Package: ggtilecal (via r-universe)

September 8, 2024

Title Create static (and interactive) overview calendars using ggplot2

Version 0.1.0

Description Creates tile-based calendar layouts using ggplot2 facets and tiles. Calculates calendar layout positioning variables, and generates customisable ggplot2 calendars for specified time periods and calendar units.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports anytime, cli, dplyr, ggplot2 (>= 3.5.0), lifecycle, lubridate, rlang, tibble, tidyselect

Suggests emoji, lorem, ggiraph

URL `https://cynthiahqy.github.io/ggtilecal/`, `https://github.com/cynthiahqy/ggtilecal`

BugReports `https://github.com/cynthiahqy/ggtilecal`

Depends R (>= 2.10)

LazyData true

Repository https://cynthiahqy.r-universe.dev

RemoteUrl https://github.com/cynthiahqy/ggtilecal

RemoteRef HEAD

RemoteSha 5200bc3cdc3ed9053e853247581d80c63305ca1a

Contents

`calc_calendar_vars` *Calculate variables for calendar layout*

Description

Helper function for calculating calendar layout variables.

Usage

```
calc_calendar_vars(
  .data,
  date_col,
  locale = Sys.getlocale("LC_TIME"),
  week_start = 1
)
```

Arguments

`.data`	tibble or data frame containing dates to be plotted in calendar layout
`date_col`	column containing calendar unit dates
`locale`	locale to use for day names. Default to current locale.
`week_start`	day on which week starts following ISO conventions: 1 means Monday and 7 means Sunday (default). When `label = FALSE` and `week_start = 7`, the number returned for Sunday is 1, for Monday is 2, etc. When `label = TRUE`, the returned value is a factor with the first level being the week start (e.g. Sunday if `week_start = 7`). You can set `lubridate.week.start` option to control this parameter globally.

Value

Tibble with additional calendar layout variables: `TC_year`, `TC_month_label`, `TC_mday`, `TC_wday_label`, `TC_wday`, `TC_month_week`, `TC_is_weekend`

Examples

```
make_empty_month_days(c("2024-01-01", "2024-02-01")) |>
  calc_calendar_vars(unit_date)
```

`demo_events_gpt` *5 Sample Events generated by ChatGPT 3.5*

Description

A set of 5 demo events generated by ChatGPT 3.5 on 13 Apr, 2024. The following field descriptions were also generated in the same session.

Usage

```
demo_events_gpt
```

Format

A data frame with 5 (event) rows and 7 columns:

event_id Unique identifier for the event.

startDate Start date of the event.

endDate End date of the event.

duration Duration of the event.

event_title Title or name of the event.

event_descr Description of the event.

event_emoji Emoji representing the event theme or type.

event_link URL link to the event's website or page.

Source

https://chat.openai.com/share/c68b7a82-5378-45c8-bf41-7fa134f0b74a

`demo_events_overlap` *10 Overlapping Events*

Description

A set of 10 demo events where Events 6,7,8 and 9 overlap, with placeholder details generated using the zoo and ipsum.

Usage

```
demo_events_overlap
```

Format

A data frame with 10 (event) rows and 6 columns:

event_id Unique identifier for the event.

title Plaeholder title for the event

start Start date of the event.

end End date of the event.

duration Duration of the event in days.

emoji Emoji representing the event theme or type.

details Loreum Ipsum placeholder details of the event.

Source

Package Authors

`fill_missing_units` *Fill out long format event table with missing dates*

Description

Helper function for filling out event table with any missing calendar units.

Usage

```
fill_missing_units(
  .events_long,
  date_col,
  adjust_months = NULL,
  cal_unit = "day"
)
```

Arguments

`.events_long`	long format calendar event data
`date_col`	column containing calendar unit dates
`adjust_months`	how many months to add before and after
`cal_unit`	increment of calendar sequence passed to by argument in `seq.Date`

Value

tibble

Examples

```
demo_events_gpt |>
  reframe_events(startDate, endDate) |>
  fill_missing_units(unit_date)
```

`gg_facet_wrap_months` *Make Monthly Calendar Facets*

Description

Generates calendar with monthly facets by:

- Padding event list with any missing days via `fill_missing_units()`
- Calculating variables for calendar layout via `calc_calendar_vars()`
- Returning a ggplot object as per Details.

Usage

```
gg_facet_wrap_months(
  .events_long,
  date_col,
  locale = NULL,
  week_start = NULL,
  nrow = NULL,
  ncol = NULL,
 .geom = list(geom_tile(color = "grey70", fill = "transparent"), geom_text(nudge_y =
    0.25)),
  .scale_coord = list(scale_y_reverse(), scale_x_discrete(position = "top"),
    coord_fixed(expand = TRUE)),
  .theme = list(theme_bw_tilecal()),
  .other = list()
)
```

Arguments

`.events_long` long format calendar event data

`date_col` column containing calendar unit dates

`locale` locale to use for day names. Default to current locale.

`week_start` day on which week starts following ISO conventions: 1 means Monday and 7 means Sunday (default). When `label = FALSE` and `week_start = 7`, the number returned for Sunday is 1, for Monday is 2, etc. When `label = TRUE`, the returned value is a factor with the first level being the week start (e.g. Sunday if `week_start = 7`). You can set `lubridate.week.start` option to control this parameter globally.

`nrow, ncol` Number of rows and columns.

`.geom, .scale_coord, .theme, .other`
Customisable lists of ggplot2 components to add to the plot. An empty `list()` leaves the plot unmodified.

Details

Returns a ggplot with the following fixed components using calculated layout variables:

- `aes()` mapping:
  - `x` is day of week,
  - `y` is week in month,
  - `label` is day of month
- `facet_wrap()` by month
- `labs()` to remove axis labels for calculated layout variables

and default customisable components:

- `geom_tile()`, `geom_text()` to label each day which inherit calculated variables
- `scale_y_reverse()` to order day in month correctly
- `scale_x_discrete()` to position weekday labels
- `coord_fixed()` to square each tile
- `theme_bw_tilecal()` to apply sensible theme defaults

To modify components alter the `.geom` and `.scale_coord`, which inherit the calculate layout mapping by default (via the ggplot2 `inherit.aes` argument).

To additional components use the ggplot + function as normal, or pass components to the `.other` argument. This can be used to add interactive geoms (e.g. from `ggiraph`)

To modify the theme, use the ggplot + function as normal, or add additional elements to the list in `.theme`.

To remove any of the optional components, set the argument to any empty `list()`

Value

ggplot

Examples

```
library(dplyr)
library(ggplot2)

demo_events_gpt |>
  reframe_events(startDate, endDate) |>
  group_by(unit_date) |>
  slice_min(order_by = duration) |>
  gg_facet_wrap_months(unit_date) +
  geom_text(aes(label = event_emoji), nudge_y = -0.25, na.rm = TRUE)
```

`make_empty_month_days` *Make padding day unit entries for specified months*

Description

Creates empty day units using `make_empty_units()` that span the months encompassing `cal_range` with the option to adjust the number of months either side of `cal_range`

Usage

```
make_empty_month_days(
  month_range,
  dates_to = "unit_date",
  adjust_months = c(-0, 0)
)
```

Arguments

`month_range`	vector of dates.
`dates_to`	string name for column to output calendar unit dates
`adjust_months`	how many months to add before and after

Value

tibble with one row per day in specified months

Examples

```
make_empty_month_days(c("2024-03-05", "2024-04-15"))
```

`make_empty_units` *Make empty sequence of calendar units between specified date range*

Description

Make empty sequence of calendar units between specified date range

Usage

```
make_empty_units(cal_range, dates_to = "unit_date", cal_unit = "day")
```

Arguments

`cal_range`	`c(start,end)` range for which to create sequence of calendar units
`dates_to`	string name for column to output calendar unit dates
`cal_unit`	increment of calendar sequence passed to by argument in `seq.Date`

Value

tibble

Examples

```
make_empty_units(c("2024-03-05", "2024-04-15"))
```

`reframe_events` *Reframe event intervals to calendar units*

Description

[Experimental]

Usage

```
reframe_events(
  .data,
  event_start,
  event_end,
  id_cols = NULL,
  dates_to = "unit_date",
  cal_unit = "day"
)
```

Arguments

`.data`	A data frame or tibble containing event details
`event_start`	column containing event start date
`event_end`	column containing event end date
`id_cols`	(optional) set of columns that uniquely identify each observation. Ignored if `.data` is grouped (i.e. using group_by). If `.data` is not grouped and `id_cols` is `NULL`, all columns except for `event_start` and `event_end` are used.
`dates_to`	string name for column to output calendar unit dates
`cal_unit`	increment of calendar sequence passed to by argument in `seq.Date`

Details

Convert rows of events to sequence of calendar units for plotting. Duplicates all columns except for `event_start` and `event_date`. Requires a unique id key for each event.

This is a thin wrapper around `dplyr::reframe()` that expands each uniquely identified event in `.data` to a sequence of datetimes defined by the `start`, end and the interval `cal_unit`.

Value

See return value of reframe

Examples

```
event <- data.frame(
  id = 1,
  start = as.Date("2024-01-05"),
  end = as.Date("2024-01-10")
)
reframe_events(event, start, end)
demo_events_gpt |>
  reframe_events(startDate, endDate)
```

`theme_bw_tilecal` *A modified version of* `ggplot2::theme_bw()` *for calendar layouts*

Description

A modified version of `ggplot2::theme_bw()` for calendar layouts

Usage

```
theme_bw_tilecal(
  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

Index