new = TRUE) %>%
          add_label_text(region = "B",
                         year = 50,
                         position = 0.9,
                         label = "Earthquake")
```

---

arrange_regions *Arranging the regions (sections) of a chronological chart*

Description

This function ensures that the regions/sections of a chronological chart and of the accompanying labels are arranged in the desired order, not necessarily in an alphabetical one (the default plotting order).

Usage

```
arrange_regions(data, order)
```

Arguments

data	A data set with a column named "region".
order	A character vector with the desired order of the region/section titles. Each title must be given only once.

Value

A tibble with data ready-to-use for plotting with `plot_chronochrt`.

Examples

```
# Create example data set

chrons <- add_chron(region = c("A", "B"),
                    name = c("a", "a"),
                    start = -100,
                    end = c(200, 150),
                    level = c(1, 1),
                    add = FALSE,
                    new_table = TRUE)

# Arrange regions

chrons <- arrange_regions(data = chrons, order = c("B", "A"))
```

`chronochrt` *ChronochRt*

Description

ChronochRt offers an easy way to draw chronological charts from tables. It aims to provide an intuitive environment for anyone new to R.

Features

- Slim structure of chronological datasets
- Import tabular data files and
- Import Excel files (requires the package **readxl**
- Possibility to display up to 2 chronological systems within the same region (e.g. long and short chronologies)
- Layout of the chronological chart optimised for easy readability and comprehensibility
- Years in BCE must be negative - that's all you need to care about for dates
- Handling of insecure dates
- Handling of gaps, e.g. abandonment phases of sites
- Optional text labels
- Optional image labels to e.g. display key finds or show typological developments
- Geoms for the chronological chart and image labels
- Export of the chronological chart in different file formats (raster and vector graphics)
- Easy customisation of the chronological chart
- Based on the tidyverse: Seamless integration in pipes, enhanced customisation with **ggplot2**

Getting started

- Cheatsheet
- Vignettes

`convert_to_chron` *Prepare an existing data set for plotting*

Description

Convert an existing data set in a ready-to-plot data set.

Usage

```
convert_to_chron(data, region, name, start, end, level, add)
```

Arguments

`data`	A data frame or tibble
`region`	The column name of the regions/sections in the plot.
`name`	The column name of the names of the chronological units. Must be a character string.
`start`	The column name of the start dates of the chronological units.
`end`	The column name of the end date of the chronological units.
`level`	The column name of the level of the chronological unit.
`add`	The column name of the columns which signals whether the chronological units within a geographical area should be drawn separately.

Details

Additional columns if the data set are directly passed to the output.

Value

A tibble with chronological data ready-to-use for plotting with `plot_chronochrt`.

Examples

```
# Create example data set
data <- data.frame(Location = c("A", "B"),
                   Dynasty = c("a", "a"),
                   Begin = -100,
                   End = c(200, 150),
                   Subunit = 1,
                   Parallel = FALSE)

# Convert to chronological data set
```

```
chrons <- convert_to_chron(data = data,
                           region = Location,
                           name = Dynasty,
                           start = Begin,
                           end = End,
                           level = Subunit,
                           add = Parallel)
```

`geom_chronochRt` *A chronological chart*

Description

Computes and draws a chronological chart.

Usage

```
geom_chronochRt(
  mapping = NULL,
  data = NULL,
  inherit.aes = TRUE,
  year_lim = NULL,
  minimal = FALSE,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`year_lim`	A numeric vector of length 2 to define the lower and upper limit of the chronological chart.

`minimal`	Should chrons be optically separated by vertical lines? If `TRUE` only vertical lines between around the chronological columns will be drawn.
`...`	Other arguments passed on to `layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `hjust = 0.1`.

Details

This geom is special because no x and y coordinates are provided in the input. Therefore, the following aesthetics must be provided only in the `aes` function: `region`, `level`, `start`, `end`, `add` (i.e. all required aesthetics).

- `region` The title(s) of the section(s) the chronological chart is subdivided into
- `name` The name(s) of the chronological unit(s). To maintain compatibility with other geoms, the aesthetic `label` can be used instead.
- `start, end` The start and end date of a chronological unit, respectively.
- `level` The levels of the chronological units.
- `add` Logical value indicating whether chronological units within a `region` should be plotted in an *add*itional column.

Usually, the names of the chronological units are placed in their middle. They can be arbitrarily placed by the aesthetics `name_x, name_y`:

- `name_x` The horizontal position within an chronological column, i.e. a value between 0 and 1 if `add = FALSE` and between 1 and 2 if `add = TRUE`.
- `name_y` The vertical position given as a year.

See `vignette("ChronochRt")` below for further details.

The geom aims to preserve access to as much of the underlying aesthetics as possible. To achieve this aim, ambiguous names were resolved (e.g. `size` to `size_line` and `size_text`).

Value

Layer of a ggplot2 object.

Aesthetics

`geom_ChronochRt()` understands the following aesthetics (required aesthetics are in bold):

- `region`
- `level`
- `end`
- `start`
- `add`
- `alpha`
- `angle`
- `colour`

- family
- fill
- fontface
- group
- hjust
- lineheight
- name|label
- name_x
- name_y
- size_line
- size_text
- vjust

See Details for how aesthetics specific for this geom work and learn more about setting aesthetics in vignette("ggplot2-specs").

Examples

```
# Create example data
library(ggplot2)

chrons <- data.frame(region = c("A", "B", "B", "B", "A"),
                     name = c("a", "a", "1", "2", "b"),
                     start = c(-100, -100, -100, "0/50", "50_100"),
                     end = c("50_100", 150, "0/50", 150, 200),
                     level = c(1, 1, 2, 2, 1),
                     add = FALSE)

ggplot(chrons) +
geom_chronochRt(aes(name = name, region = region, level = level,
                    start = start, end = end, add = add))

ggplot(chrons, aes(name = name, region = region, level = level,
                   start = start, end = end, add = add)) +
   geom_chronochRt()

# If more than one region should be plotted, they must be separated with facet_grid:
ggplot(chrons) +
   geom_chronochRt(aes(name = name, region = region, level = level,
                       start = start, end = end, add = add)) +
   facet_grid(cols = vars(region), scales = "free_x", space = "free_x")

# Adjust upper and lower end of a chronological chart with year_lim:
q <- ggplot(chrons, aes(name = name, region = region, level = level,
                        start = start, end = end, add = add)) +
   facet_grid(cols = vars(region), scales = "free_x", space = "free_x")
```

```
q + geom_chronochRt(year_lim = c(-50, 100))

# Change aesthetics of the plot:
q + geom_chronochRt(fill = "black", colour = "white")
q + geom_chronochRt(aes(fill = level, size_line = 3))

# Change position of the names:
q + geom_chronochRt(aes(name_x = 0.75))

# To remove vertical lines within a chronological column:
q + geom_chronochRt(minimal = TRUE)
```

`geom_chronochRtImage` *Add image labels to plot*

Description

Plot images in a ggplot2 object. Supported file types are: `.png`, `.jpg`, `.tif`, `.bmp.`, `.svg` (see `image_read` for further details).

Usage

```
geom_chronochRtImage(mapping = NULL, data = NULL, inherit.aes = TRUE, ...)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`...`	Other arguments passed on to `layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `height = 1`.

Details

The images are provided by their paths (local files or URLs) via the aesthetic `image_path`. Rows with invalid file paths are silently dropped, invalid URLs will throw an error.

The absolute size in cm of the images can be specified via the aesthetics `height` and `width`. If only one is specified, the image is scaled under preservation of its aspect ratio. If both are given, the image might appear distorted. See examples for further details.

Value

Layer of a ggplot2 object.

Aesthetics

`geom_ChronochRtImage()` understands the following aesthetics (required aesthetics are in bold):

- `image_path`
- `x`
- `y`
- `group`
- `height`
- `width`

See Details for how these aesthetics work.

Author(s)

This geom is a modified version of the `geom_custom()` from Baptiste Auguié's egg package.

Examples

```
library(ggplot2)

# Create example data
data <- data.frame(x = c(2, 4), y = c(2, 4),
                   image_path = "https://www.r-project.org/logo/Rlogo.png",
                   height = c(1,2), width = c(3,0.5))

q <- ggplot(data) + lims(x = c(0, 6), y = c(0, 6))

# Without size specifications
q + geom_chronochRtImage(aes(image_path = image_path, x = x, y = y))

# Scale images to individual heights/widths by specifying one of them:
q + geom_chronochRtImage(aes(image_path = image_path, x = x, y = y, height = height))

# Scale images to uniform height/width (i.e. independent of input data):
q + geom_chronochRtImage(aes(image_path = image_path, x = x, y = y, height = 1))
```

```
# Specifying height and width might result in distorted images:
q + geom_chronochRtImage(aes(image_path = image_path, x = x, y = y, height = height, width = width))
```

`import_chron` *Import data for Chronological chart*

Description

The function imports and converts chronological information from tabular data saved as `.csv`, `.xlsx` and `.xls`, and any other kind of delimited file format into a ready-to-use data set for plotting with ChronochRt. It automatically selects the appropriate import function from the file extension and the argument `delim`. To import excel files, the package **readxl** must be installed.

Usage

```
import_chron(
  path,
  region = "region",
  name = "name",
  start = "start",
  end = "end",
  level = "level",
  add = "add",
  delim,
  ...
)
```

Arguments

`path`	A character string with either the path or a URL to the file to be imported.
`region`	Character string (case sensitive) with the column name of the region/section, default to `"region"`.
`name`	Character string (case sensitive) with the column name of the chronological units' names, default to `"name"`.
`start`	Character string (case sensitive) with the column name of the chronological units' start dates, default to `"start"`.
`end`	Character string (case sensitive) with the column name of the chronological units' end dates, default to `"end"`.
`level`	Character string (case sensitive) with the column name of the chronological units' levels, default to `"level"`.
`add`	Character string (case sensitive) with the column name of the information whether the chronological units within a region/section should be drawn separately or not, default to `"add"`.
`delim`	A character string with the separator for tabular data. Use `delim = "\t"` for tab-separated data. Must be provided for all file types except `.xlsx` or `.xls`.

`...` Additional arguments passed to the respective import functions. See their documentation for details:

- `read_excel` for file formats `.xlsx` and `.xls`,
- `read_csv` for the file format `.csv`,
- `read_delim` for all other file formats.

Details

Additional columns in the import file will be imported as they are. Among these might be e.g. columns specifying the x and y position of the names to place them at an arbitrary spot.

Value

A tibble containing the desired chronological information.

Examples

```
## Not run:

# Import of Excel files
chrons <- import_chron(path = "ex_urnfield_periods.xlsx",
                       region = "Region",
                       name = "Name",
                       start = "Start",
                       end = "End",
                       level = "Level",
                       add = "Add")

# Import of delimited tabular data
chrons <- import_chron(path = "ex_urnfield_periods.csv",
                       region = "Region",
                       name = "Name",
                       start = "Start",
                       end = "End",
                       add = "Add",
                       delim = ",")

chrons <- import_chron(path = "ex_urnfield_periods.txt",
                       region = "Region",
                       name = "Name",
                       start = "Start",
                       end = "End",
                       add = "Add",
                       delim = "\t")

# Include additional parameters of the import function
chrons <- import_chron(path = "ex_urnfield_periods.xlsx",
                       region = "Region",
                       name = "Name",
                       start = "Start",
                       end = "End",
```

```
                        level = "Level",
                        add = "Add",
                        sheet = "data")

## End(Not run)
```

`plot_chronochrt` *Plot a chronological chart*

Description

This function converts a chronological data set into a chronological chart. It provides basic features for the export of the plot and for its customisation.

Usage

```
plot_chronochrt(
  data,
  labels_text = NULL,
  labels_image = NULL,
  year_lim = NULL,
  axis_title = "Years",
  minimal = FALSE,
  size_text = 6,
  height_image = 2,
  size_line = 0.5,
  fill_chron = "white",
  color_chron = "black",
  color_label = "black",
  line_break = 10,
  filename = NULL,
  plot_dim,
  background = NULL,
  ...
)
```

Arguments

`data`	A data set with chronological data.
`labels_text`	A data set containing the text labels.
`labels_image`	A data set containing the image labels.
`year_lim`	A numeric vector of length 2 to define the lower and upper limit of the chronological chart.
`axis_title`	A character string with the axis label of the vertical axis. Default is `"Years"`.
`minimal`	Should chrons be optically separated by vertical lines? If `TRUE` only vertical lines between around the chronological columns will be drawn.

size_text	Font size of the names of the chronological units in mm. All other text elements will be scaled accordingly. The default is 6 mm.
height_image	The absolute height of the image labels in cm. The default is 2 cm.
size_line	Thickness of the line in mm. The default is 0.5 mm.
fill_chron	Fill colour of the chronological units. The default is "white". See the colour specification section of par for how to specify colours in R.
color_chron	Line (border) colour of the chronological units. The default is "black". See the colour specification section of par for how to specify colours in R.
color_label	Colour of the text labels. The default is "black". See the colour specification section of par for how to specify colours in R.
line_break	Line length of the section labels in characters. Text will be wrapped at the blank closest to the specified number of characters. Default is 10 characters.
filename	A character string with the filename or path. If specified, the plot will be saved in the given location. The file format is automatically recognised from the file extension. The most common file types are supported, e.g. .tiff, .png, .jpg, .eps, and .pdf. To export as .svg installation of the package **svglite** is required. See ggsave for more details about the supported file formats.
plot_dim	Dimensions of the plot as a vector in the format c(width, height, units). Supported units are "cm", "mm", in". For example, plot_dim = c(5,5,"cm") will produce a plot of the dimensions 5 x 5 cm. If unspecified, the standard values of the respective graphic device are used.
background	Optional specifications for the background of the chronological chart as vector in the format c(background colour,linetype of grid lines) to overwrite the default behaviour of theme_chronochrt. Any valid colour and line type specifications are accepted, e.g. c("grey90", "dotted") (these are the default values of theme_chronochrt. See the sections "colour specification" and "line type specification" in par for how to specify colours and line types in R.
...	Additional arguments passed to ggsave to enhance the saved plot like dpi to specify its resolution. See ggsave for detailed information.

Details

This function is wrapper around various functions for an quick and convenient way to draw chronological charts. It relies on the common data structure of ChronochRt (see vignette("ChronochRt") for details). For full customisation use the respective geoms to build your own plot.

It is assumed that the majority of the text labels will be placed on the right side of each column. Therefore they are right aligned to prevent them from running outside the plotting area. Vertically, it will be placed centred on the year given. Text in labels can be wrapped by inserting "\n" (without blanks).

Value

A ggplot2 object with the chronological chart.

Examples

```
# Create Example data
chrons <- data.frame(region = c("A", "B", "B", "B", "A"),
                     name = c("a", "a", "1", "2", "b"),
                     start = c(-100, -100, -100, "0/50", "50_100"),
                     end = c("50_100", 150, "0/50", 150, 200),
                     level = c(1, 1, 2, 2, 1),
                     add = FALSE)

# Plot with default options
plot_chronochrt(chrons)

# Add labels
labels <- data.frame(region = "A",
                     year = -50,
                     position = 0.5,
                     label = "Event")

images <- data.frame(region = "B",
                     year = 100,
                     position = 0.5,
                     image_path = "https://www.r-project.org/logo/Rlogo.png")

plot_chronochrt(chrons, labels, images)

# Customise plot
plot_chronochrt(chrons, axis_title = "BC/AD", year_lim = c(-50,100),
                fill_chron = "black", color_chron = "white", size_line = 5)
plot_chronochrt(chrons, labels, images, color_label = "red", size_text = 5, height_image = 4)

# Export plot

file <- tempfile(fileext = ".jpg")

plot_chronochrt(chrons, filename = tempfile(fileext = ".jpg"),
  plot_dim = c(10, 15, "cm"))

  # with additional parameters
  plot_chronochrt(chrons, filename = tempfile(fileext = ".jpg"),
    plot_dim = c(10, 15, "cm"), dpi = 300)

unlink(file)

# Additional customisation with ggplot2
plot_chronochrt(chrons) + ggplot2::theme_bw()
```

---

theme_chronochrt *The ChronochRt ggplot theme*

Description

This is a theme to provide a ready-to-use layout of chronological charts. See theme for how to modify selected elements of it.

Usage

```
theme_chronochrt(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
base_line_size	base size for line elements
base_rect_size	base size for rect elements

Value

A ggplot-theme

Index

add_chron, 2
add_label_image, 4
add_label_text, 6
add_row, *3*, *5, 6*
aes, *11*
aes(), *10*, *13*
arrange_regions, 7

borders(), *10*, *13*

chronochrt, 8
convert_to_chron, 9

fortify(), *10*, *13*

geom_chronochRt, 10
geom_chronochRtImage, 13
ggplot(), *10*, *13*
ggsave, *18*

image_read, *13*
import_chron, 15

layer, *11*, *13*

par, *18*
plot_chronochrt, *3*, *5, 6*, *8, 9*, 17

read_csv, *16*
read_delim, *16*
read_excel, *16*

theme, *20*
theme_chronochrt, *18*, 19
tibble, *3*, *5, 6*