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 correspond to 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.

```
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

```
data <- load_qualtrics_data("deidentified_no_qual.tsv")
```

Wrangle data

```
challenges <- data %>%
  select(
    starts_with("challenges")
  )

head(challenges)
```

```
  challenges_1 challenges_2 challenges_3 challenges_4 challenges_5 challenges_6
1       Always       Always       Always       Always       Always       Always
2   Frequently Occasionally Occasionally Occasionally Occasionally       Rarely
3   Frequently       Always Occasionally       Always Occasionally   Frequently
4       Always       Always   Frequently Occasionally   Frequently       Always
5       Always       Always       Rarely Occasionally   Frequently        Never
6
  challenges_7 challenges_8 challenges_9 challenges_10 challenges_11
1       Always       Always       Always        Always        Always
2   Frequently Occasionally   Frequently    Frequently    Frequently
3   Frequently Occasionally Occasionally        Rarely        Rarely
4 Occasionally       Rarely       Rarely    Frequently        Rarely
5        Never        Never        Never        Always  Occasionally
6
  challenges_12 challenges_13 challenges_14
1        Always        Always        Always
2    Frequently    Frequently  Occasionally
3        Always        Always        Always
4  Occasionally    Frequently    Frequently
5  Occasionally        Rarely        Always
6
```

STOP!! Presumably, “challenges_1” corresponds to the first option, “challenges_2” corresponds to the second option, etc., but we still need to check. I am manually comparing the answers in this data frame to those in the Qualtrics interface, which shows the whole response, i.e. “Limited time for writing new code”, not just “challenges_1”. To be extra confident that I am comparing the same rows between the two tables, I am looking at responses associated with a particular email. After this code chunk, I go back to using the data frame that doesn’t contain the emails.

Since this code only needed to be run once, I’ve commented it out.

```
# pii <- load_qualtrics_data("pii.tsv")
# emails <- pii %>%
#     select(starts_with("stay_in_touch_email"))

# t <- cbind(emails, challenges)

# # Next, I run this line repeatedly with different emails,
# # to make sure that this person's response to "challenges_1"
# # matches their response to "Limited time for writing new code", etc.
# subset(t, startsWith(stay_in_touch_email, "PERSON_EMAIL_HERE"))
```

My assumption above was correct; the options are ordered as expected. Let's rename the columns accordingly.

```r
challenge_codes <- c(
  "Coding time" = "challenges_1",
  "Documentation time" = "challenges_2",
  "Managing issues" = "challenges_3",
  "Attracting users" = "challenges_4",
  "Recognition" = "challenges_5",
  "Hiring" = "challenges_6",
  "Security" = "challenges_7",
  "Finding peers" = "challenges_8",
  "Finding mentors" = "challenges_9",
  "Education time" = "challenges_10",
  "Educational resources" = "challenges_11",
  "Legal" = "challenges_12",
  "Finding funding" = "challenges_13",
  "Securing funding" = "challenges_14"
)
challenges <- rename(challenges, any_of(challenge_codes))
```

Next, 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.

```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 ("Never" = 0, "Non-applicable" = 0,"Rarely" = 1, "Occasionally" = 2, "Frequently" = 3, "Always" = 4) * 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 don't know.) * 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),
    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 5
   challenge              total  mean  mode st_dev
   <chr>                  <int> <dbl> <int>  <dbl>
 1 Documentation time       686  2.94     3   1.08
 2 Coding time              606  2.60     3   1.24
 3 Education time           539  2.31     3   1.26
 4 Managing issues          451  1.94     2   1.29
 5 Attracting users         442  1.90     0   1.45
 6 Securing funding         438  1.88     0   1.74
 7 Finding funding          432  1.85     0   1.68
 8 Educational resources    369  1.58     1   1.19
 9 Recognition              334  1.43     0   1.35
10 Legal                    333  1.43     0   1.24
11 Finding mentors          323  1.39     0   1.31
12 Security                 307  1.32     0   1.31
13 Hiring                   291  1.25     0   1.53
14 Finding peers            267  1.15     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”).

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

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

sd_df
```

```
# A tibble: 14 x 5
   challenge              total  mean  mode st_dev
   <chr>                  <int> <dbl> <int>  <dbl>
 1 Securing funding         438  1.88     0   1.74
 2 Finding funding          432  1.85     0   1.68
 3 Hiring                   291  1.25     0   1.53
 4 Attracting users         442  1.90     0   1.45
 5 Recognition              334  1.43     0   1.35
 6 Security                 307  1.32     0   1.31
 7 Finding mentors          323  1.39     0   1.31
 8 Managing issues          451  1.94     2   1.29
 9 Education time           539  2.31     3   1.26
10 Legal                    333  1.43     0   1.24
11 Coding time              606  2.60     3   1.24
12 Educational resources    369  1.58     1   1.19
13 Finding peers            267  1.15     0   1.13
14 Documentation time       686  2.94     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

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

to_plot <- long_data %>%
  mutate(challenge_level = factor(challenge_level, levels = ordered_levels)) %>%
  count(
    challenge,
    challenge_level,
    name = "total"
  ) %>%
  ungroup()

to_plot
```

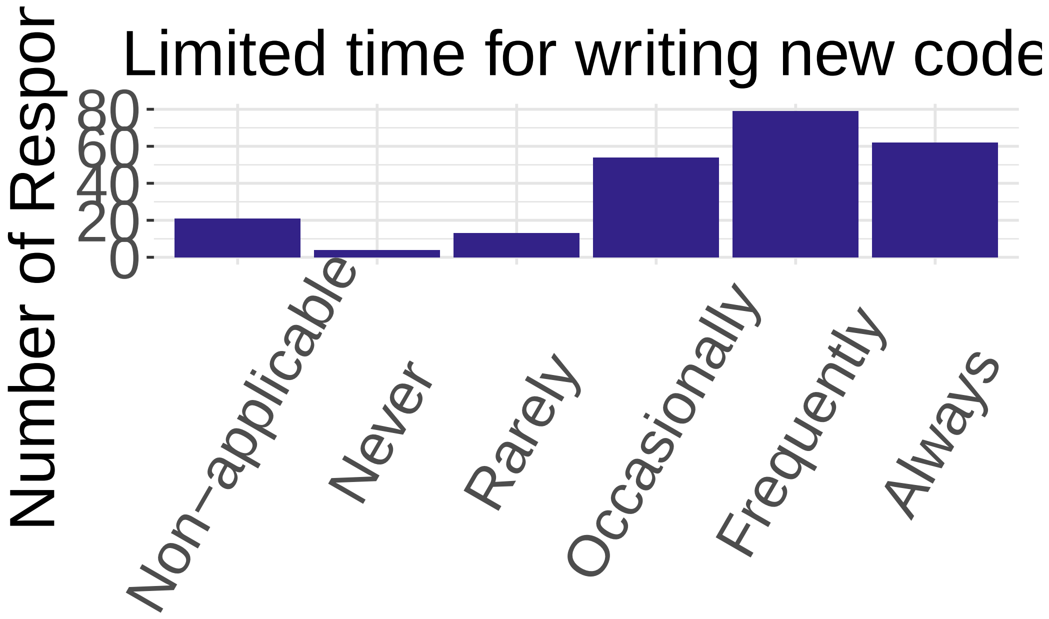
```
# A tibble: 84 x 3
   challenge        challenge_level total
   <chr>            <fct>           <int>
 1 Attracting users Non-applicable     50
 2 Attracting users Never              15
 3 Attracting users Rarely             24
 4 Attracting users Occasionally       53
 5 Attracting users Frequently         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
```

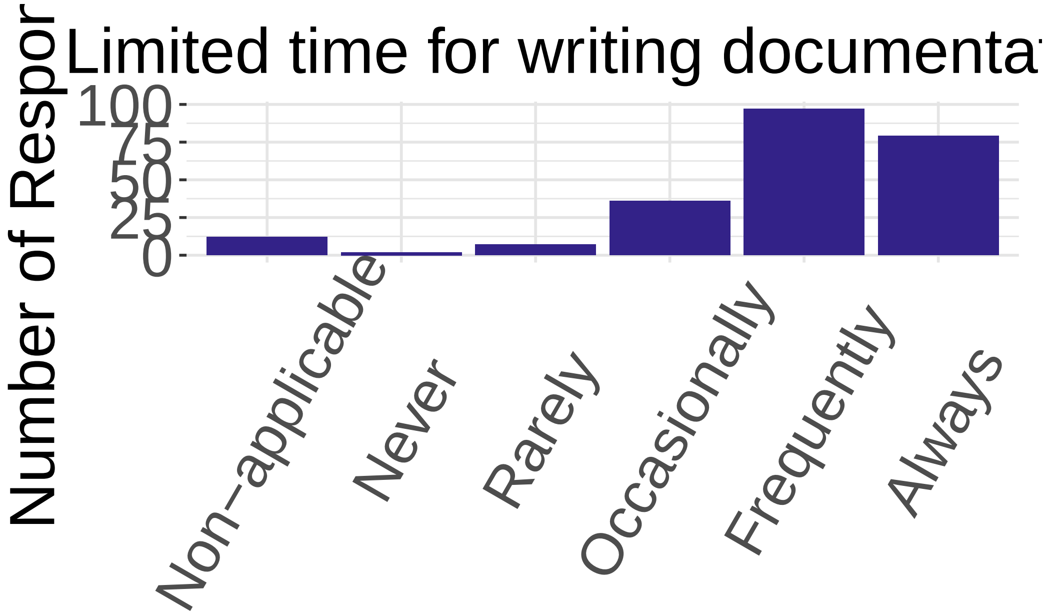
Create a plot for each "challenge". After inspecting the plots, I attempted to order them into groups based on the shape of their distribution. These are the shapes I observed (this is extremely subjective): * Right-skewed: Documentation time, coding time, education time * Interpretation: Common tasks 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.

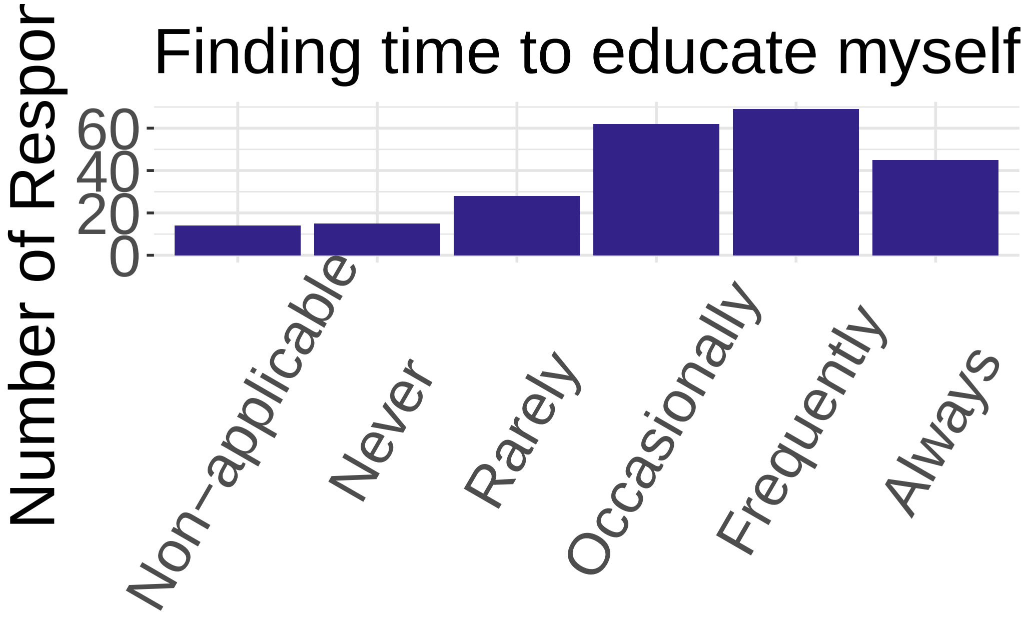
```
titles <- list(
    "Coding time" = "Limited time for writing new code",
    "Documentation time" = "Limited time for writing documentation",
    "Managing issues" = "Managing issues and pull requests",
    "Attracting users" = "Attracting users and/or contributors",
    "Recognition" = "Receiving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

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

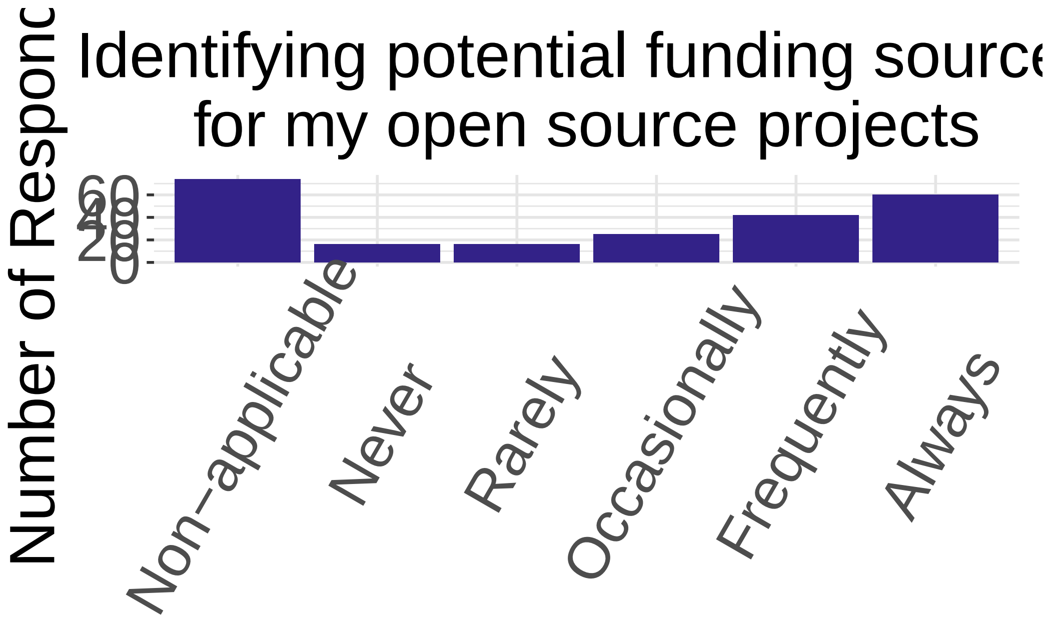


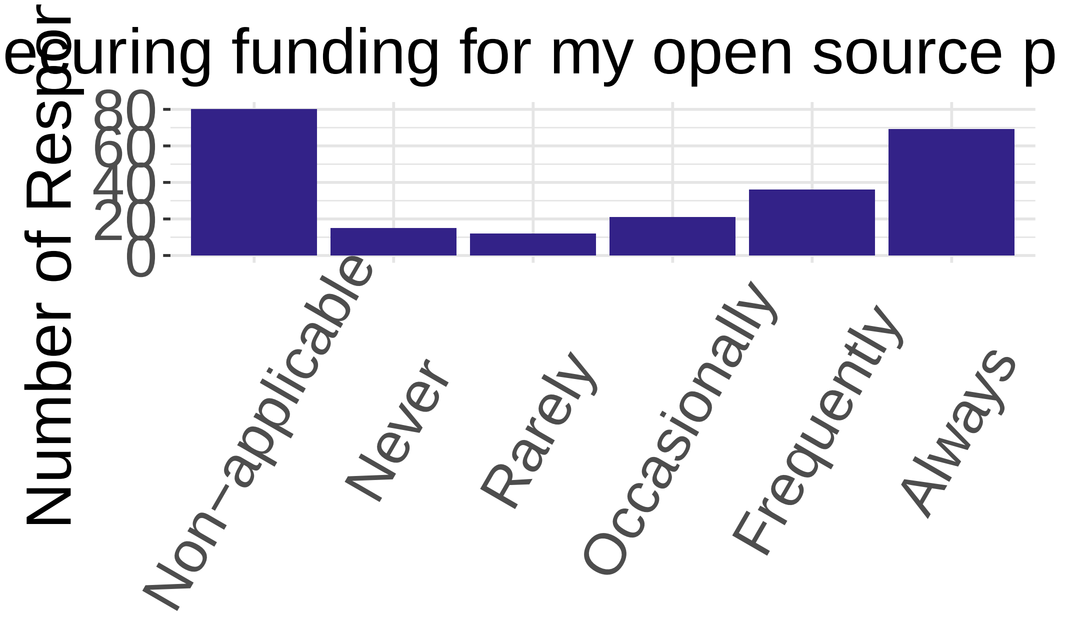


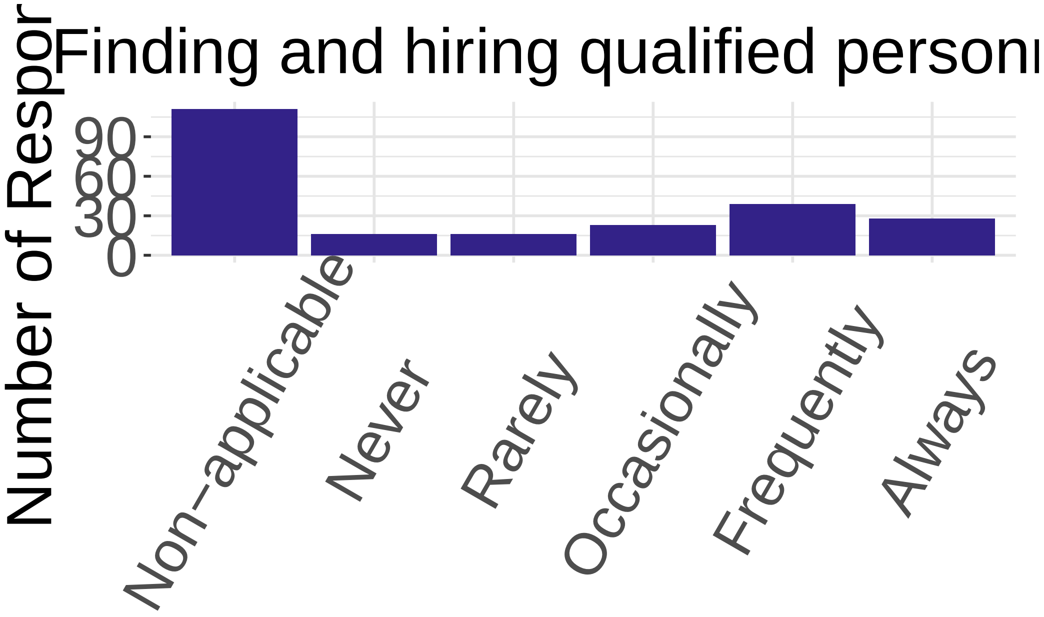


"highly bimodal"

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

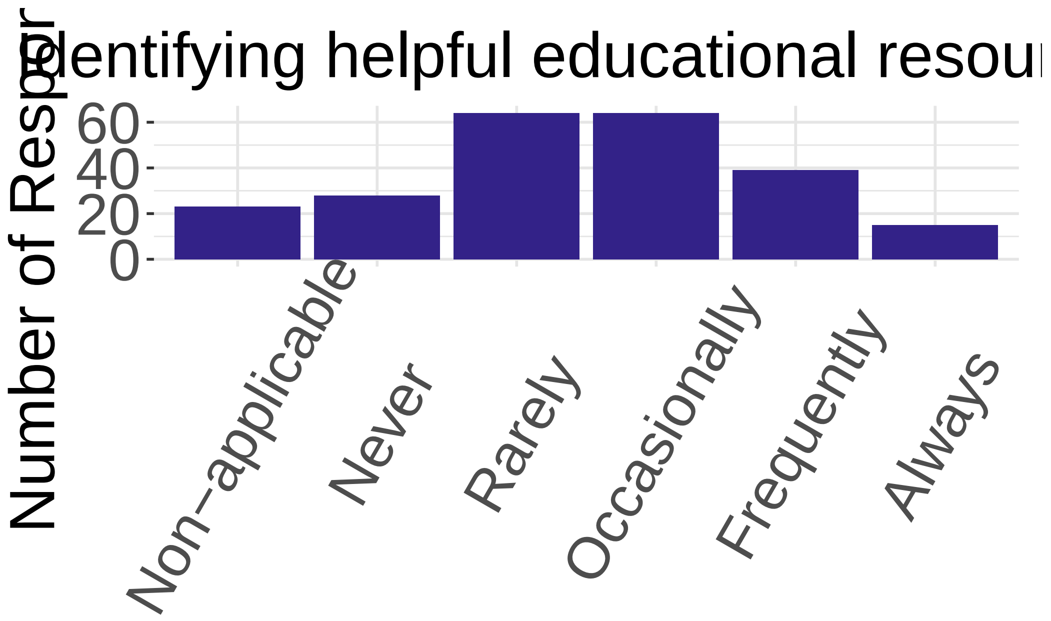

Identifying potential funding source
for my open source projects
Number of Respon
60
40
20
0
Non−applicable
Never
Rarely
Occasionally
Frequently
Always

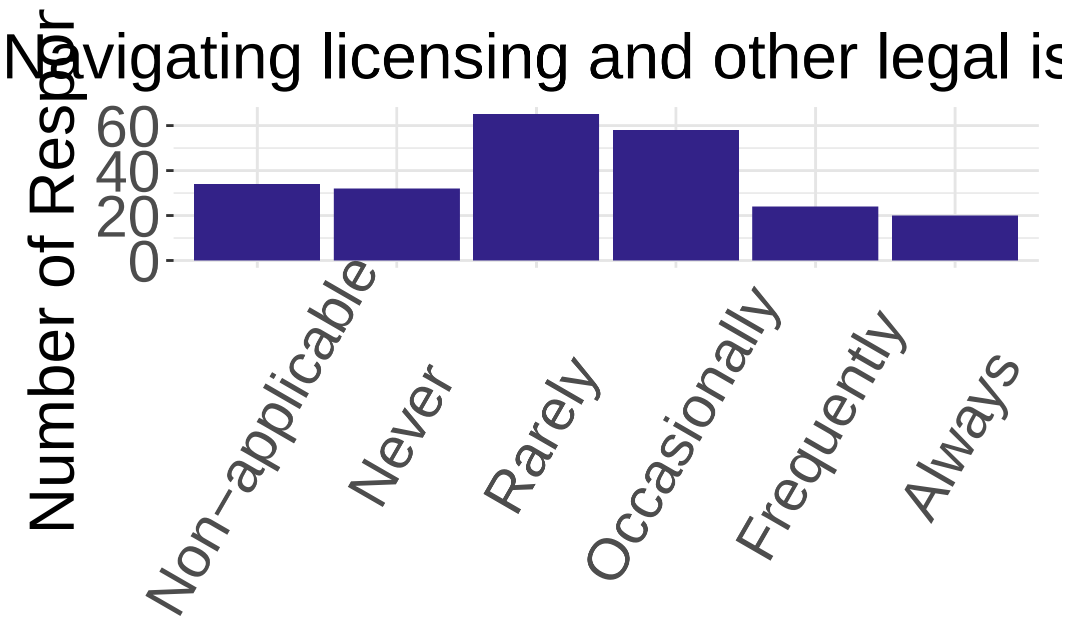

ecuring funding for my open source p
Number of Respor
80
60
40
20
0
Non−applicable
Never
Rarely
Occasionally
Frequently
Always



"normal"

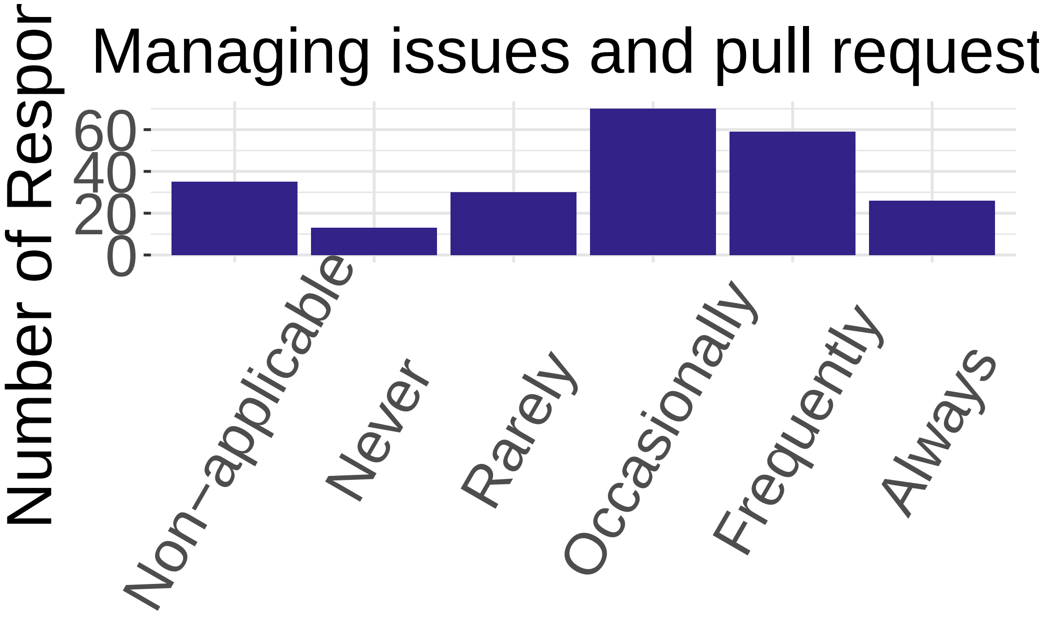
```
multiple_plots(to_plot, titles, normal)
```

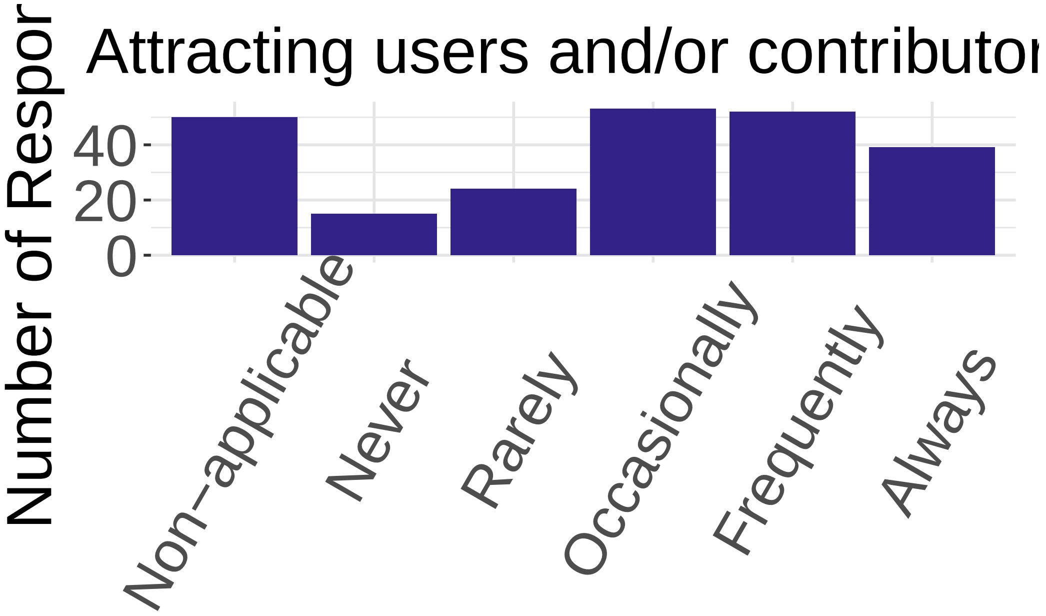

Identifying helpful educational resour
Number of Respor
60
40
20
0
Non−applicable
Never
Rarely
Occasionally
Frequently
Always

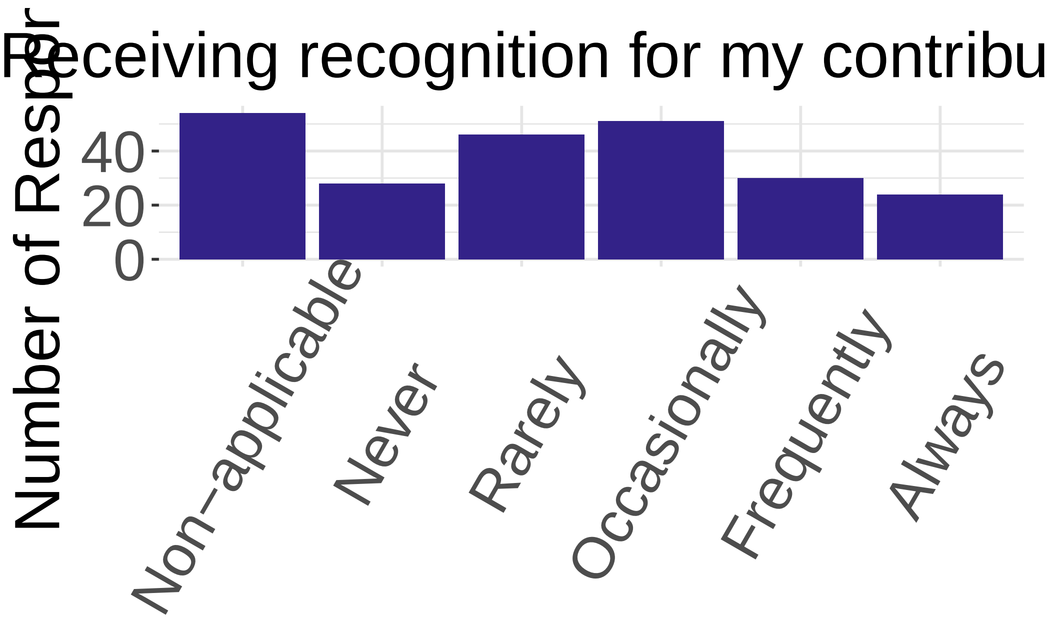

Navigating licensing and other legal is
Number of Respor
60
40
20
0
Non−applicable
Never
Rarely
Occasionally
Frequently
Always

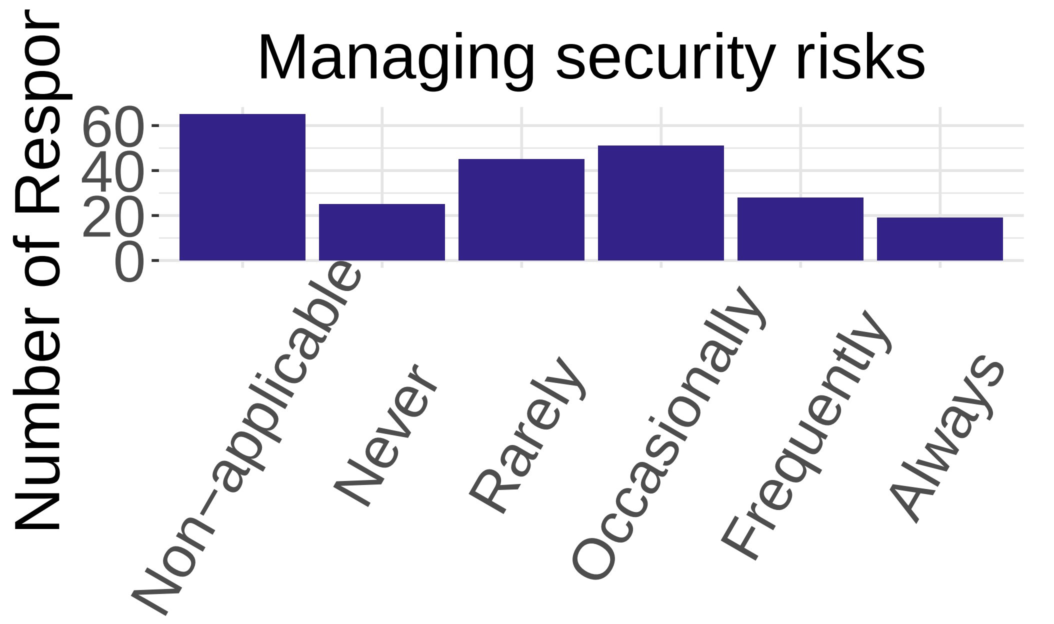
“na-skewed”

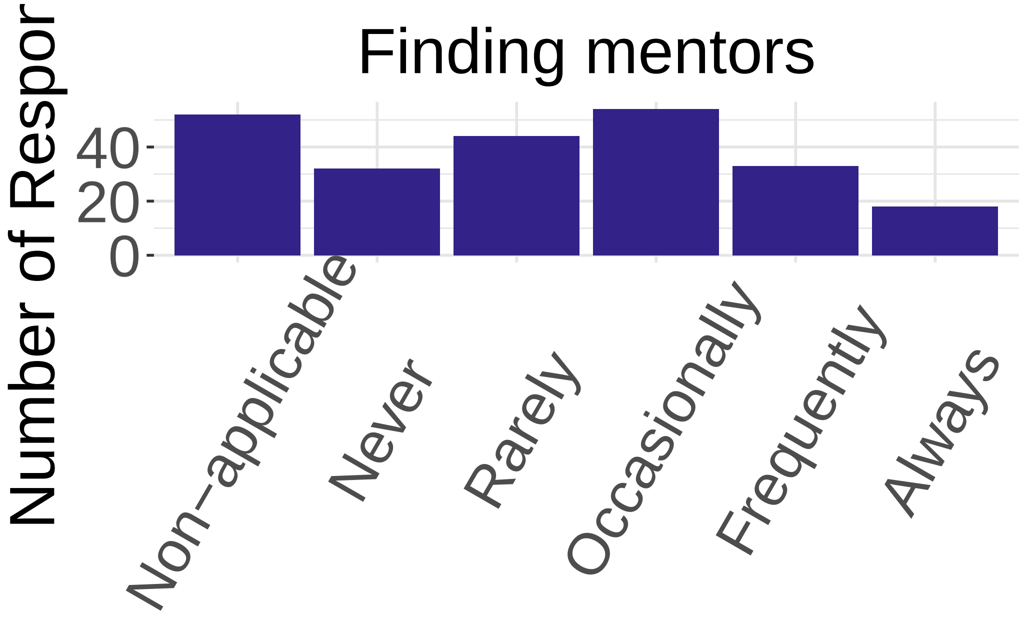
```
multiple_plots(to_plot, titles, na_skewed)
```





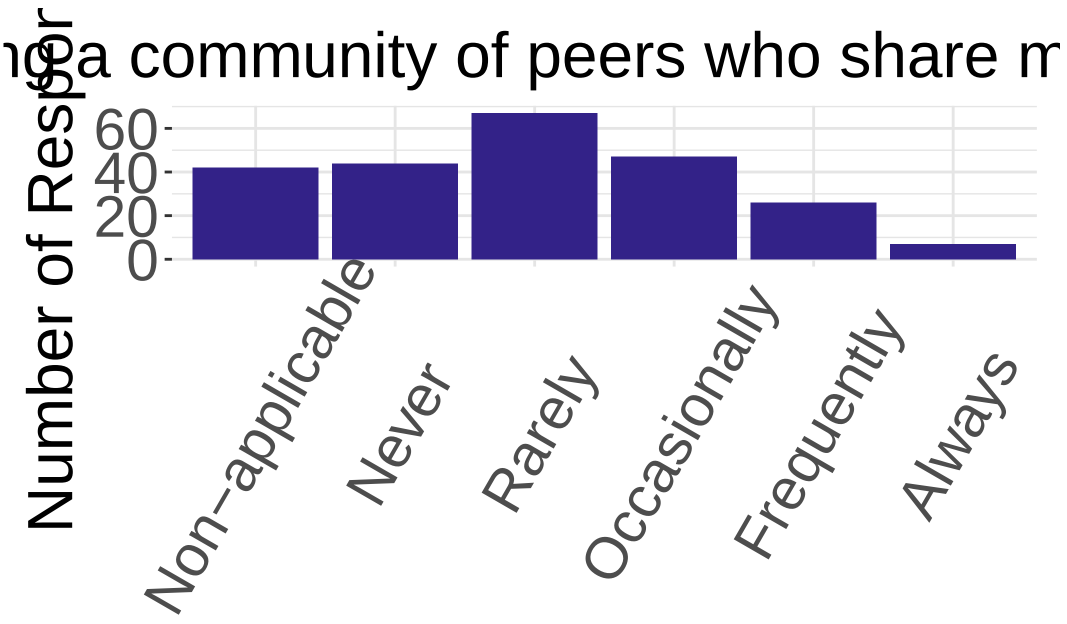

Receiving recognition for my contribu
Number of Respor
0
20
40
Non–applicable
Never
Rarely
Occasionally
Frequently
Always


Managing security risks
Number of Respor
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

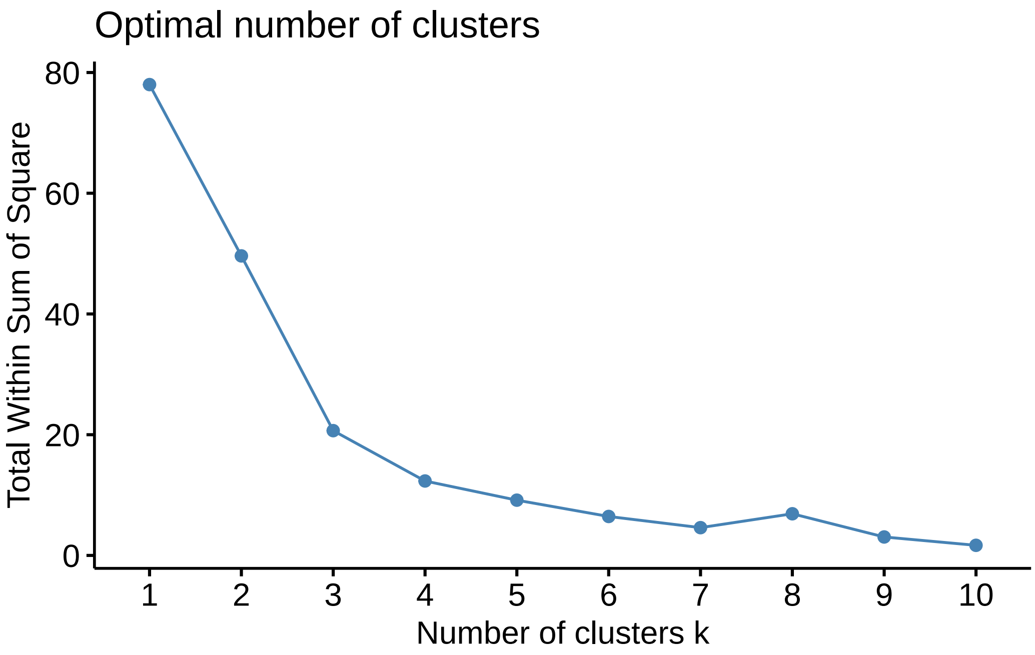
```
wide_counts <- to_plot %>%
  pivot_wider(
    names_from   = challenge_level,
    values_from  = total,
    values_fill  = 0
  )

wide_counts <- data.frame(wide_counts)
# Turn this categorical column into row names
rownames(wide_counts) <- wide_counts$challenge
wide_counts <- wide_counts[,2:(ncol(wide_counts))]

# 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
```

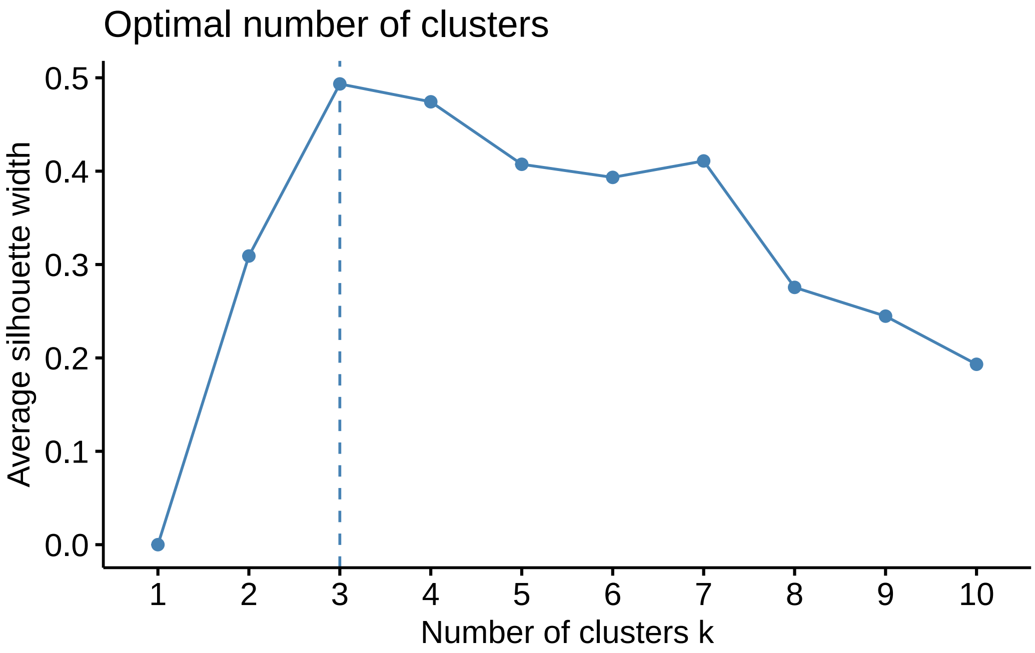
Plot an elbow plot to find the point of diminishing returns.

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



I seem to get diminishing returns around k=4.

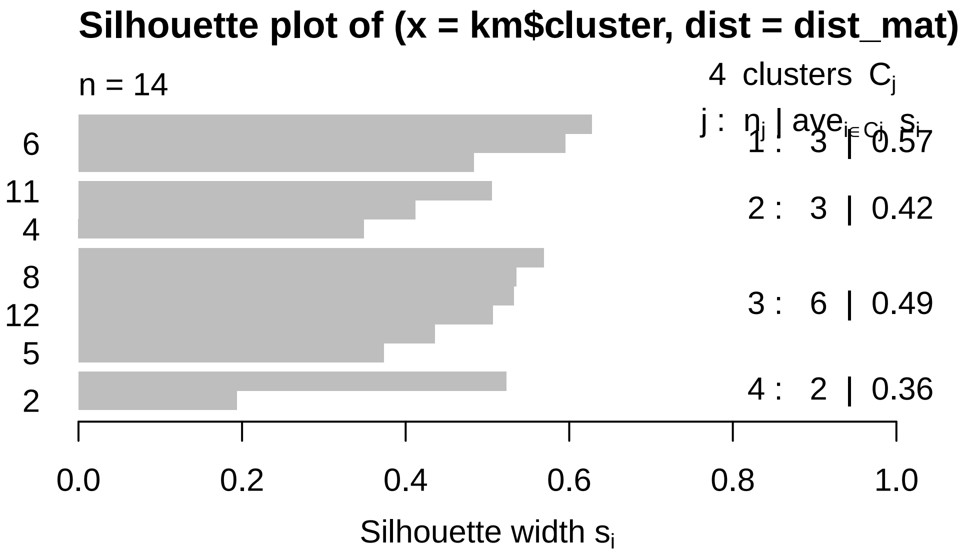
```
factoextra::fviz_nbclust(scaled, kmeans, method = "silhouette")
```



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.

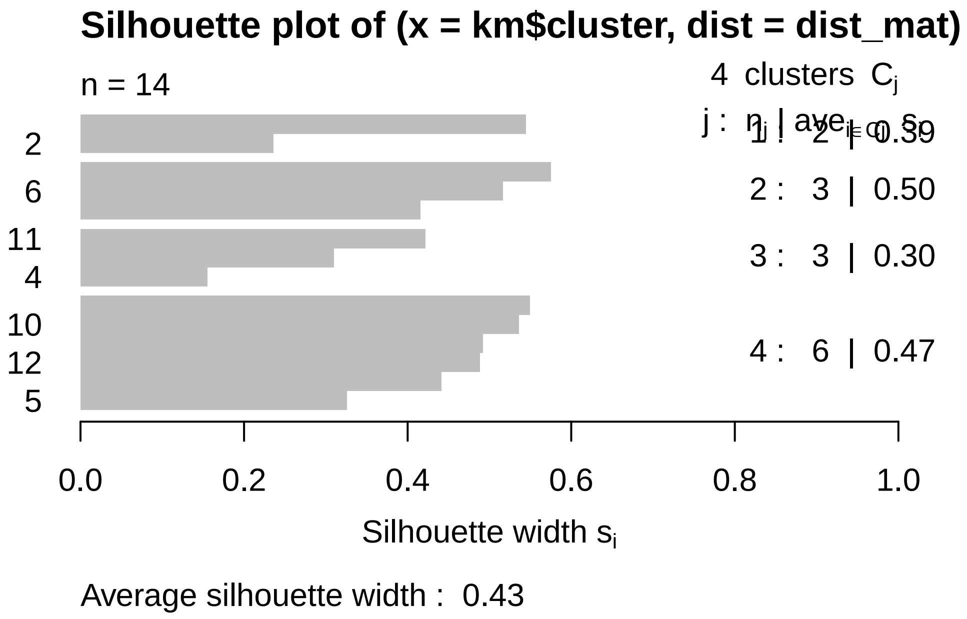
```r
km <- kmeans(scaled, centers = 4, nstart = 25)
dist_mat <- dist(scaled)
sil <- cluster::silhouette(km$cluster, dist_mat)
plot(sil)
```



Average silhouette width : 0.47

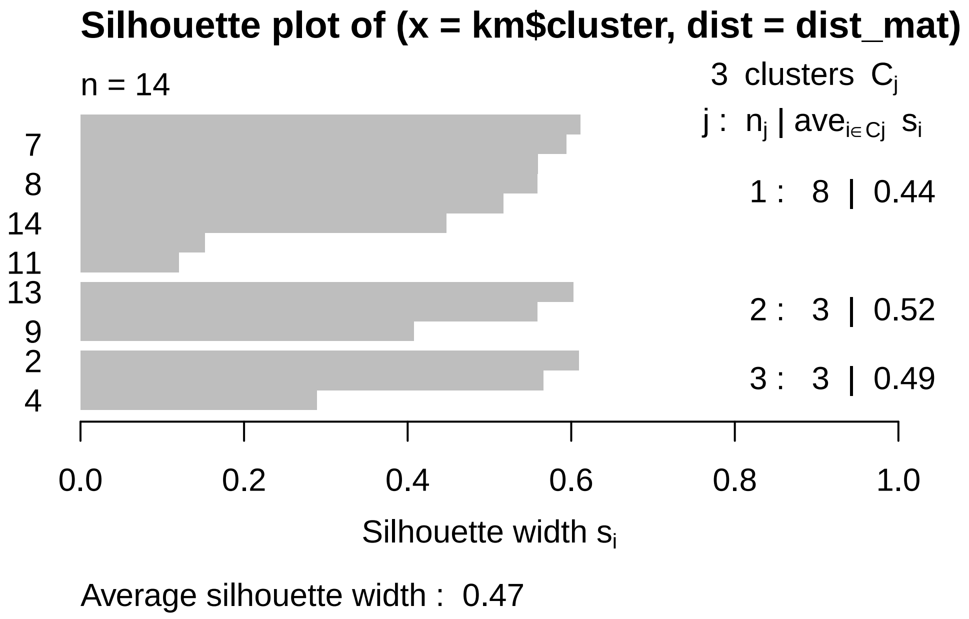
Hm. Looks… acceptable. From Wikipedia: "A clustering with an average silhouette width of over 0.7 is considered to be"strong", a value over 0.5 "reasonable" and over 0.25 "weak"." Let's try unscaled data.

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



Looks slightly worse. 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 <- 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 0.43-0.47, our clusters aren't looking amazing. Still, the average silhouette score is 0.47 with unscaled data and k=3. This is not terrible. I think it's consistent with my hunch that the data for the challenges are not all drawn from the same distribution. These are the cluster assignments:

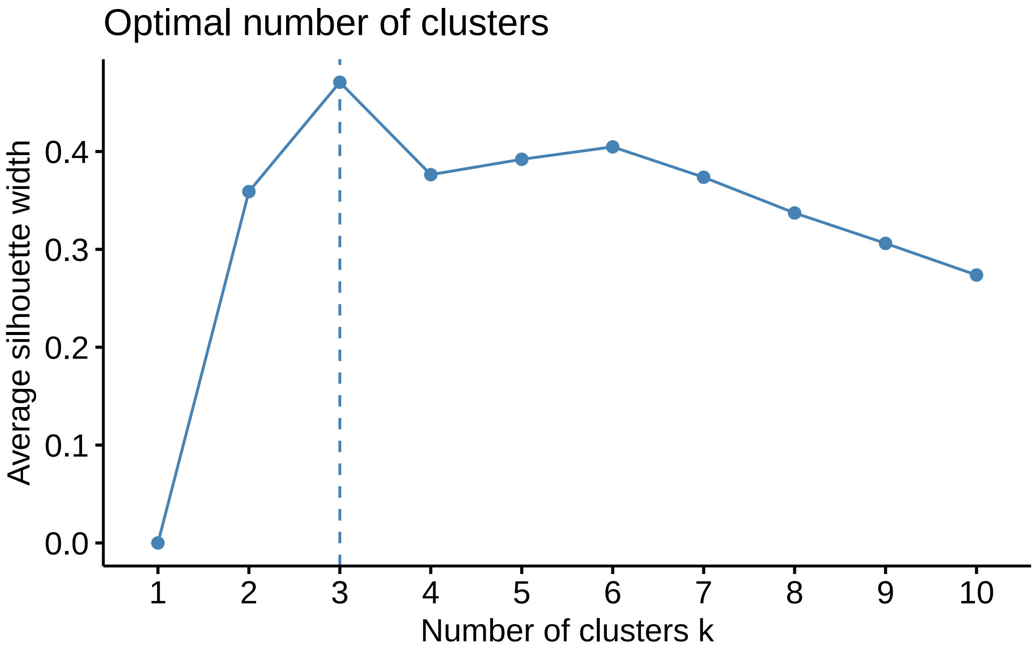
```
# A little extra code to achieve prettier printing
cluster_df <- data.frame(sort(km$cluster))
cluster_df$challenge <- rownames(cluster_df)
clusters <- unique(cluster_df[,1])
for (cl in clusters) {
  print(cluster_df[cluster_df[,1] == cl,], row.names = FALSE)
  cat("\n")
}
```

```
 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.

```r
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.

```r
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, 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)
```

```
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
```

```
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.

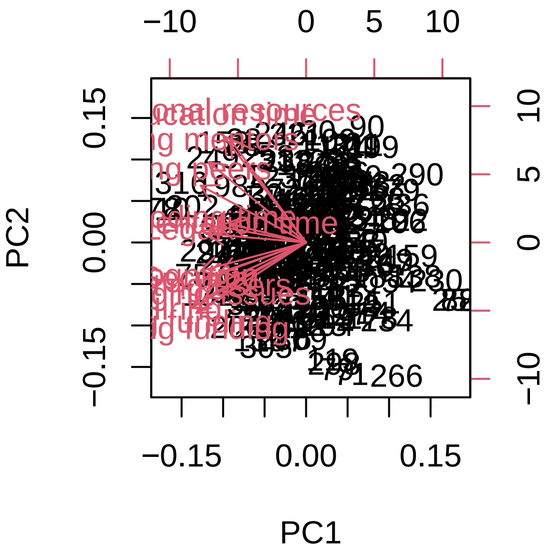
PCA (Abandoned)

In the above sections, I was clustering challenges into groups. Here, I started clustering people into groups. However, this didn't seem too promising, and I don't know if I care enough to pursue it. NOTE THE DIFFERENT CODING SCHEME

```
challnumeric <- challenges %>%
  mutate(
    across(
      everything(),
      ~ recode(
        .x,
        "Never" = 0L,
        "Non-applicable" = -1L, # THIS IS DIFFERENT (-1, not 0)
        "Rarely" = 1L,
        "Occasionally" = 2L,
        "Frequently" = 3L,
        "Always" = 4L
      )
    )
  )


pca <- prcomp(challnumeric, scale = TRUE)
summary(pca)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4    PC5    PC6     PC7
Standard deviation     2.1561 1.3366 1.1930 1.06788 0.9642 0.8606 0.79525
Proportion of Variance 0.3321 0.1276 0.1017 0.08146 0.0664 0.0529 0.04517
Cumulative Proportion  0.3321 0.4597 0.5613 0.64279 0.7092 0.7621 0.80726
                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
Standard deviation     0.74086 0.70184 0.67486 0.64866 0.62517 0.56317 0.26958
Proportion of Variance 0.03921 0.03518 0.03253 0.03005 0.02792 0.02265 0.00519
Cumulative Proportion  0.84647 0.88165 0.91418 0.94424 0.97215 0.99481 1.00000
```

```
biplot(pca)
```


PC2
PC1