

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 unmodified data frame with all the (deidentified) survey data.
 - `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.

```
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,
  white_labels = FALSE
) {

  if (white_labels) {
    label_x      <- 3.5
    label_size   <- 8
    label_color  <- "white"
    # We only need xlim up to 4 (labels sit inside the ring)
    #x_limits    <- c(2, 4)
  } else {
    label_x      <- 4.3
    label_size   <- 9
    label_color  <- "black"
    # Extend x-axis so outside labels aren't cut off
    #x_limits    <- c(2, 4.6)
  }

  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") +
xlim(c(2, 4)) +
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 = 5, 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")
```

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 job category (Undergrad, Post-Doc, Non-research Staff, etc.).

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

p1 <- donut_chart(job_data) +
  labs(title = sprintf("Job Category of %s", population))
```

Plot #2 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")
  )

p2 <- donut_chart(campus_data) +
  labs(title = sprintf("Campus of %s", population))
```

Plot #3 is of their field of study. Qualitative answers are not included in this plot.

```
field_data <- create_df_for_plotting(other_quant, "field_of_study")
p3 <- donut_chart(field_data) +
  labs(title = sprintf("Academic %s' fields of study", tolower(population)))
```

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", tolower(population)))

```

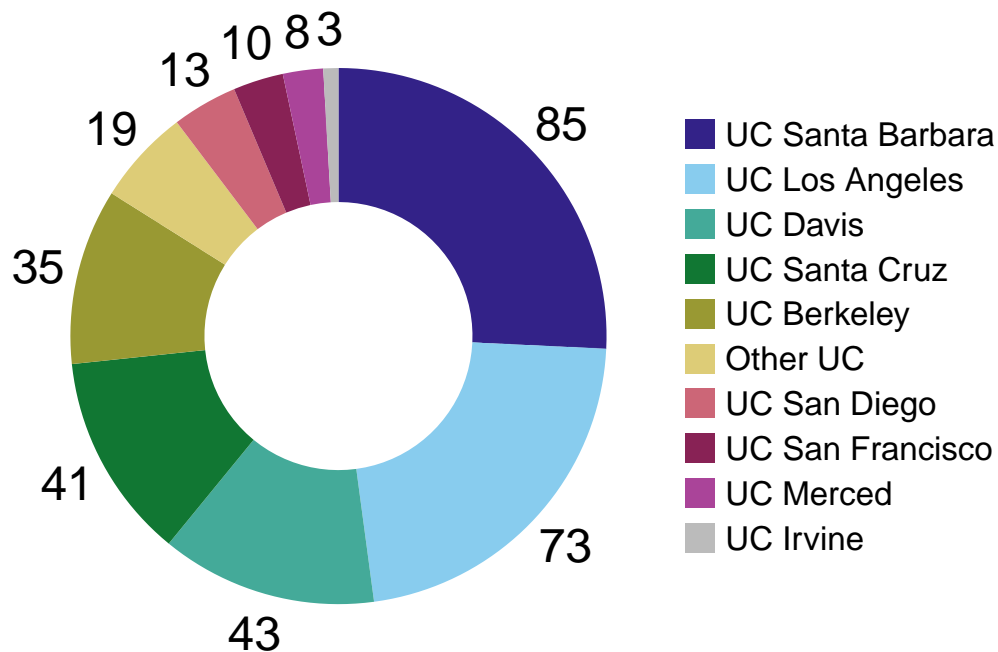
View and save combined plot

First let's view and save one plot.

Individual plot: campus

```
p2
```

Campus of Respondents



Save if you wish.

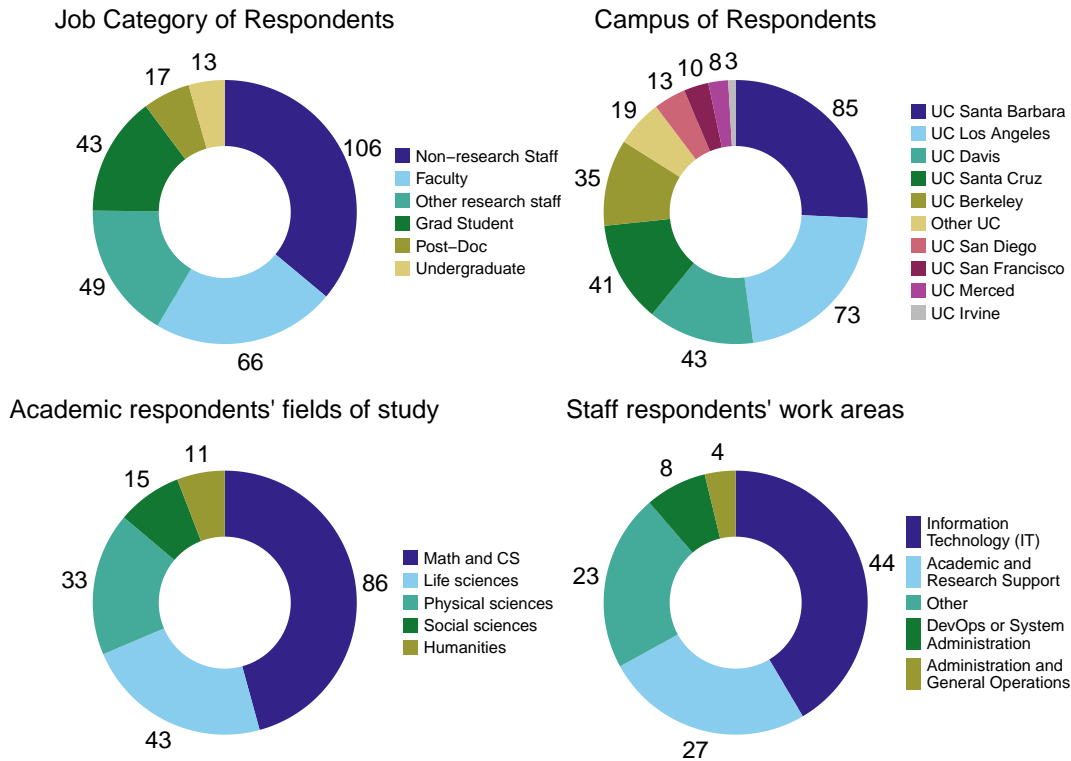
```
# Function from utils.R
# save_plot(
#   sprintf("campus_donut_%.s.tiff", tolower(population)),
#   9,
#   6,
#   p = p2
# )
```

All four plots combined

It's also nice to see them all in one figure.

```
combined_donuts <- patchwork::wrap_plots(p1, p2, p3, p4, ncol = 2)
```

```
combined_donuts
```

```
# Function from utils.R
# save_plot(
#   sprintf("combined_donuts_%s.tiff", tolower(population)),
#   18,
#   12,
#   p = combined_donuts
# )
```

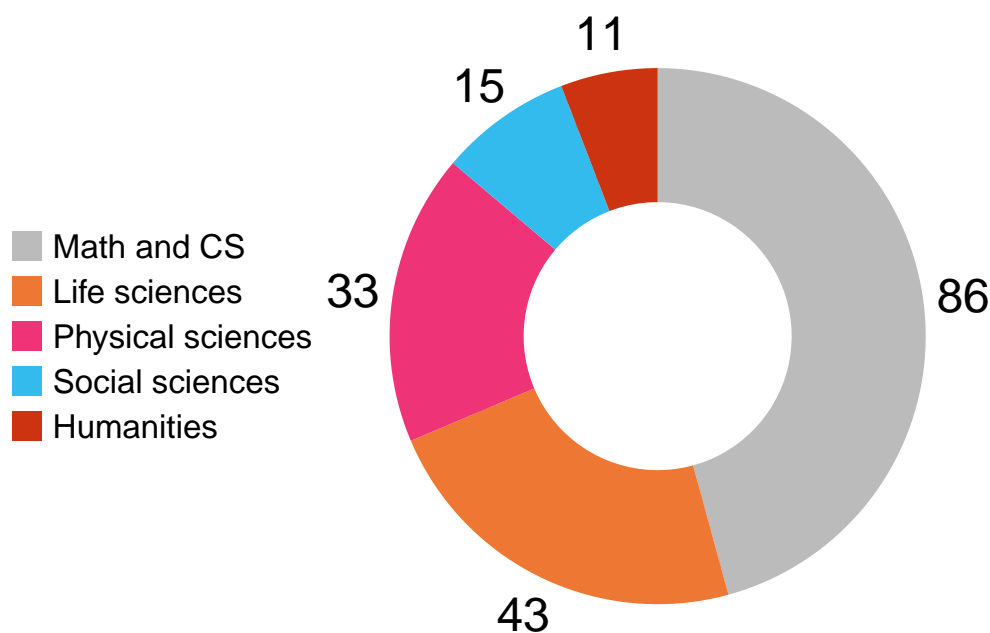
Refine plot aesthetics

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.)

Left plot: fields of study

```
p3 <- donut_chart(
  field_data,
  cpalette = c(
    '#BBBBBB',
    '#EE7733',
    '#EE3377',
    '#33BBEE',
    '#CC3311'
  ),
  legendpos="left"
) +
  labs(title = sprintf("Academic %s' fields of study", tolower(population)))
p3
```

Academic respondents' fields of study

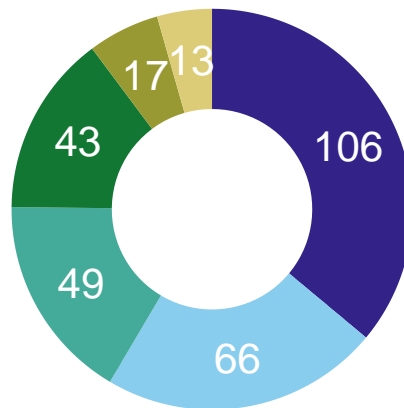


```
# Function from utils.R
save_plot(sprintf("field_donut_%s.tiff", tolower(population)), 10, 6, p = p3)
```

Central plot: job categories

```
p1 <- donut_chart(
  job_data,
  legendpos = "bottom",
  title_bottom_margin = 40,
  legend_top_margin = 20,
  white_labels = TRUE
) +
  labs(title = sprintf("Job category of %s", tolower(population)))
p1
```

Job category of respondents



Non-research Staff
 Other research staff
 Post-Doc
 Faculty
 Grad Student
 Undergraduate

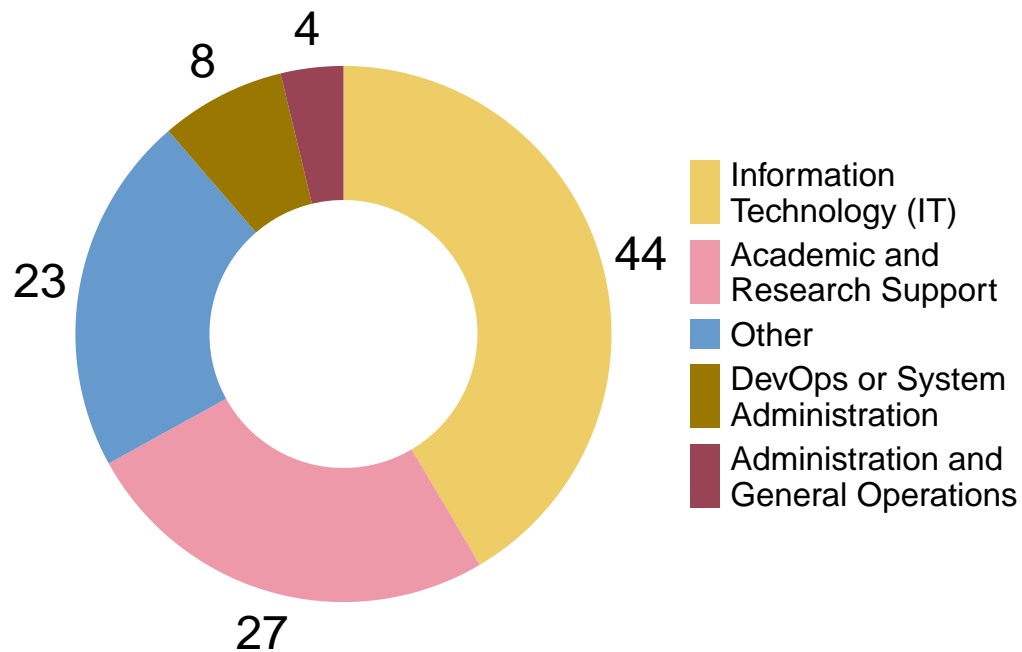
Save plot

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

Right plot: non-research staff categories

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

Staff respondents' work areas



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

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.4.1
```

```
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;
```

```
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    mvabund_4.2.1      languageserver_0.3.16
[10] here_1.0.1         gtools_3.9.5       fpc_2.2-13
[13] forcats_1.0.0      factoextra_1.0.7   ggplot2_3.5.2
[16] dplyr_1.1.4        corrplot_0.95      cluster_2.1.8.1
```

```
loaded via a namespace (and not attached):
```

```
[1] gtable_0.3.6      xfun_0.52          ggrepel_0.9.6
[4] processx_3.8.6    lattice_0.22-6     callr_3.7.6
[7] tzdb_0.5.0        vctrs_0.6.5        ps_1.9.1
[10] generics_0.1.4    stats4_4.4.2       parallel_4.4.2
[13] flexmix_2.3-20    tibble_3.2.1       DEoptimR_1.1-3-1
```

[16]	pkgconfig_2.0.3	data.table_1.17.6	RColorBrewer_1.1-3
[19]	lifecycle_1.0.4	compiler_4.4.2	farver_2.1.2
[22]	statmod_1.5.0	httpuv_1.6.16	htmltools_0.5.8.1
[25]	class_7.3-22	yaml_2.3.10	later_1.4.2
[28]	pillar_1.10.2	prabclus_2.3-4	MASS_7.3-61
[31]	diptest_0.77-1	mclust_6.1.1	mime_0.13
[34]	robustbase_0.99-4-1	tidyselect_1.2.1	digest_0.6.37
[37]	stringi_1.8.7	purrr_1.0.4	kernlab_0.9-33
[40]	labeling_0.4.3	rprojroot_2.0.4	fastmap_1.2.0
[43]	grid_4.4.2	colorspace_2.1-1	cli_3.6.5
[46]	magrittr_2.0.3	withr_3.0.2	promises_1.3.3
[49]	tweedie_2.3.5	rmarkdown_2.29	igraph_2.1.4
[52]	nnet_7.3-19	modeltools_0.2-24	hms_1.1.3
[55]	shiny_1.11.0	evaluate_1.0.3	knitr_1.50
[58]	rlang_1.1.6	Rcpp_1.0.14	xtable_1.8-4
[61]	gridBase_0.4-7	glue_1.8.0	xml2_1.3.8
[64]	renv_1.1.4	jsonlite_2.0.0	R6_2.6.1