Reusing ‘ggplot2’ code: how to design better plot helper functions

useR! 2026, Warsaw, Poland, 6-9 Jul 2026

Cynthia A. Huang

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

About Me!

- 👩 Post-Doctoral Researcher with *Prof. Frauke Kreuter*, Social Data and AI Lab, LMU Munich
- 🎨 Organiser of 'Grand Unified Grammar of Graphics' workshop at IEEE VIS 2026 🔗 gugog-vis.github.io/2026
- 📊 Research Interests
  - 🌰 "alternative" data for empirical social science research
  - 🤖 Leveraging LLMs and genAI in data science
  - 🛠️ Grammar of Graphics visualisation systems

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Ways to reuse ggplot code

- Copy & paste
- Component objects or lists
- Complete plot helpers
- Decorator and combination functions
- Extending ggproto objects

Copy & Paste

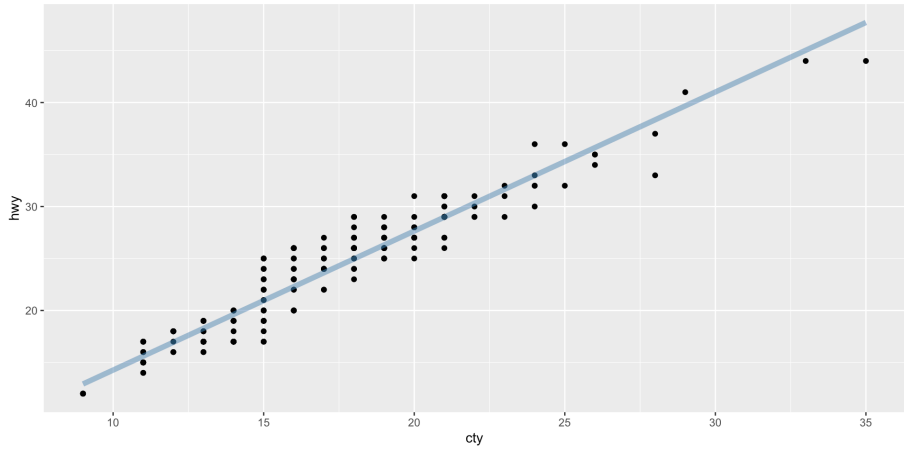
```
ggplot(
  data = penguins,
  mapping = aes(
    x = bill_length_mm,
    y = bill_depth_mm,
    color = species)) +
  geom_point()

## -- SPOT THE DIFFERENCE --

ggplot(
  data = penguins,
  mapping = aes(
    x = bill_length_mm,
    y = flipper_length_mm,
    color = species)) +
  geom_point()
```

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Component Objects

💡 Useful if components are exactly the same each time!

```
bestfit <- list(
  geom_smooth(
    method = "lm",
    se = FALSE,
    colour = alpha("steelblue", 0.5
    linewidth = 2
  ))
ggplot(mpg, aes(cty, hwy)) +
  geom_point() +
  bestfit
```



Example from *ggplot2: Elegant Graphics for Data Analysis (3e)*

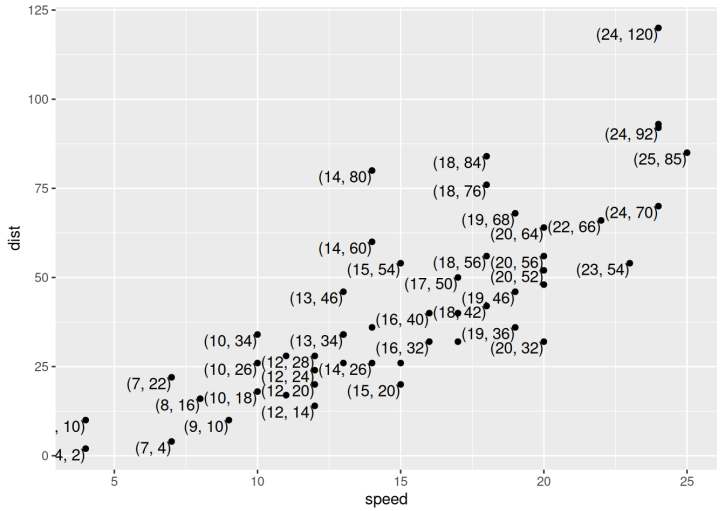
🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Decorators and combinations

- Combine multiple components in a list
- Specify new parameter options

```
1 geom_mean <- function(mean.fill = "grey70") {
2   list(
3     stat_summary(fun = "mean", geom = "bar", fill = mean.fill),
4     stat_summary(fun.data = "mean_cl_normal", geom = "errorbar", width = 0.
5   )
6 }
7 ggplot(mpg, aes(class, cty)) + geom_mean()
```

New components via ggproto



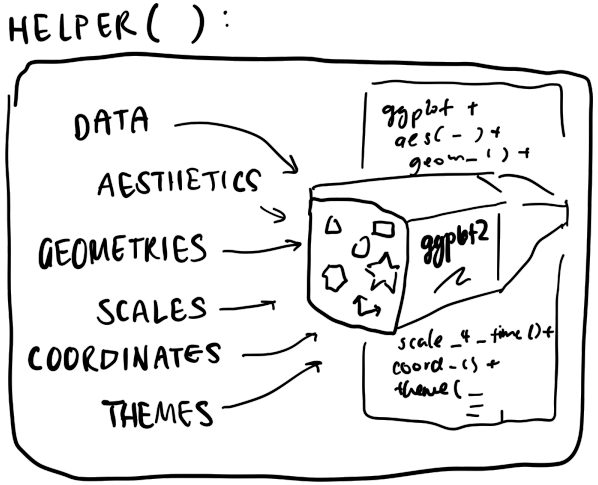

```
compute_group_coordinates  <- function(data
  data |>
    mutate(label = paste0("(", x ", ", y, "
}
StatCoordinates =
ggproto(`_class` = "StatCoordinates",
                    `_inherit` = Stat,
                    required_aes = c("x", "
                    compute_group = compute
...
```

Example from: Easy geom_*() recipes, Gina Reynolds

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Complete Plot Helpers

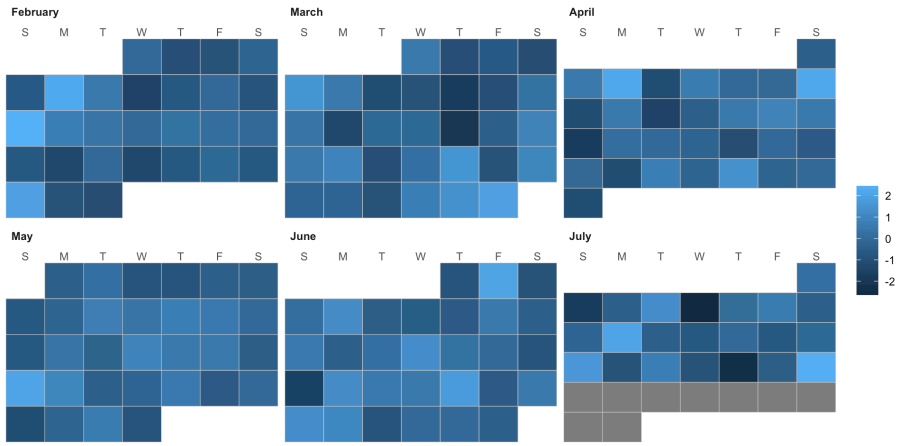
- may take as input:
  - tidy or not-tidy data inputs, and/or
  - multiple parameters arguments
- usually, transform or create data for plotting,
- return a complete *ggplot2* plot (not just components) as output



🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Calendars: two data inputs

```
1  library(ggcal)
2
3  mydate <-
4    seq(
5      as.Date("2017-02-01"),
6      as.Date("2017-07-22"),
7      by="1 day")
8  myfills <-
9    rnorm(length(mydate))
10
11 print(
12   ggcal(mydate, myfills)
13   )
```

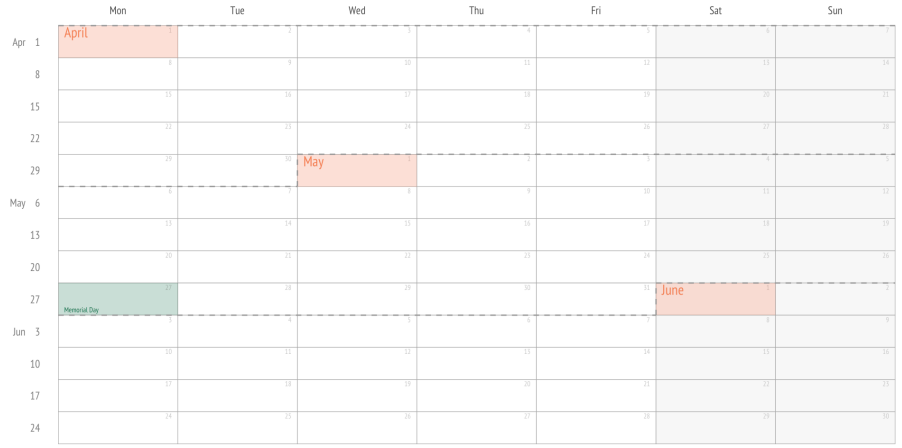


- how many arguments does `ggcal()` have?
- what format should `dates`, `fills` be?

Example from `github.com/jayjacobs/ggcal`

Calendars: plot specific args

```
library(ggweekly)
ggweek_planner(
  start_day = "2019-04-01",
  end_day = "2019-06-30",
)
```



What about customisation? e.g. text, colors, font, extra data

```
ggweekly::ggweek_planner(
    start_day = lubridate::today(),
    end_day = start_day +
        lubridate::weeks(8) - lubridate::days(1
    highlight_days = NULL,
    week_start = c("isoweek", "epiweek"),
    week_start_label = c("month day", "week", "
    show_day_numbers = TRUE,
    show_month_start_day = TRUE,
    show_month_boundaries = TRUE,
    highlight_text_size = 2,
    month_text_size = 4,
    day_number_text_size = 2,
    month_color = "#f78154",
    day_number_color = "grey80",
    weekend_fill = "#f8f8f8",
    holidays = ggweekly::us_federal_holidays,
    font_base_family = "PT Sans",
    font_label_family = "PT Sans Narrow",
    font_label_text = NULL
)
```

Example from `github.com/gadenbuie/ggweekly`

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

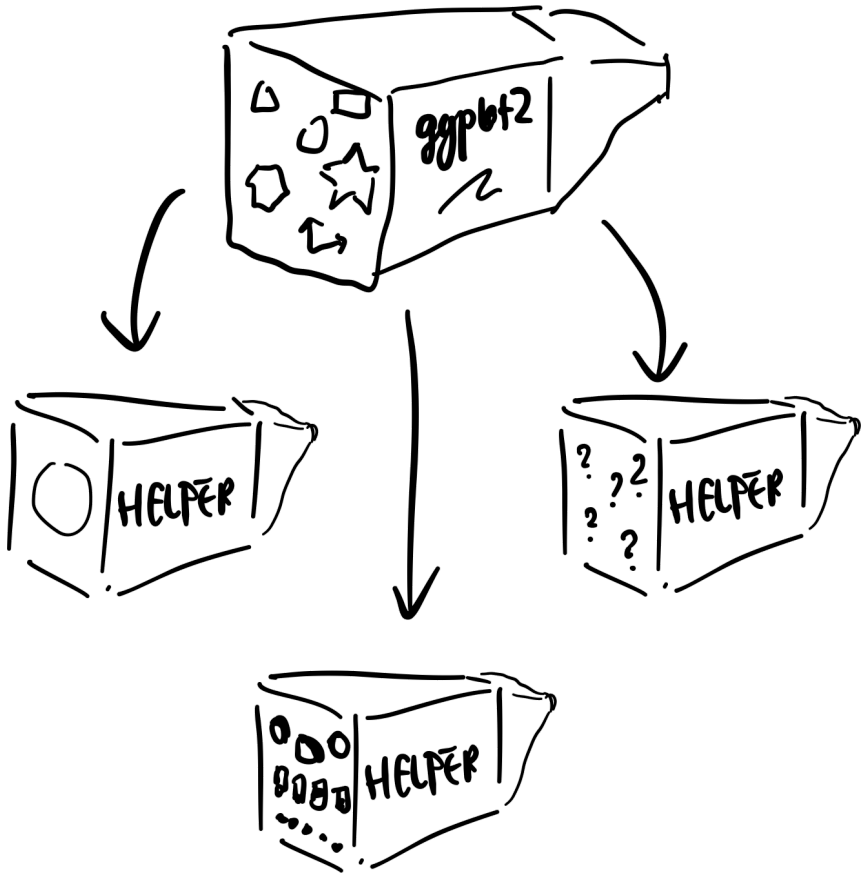
Calendars: ggplot sub-arguments

```r
calendR(
    year = format(Sys.Date(), "%Y")
    month = NULL,
    from = NULL,
    to = NULL,
    start = c("S", "M"),
    orientation = c("portrait", "la
    title,
    title.size = 20,
    title.col = "gray30",
    subtitle = "",
    subtitle.size = 10,
    subtitle.col = "gray30",
    text = NULL,
    text.pos = NULL,
    text.size = 4,
    text.col = "gray30",
    special.days = NULL,
```

- Is this using ggplot2?
- What are the inputs?
- Internal data preparation?
- …?

Example from `github.com/R-CoderDotCom/calendR`

Beware false savings!



Sometimes

“saved” lines of code

turn into

confusing lists of plot specific arguments

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Better complete plot helpers

Design goals

- Quick ‘autoplotting’
- Minimise plot-specific arguments
- Retain ggplot syntax & error messages
- Integrate well with other ggplot2 components

Helpers with Windows

1. **Expose** ggplot syntax & recipe
2. **Separate** data preparation from plotting
3. **Document** grammatical customisation options

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Case study: Another Calendar!

very-long-ggplot-script.R

```
# source(here::here("_utilities/travel_dates_load.R"))
library(ggplot2)
library(ggiraph)

## ---- daily-loc-prep ----

## expand dates: https://stackoverflow.com/a/54728153

travel_days <- travel_dates |>
  mutate(nights = interval(startDate, endDate) / days(1)) |>
  arrange(startDate, desc(nights)) |>
  rowid_to_column() |>
  rowwise() |>
  reframe(rowid, location, details, nights,
    date = seq(startDate, endDate, by = "day")
  ) |>
  group_by(date) |>
  slice_min(order_by = nights, n = 1) |>
  ungroup() |>
  mutate(country = countrycode::countryname(location, "iso2c"))

away_days <- travel_days |>
  summarise(
    startDate = min(date),
    endDate = max(date)
  ) |>
  mutate(rowid = 0, location = "TBC") |>
  reframe(rowid, location, date = seq(startDate, endDate, by = "day"))

daily_loc <- away_days |>
  anti_join(travel_days, by = "date") |>
  bind_rows(travel_days) |>
  mutate(
    flag = countrycode::countrycode(country, "iso2c", "unicode.symbol"),
    continent = countrycode::countrycode(country, "iso2c", "continent")
  ) |>
```

- data wrangling:
  - reshaping via `reframe()`
  - creating new variables
  - fill in missing days
  - calculating facet and layout (e.g. `Month`, `Month_week`)
- `ggplot2` components:
  - `scale`, `coord` and `facet` to layout the calendar
  - `theme` modification to style the plot as a calendar
  - interactive `geom` from `{ggiraph}`

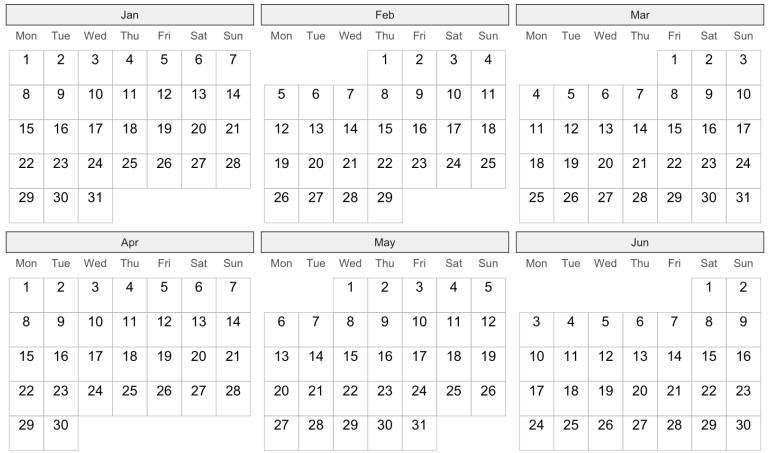
🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Convenient autoplot

```r
library(ggtilecal)
dates <-
ggtilecal::make_empty_month_days(
  c("2024-01-05", "2024-06-30"))
head(dates, 2)
```

```
# A tibble: 2 × 1
  unit_date
  <date>
1 2024-01-01
2 2024-01-02
```

```r
ggtilecal::gg_facet_wrap_months(
  .events_long = dates,
  date_col = unit_date)
```



🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

List arguments as ‘window’ interface

function-interface

```
gg_facet_wrap_months(
  ## --- data ----
  .events_long,
  date_col,
  locale = NULL,
  ## --- facets / layout ----
  week_start = NULL,
  nrow = NULL,
  ncol = NULL,
  ## --- default components ---
  .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)),
```

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Modular internals

internals

```
gg_facet_wrap_months <- function(...){
  ## Data Helpers
  cal_data <- .events_long |>
    fill_missing_units({{ date_col }}) |>
    calc_calendar_vars({{ date_col }})
  ## Plot Construction
  cal_data |>
    ggplot2::ggplot(mapping = aes_string(
      x = "TC_wday_label",
      y = "TC_month_week",
      label = "TC_mday" )) +
    facet_wrap(c("TC_month_label"), axes = "all_x", nrow = nrow, ncol = nco
    labs(y = NULL, x = NULL) +
    .geom +
    .scale_coord +
    .theme +
    .others
}
```

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

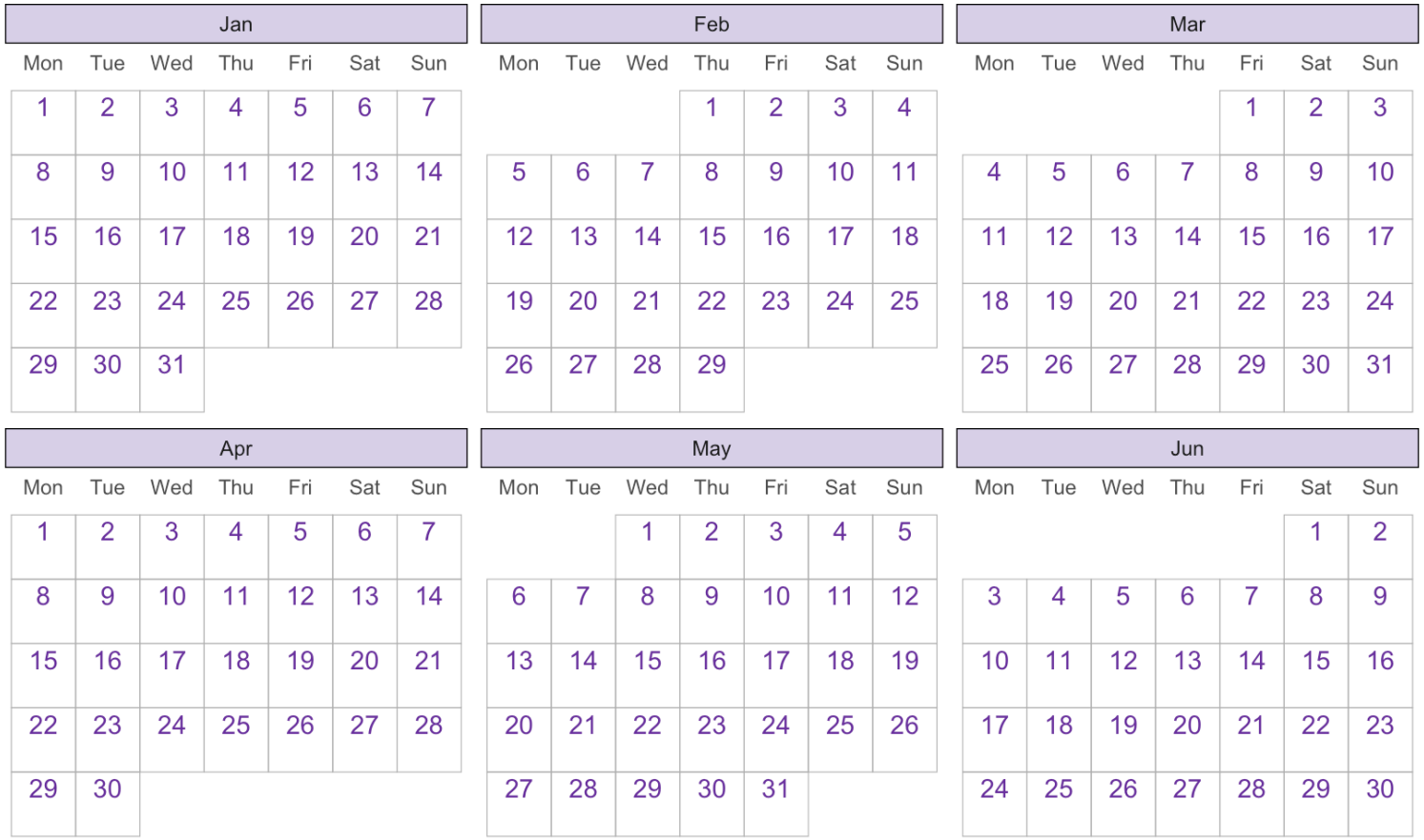
Grammatical customisation

Reuses existing ggplot2 syntax knowledge for customisations instead of plot-specific arguments!

```
gg_facet_wrap_months(dates, unit_date,
    .geom = list(
      geom_tile(color = "grey70", fill = "transparent"),
      geom_text(nudge_y = 0.25,
                color = "#6a329f")),
    .theme = list(theme_bw_tilecal(),
      theme(
        strip.background = element_rect(fill = "#d9d2e9")))
    )
```

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Grammatical customisation



Jan						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Feb						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

Mar						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Apr						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

May						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Jun						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Documentation support

ggtilecal 0.1.0

Make Monthly Calendar Facets

Source: R/gg_facet_wrap_months.R

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
```

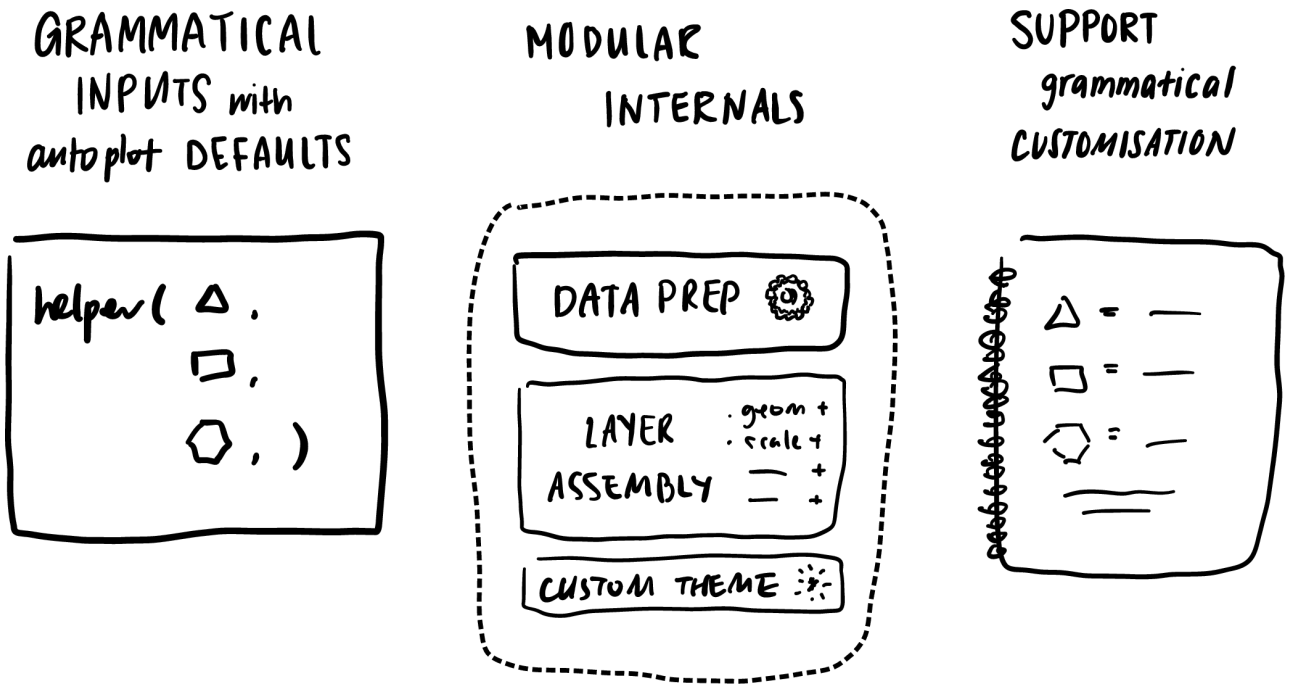
Function docs explains:

1. Data preparation steps
2. How the ggplot2 recipe works
3. How to modify the plot

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Window plot helpers

1. **Expose** ggplot syntax & recipe
2. **Separate** functionality
3. **Document** grammatical customisation



🔗 Read more at cynthiahqy.com

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers

Share your ggplot2 helper habits!

- 5-10 minute survey
- Share what strategies you use for re-using ggplot2 code
- Contribute to research on teaching and developing ggplot2 extensions
- Optional EOI for follow-up interview

🔗 cynthiahqy.com/survey

🔗 cynthiahqy.quarto.pub/user26-ggplot2-helpers