Challenges

Overview

Initial analysis of survey Q9: "How frequently have you encountered the following challenges while working on open-source projects?"

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")))
```

Set seed

```
set.seed(42)
```

Define functions

multiple_plots

- Arguments:
  - `df`: In this script, this will always be the `to_plot` data frame. Must contain (at least) three columns: `challenge`, `challenge_level` (a character column), and `total`.

  - `title_codes`: In this script, this will always be the `titles` list. Keys are shorthand codes for each challenge, and values are the full challenge from the survey.
  - `challenges_of_interest`: A character vector of the challenges you want to plot.

- Details:

  - A simple function to call my `basic_bar_chart` function (from scripts/utils.R) on multiple challenges, producing multiple plots.

- Outputs:

  - Prints n plots, where n is the number of challenges of interest.

```r
multiple_plots <- function(df, title_codes, challenges_of_interest) {
  for (ch in challenges_of_interest) {
    df_ch <- filter(df, challenge == ch)
    plot_title <- title_codes[[ch]]
    p <- basic_bar_chart(
      df_ch,
      x_var = "challenge_level",
      y_var = "total",
      title = plot_title,
      show_grid = TRUE
    )
    print(p)
  }
}
```

Load data

```r
challenges <- load_qualtrics_data("clean_data/challenges_Q9.tsv")
```

Wrangle data

Remove empty rows, i.e. rows from respondents who didn't receive this question. As with many questions in this survey, we can cut some corners in the code because the question was mandatory. For example, no need to worry about incomplete answers.

```r
nrow(challenges)
```

```
[1] 332
```

```r
challenges <- exclude_empty_rows(challenges) # from scripts/utils.R
nrow(challenges)
```

```
[1] 233
```

Let's reshape the data from wide to long format for easier plotting later. We'll also recode the Likert values to integers, so we can get descriptive statistics of the responses. ("Never" = 0, "Non-applicable" = 0,"Rarely" = 1, "Occasionally" = 2, "Frequently" = 3, "Always" = 4)

```r
long_data <- challenges %>%
  pivot_longer(
    cols = everything(),
    names_to = "challenge",
    values_to = "challenge_level"
  )

long_data <- long_data %>%
  mutate(
    challenge_score = recode(
      challenge_level,
      "Never"           = 0L,
      "Non-applicable"  = 0L,
      "Rarely"          = 1L,
      "Occasionally"    = 2L,
      "Frequently"      = 3L,
      "Always"          = 4L
    )
  )
# Using interger literals 0L, 1L, etc., ensures that
# the new column will be integers, not doubles.

long_data
```

```
# A tibble: 3,262 x 3
   challenge          challenge_level challenge_score
   <chr>              <chr>                     <int>
 1 Coding time        Always                        4
 2 Documentation time Always                        4
 3 Managing issues    Always                        4
 4 Attracting users   Always                        4
 5 Recognition        Always                        4
```

```
 6 Hiring              Always                        4
 7 Security            Always                        4
 8 Finding peers       Always                        4
 9 Finding mentors     Always                        4
10 Education time      Always                        4
# i 3,252 more rows
```

Next, let's calculate some simple descriptive statistics. I will choose:
* The total "score", that is, the total number of "points" a challenge received * The mean (which might be misleading if 0s drag it down, and also, who's to say what a 2.5 really means? Are the distances between the Likert points equal? We assume so, but this is hand-wavy.) * The median * The mode * The standard deviation

```r
# Helper to compute the (numeric) mode
get_mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

summary_df <- long_data %>%
  group_by(challenge) %>%
  summarise(
    total  = sum(challenge_score),
    mean   = mean(challenge_score, na.rm = TRUE),
    median = median(challenge_score),
    mode   = get_mode(challenge_score),
    st_dev = sd(challenge_score, na.rm = TRUE)
  ) %>%
  ungroup()

# Order by highest total "score"
summary_df <- summary_df %>%
    arrange(desc(total))

summary_df
```

```
# A tibble: 14 x 6
   challenge             total  mean median  mode st_dev
   <chr>                 <int> <dbl>  <int> <int>  <dbl>
 1 Documentation time      686  2.94      3     3   1.08
 2 Coding time             606  2.60      3     3   1.24
 3 Education time          539  2.31      2     3   1.26
```

```
 4 Managing issues         451  1.94      2     2   1.29
 5 Attracting users        442  1.90      2     0   1.45
 6 Securing funding        438  1.88      2     0   1.74
 7 Finding funding         432  1.85      2     0   1.68
 8 Educational resources   369  1.58      2     1   1.19
 9 Recognition             334  1.43      1     0   1.35
10 Legal                   333  1.43      1     0   1.24
11 Finding mentors         323  1.39      1     0   1.31
12 Security                307  1.32      1     0   1.31
13 Hiring                  291  1.25      0     0   1.53
14 Finding peers           267  1.15      1     0   1.13
```

Cool! It looks like finding the time for documentation, coding, and self-education are the challenges encountered most frequently. These are the only responses that had a mode of 3 ("Frequently") and a mean of **greater** than 2 ("Occasionally").

```r
# Prettify
summary_df_to_print <- summary_df %>%
  rename(
    `Challenge` = `challenge`,
    `Total` = `total`,
    `Mean` = `mean`,
    `Median` = `median`,
    `Mode` = `mode`,
    `Standard Deviation` = `st_dev`
  )

write_df_to_file(summary_df_to_print, "challenges.tsv")
```

Out of curiosity, how does it look when we order by variability?

```r
sd_df <- summary_df %>%
    arrange(desc(st_dev))

sd_df
```

```
# A tibble: 14 x 6
   challenge             total  mean median  mode st_dev
   <chr>                 <int> <dbl>  <int> <int>  <dbl>
 1 Securing funding        438  1.88      2     0   1.74
 2 Finding funding         432  1.85      2     0   1.68
```

```
 3 Hiring                   291  1.25      0     0   1.53
 4 Attracting users         442  1.90      2     0   1.45
 5 Recognition              334  1.43      1     0   1.35
 6 Security                 307  1.32      1     0   1.31
 7 Finding mentors          323  1.39      1     0   1.31
 8 Managing issues          451  1.94      2     2   1.29
 9 Education time           539  2.31      2     3   1.26
10 Legal                    333  1.43      1     0   1.24
11 Coding time              606  2.60      3     3   1.24
12 Educational resources    369  1.58      2     1   1.19
13 Finding peers            267  1.15      1     0   1.13
14 Documentation time       686  2.94      3     3   1.08
```

Fascinating! The greatest standard deviations are from securing funding, finding funding, and hiring. This makes sense, as these are, at least in my perception, "manager tasks"–tasks that only some people face, but they're likely to be a big challenge for those who face them. I would guess that these might show a bimodal distribution. Let's plot them and find out!

Plot the distributions

Prepare data for plotting

```r
ordered_levels <- c(
  "Non-applicable",
  "Never",
  "Rarely",
  "Occasionally",
  "Frequently",
  "Always"
)

to_plot <- long_data %>%
  mutate(
    challenge_level = factor(
      challenge_level,
      levels = ordered_levels
    )
  ) %>%
  count( # Count number of occurrences,
  #i.e. number of people who selected that challenge level
    challenge,
```

```
    challenge_level,
    name = "total"
  ) %>%
  ungroup()

to_plot
```

```
# A tibble: 84 x 3
   challenge        challenge_level total
   <chr>            <fct>           <int>
 1 Attracting users Non-applicable     50
 2 Attracting users Never              15
 3 Attracting users Rarely             24
 4 Attracting users Occasionally       53
 5 Attracting users Frequently         52
 6 Attracting users Always             39
 7 Coding time      Non-applicable     21
 8 Coding time      Never               4
 9 Coding time      Rarely             13
10 Coding time      Occasionally       54
# i 74 more rows
```

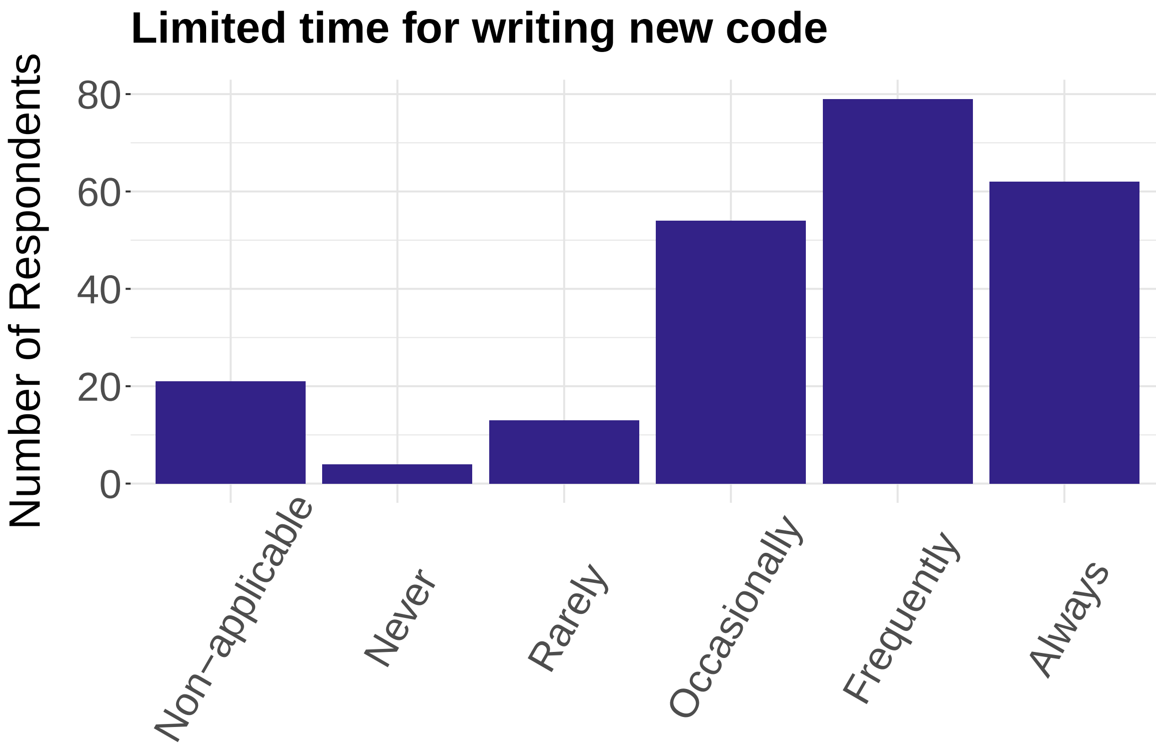
Create a plot for each "challenge". After inspecting the plots, I attempted to order them into groups based on the shape of their distribution. These are the shapes I observed (this is extremely subjective):
* Right-skewed: Documentation time, coding time, education time * Interpretation: Common tasks that are frequently challenging * Highly bimodal: Securing funding, identifying funding, hiring * Interpretation: Tasks that are not as common, but they are frequently challenging for the people tasked with them. * Normal: Educational resources, Legal * Interpretation: Moderately common tasks that are challenging with moderate frequency. * NA-skewed but otherwise normal: Attracting users, Receiving recognition, finding mentors, managing security risks, managing issues * Interpretation: Less-common tasks that are challenging with moderate frequency. * Left-skewed: Finding peers * Interpretation: Moderately common tasks that are infrequently challenging.

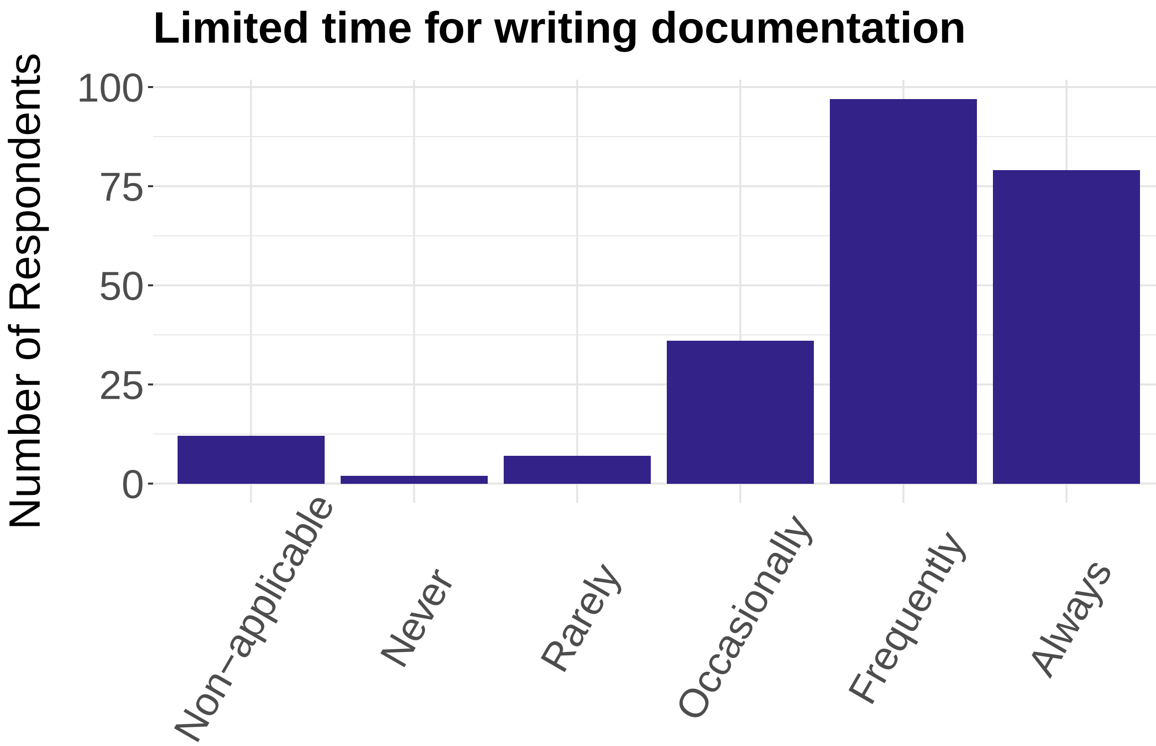
```
titles <- list(
    "Coding time" = "Limited time for writing new code",
    "Documentation time" = "Limited time for writing documentation",
    "Managing issues" = "Managing issues and pull requests",
    "Attracting users" = "Attracting users and/or contributors",
    "Recognition" = "Receiving recognition for my contributions",
```

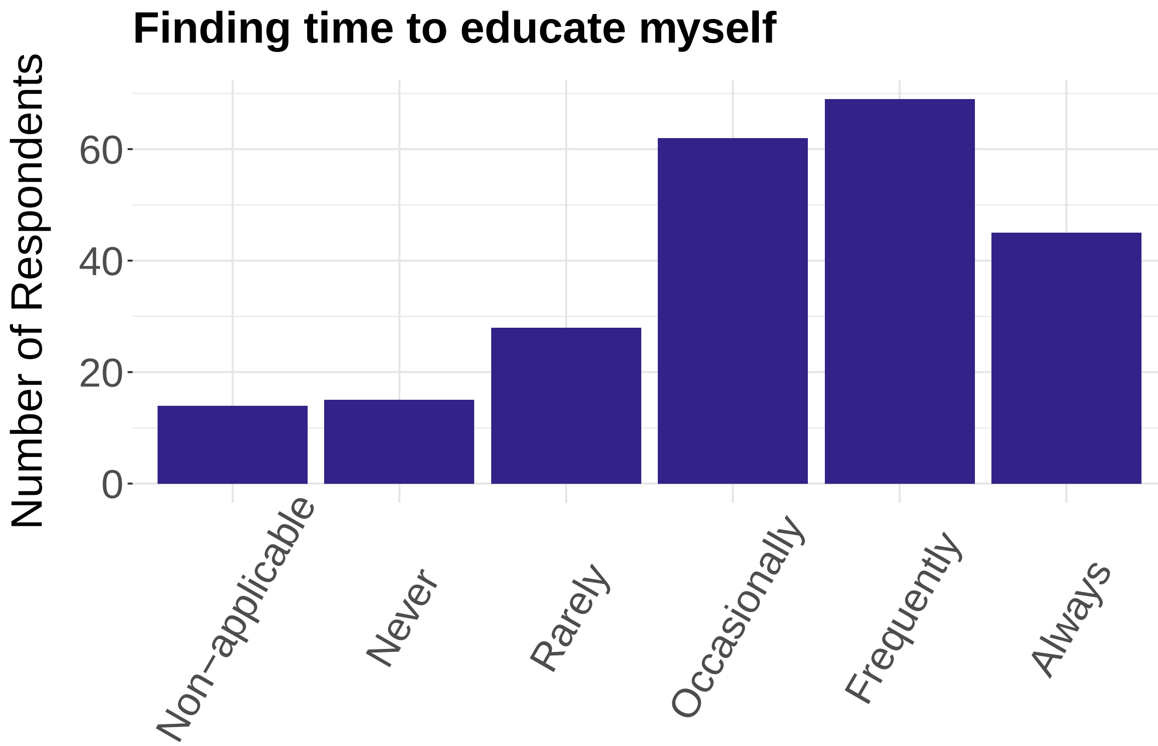
```r
    "Hiring" = "Finding and hiring qualified personnel",
    "Security" = "Managing security risks",
    "Finding peers" = "Finding a community of peers who share my interests",
    "Finding mentors" = "Finding mentors",
    "Education time" = "Finding time to educate myself",
    "Educational resources" = "Identifying helpful educational resources",
    "Legal" = "Navigating licensing and other legal issues",
    "Finding funding" = "Identifying potential funding sources\nfor my open source projects"
    "Securing funding" = "Securing funding for my open source projects"
)

right_skewed <- c(
    "Coding time",
    "Documentation time",
    "Education time"
)
bimodal <- c(
    "Finding funding",
    "Securing funding",
    "Hiring"
)
normal <- c(
    "Educational resources",
    "Legal"
)
na_skewed <- c(
    "Managing issues",
    "Attracting users",
    "Recognition",
    "Security",
    "Finding mentors"
)
left_skewed <- c(
    "Finding peers"
)
```

"right-skewed"

```r
multiple_plots(to_plot, titles, right_skewed)
```

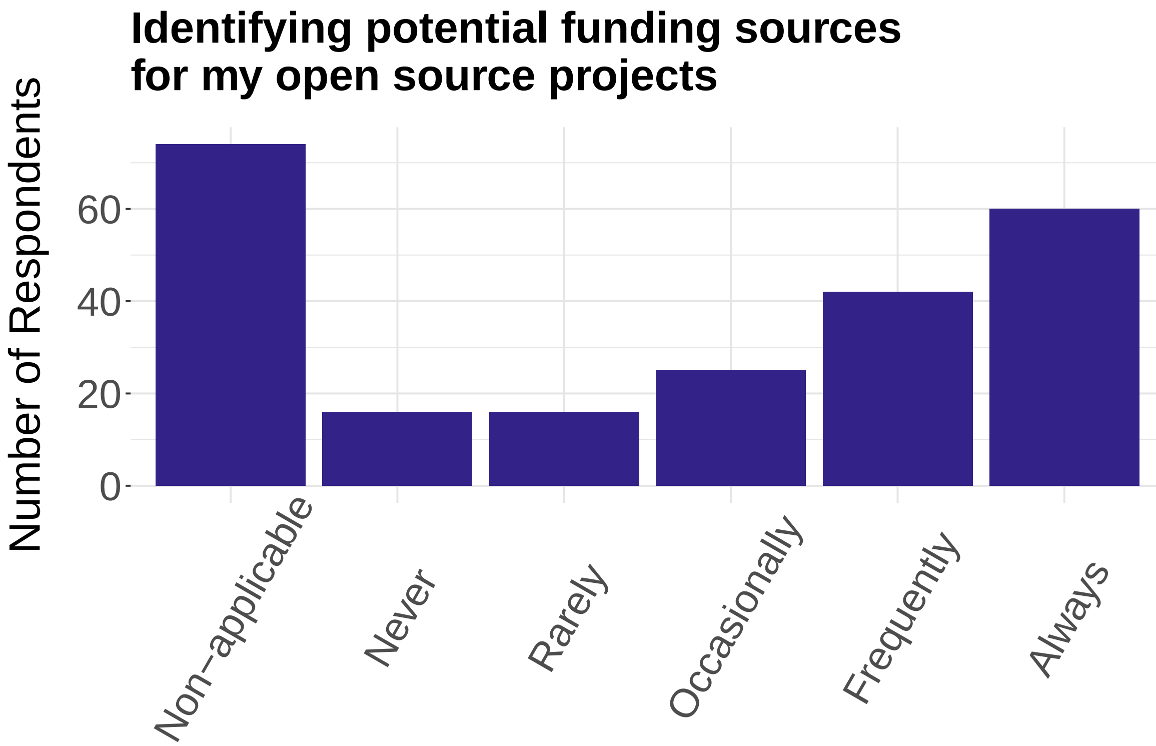

Limited time for writing new code
Number of Respondents
0
20
40
60
80
Non–applicable
Never
Rarely
Occasionally
Frequently
Always

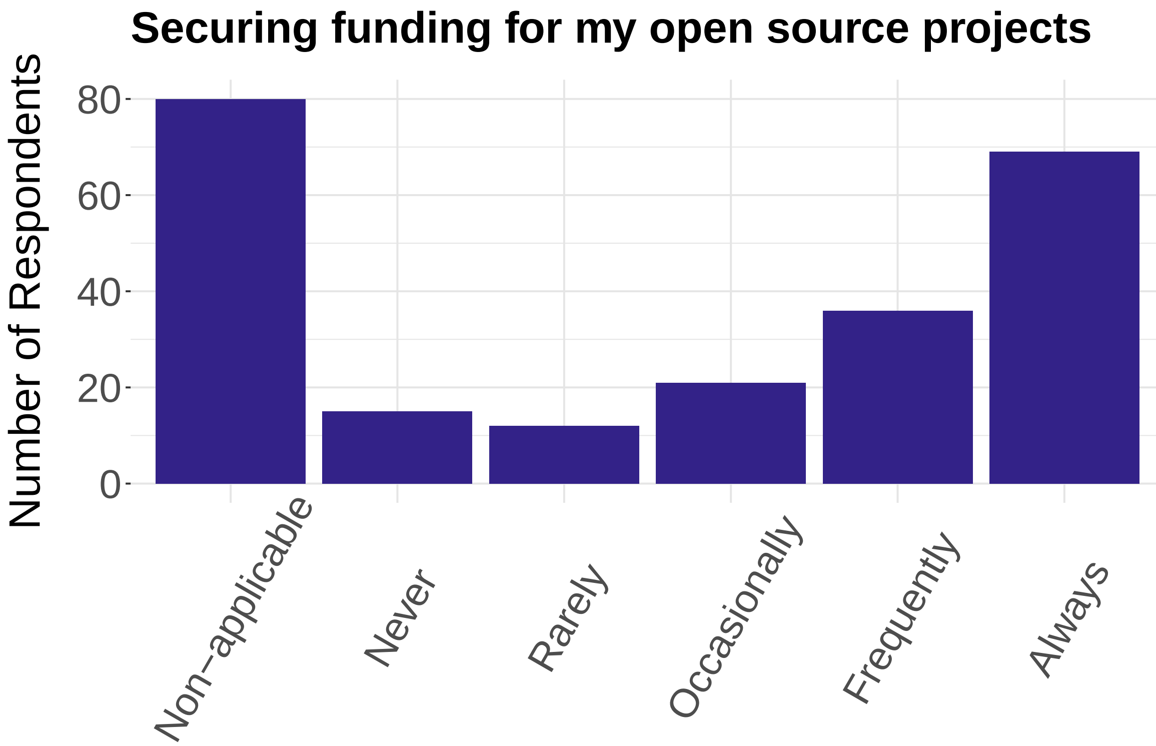

Limited time for writing documentation
Number of Respondents
100
75
50
25
0
Non-applicable
Never
Rarely
Occasionally
Frequently
Always



"highly bimodal"

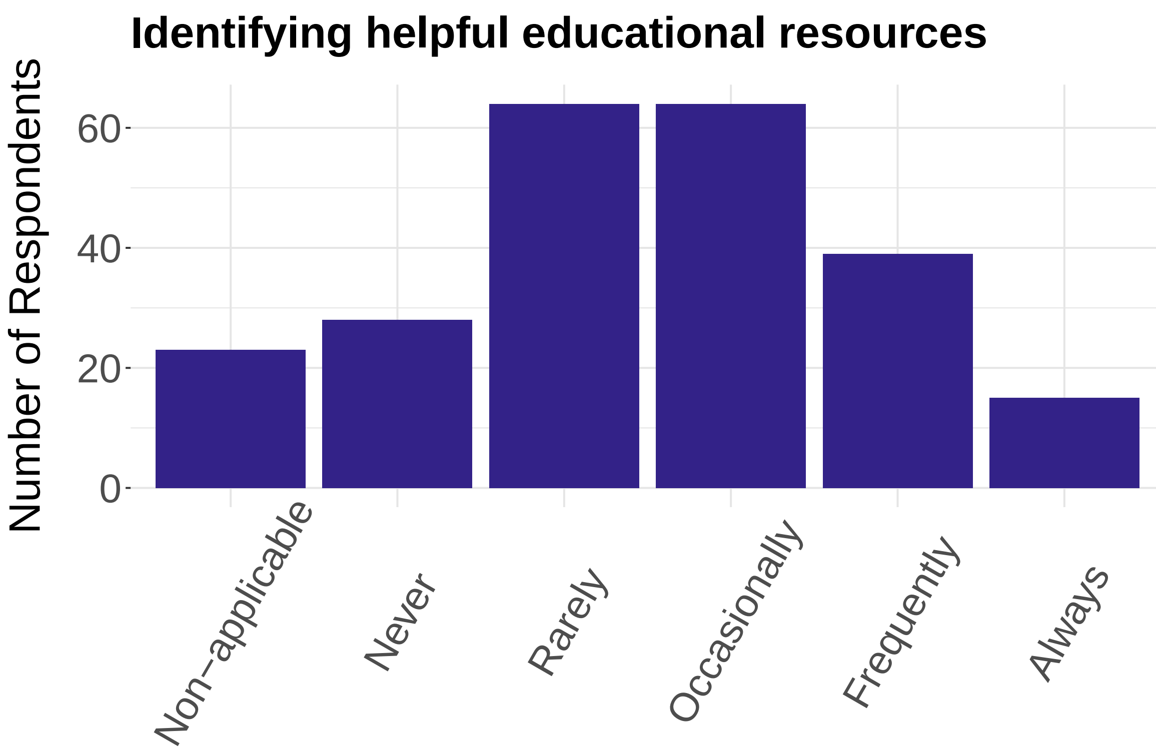
```
multiple_plots(to_plot, titles, bimodal)
```

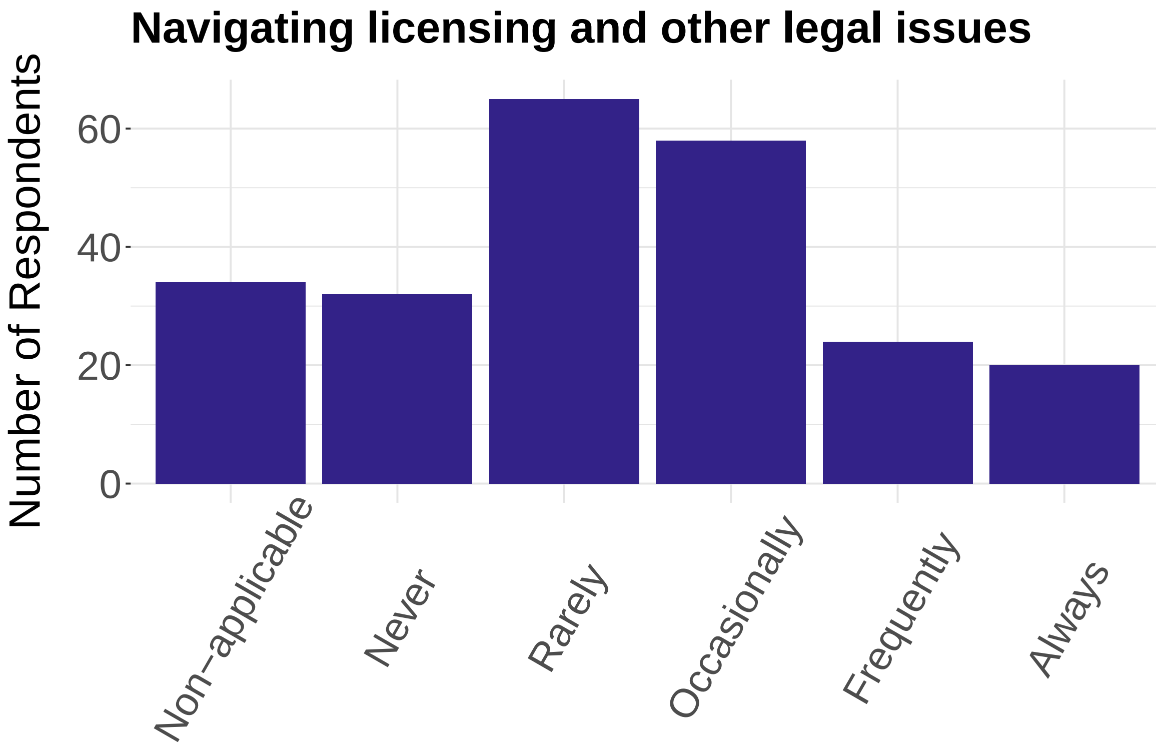

Identifying potential funding sources
for my open source projects
Number of Respondents
0
20
40
60
Non–applicable
Never
Rarely
Occasionally
Frequently
Always


Securing funding for my open source projects
Number of Respondents
80
60
40
20
0
Non–applicable
Never
Rarely
Occasionally
Frequently
Always

“normal”

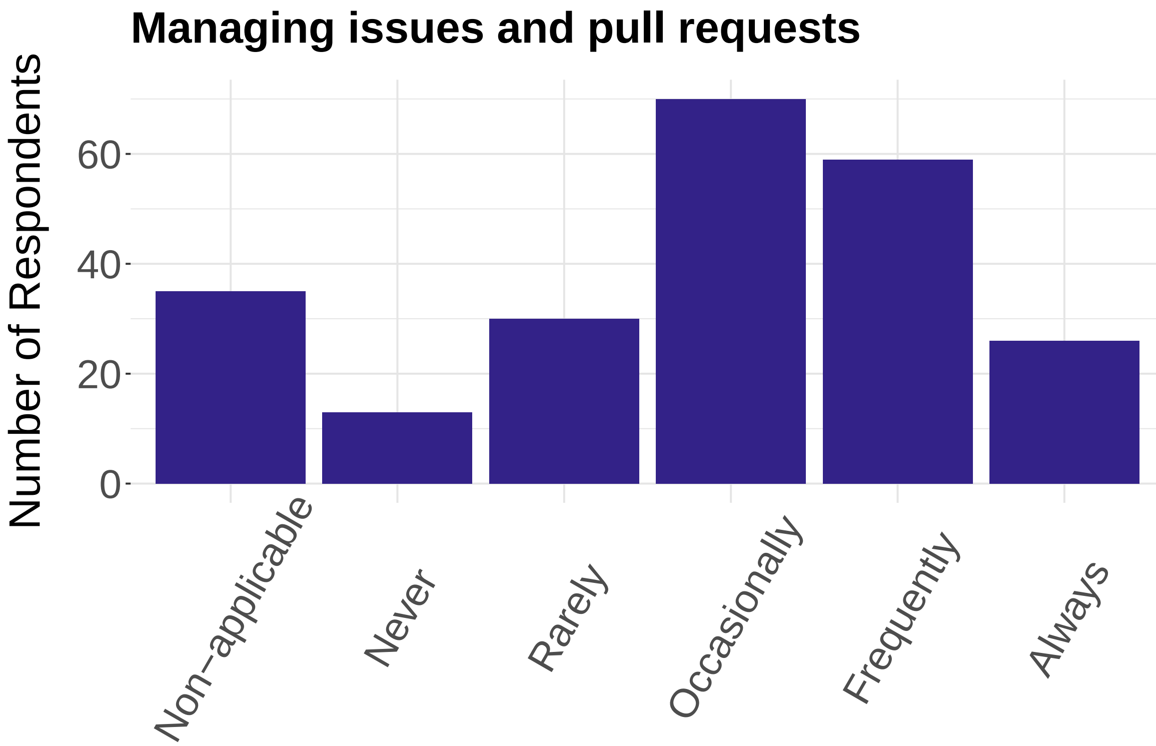
```
multiple_plots(to_plot, titles, normal)
```

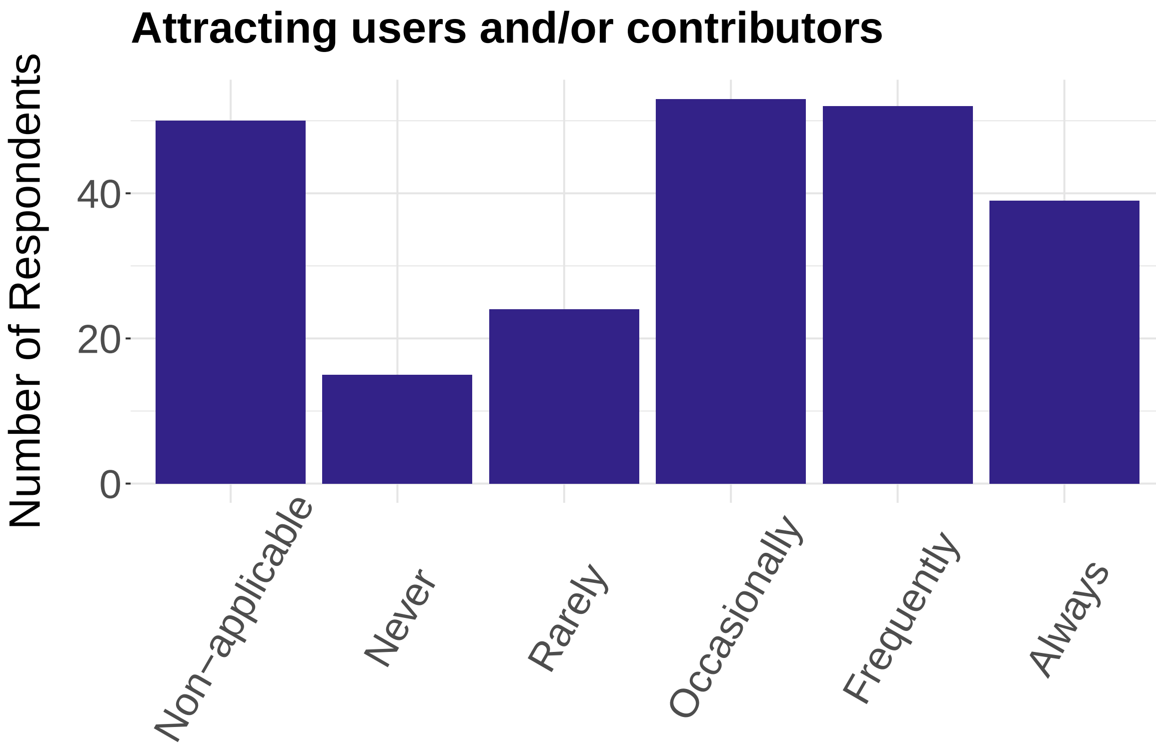

Identifying helpful educational resources
Number of Respondents
0
20
40
60
Non–applicable
Never
Rarely
Occasionally
Frequently
Always

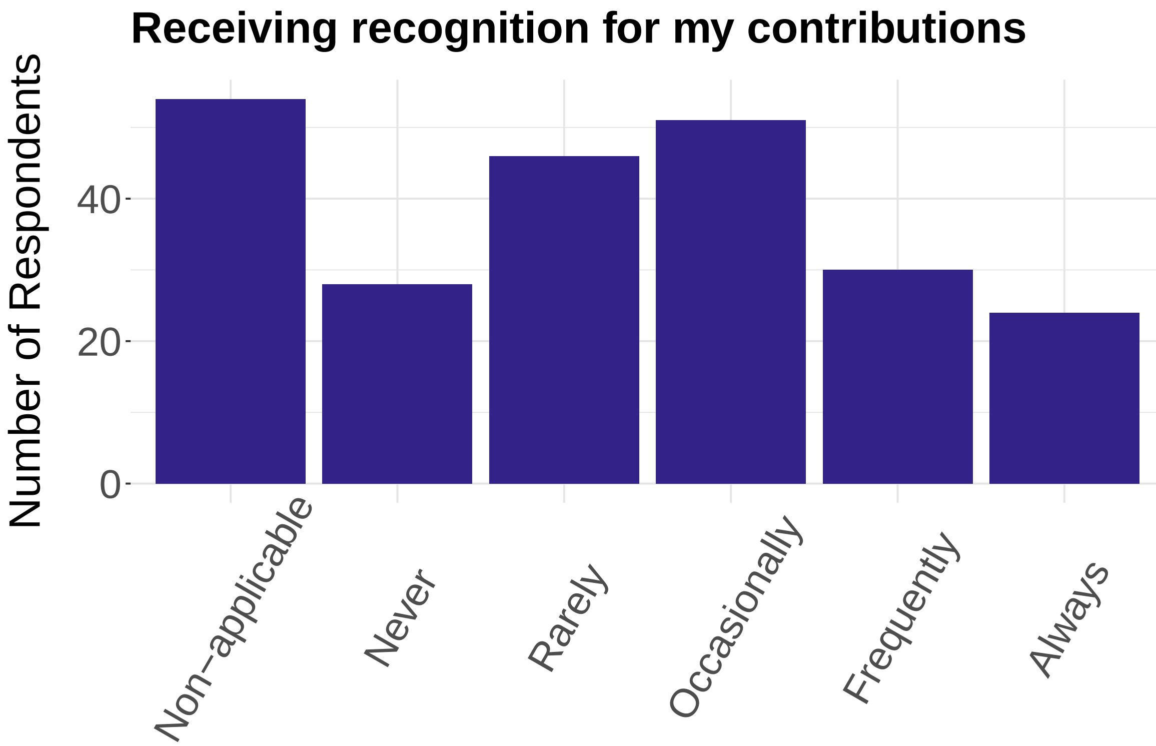


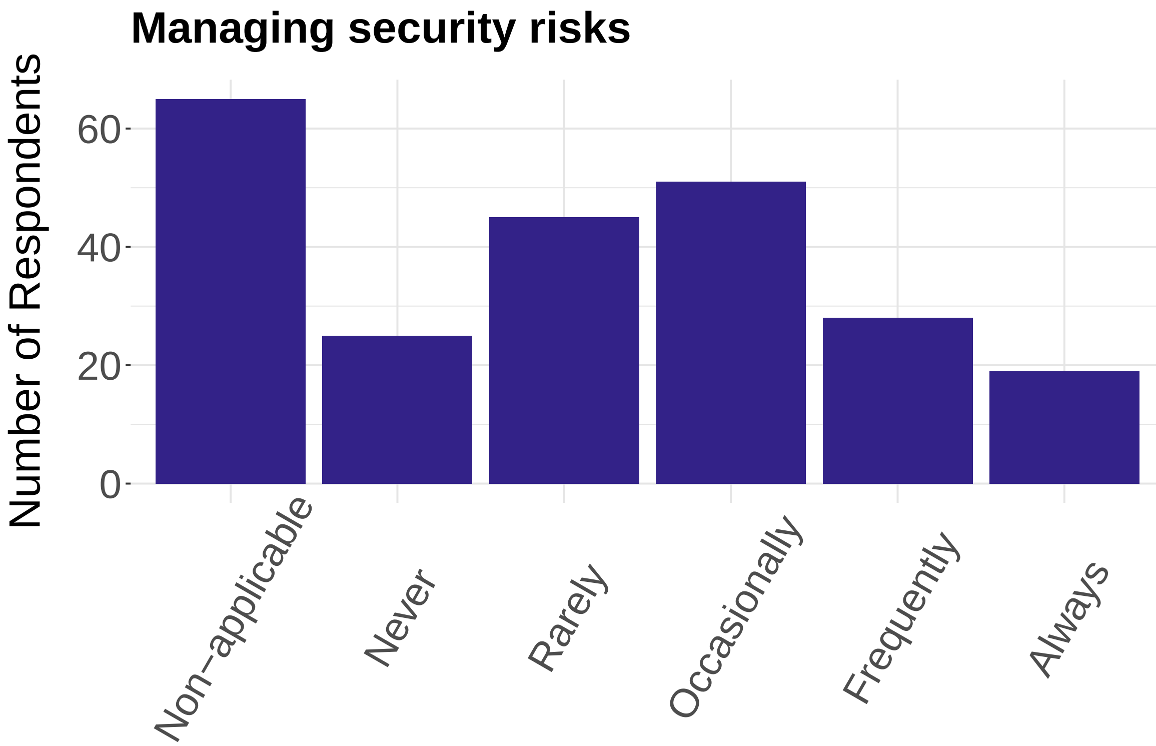
"na-skewed"

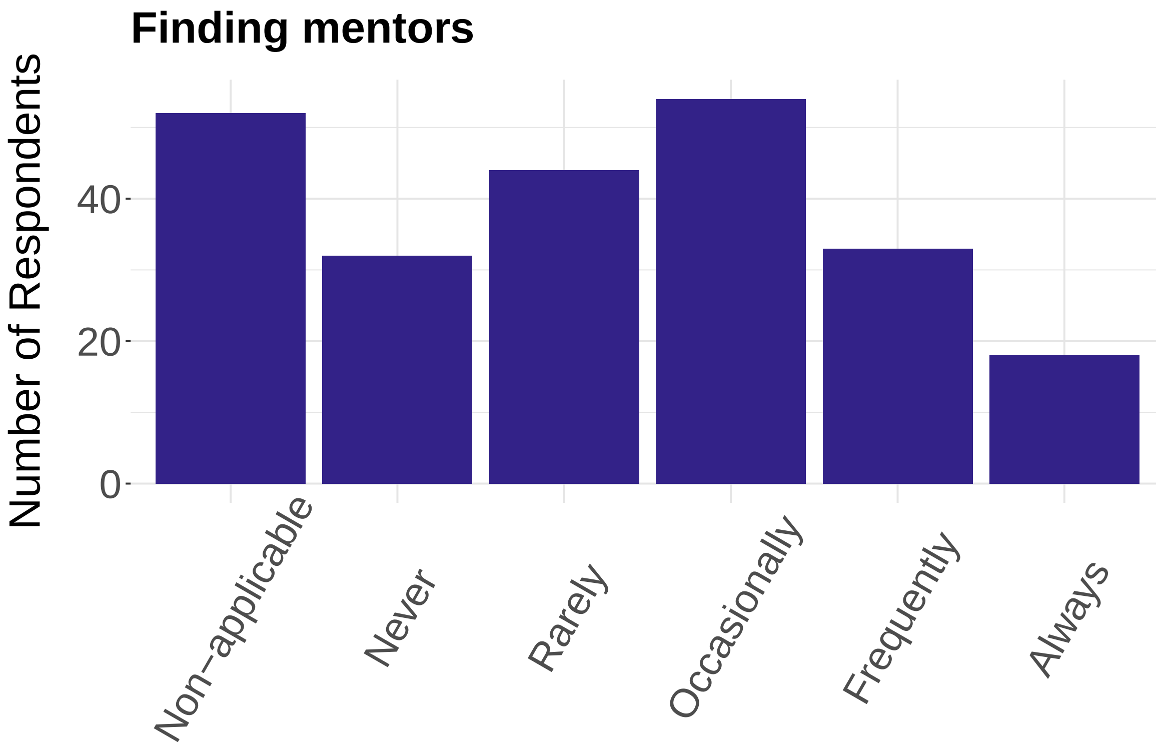
```
multiple_plots(to_plot, titles, na_skewed)
```


Managing issues and pull requests
Number of Respondents
0
20
40
60
Non–applicable
Never
Rarely
Occasionally
Frequently
Always


Attracting users and/or contributors
Number of Respondents
0
20
40
Non–applicable
Never
Rarely
Occasionally
Frequently
Always

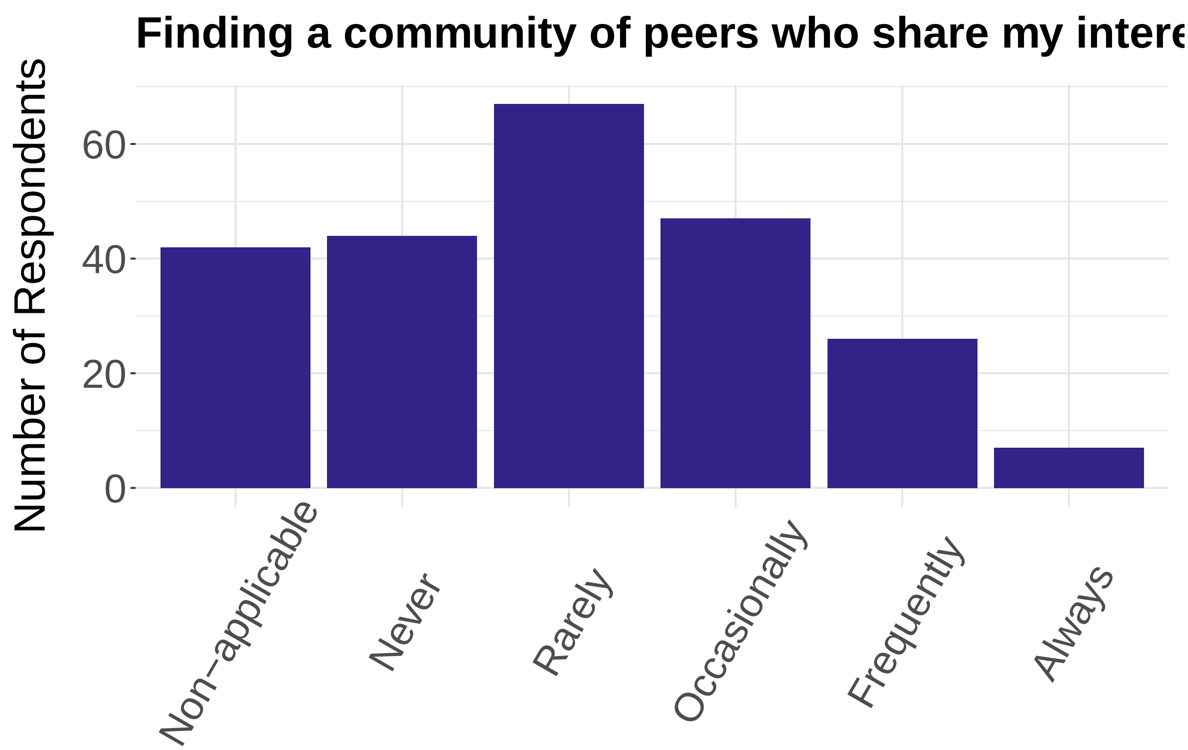

Receiving recognition for my contributions
Number of Respondents
0
20
40
Non–applicable
Never
Rarely
Occasionally
Frequently
Always


Managing security risks
Number of Respondents
0
20
40
60
Non–applicable
Never
Rarely
Occasionally
Frequently
Always



“left-skewed”

```
multiple_plots(to_plot, titles, left_skewed)
```



K-means clustering of distributions

This seems like an interesting line of inquiry. Let's make it a little more rigorous by clustering the challenges based on the response rates (actually, the absolute response numbers).

Wrangle data

```
wide_counts <- to_plot %>%
  pivot_wider(
    names_from    = challenge_level,
    values_from   = total,
    values_fill   = 0
  )

wide_counts <- data.frame(wide_counts)
# Turn this categorical column into row names
```

```r
rownames(wide_counts) <- wide_counts$challenge
wide_counts <- wide_counts[,2:(ncol(wide_counts))]
head(wide_counts)
```

```
                       Non.applicable Never Rarely Occasionally Frequently
Attracting users                   50    15     24           53         52
Coding time                        21     4     13           54         79
Documentation time                 12     2      7           36         97
Education time                     14    15     28           62         69
Educational resources              23    28     64           64         39
Finding funding                    74    16     16           25         42
                       Always
Attracting users           39
Coding time                62
Documentation time         79
Education time             45
Educational resources      15
Finding funding            60
```

```r
# Scaling probably isn't necessary?
# We have the same number of responses throughout,
# so the units for each challenge are the same
# (number of responses).
scaled <- scale(wide_counts)
scaled
```

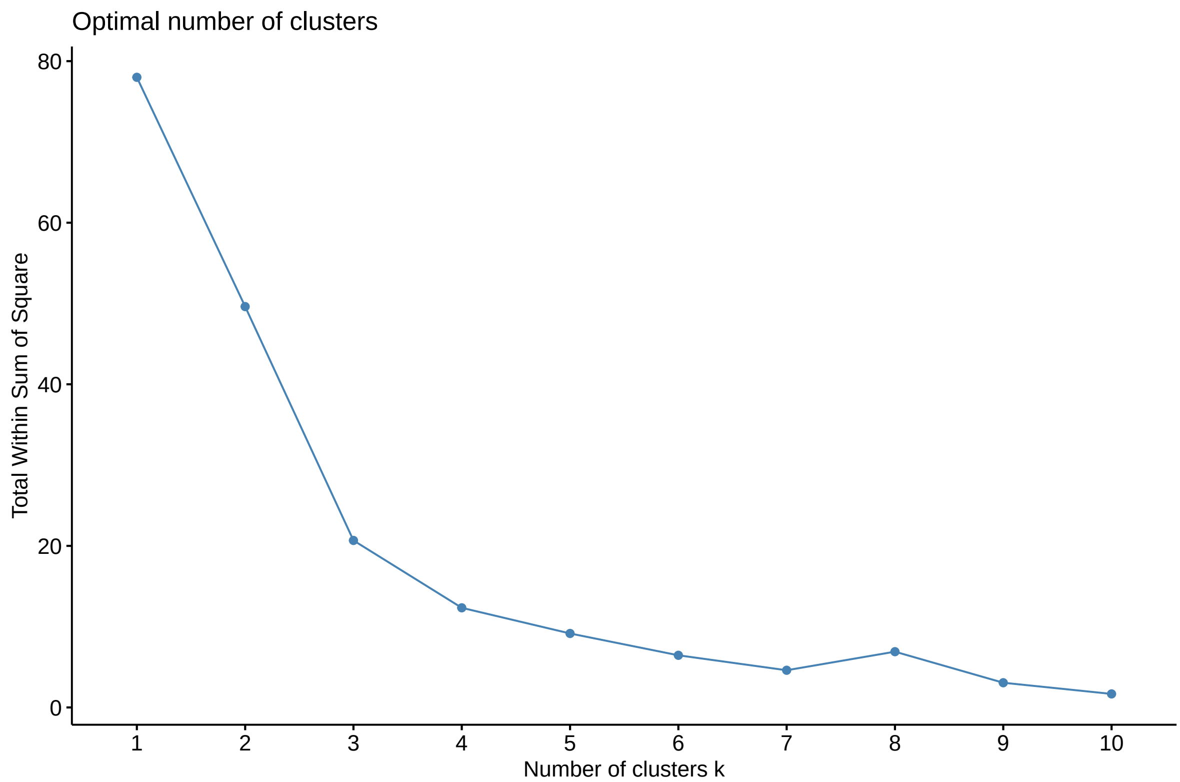
```
                       Non.applicable      Never     Rarely Occasionally
Attracting users           0.08407151 -0.4615675 -0.4780961   0.33347644
Coding time               -0.95026278 -1.4093196 -1.0002719   0.39743082
Documentation time        -1.27126308 -1.5816381 -1.2850951  -0.75374811
Education time            -1.19992968 -0.4615675 -0.2882139   0.90906590
Educational resources     -0.87892938  0.6585030  1.4207252   1.03697467
Finding funding            0.94007229 -0.3754083 -0.8578603  -1.45724635
Finding mentors            0.15540491  1.0031401  0.4713146   0.39743082
Finding peers             -0.20126209  2.0370514  1.5631368  -0.05024987
Hiring                     2.25974018 -0.3754083 -0.8578603  -1.58515512
Legal                     -0.48659569  1.0031401  1.4681957   0.65324836
Managing issues           -0.45092899 -0.6338861 -0.1932729   1.42070098
Recognition                0.22673830  0.6585030  0.5662556   0.20556767
Securing funding           1.15407249 -0.4615675 -1.0477424  -1.71306389
Security                   0.61907200  0.4000252  0.5187851   0.20556767
```

```
                      Frequently     Always
Attracting users       0.2440265  0.1098315
Coding time            1.4739199  1.1202809
Documentation time     2.2938488  1.8671349
Education time         1.0184038  0.3734270
Educational resources -0.3481444 -0.9445506
Finding funding       -0.2114896  1.0324157
Finding mentors       -0.6214541 -0.8127528
Finding peers         -0.9403153 -1.2960113
Hiring                -0.3481444 -0.3734270
Legal                 -1.0314186 -0.7248876
Managing issues        0.5628877 -0.4612921
Recognition           -0.7581089 -0.5491573
Securing funding      -0.4847993  1.4278090
Security              -0.8492121 -0.7688202
attr(,"scaled:center")
Non.applicable          Never         Rarely   Occasionally     Frequently
      47.64286       20.35714       34.07143       47.78571       46.64286
        Always
      36.50000
attr(,"scaled:scale")
Non.applicable          Never         Rarely   Occasionally     Frequently
      28.03736       11.60641       21.06570       15.63614       21.95312
        Always
      22.76215
```

Plot an elbow plot to find the point of diminishing returns.

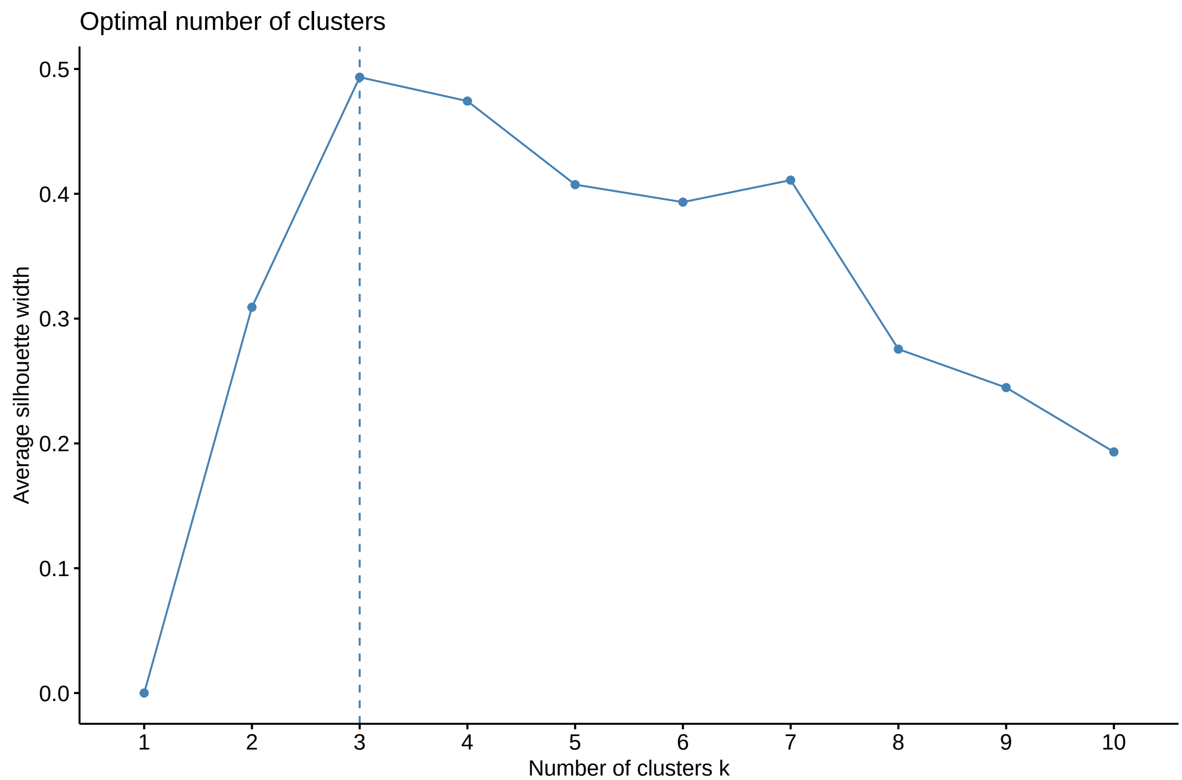
```r
factoextra::fviz_nbclust(scaled, kmeans, method = "wss")
```

```
Registered S3 methods overwritten by 'car':
  method       from
  hist.boot    FSA
  confint.boot FSA
```



I seem to get diminishing returns around k=4.

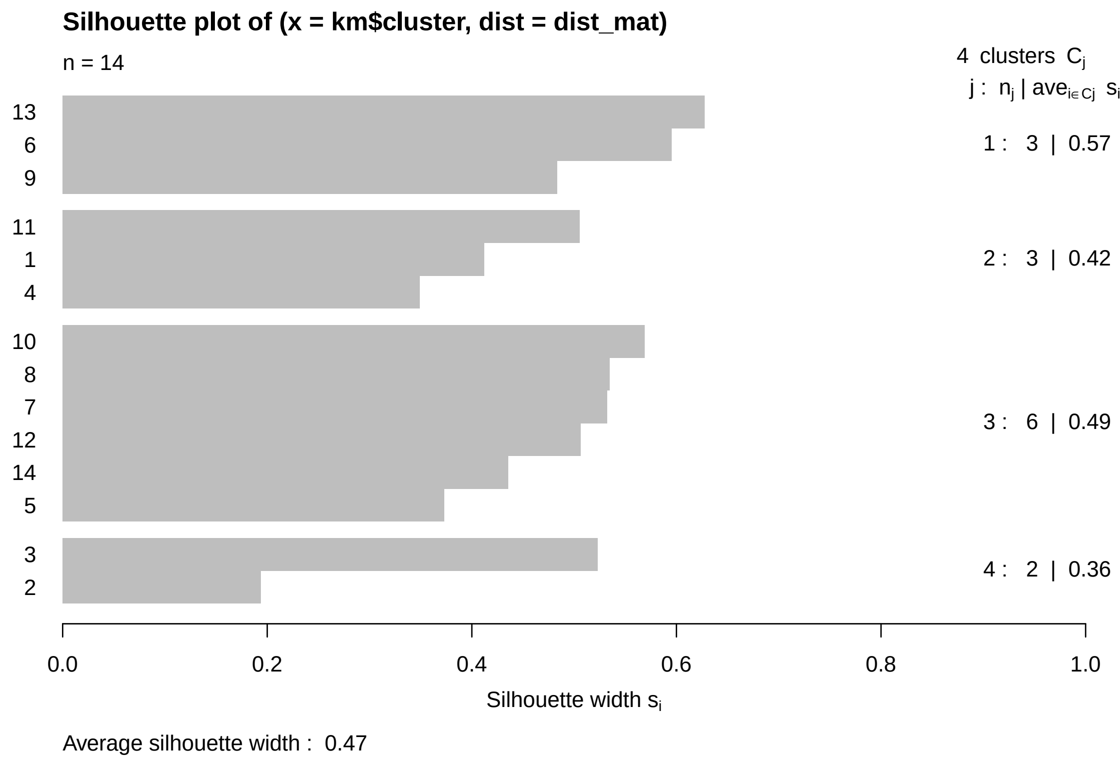
```
factoextra::fviz_nbclust(scaled, kmeans, method = "silhouette")
```



Hm. The silhouette plot indicates I should use k=3.

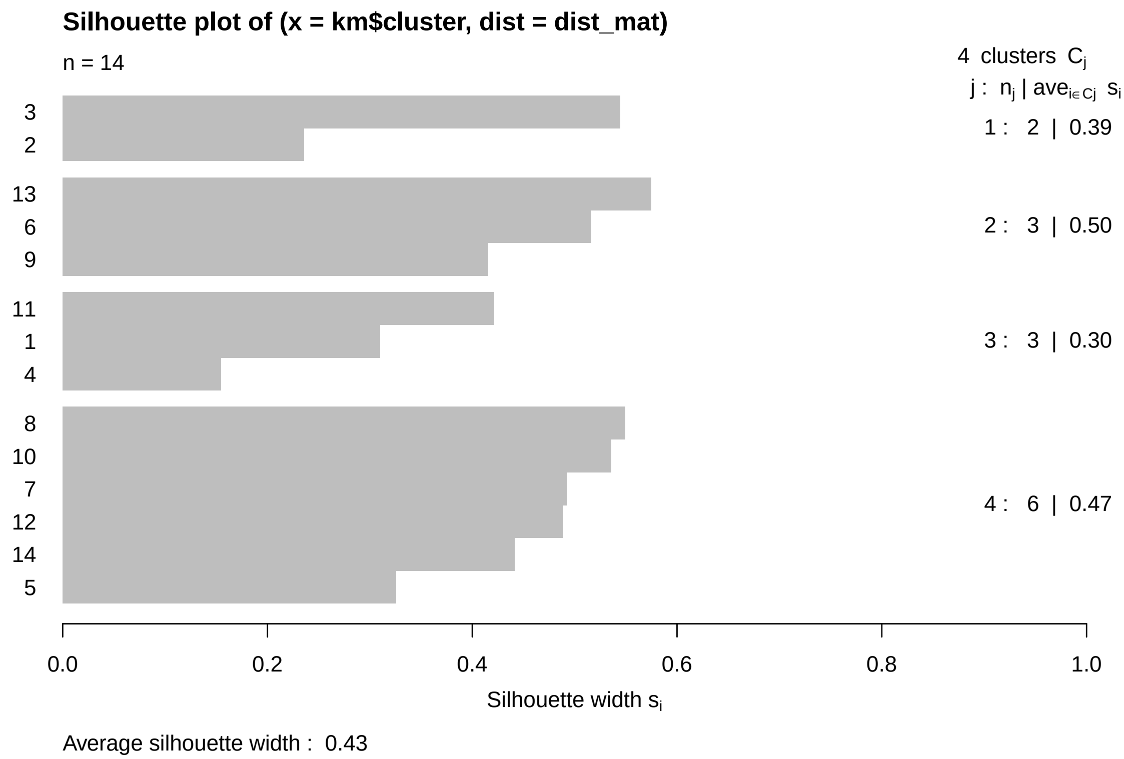
I think I'll try k=4 first, since it's closer to the number I got from eyeballing. Let's look at a different type of silhouette plot, which shows us the silhouette width of each cluster and on average across the clusters.

```
km <- stats::kmeans(scaled, centers = 4, nstart = 25)
dist_mat <- dist(scaled)
sil <- cluster::silhouette(km$cluster, dist_mat)
plot(sil)
```



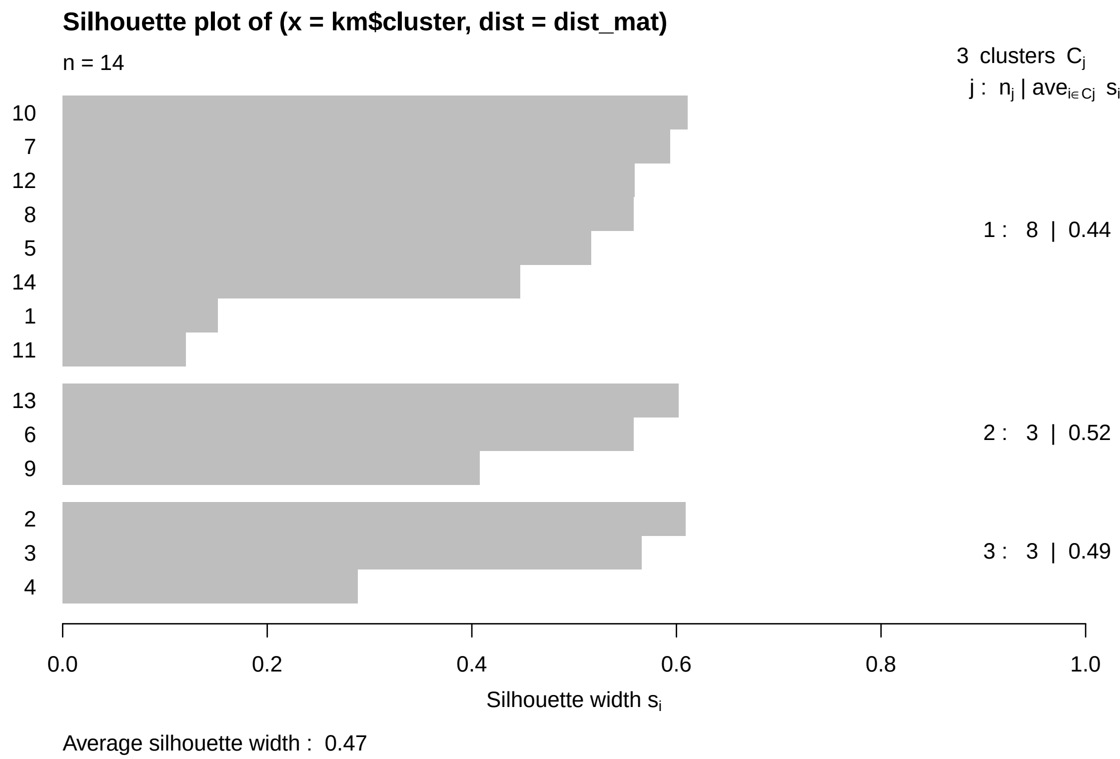
Hm. Looks… acceptable. Average silhouette width of 0.47. From Wikipedia: "A clustering with an average silhouette width of over 0.7 is considered to be"strong", a value over 0.5 "reasonable" and over 0.25 "weak"." Let's try unscaled data.

```
km <- stats::kmeans(wide_counts, centers = 4, nstart = 25)
dist_mat <- dist(wide_counts)
sil <- cluster::silhouette(km$cluster, dist_mat)
plot(sil)
```



Looks slightly worse: average silhouette width = 0.43. Still, I think we should probably stick with unscaled data because it's simpler, and I don't think we should add extra unnecessary procedures. What if we try 3 clusters?

```
km <- stats::kmeans(wide_counts, centers = 3, nstart = 25)
dist_mat <- dist(wide_counts)
sil <- cluster::silhouette(km$cluster, dist_mat)
plot(sil)
```



With an average silhouette width of 0.44-0.52, our clusters aren't looking amazing (unscaled data, 3 clusters). But they're not terrible, either. I prefer to use unscaled data with k=3, which results in an average silhouette width of 0.47. I think these results are consistent with my hunch that the data for the challenges are not all drawn from the same distribution. These are the cluster assignments:

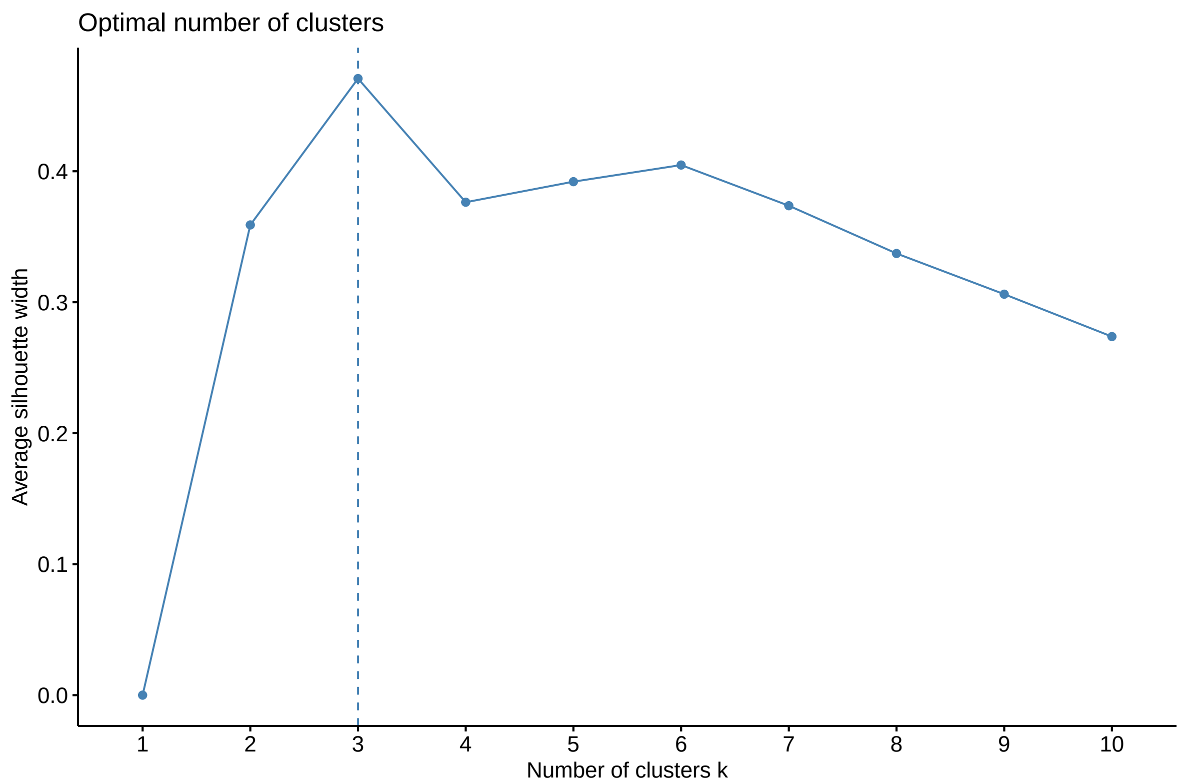
```
# A little extra code to achieve prettier printing
cluster_df <- data.frame(sort(km$cluster))
cluster_df$challenge <- rownames(cluster_df)
clusters <- unique(cluster_df[,1])
for (cl in clusters) {
  print(cluster_df[cluster_df[,1] == cl,], row.names = FALSE)
  cat("\n")
}
```

```
 sort.km.cluster.             challenge
                1       Attracting users
                1 Educational resources
                1        Finding mentors
                1          Finding peers
```

```
                1                  Legal
                1       Managing issues
                1           Recognition
                1              Security

sort.km.cluster.        challenge
                2  Finding funding
                2           Hiring
                2 Securing funding

sort.km.cluster.          challenge
                3        Coding time
                3 Documentation time
                3     Education time
```

Let's look at a silhouette plot for the PAM method, too.

```
factoextra::fviz_nbclust(wide_counts, FUNcluster = pam, method = "silhouette")
```



This also says that 3 clusters is ideal.

Let's try PAM clustering on the unscaled data with k=3.

```
pm <- cluster::pam(wide_counts, k=3)
```

Print the clusters in a more readable format.

```
cluster_df <- data.frame(sort(pm$cluster))
cluster_df$challenge <- rownames(cluster_df)
clusters <- unique(cluster_df[,1])
for (cl in clusters) {
  print(cluster_df[cluster_df[,1] == cl,], row.names = FALSE)
  cat("\n")
}
```

```
 sort.pm.cluster.              challenge
                1       Attracting users
                1 Educational resources
                1        Finding mentors
                1          Finding peers
                1                  Legal
                1        Managing issues
                1            Recognition
                1               Security

 sort.pm.cluster.           challenge
                2         Coding time
                2 Documentation time
                2      Education time

 sort.pm.cluster.         challenge
                3  Finding funding
                3           Hiring
                3 Securing funding
```

We see the same groups we saw with k-means clustering. Good!

One last check: what about a stability assessment by bootstrap resampling?

```
# Note I'm hiding the printed status update from each iteration
boot_res <- fpc::clusterboot(
  wide_counts,
  clustermethod = fpc::kmeansCBI,
```

```r
  krange = 3
)
# Annoyingly, the documentation doesn't explain 'krange',
# but I'm pretty sure that this argument lets you specify
# a desired k or range of k values (e.g. 5:7)
```

```r
boot_res
```

```
* Cluster stability assessment *
Cluster method:  kmeans
Full clustering results are given as parameter result
of the clusterboot object, which also provides further statistics
of the resampling results.
Number of resampling runs:  100

Number of clusters found in data:  3

 Clusterwise Jaccard bootstrap (omitting multiple points) mean:
[1] 0.7761190 0.8803333 0.8630714
dissolved:
[1] 30 16  1
recovered:
[1] 61 80 71
```

```r
mean(boot_res$bootmean)
```

```
[1] 0.8398413
```

The clusterwise Jaccard bootstrap means are around 0.8-0.9, which is pretty respectable. Although this analysis was brief, I think we can conclude that these three clusters are reasonably stable and meaningful.

Stacked bar plots

A request from Amber: how about color-coded stacked bar plots, instead of the bar charts I made above?

```r
# Rename this column, poorly named by the cluster package
# From `sort.km.cluster.` to sort_km_cluster
names(cluster_df)[1] <- "sort_km_cluster"

temp <- to_plot
temp$challenge_level <- factor(temp$challenge_level, levels = rev(ordered_levels))

data_cluster1 <- temp %>%
  filter(
      challenge %in% rownames(subset(cluster_df, sort_km_cluster == 1)
    )
  )

data_cluster2 <- temp %>%
  filter(
      challenge %in% rownames(subset(cluster_df, sort_km_cluster == 2)
    )
  )

data_cluster3 <- temp %>%
  filter(
      challenge %in% rownames(subset(cluster_df, sort_km_cluster == 3)
    )
  )
```

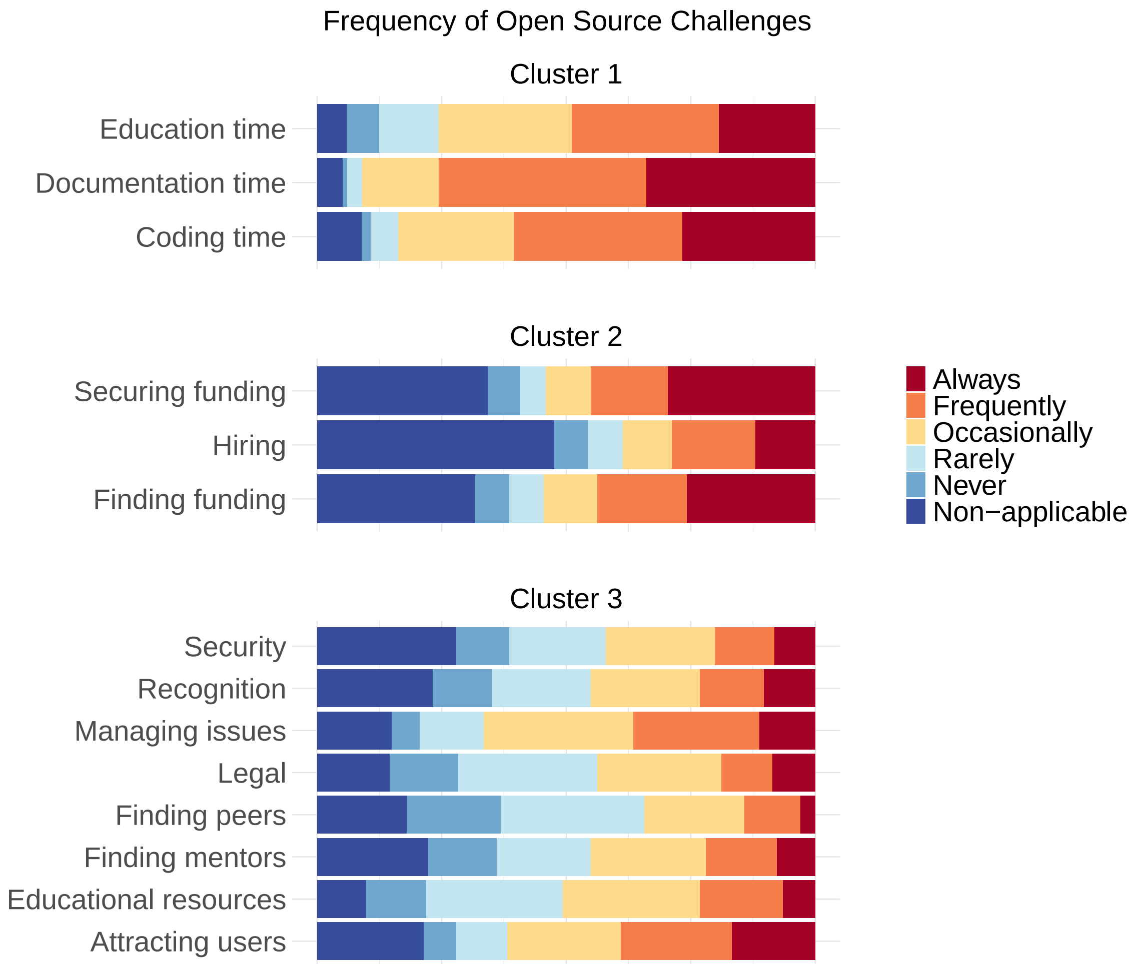
WARNING! I am changing the cluster names ONLY on the plot, not in the rest of the code. The plot looks better when the two small clusters–clusters 1 and 3–are next to each other (we want to visually compare them, and it's easier to do so when they're adjacent). However, the plot looks dumb when I have them in order 1, 3, 2. It looks like I made a mistake. So I am renaming the plot panel titles ONLY.

```r
# Paul Tol's sunset theme
#https://sronpersonalpages.nl/~pault/
sunset <- c(
  "#A50026",
  "#F67E4B",
  "#FEDA8B",
  "#C2E4EF",
  "#6EA6CD",
  "#364B9A"
)
```

```
p1 <- stacked_bar_chart(
  df = data_cluster2,
  x_var = "challenge",
  y_var = "total",
  fill = "challenge_level",
  title = "Cluster 1", # NAME CHANGE, SEE TEXT ABOVE
  ylabel = NULL,
  show_axis_title_y = FALSE,
  show_x_axis_text = FALSE,
  show_grid = TRUE,
  show_legend = FALSE, # don't show legend
  horizontal = TRUE,
  proportional = TRUE,
  cpalette = sunset
)

p2 <- stacked_bar_chart(
  df = data_cluster3,
  x_var = "challenge",
  y_var = "total",
  fill = "challenge_level",
  title = "Cluster 2", # NAME CHANGE, SEE TEXT ABOVE
  ylabel = NULL,
  legend_left_margin = 45, # show legend, with a wide margin
  show_axis_title_y = FALSE,
  show_x_axis_text = FALSE,
  show_grid = TRUE,
  horizontal = TRUE,
  proportional = TRUE,
  cpalette = sunset
)

p3 <- stacked_bar_chart(
  df = data_cluster1,
  x_var = "challenge",
  y_var = "total",
  fill = "challenge_level",
  title = "Cluster 3", # NAME CHANGE, SEE TEXT ABOVE
  ylabel = NULL,
  show_axis_title_y = FALSE,
  show_x_axis_text = FALSE,
  show_grid = TRUE,
```

```r
  show_legend = FALSE, # don't show legend
  horizontal = TRUE,
  proportional = TRUE,
  cpalette = sunset
)
```

```r
combined <- patchwork::wrap_plots(p1, p2, p3, ncol = 1) +
  patchwork::plot_layout(heights = c(1, 1, 2)) +
  patchwork::plot_annotation(
    title = "Frequency of Open Source Challenges",
    theme = theme(plot.title = element_text(
      size = 24,
      margin = margin(t = 15),
      hjust = 0.5)
      )
  )
combined
```



Actually, I think the cluster assignments might be stochastic? I.e., they are labeled randomly each time I run this script? I am too lazy to make sure that e.g. the cluster that contains "Documentation time" is always cluster #1. In fact, I think it would be poor practice to rename an intermediate data structure for the sake of the plot. I'd rather just change the plot titles. My solution is, I am just going to save the plot that I like, and not re-create it every time I run this script. Yes, this is less reproducible, but I'm not going to spend my time fixing it because it's a purely cosmetic problem that will only come up if someone tries to recreate my figures, and doesn't affect the underlying data.

```
#save_plot("challenge_stacks.tiff", 14, 12, p=combined)
```

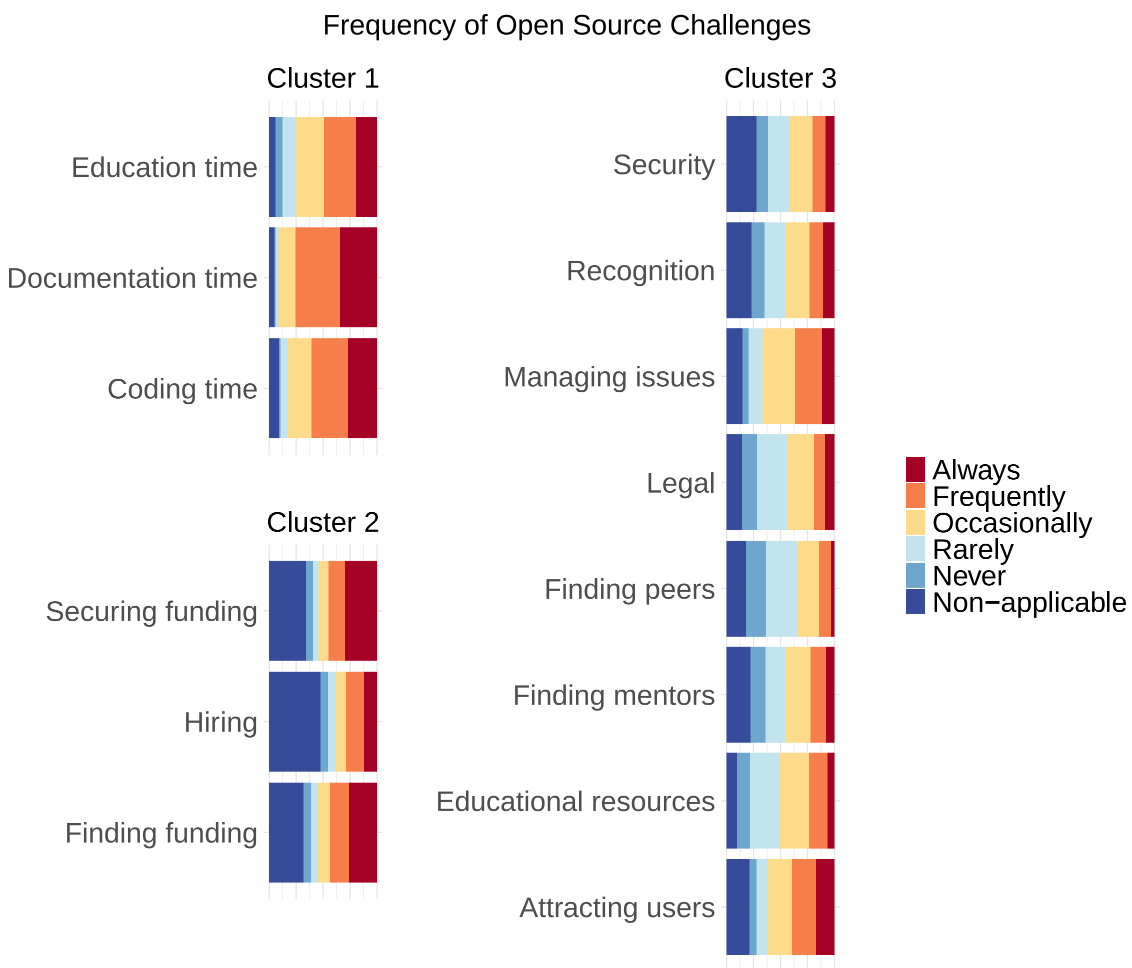
How does it look if we arrange these plots horizontally? We'll need to put the lenged on the right-most plot in this case.

```
p2_hor <- stacked_bar_chart(
  df = data_cluster3,
  x_var = "challenge",
  y_var = "total",
  fill = "challenge_level",
  title = "Cluster 2", # NAME CHANGE, SEE TEXT ABOVE
  ylabel = NULL,
  show_legend = FALSE, # don't show legend
  show_axis_title_y = FALSE,
  show_x_axis_text = FALSE,
  show_grid = TRUE,
  horizontal = TRUE,
  proportional = TRUE,
  cpalette = sunset
)

p3_hor <- stacked_bar_chart(
  df = data_cluster1,
  x_var = "challenge",
  y_var = "total",
  fill = "challenge_level",
  title = "Cluster 3", # NAME CHANGE, SEE TEXT ABOVE
  ylabel = NULL,
  show_axis_title_y = FALSE,
  show_x_axis_text = FALSE,
  show_grid = TRUE,
  legend_left_margin = 45, # show legend, with a wide margin
  horizontal = TRUE,
  proportional = TRUE,
  cpalette = sunset
)
```

Tried plotting horizontally a couple different ways. I like the second way better, for now.

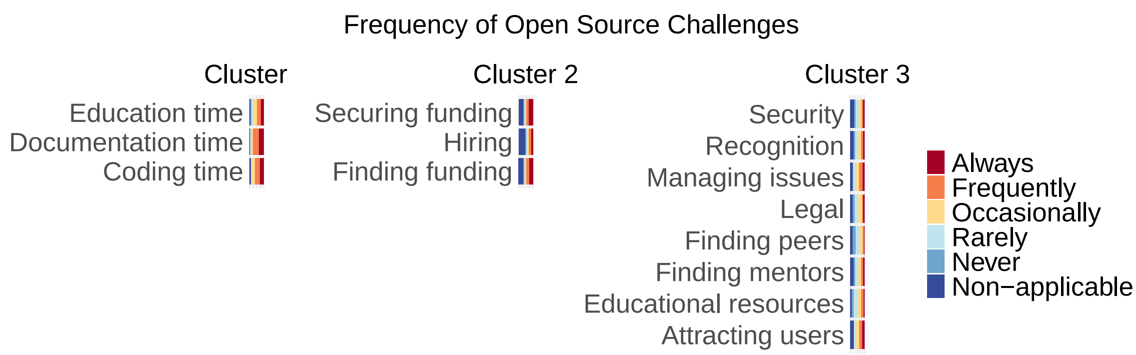
```
# left column: p1 over p2; right column: p3 spanning full height
combined_hor <- patchwork::wrap_plots(
  (p1 / p2_hor / patchwork::plot_spacer()) + plot_layout(heights = c(1,1,0.1)) | p3_hor
) +
  #patchwork::plot_layout(heights = c(1, 1, 1)) +
  patchwork::plot_annotation(
    title = "Frequency of Open Source Challenges",
    theme = theme(
```

```
      plot.title = element_text(
        size = 24,
        margin = margin(t = 15),
        hjust = 0.5
      )
    )
  )
combined_hor
```




```
# three plots in a row
combined_hor <- patchwork::wrap_plots(
  (p1 / patchwork::plot_spacer() ) + plot_layout(heights = c(1,1.4)) |
  (p2_hor / patchwork::plot_spacer() ) + plot_layout(heights = c(1,1.4)) |
```

```
  p3_hor
) +
  #patchwork::plot_layout(heights = c(1, 1, 1)) +
  patchwork::plot_annotation(
    title = "Frequency of Open Source Challenges",
    theme = theme(
      plot.title = element_text(
        size = 24,
        margin = margin(t = 15),
        hjust = 0.5
      )
    )
  )
combined_hor
```




```
save_plot("challenge_stacks_horizontal.tiff", 36, 8, p=combined_hor)
```

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.6.1

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     grid      stats     graphics  grDevices datasets  utils
[8] methods   base

other attached packages:
 [1] treemapify_2.5.6      tidyr_1.3.1           svglite_2.2.1
 [4] stringr_1.5.1         scales_1.4.0          readr_2.1.5
 [7] pwr_1.3-0             patchwork_1.3.2       ordinal_2023.12-4.1
[10] lme4_1.1-37           Matrix_1.7-1          languageserver_0.3.16
[13] here_1.0.1            gtools_3.9.5          ggforce_0.5.0
[16] FSA_0.10.0            fpc_2.2-13            forcats_1.0.0
[19] factoextra_1.0.7      ggplot2_3.5.2         emmeans_1.11.2
[22] dplyr_1.1.4           corrplot_0.95         ComplexHeatmap_2.22.0
[25] cluster_2.1.8.1       BiocManager_1.30.26

loaded via a namespace (and not attached):
 [1] Rdpack_2.6.4        rlang_1.1.6         magrittr_2.0.3
 [4] clue_0.3-66         GetoptLong_1.0.5    matrixStats_1.5.0
 [7] compiler_4.4.2      flexmix_2.3-20      systemfonts_1.2.3
[10] png_0.1-8           callr_3.7.6         vctrs_0.6.5
[13] pkgconfig_2.0.3     shape_1.4.6.1       crayon_1.5.3
[16] fastmap_1.2.0       backports_1.5.0     labeling_0.4.3
[19] utf8_1.2.6          rmarkdown_2.29      ggfittext_0.10.2
[22] tzdb_0.5.0          ps_1.9.1            nloptr_2.2.1
[25] purrr_1.1.0         xfun_0.53           modeltools_0.2-24
[28] jsonlite_2.0.0      tweenr_2.0.3        broom_1.0.9
[31] parallel_4.4.2      prabclus_2.3-4      R6_2.6.1
[34] stringi_1.8.7       RColorBrewer_1.1-3  car_3.1-3
[37] boot_1.3-31         diptest_0.77-2      numDeriv_2016.8-1.1
[40] estimability_1.5.1  Rcpp_1.1.0          iterators_1.0.14
[43] knitr_1.50          IRanges_2.40.1      splines_4.4.2
[46] nnet_7.3-19         tidyselect_1.2.1    abind_1.4-8
[49] yaml_2.3.10         doParallel_1.0.17   codetools_0.2-20
[52] processx_3.8.6      lattice_0.22-6      tibble_3.3.0
[55] withr_3.0.2         evaluate_1.0.4      polyclip_1.10-7
[58] xml2_1.4.0          circlize_0.4.16     mclust_6.1.1
[61] kernlab_0.9-33      ggpubr_0.6.1        pillar_1.11.0
[64] carData_3.0-5       renv_1.1.5          foreach_1.5.2
[67] stats4_4.4.2        reformulas_0.4.1    generics_0.1.4
```

```
[70] rprojroot_2.1.1     S4Vectors_0.44.0    hms_1.1.3
[73] minqa_1.2.8         xtable_1.8-4        class_7.3-22
[76] glue_1.8.0          robustbase_0.99-4-1 ggsignif_0.6.4
[79] mvtnorm_1.3-3       rbibutils_2.3       colorspace_2.1-1
[82] nlme_3.1-166        Formula_1.2-5       cli_3.6.5
[85] textshaping_1.0.1   gtable_0.3.6        DEoptimR_1.1-4
[88] rstatix_0.7.2       digest_0.6.37       BiocGenerics_0.52.0
[91] ucminf_1.2.2        ggrepel_0.9.6       rjson_0.2.23
[94] farver_2.1.2        htmltools_0.5.8.1   lifecycle_1.0.4
[97] GlobalOptions_0.1.2 MASS_7.3-61
```