Donut plots

Overview

This script creates donut plots that show survey participation rates across various groups (demographics). If you want to look at just demographics for experienced OS contributors, you can change the `contributor_mode` variable to `TRUE`, around line 182(ish).

Import packages and utilities

```
project_root <- here::here() # requires that you be somewhere in the
# project directory (not above it)
# packages
suppressMessages(source(file.path(project_root, "scripts/packages.R")))
# functions and objects used across scripts
suppressMessages(source(file.path(project_root, "scripts/utils.R")))
```

Define functions

create_df_for_plotting

- Arguments:
  - `data`: The data frame with the survey data. This function will count rows to determine the count for each category, so don't tally up the group counts in advance. It will also remove rows where this column is an empty string, so no need to clean the df at all.
  - `column`: The name of the column with the data to be plotted.
- Details:

  - This function creates a new data frame suitable for plotting as a donut plot. It essentially creates a single stacked bar with all our data, and then plots that on a polar coordinate system to make it a donut. Method from https://r-graph-gallery.com/128-ring-or-donut-plot.html. This function extracts the data of interest from the larger data frame, and puts them into a new data frame along with relevant breakpoints for and label locations.

- Outputs:
  - `long_data`: a new data frame with columns values, Freq, fraction, ymax, ymin, labelPosition, and label.

```r
create_df_for_plotting <- function(data, column) {
  if (!column %in% names(data)) {
    stop("Column not found in data frame")
  }
  # Extract specified column and remove rows where this column
  # is an empty string, indicating the participant didn't answer
  values <- data[[column]][nzchar(data[[column]])]
  # Count occurrences of each unique value
  values_table <- table(values)

  # Convert to data frame and compute fractions
  long_data <- as.data.frame(values_table) %>%
    mutate(fraction = Freq / sum(Freq)) %>%
    arrange(desc(fraction))

  # Compute the cumulative percentages (top of each rectangle)
  long_data$ymax <- cumsum(long_data$fraction)
  long_data$ymin <- c(0, head(long_data$ymax, n = -1))

  # Compute label position
  long_data$labelPosition <- (long_data$ymax + long_data$ymin) / 2

  # Create label column
  long_data$label <- paste0(long_data$Freq)

  # Make the text wrap so the legend is more compact
  long_data <- long_data %>%
    mutate(values = str_wrap(values, width = 20))

  # from scripts/utils.R
  long_data <- reorder_factor_by_column(
```

```
    long_data,
    values,
    Freq
  )

  return(long_data)
}
```

donut_chart

- Arguments:
  - `df`: The data frame ready for plotting, such as the one produced by create_df_for_plotting.
- Details:
  - This function creates a donut plot.
- Outputs:
  - A ggplot object.

```
donut_chart <- function(
  df,
  cpalette = COLORS, # default: colors from utils.R
  legendpos = "right",
  title_bottom_margin = 15,
  legend_top_margin = 0,
  label_dist = 4.3, # controls how close the labels are to the ring
  white_labels = FALSE
) {

    if (white_labels) {
    label_x     <- 3.5
    label_size  <- 8
    label_color <- "white"
  } else {
    label_x     <- label_dist
    label_size  <- 9
    label_color <- "black"
  }

  ggplot(
```

```
    df,
    aes(
      ymax = ymax,
      ymin = ymin,
      xmax = 4,
      xmin = 3,
      fill = values
    )
  ) +
    geom_rect() +
    # Add labels
    geom_text(
      x = label_x,
      aes(y = labelPosition, label = label),
      size = label_size,
      color = label_color
    ) +
    scale_fill_manual(values = cpalette) +
    theme_void() +
    coord_polar(theta = "y", clip = "off") +
    xlim(c(2, label_x + 0.05)) +
    theme(
      legend.text = element_text(size = 18),
      legend.title = element_blank(),
      legend.position = legendpos,
      legend.key.spacing.y = unit(0.5, "lines"),
      legend.margin = margin(t = unit(legend_top_margin, "lines")),
      plot.title = element_text(
        hjust = 0.5,
        size = 28,
        margin = margin(b = title_bottom_margin)
      ),
      plot.margin = margin(t = 20, r = 5, b = 10, l = 5),
      plot.background = element_rect(fill = "white", color = "white")
    )
}
```

```
filter_contribs <- function(df, cmode) {
  if (cmode) {
    new_df <- df %>% filter(favorite_solution != "")
    return(new_df)
  } else {
```

```
    return(df)
  }
}
```

Load data

```
other_quant <- load_qualtrics_data("clean_data/other_quant.tsv")
status <- load_qualtrics_data("clean_data/contributor_status_Q3.tsv")
```

Filter for contributors, if desired

Here, I use a boolean variable called `contributor_mode` to control whether I want to create donut plots for all respondents or contributors only. Because only experienced contributors saw the "favorite_solution" question, I can use this column to filter out everybody except experienced contributors. (Also, if this code seems weird, it's probably because `contributor_mode` used to be a command-line argument, before I turned this script into a notebook.)

I generally keep this set to FALSE. Since 70% of our respondents are contributors, the contributor responses are fairly similar to the overall population responses.

```
# EDIT ME, IF YOU WISH
contributor_mode <- FALSE

other_quant <- filter_contribs(other_quant, contributor_mode)

population <- ""
# For plot axis labels
if (contributor_mode) {
  population <- "Contributors"
} else {
  population <- "Respondents"
}
```

Donut charts of participation by groups

This script creates 4 plots, called `p1`-`p4`.

Plot #1 is of campus.

```
campus_data <- create_df_for_plotting(other_quant, "campus")
campus_data <- campus_data %>%
  filter(
    !startsWith(as.character(values), "I'm not affiliat")
  )

p1 <- donut_chart(campus_data) +
  labs(title = sprintf("Campus of %s", population))
```

Save data to file for fine-tuning of plot.

```
cd <- campus_data
names(cd)[1] <- "categories"
names(cd)[2] <- "n_responses"
write_df_to_file(
  cd,
  file.path("data_for_plots/fig_campus_donut.tsv")
)
```

Plot #2 is academics' field of study. Qualitative answers are not included in this plot.

```
field_data <- create_df_for_plotting(other_quant, "field_of_study")
p2 <- donut_chart(field_data) +
  labs(title = sprintf("Academic %s' Fields of Study", population))
```

Save data to file for fine-tuning of plot.

```
fd <- field_data
names(fd)[1] <- "categories"
names(fd)[2] <- "n_responses"
write_df_to_file(
  fd,
  file.path("data_for_plots/fig_fields_donut.tsv")
)
```

Plot #3 is of all respondents' job categories.

```
job_data <- create_df_for_plotting(other_quant, "job_category")

p3 <- donut_chart(job_data) +
  labs(title = sprintf("%s' Job Categories", population))
```

Save the count data to a file for fine-tuning of plot. Since this plot is more complicated, I am not pre-calculating the label positions; I am just saving the simple count data.

```
jd <- job_data %>% select(c(values, Freq))
names(jd) <- c("categories", "n_responses")
write_df_to_file(
  jd,
  file.path("data_for_plots/fig_job_donut.tsv")
)
```

Plot #4 is of staff categories (IT, Research Support, etc.). This one is more complicated. I'm not using my create_df_for_plotting function for this group because I want to combine the jobs that have only 1 or 2 responses into the existing "Other" category. So the code is similar but not the same.

```
staff_data <- other_quant[["staff_categories"]][nzchar(other_quant[[
  "staff_categories"
]])]
# Count occurrences of each unique value
staff_data <- as.data.frame(table(staff_data))
names(staff_data) <- c("job", "count")
staff_data$job <- as.character(staff_data$job)

staff_data_clean <- staff_data %>%
  mutate(job = if_else(count < 3, "Other", job)) %>% # relabel rare jobs as "Other"
  group_by(job) %>% # gather all "Other" rows together
  summarise(Freq = sum(count), .groups = "drop")

staff_long_data <- as.data.frame(staff_data_clean) %>%
  mutate(fraction = Freq / sum(Freq)) %>%
  arrange(desc(fraction))

# Compute the cumulative percentages (top of each rectangle)
staff_long_data$ymax <- cumsum(staff_long_data$fraction)
staff_long_data$ymin <- c(0, head(staff_long_data$ymax, n = -1))

# Compute label position
staff_long_data$labelPosition <- (staff_long_data$ymax + staff_long_data$ymin) /
  2

# Create label column
staff_long_data$label <- paste0(staff_long_data$Freq)
```

```
# Rename this one column to match the donut_chart function
names(staff_long_data)[names(staff_long_data) == "job"] <- "values"

# Wrap text
staff_long_data <- staff_long_data %>%
  mutate(values = str_wrap(values, width = 20))

# from scripts/utils.R
staff_long_data <- reorder_factor_by_column(
  staff_long_data,
  values,
  Freq
)

p4 <- donut_chart(staff_long_data) +
  labs(title = sprintf("Staff %s' Work Areas", population))
```

Save data to file for fine-tuning of plot.

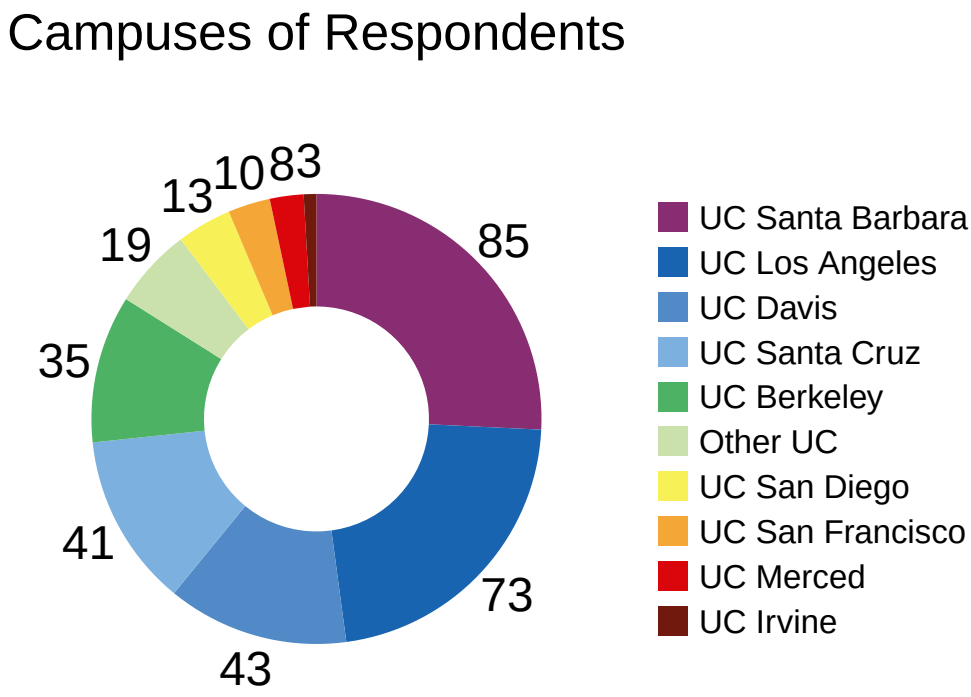
```
sd <- staff_long_data
names(sd)[1] <- "categories"
names(sd)[2] <- "n_responses"
# I put newline characters in some of these categories so they would
# fit on the plot; remove them before saving to file
sd$categories <- sapply(sd$categories, function(x) gsub("\n", " ", x))
write_df_to_file(
  sd,
  file.path("data_for_plots/fig_staff_donut.tsv")
)
```

Refine plot aesthetics

I'm envisioning a figure where I'd like to have some variety in the color palettes and the legend position. I'll put three donuts onto one canvas in Powerpoint, and then make some additional tweaks: I'll add some lines to illustrate which slices of the donut I'm focusing on, and also I will need to fix this bug where the legend keys stretch to multiple lines. (https://github.com/tidyverse/ggplot2/issues/3669) (A fix has been merged, but it's not yet available in the latest release of ggplot2.)

Plot 1: campus

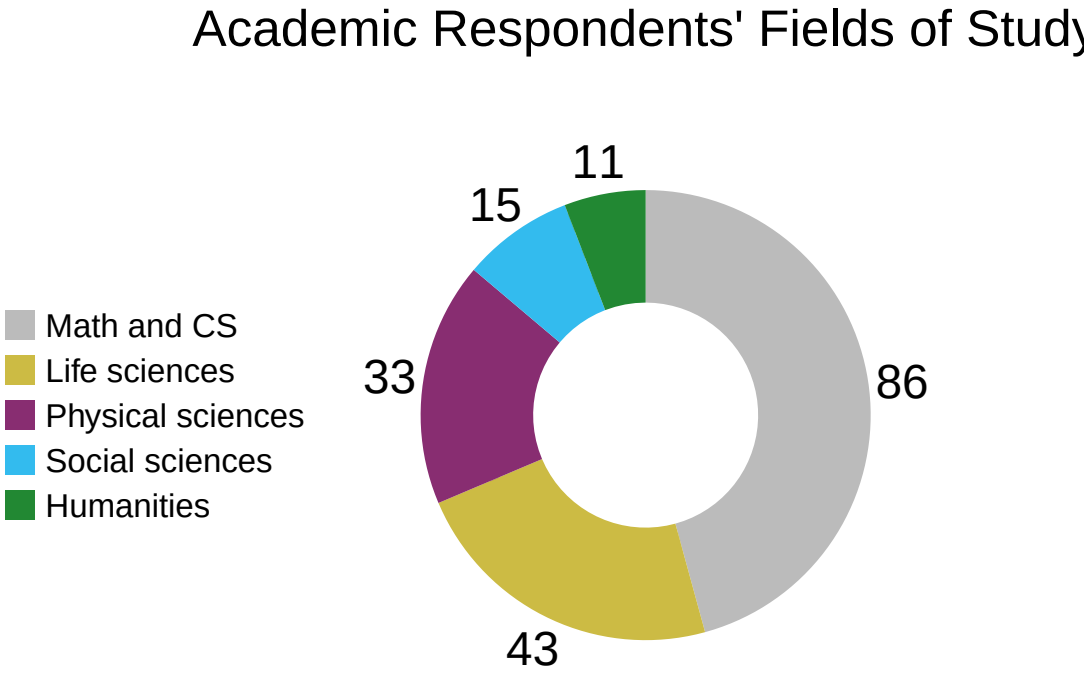
```
p1 <- donut_chart(
  campus_data,
  # Paul Tol's "rainbow" color scheme
  cpalette = c(
  "#882d71",
  "#1964b0",
  "#518ac6",
  "#7bb0df",
  "#4db264",
  "#cae1ac",
  "#f7f057",
  "#f4a637",
  "#db060b",
  "#71190d"
  )
 ) +
  labs(title = sprintf("Campuses of %s", population))
p1
```



```r
# Function from utils.R
save_plot(sprintf("donut_p1_%s.tiff", population), 10, 6, p = p1)
```

Plot 2: fields of study

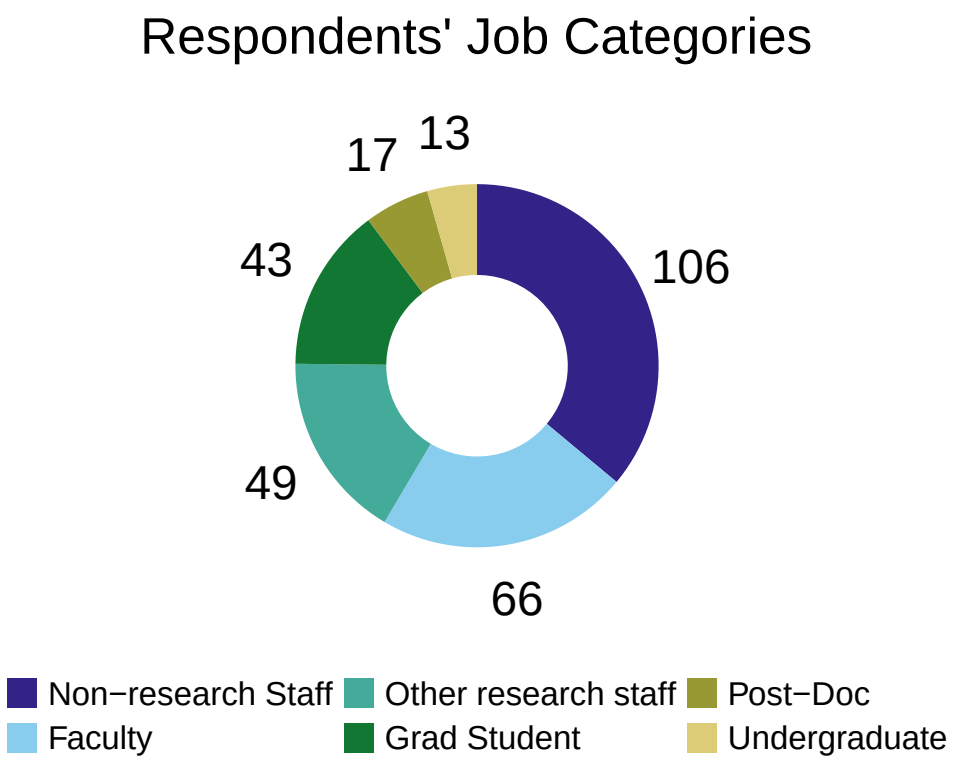
```r
p2 <- donut_chart(
  field_data,
  cpalette = c(
    '#BBBBBB',
    '#CCBB44',
    '#882d71',
    '#33BBEE',
    '#228833'
  ),
  legendpos="left"
) +
  labs(title = sprintf("Academic %s' Fields of Study", population))
p2
```



```
# Function from utils.R
save_plot(sprintf("donut_p2_%s.tiff", population), 10, 6, p = p2)
```

Plot3: job categories

```
p3 <- donut_chart(
  job_data,
  legendpos = "bottom",
  title_bottom_margin = 0,
  legend_top_margin = 0,
  label_dist = 4.6,
  white_labels = FALSE
) +
  labs(title = sprintf("%s' Job Categories", population)
  )
p3
```

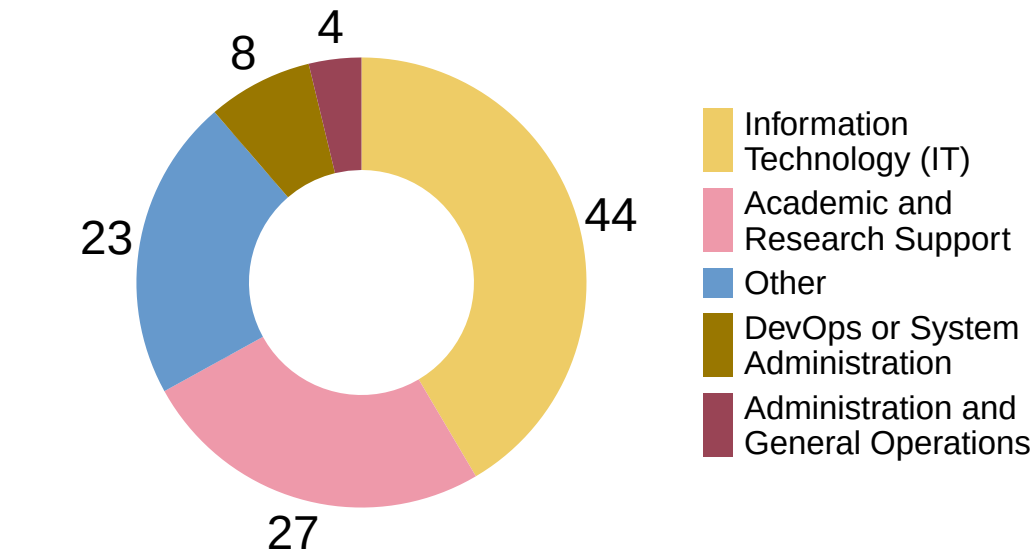


Save plot

```r
# Function from utils.R
save_plot(
  sprintf("donut_p3_%s.tiff", tolower(population)),
  12,
  8,
  p = p3
)
```

Plot 4: non-research staff categories

```r
p4 <- donut_chart(
  staff_long_data,
  cpalette = c(
    '#EECC66',
    '#EE99AA',
    '#6699CC',
    '#997700',
    '#994455'
  )
) +
  labs(title = sprintf("Staff %s' Work Areas", population))
p4
```




```
# Function from utils.R
save_plot(
  sprintf("donut_p4_%s.tiff", tolower(population)),
  10,
  6,
  p = p4
)
```

Experienced vs. aspiring contributors

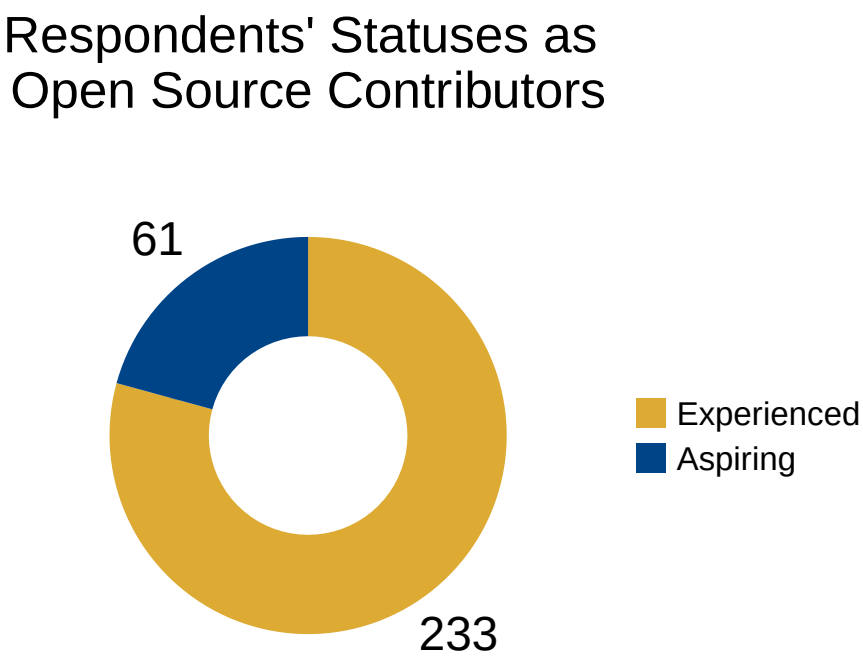
A last-minute add-on: let's also make a simple donut plot with aspiring vs. experienced contributors.

```
# Filter duds
status_clean <- status %>%
    filter(!(Past == "" | Future == "")) %>%
    filter(!(Past == "False" & Future == "False"))
```

```r
status_clean <- status_clean %>%
  mutate(status = if_else(Past == "True", "Experienced", "Aspiring")) %>%
  select(status)

head(status_clean)
```

```
       status
1 Experienced
2 Experienced
3 Experienced
4 Experienced
5 Experienced
6    Aspiring
```

```r
status_data <- create_df_for_plotting(status_clean, "status")

p5 <- donut_chart(
  status_data,
  label_dist = 4.5,
  cpalette = c(
    "#ddaa33",
    "#004488"
  ),
  title_bottom_margin = 0
  ) +
  labs(title = sprintf("%s' Statuses as \nOpen Source Contributors", population))

p5
```




```
# Function from utils.R
save_plot(
  sprintf("donut_p5_%s.tiff", tolower(population)),
  10,
  6,
  p = p5
)
```

Save data to file for fine-tuning of plot.

```
statd <- status_data
names(statd)[1] <- "categories"
names(statd)[2] <- "n_responses"
write_df_to_file(
  statd,
  file.path("data_for_plots/fig_status_donut.tsv")
)
```

```r
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Los_Angeles
tzcode source: internal

attached base packages:
[1] tools     stats     graphics  grDevices datasets  utils     methods
[8] base

other attached packages:
 [1] treemap_2.4-4         tidyr_1.3.1           stringr_1.5.1
 [4] scales_1.4.0          readr_2.1.5           pwr_1.3-0
 [7] patchwork_1.3.0       ordinal_2023.12-4.1   languageserver_0.3.16
[10] here_1.0.1            gtools_3.9.5          ggforce_0.5.0
[13] fpc_2.2-13            forcats_1.0.0         factoextra_1.0.7
[16] ggplot2_3.5.2         emmeans_1.11.1        dplyr_1.1.4
[19] corrplot_0.95         cluster_2.1.8.1

loaded via a namespace (and not attached):
 [1] tidyselect_1.2.1    gridBase_0.4-7      farver_2.1.2
 [4] fastmap_1.2.0       tweenr_2.0.3        promises_1.3.3
 [7] digest_0.6.37       mime_0.13           estimability_1.5.1
[10] lifecycle_1.0.4     processx_3.8.6      magrittr_2.0.3
[13] kernlab_0.9-33      compiler_4.4.2      rlang_1.1.6
[16] igraph_2.1.4        yaml_2.3.10         data.table_1.17.6
[19] knitr_1.50          labeling_0.4.3      mclust_6.1.1
[22] xml2_1.3.8          RColorBrewer_1.1-3  withr_3.0.2
[25] purrr_1.0.4         numDeriv_2016.8-1.1 nnet_7.3-19
[28] grid_4.4.2          polyclip_1.10-7     stats4_4.4.2
[31] xtable_1.8-4        colorspace_2.1-1    MASS_7.3-61
[34] prabclus_2.3-4      cli_3.6.5           mvtnorm_1.3-3
```

```
[37] rmarkdown_2.29      generics_0.1.4      robustbase_0.99-4-1
[40] tzdb_0.5.0          modeltools_0.2-24   parallel_4.4.2
[43] vctrs_0.6.5         Matrix_1.7-1        jsonlite_2.0.0
[46] callr_3.7.6         hms_1.1.3           ggrepel_0.9.6
[49] diptest_0.77-1      glue_1.8.0          DEoptimR_1.1-3-1
[52] ps_1.9.1            stringi_1.8.7       gtable_0.3.6
[55] later_1.4.2         tibble_3.2.1        pillar_1.10.2
[58] htmltools_0.5.8.1   R6_2.6.1            ucminf_1.2.2
[61] rprojroot_2.0.4     evaluate_1.0.3      shiny_1.11.0
[64] lattice_0.22-6      renv_1.1.4          httpuv_1.6.16
[67] class_7.3-22        Rcpp_1.0.14         flexmix_2.3-20
[70] nlme_3.1-166        xfun_0.52           pkgconfig_2.0.3
```