3106 Project 2: Kate and Jiaxin

```
library(jsonlite)
library(tm)
```

```
## Loading required package: NLP
```

```
library(qdap)
```

```
## Loading required package: qdapDictionaries
```

```
## Loading required package: qdapRegex
```

```
##
## Attaching package: 'qdapRegex'
```

```
## The following object is masked from 'package:jsonlite':
##
##     validate
```

```
## Loading required package: qdapTools
```

```
## Loading required package: RColorBrewer
```

```
##
## Attaching package: 'qdap'
```

```
## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, as.TermDocumentMatrix
```

```
## The following object is masked from 'package:NLP':
##
##     ngrams
```

```
## The following object is masked from 'package:base':
##
##     Filter
```

```
library(overlap)
library(methods)
library(quanteda)
```

```
## Package version: 2.1.2
```

```
## Parallel computing: 2 of 8 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
##
## Attaching package: 'quanteda'
```

```
## The following objects are masked from 'package:qdap':
##
##     %>%, as.DocumentTermMatrix, as.wfm
```

```
## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, stopwords
## The following objects are masked from 'package:NLP':
##
##     meta, meta<-
## The following object is masked from 'package:utils':
##
##     View
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following object is masked from 'package:qdap':
##
##     %>%
## The following object is masked from 'package:qdapTools':
##
##     id
## The following object is masked from 'package:qdapRegex':
##
##     explain
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidytext)
library(stringr)
```

```
##
## Attaching package: 'stringr'
## The following object is masked from 'package:qdap':
##
##     %>%
```

```
library(officer)
```

```
##
## Attaching package: 'officer'
## The following object is masked from 'package:qdapTools':
##
##     read_docx
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:qdap':
##
##     %&%
```

```
## Loaded glmnet 4.1
```

```
library(knitr)
library(stringi)
library(formattable)
library(wordcloud)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:qdapRegex':
##
##     %+%
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
library(waffle)
library(ggpubr)
```

```
##
## Attaching package: 'ggpubr'
```

```
## The following object is masked from 'package:qdap':
##
##     %>%
```

```
library(text2vec)
```

```
##
## Attaching package: 'text2vec'
```

```
## The following object is masked from 'package:formattable':
##
##     normalize
```

```
job_desc <- jsonlite::fromJSON('~/Downloads/indeed_job_descs_2021_03_16.json')
names(job_desc)
```

```
## [1] "request_params"   "job_descriptions"
```

```
job_desc$request_params[[1]] #not na 1
```

```
##  [1] "ux+designer"               "recruiter"
##  [3] "marketing"                 "sales"
##  [5] "office+manager"            "frontend+developer"
##  [7] "fullstack+engineer"        "test+engineer"
##  [9] "site+reliability+engineer" "data+architect"
## [11] "human+resource+specialist" "business+analyst"
## [13] "engineering+manager"       "researcher"
## [15] "researcher"                "data+scientist"
## [17] "software+developer"        "statistician"
## [19] "deep+learning"             "machine+learning+engineer"
## [21] "actuary"                   "financial+analyst"
```

```
## [23] "econmist"                  "financial+engineer"
## [25] "political+analyst"         "data+journalist"
```

```
job_desc$job_descriptions[[45]]  # not na 2
```

```
##  [1] NA
##  [2] "Job detailsSalary$79,000 - $119,000 a yearJob TypeFull-timeFull Job DescriptionYour Job\nIndee
##  [3] NA
##  [4] NA
##  [5] NA
##  [6] NA
##  [7] NA
##  [8] NA
##  [9] NA
## [10] NA
## [11] NA
## [12] NA
## [13] NA
## [14] NA
## [15] NA
## [16] NA
## [17] NA
## [18] NA
## [19] NA
## [20] NA
## [21] NA
## [22] NA
## [23] NA
## [24] NA
## [25] NA
## [26] NA
```

```
len <- function(x) {
  return(length(x))
}

l <- sapply(job_desc$job_descriptions,length)
l_p <- sapply(job_desc$request_params,length)
description <- job_desc$job_descriptions

desc = c()
for (i in 1:length(description)){
  for (j in 1:length(description[[i]])){
    if (is.na(description[[i]][j])){
      next
    }
    desc <- append(desc,description[[i]][j])
  }
}


# data("data_corpus_inaugural", package = "quanteda")
d <- quanteda::dfm(desc, verbose = FALSE)

dim(d)
```

```
## [1]   868 20678
```

```
#l <- sapply(desc,len)
#desc_dtm <- DocumentTermMatrix(desc[[1]])
target_freq <- as.numeric(d)
freqs_mat <- as.matrix(d)
doc_freq <- apply(freqs_mat,2,function(x) mean(x>0))
idf <- 1/doc_freq
idf_mat <- rep(idf,nrow(freqs_mat), byrow = TRUE, nrow = nrow(freqs_mat))
tf_idf <- freqs_mat * idf_mat
```

```
desc_cleaned <- c()
for (i in seq_along(desc)){
    without_stopwords <- rm_stopwords(
    desc[i],
    stopwords = qdapDictionaries::Top200Words,
    unlist = FALSE,
    separate = TRUE,
    strip = FALSE,
    unique = FALSE,
    char.keep = NULL,
    names = FALSE,
    ignore.case = TRUE,
    apostrophe.remove = FALSE
  )
  text <- unlist(without_stopwords)
  text <- str_replace_all(text, pattern = '\n', replacement = "") # Remove \n
  text <- str_replace_all(text, pattern = '\u0092', replacement = "'") #Replace with quote
  text <- str_replace_all(text, pattern = '\u0091', replacement = "'") #Replace with quote
  text <- str_replace_all(text, pattern = '[:punct:]', replacement = "") #Remove punctuation
  text <- str_replace_all(text, pattern = '$', replacement = "") #Remove punctuation
  text <- str_replace_all(text, pattern = '-', replacement = "") #Remove punctuation
  text <- str_replace_all(text, pattern = '·', replacement = "") #Remove punctuation
  text <- str_replace_all(text, pattern = '[0-9]+', replacement = "") #Remove numbers
  #text <- str_replace_all(text, pattern = 'per', replacement = "") #Remove numbers
  without_stopwords <- as.list(text)
  combine_1 <- combine_words(
    without_stopwords,
    sep = " "
  )
  combine =paste("", combine_1,"")
  desc_cleaned <- append(desc_cleaned, removePunctuation(combine))
}

# the dim of d is 590*16202. After removing all the punctuations and stopwords, its dimension is 590*13
dim(d)
```

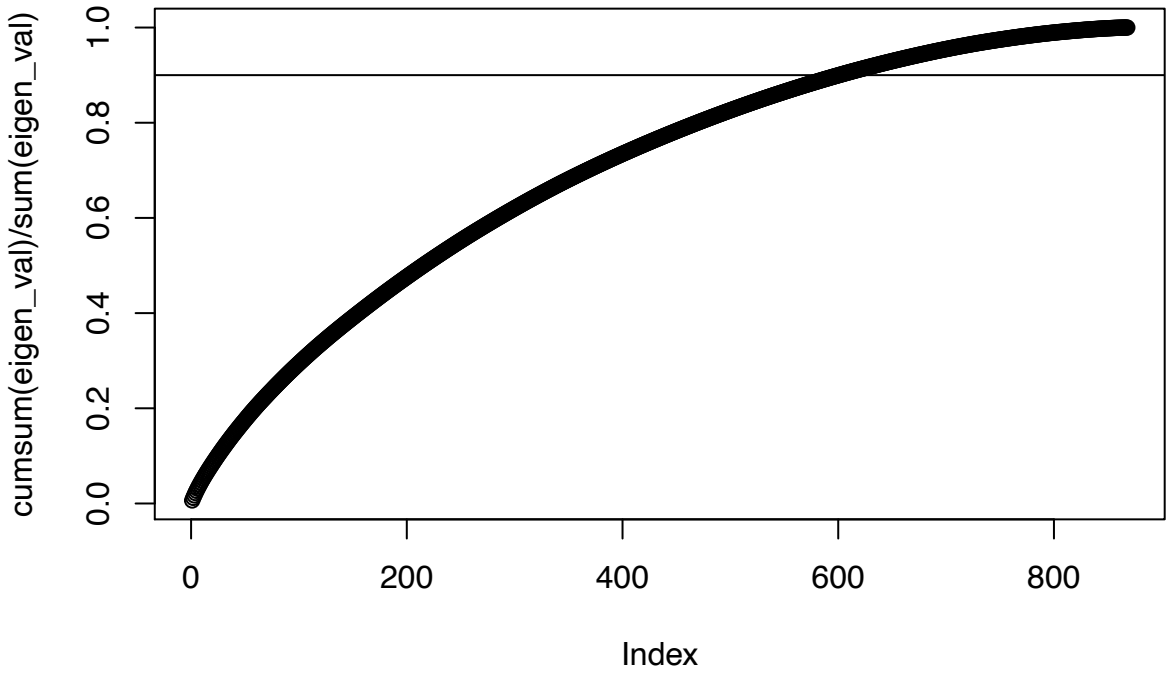
```
## [1]   868 20678
```

```
d_cleaned <- quanteda::dfm(desc_cleaned, verbose = FALSE)

dim(d_cleaned)
```

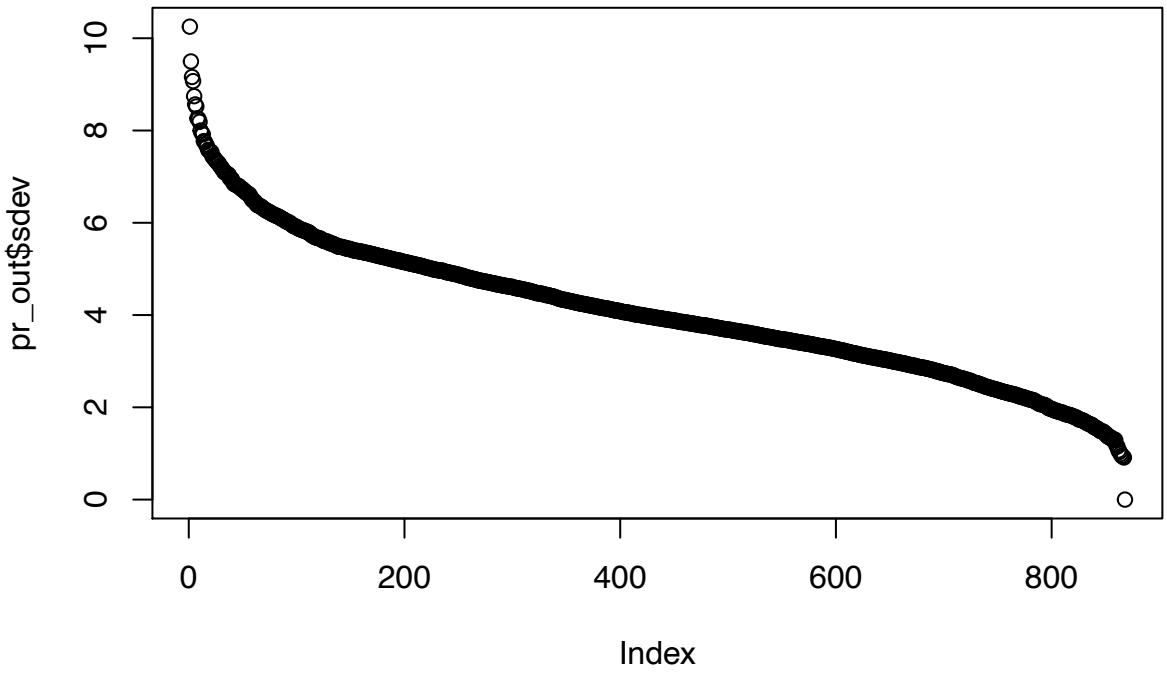
```
## [1]   868 16433
```

```r
#l <- sapply(desc,len)
#desc_dtm <- DocumentTermMatrix(desc[[1]])
target_freq_1 <- as.numeric(d_cleaned)
freqs_mat_1 <- as.matrix(d_cleaned)
doc_freq_1 <- apply(freqs_mat_1,2,function(x) mean(x>0))
idf_1 <- 1/doc_freq_1
idf_mat_1 <- rep(idf_1,nrow(freqs_mat_1), byrow = TRUE, nrow = nrow(freqs_mat_1))
tf_idf_1 <- freqs_mat_1 * idf_mat_1
```

```r
pr_out <- prcomp(tf_idf_1, scale=TRUE) #look at names of pr_out
eigen_val <- pr_out$sdev^2
plot(cumsum(eigen_val) / sum(eigen_val))
abline(h=.9)
```



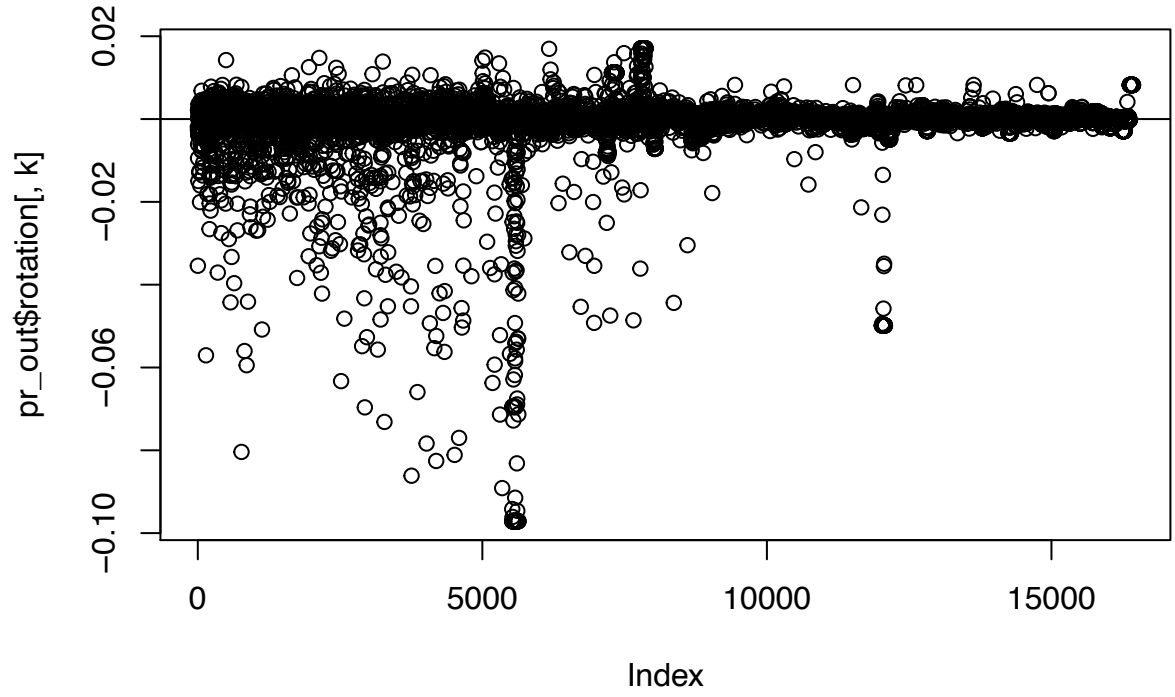

```r
plot(pr_out$sdev)
```




```
# data mining approach
k <- 5 # could be anywhere from 2-4
plot(pr_out$rotation[, k])
head(pr_out$rotation[,k]) #these are the loadings
```

```
##           job detailssalary       yearjob      typefull    timenumber
##  -0.035426887  -0.001880251  -0.002650382   0.002083729  -0.009461063
##         hires
##  -0.006112571
```

```
abline(h = 0)
```



```
which(abs(pr_out$rotation[, k]) > 0.06)
```

```
##                flex                walk              calmly              exceed
##                 768                2518                2935                3280
##          implements           congenial             periods               rooms
##                3756                3860                4020                4189
##               neat            oversees         hospitality      failinnovative
##               4513                4591                5179                5311
##       applycompanys   descriptionmohonk            mountain           victorian
##               5349                5524                5525                5526
##             castle              resort             nestled             scenic
##               5527                5528                5529                5530
##           landmark              smiley         spectacular            timeless
##               5533                5535                5536                5537
##             cliffs            pristine            woodland               acres
##               5538                5539                5541                5542
##             mohonk              hiking              trails              tennis
##               5543                5545                5546                5549
##               golf           horseback              riding            carriage
##               5550                5551                5552                5553
##              rides             boating            swimming            climbing
##               5554                5555                5556                5559
##         snowshoeing             skating               charm           evaluates
##               5560                5562                5569                5572
##            printed              hosts             behaves              houses
##               5575                5577                5580                5581
##            travels             attends               theme       reforecasting
##               5583                5584                5587                5595
##          exhibiting         subtraction      multiplication            warnings
##               5598                5600                5601                5603
##              stoop              twist            inclines             counter
##               5607                5608                5610                5611
##          peripheral          variations              shades             groomed
##               5613                5614                5615                5620
##       interruptions             noises          overnights            weekdays
##               5622                5623                5626                5627
## shiftholidaysmonday      creativitythis      careersbenefit
##               5632                5634                5635
```

The some of the loadings are who, we, are, :, cardinal, and financial. NOTE: I am wondering if our word-separation method is the best we could use because the loadings include a lot of things like punctuation marks.

There are a lot of words with high PCA values. NOTE: should we keep the websites in? TF-IDF does seem to be important though because it has revealed that the job descriptions are asking for very specific things like "agtech" or "adventurous" very frequently.

```
# get the top k tokens with the highest tf-idf value
k <- 15
i <- 1
keyword_lists <- data.frame(matrix(NA, nrow = nrow(tf_idf_1), ncol = k))
for (i in 1:nrow(tf_idf_1)){
  keyword_lists[i,] <- names(tf_idf_1[i,][order(tf_idf_1[i,],decreasing = TRUE)[1:k]])
}
head(keyword_lists)
```

```
##           X1         X2         X3          X4         X5         X6
## 1   creative     strong experience       years   prestige         ui
## 2    complex    product experience experiences     create management
## 3  princeton     online         ux          ui prototypes       able
## 4 experience     remote      level experiences passionate developers
## 5     design experience   designer experiences newspapers    hearsts
## 6 experience         ui    working       cover delivering       best
##            X7          X8       X9         X10           X11          X12
## 1 environment collaborate   across      highly understanding         team
## 2        user     looking designer environment        across         team
## 3      status      review tutoring        team      products deliverables
## 4         web       years       ux    designer       develop     position
## 5        able     designs    teams      hearst     platforms     projects
## 6       teams     provide       ux    designer      projects  development
##           X13           X14        X15
## 1    products        online  animation
## 2    products        online prototypes
## 3       cross     protected  education
## 4        life understanding     mobile
## 5      highly   comfortable   identity
## 6 environment        social   flexible
```

For each job description(each row in the tf_ids_1), I get the top 15 tokens with the highest tf-idf value. We can also extract the top 5 tokens with highest tf-idf scores in our resume, and see if there will be some tokens overlapping between the job description and the resume.

```
# give priority on the resume.
# cooccurance- build up association between words
#data(stop_words) # Stop words.
real_resume <- readLines("Li_Jiaxin_resume.txt")
real_resume <- str_replace_all(real_resume, pattern = '\"', replacement = "")

# real_resume <- paste(real_resume,collapse="")
# transform it into a term document matrix
resume <- quanteda::dfm(real_resume, verbose = FALSE)

target_freq_resume <- as.numeric(resume)
freqs_mat_resume <- as.matrix(resume)
doc_freq_resume <- apply(freqs_mat_resume,2,function(x) mean(x>0))
idf_resume <- 1/doc_freq_resume
idf_mat_resume <- rep(idf_resume,nrow(freqs_mat_resume), byrow = TRUE, nrow = nrow(freqs_mat_resume))
tf_idf_resume <- freqs_mat_resume * idf_mat_resume

keywords_in_resume <- names(tf_idf_resume[1,][order(tf_idf_1[1,],decreasing = TRUE)[1:k]])
```

```
overlap_keyword <- c()
for (i in 1:nrow(tf_idf_1)){
  keyword_job_desc <- keyword_lists[i,]
  overlap_keyword[i] <- length(intersect(keywords_in_resume,keyword_job_desc))
}
```

It is awkward that none of the job desc has over lap with my resume...

```
text <- paste(real_resume,collapse="")
text <- paste(text, collapse = " ")
text <- str_replace_all(text, pattern = '\"', replacement = "") # Remove slashes
```

```
text <- str_replace_all(text, pattern = '\n', replacement = "") # Remove \n
text <- str_replace_all(text, pattern = '\u0092', replacement = "'") #Replace with quote
text <- str_replace_all(text, pattern = '\u0091', replacement = "'") #Replace with quote
text <- str_replace_all(text, pattern = '-', replacement = "") #Remove dashes
text <- str_replace_all(text, pattern = '[0-9]+', replacement = "") #Remove numbers
text <- str_replace_all(text, pattern = '\u0092', replacement = "'") #Replace with quote
text <- str_replace_all(text, pattern = '\u0091', replacement = "'") #Replace with quote
text <- str_replace_all(text, pattern = '·', replacement = "") #Remove dots


text_df <- data_frame(Text = text) # tibble aka neater data frame
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
text_words <- text_df %>% unnest_tokens(output = word, input = Text)

text_words  <- text_words  %>% anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
text_wordcounts <- text_words  %>% count(word, sort = TRUE)
keywords_in_resume <- text_wordcounts$word[1:100]

overlap_keyword <- c()
for (i in 1:nrow(tf_idf_1)){
  keyword_job_desc <- keyword_lists[i,]
  overlap_keyword[i] <- length(intersect(keywords_in_resume,keyword_job_desc))
}
sum(overlap_keyword)
```

```
## [1] 1285
```

Still doesn't work...

```
# give priority on the resume.
# cooccurance- build up association between words
data(stop_words) # Stop words.
real_resume_k <- readLines("~/Downloads/MarshResumeJan2021_Data_Mining.txt")
real_resume_k <- str_replace_all(real_resume, pattern = '\"', replacement = "")

# real_resume <- paste(real_resume,collapse="")
# transform it into a term document matrix
resume <- quanteda::dfm(real_resume_k, verbose = FALSE)
#non_stop_cols <- stopwords(colnames(resume))
stops <- stopwords("smart")
```

```
## Warning: 'stopwords(language = "smart")' is deprecated.
## Use 'stopwords(source = "smart")' instead.
## See help("Deprecated")
```

```
cols <- removePunctuation(colnames(resume), preserve_intra_word_contractions = TRUE, preserve_intra_wor
#colnames(cols["klm2244columbiaedu"]) <- "klm2244columbia.edu"
#keep_cols <- cols[!cols %in% stops]# colnames w/o stopwords
#keep_cols <- rm_number(cols, trim = TRUE, clean = TRUE)
```

```
#keep_cols <- stri_remove_empty(keep_cols)
#resume <- resume[, keep_cols]


target_freq_resume <- as.numeric(resume)
freqs_mat_resume <- as.matrix(resume)
doc_freq_resume <- apply(freqs_mat_resume,2,function(x) mean(x>0))
idf_resume <- 1/doc_freq_resume
idf_mat_resume <- rep(idf_resume,nrow(freqs_mat_resume), byrow = TRUE, nrow = nrow(freqs_mat_resume))
tf_idf_resume <- freqs_mat_resume * idf_mat_resume

keywords_in_resume <- names(tf_idf_resume[1,][order(tf_idf_1[1,],decreasing = TRUE)[1:k]])
#keywords_in_resume <- str_replace_all(keywords_in_resume, pattern = '-', replacement = "") #Remove das
```

```
overlap_keyword <- c()
for (i in 1:nrow(tf_idf_1)){
  keyword_job_desc <- keyword_lists[i,]
  overlap_keyword[i] <- length(intersect(keywords_in_resume,keyword_job_desc))
}
head(overlap_keyword)
```

```
## [1] 0 0 0 0 0 0
```

```
text <- paste(real_resume_k,collapse="")
text <- paste(text, collapse = " ")
text <- str_replace_all(text, pattern = '\"', replacement = "") # Remove slashes
text <- str_replace_all(text, pattern = '\n', replacement = "") # Remove \n
text <- str_replace_all(text, pattern = '-', replacement = "") #Remove dashes
text <- str_replace_all(text, pattern = '[0-9]+', replacement = "") #Remove dashes
text <- str_replace_all(text, pattern = '\u0092', replacement = "'") #Replace with quote
text <- str_replace_all(text, pattern = '\u0091', replacement = "'") #Replace with quote
text <- str_replace_all(text, pattern = '·', replacement = "") #Remove dashes

text_df <- data_frame(Text = text) # tibble aka neater data frame
text_words <- text_df %>% unnest_tokens(output = word, input = Text)
text_words  <- text_words  %>% anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
text_wordcounts <- text_words  %>% count(word, sort = TRUE)
keywords_in_resume <- text_wordcounts$word[1:100]

overlap_keyword <- c()
#overlaps <- c()
for (i in 1:nrow(tf_idf_1)){
  keyword_job_desc <- keyword_lists[i,]
  overlap_keyword[i] <- length(intersect(keywords_in_resume,keyword_job_desc))
  #overlaps[i] <- intersect(keywords_in_resume,keyword_job_desc)
  # rbind(overlaps[i], fill = TRUE) this
}
sum(overlap_keyword)
```

```
## [1] 1285
```

```
highly_relevant <- length(which(overlap_keyword>=4))
head(tf_idf_1[highly_relevant,])
```

```
##           job detailssalary       yearjob      typefull    timenumber
##          4.34          0.00          0.00        868.00          0.00
##         hires
##          0.00
```

```
head(desc[highly_relevant])
```

```
## [1] "Job detailsJob TypeFull-timeFull Job DescriptionJob Summary:\nLocation: NYC (remote until TBD)\n
```

```
overlap_keyword[26]
```

```
## [1] 1
```

```
desc[26]
```

```
## [1] "Job detailsJob TypeFull-timeFull Job DescriptionUI/UX Designer\n\nWe are looking for a UX Desig
```

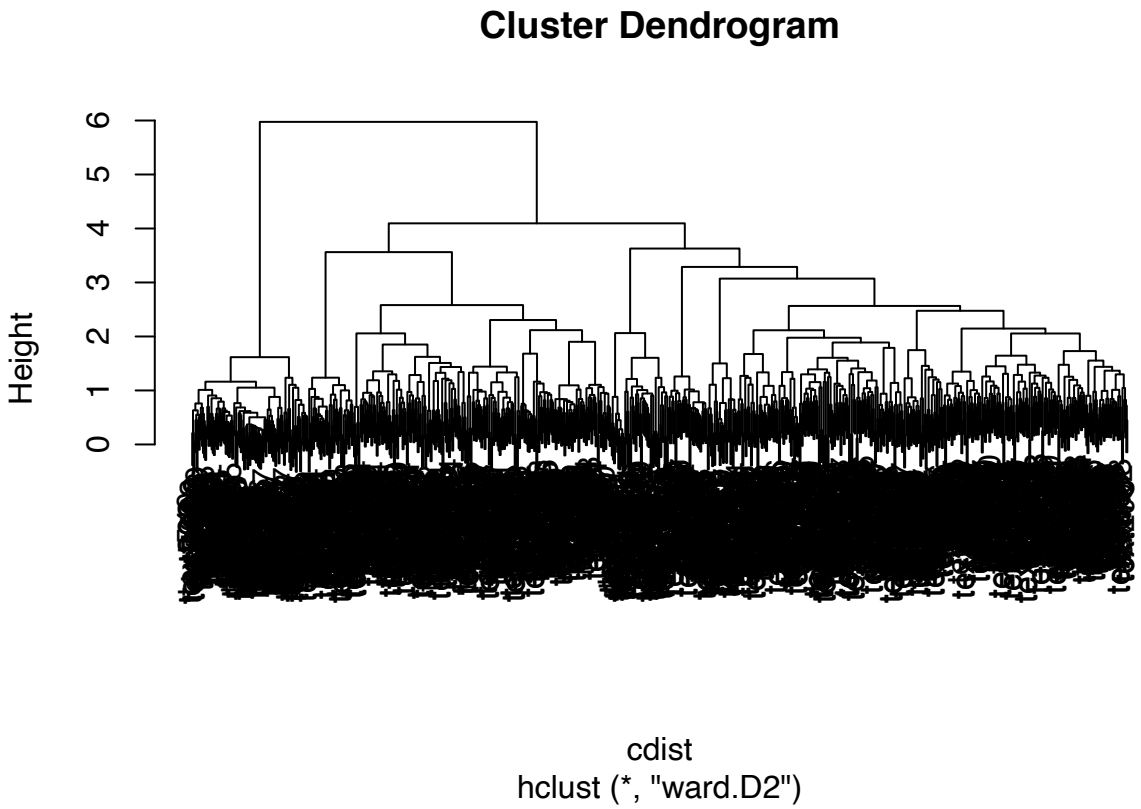
```
#intersect(which(grepl("sustainab", desc, ignore.case = TRUE)), highly_relevant)

#desc[] #checking on things that would be close to my resume
#sort(tf_idf_1[847,],decreasing=TRUE)
```

```
# bind tf_idf_1
#log_df1 = cbind(d, as.matrix(tf_idf_1))
#log_df <- log_df1[,unique(colnames(log_df1))]
#mat_tf_idf <- as.matrix(tf_idf_1)

#hclus_out <- hclust(dist(cbind(x, y)), "single") #cbind to make a matrix
#hclus_out <- hclust(dist(log_df1), "complete") #cbind to make a matrix
#plot(hclus_out)
#clus_est <- cutree(hclus_out, k=8)
#plot(tf_idf_1, pch=16,
#     col=c("red", "blue", "black", "green", "purple", "orange", "yellow", "grey", "pink", "navy","tan"

#clus_est
#desc[,c(263, 323)]
```

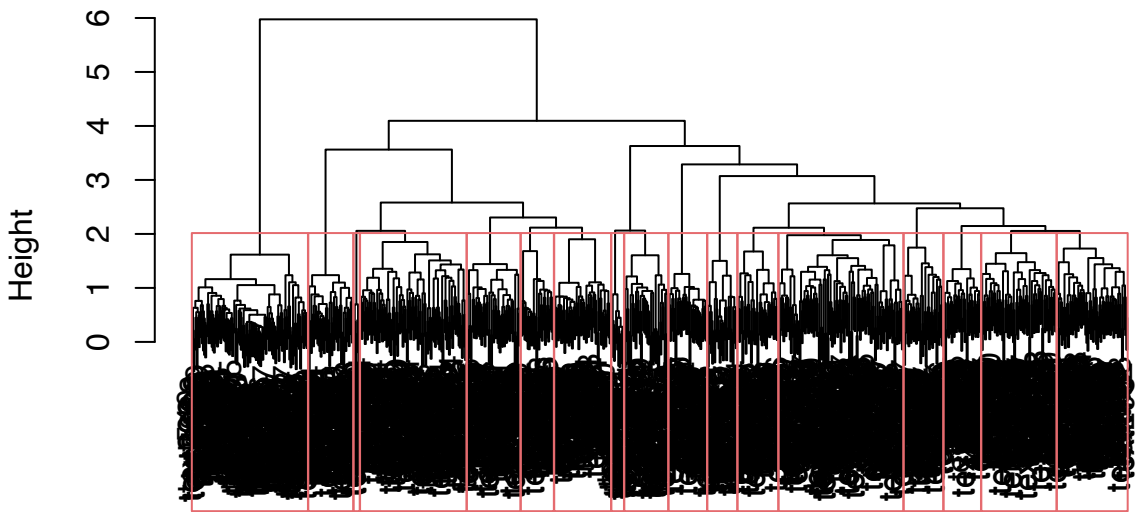
```
#km_out <- kmeans(mat_tf_idf, centers = 8)
#names(km_out) # $cluster is the assignment to the cluster
#length(km_out$cluster)
#table(km_out$cluster)
# km_out$centers
#km_out$centers
# j is the TRUE assignment
#mean(km_out$cluster==j) # xbar and ybar in the different groups
# NOT a good way to evaluate
#table(km_out$cluster, j)

#cols <- c("red", "blue", "green", "purple", "yellow", "orange", "pink", "grey")
#plot(tf_idf_1,
#     col=cols[km_out$cluster],
#     pch=16)
```

```
# TF-IDF and cosine similarity
# document clustering! with https://cran.r-project.org/web/packages/textmineR/vignettes/b_document_clus


# changing cosine similarity to a distance
```

```
csim <- tf_idf_1 / sqrt(rowSums(tf_idf_1 * tf_idf_1))
csim <- csim %*% t(csim)
cdist <- as.dist(1 - csim)

#h clust
k <- 17
hc <- hclust(cdist, "ward.D2")
plot(hc)
```




```
clustering <- cutree(hc, h=2)
plot(hc, main = "'Complete' Hierarchical Clustering of Job Description TF-IDF",
     xlab = "Cosine Similarity as Distance")
rect.hclust(hc, k, border = "#e56b6f")
```




```
p_words <- colSums(freqs_mat_1) / sum(freqs_mat_1)
cluster_words <- lapply(unique(clustering), function(x){
  rows <- freqs_mat_1[clustering == x,]
  # for memory's sake, drop all words that don't appear in the cluster
  rows <- rows[ , colSums(rows) > 0 ]
  colSums(rows) / sum(rows) - p_words[ colnames(rows) ]
})
```

```
cluster_summary <- data.frame(cluster = unique(clustering),
                              size = as.numeric(table(clustering)),
                              top_words = sapply(cluster_words, function(d){
                                paste(
                                  names(d)[ order(d, decreasing = TRUE) ][ 1:5 ],
                                  collapse = ", ")
                              }),
                              stringsAsFactors = FALSE)
formattable(cluster_summary)
```

cluster

size

top_words

1

53

design, experience, user, ux, product

2

99

technical, business, clinical, employment, status

3

70

job, full, years, per, hires

4

38

financial, investment, finance, management, analysis

5

116

actuarial, business, pricing, project, insurance

6

50

team, product, olo, engineering, spotify

7

41

research, ibm, information, insights, user

8

31

software, infrastructure, experience, cloud, systems

9

66

office, customer, equipment, service, maintain

10

35

hiring, recruiting, recruiter, recruitment, talent

11

36

sales, customer, training, per, looking

12

37

marketing, media, social, content, hbo

13

28

public, city, program, health, service

14

108

data, business, science, analytics, learning

15

6

mount, sinai, medicine, system, health

16

42

learning, machine, ml, deep, systems

17

12

research, investment, industry, equity, level

```
wordcloud::wordcloud(words = names(cluster_words[[ 2 ]]),
                     freq = cluster_words[[ 2 ]],
                     max.words = 25,
                     random.order = FALSE,
                     colors = c("#eaac8b", "#355070", "#6d597a", "grey",
                                "#b56576", "#e56b6f"),
                     main = "Top 25 words in cluster 4")
```

```
## Warning in wordcloud::wordcloud(words = names(cluster_words[[2]]), freq =
## cluster_words[[2]], : capgemini could not be fit on page. It will not be
## plotted.
```

```
## Warning in wordcloud::wordcloud(words = names(cluster_words[[2]]), freq =
## cluster_words[[2]], : requirements could not be fit on page. It will not be
## plotted.
```

```
## Warning in wordcloud::wordcloud(words = names(cluster_words[[2]]), freq =
## cluster_words[[2]], : qualified could not be fit on page. It will not be
## plotted.
```

```
## Warning in wordcloud::wordcloud(words = names(cluster_words[[2]]), freq =
## cluster_words[[2]], : opportunity could not be fit on page. It will not be
## plotted.
```

```
## Warning in wordcloud::wordcloud(words = names(cluster_words[[2]]), freq =
## cluster_words[[2]], : software could not be fit on page. It will not be plotted.
```


```
groups = c()
  for (i in 1:length(description)){
    for (j in 1:length(description[[i]])){
      if (is.na(description[[i]][j])){
        next
      }
    groups <- append(groups,j)
    }
  }
researchers_2 <- which(groups == 15)
groups[researchers_2] <- 14
length(unique(groups))
```

```
## [1] 25
```

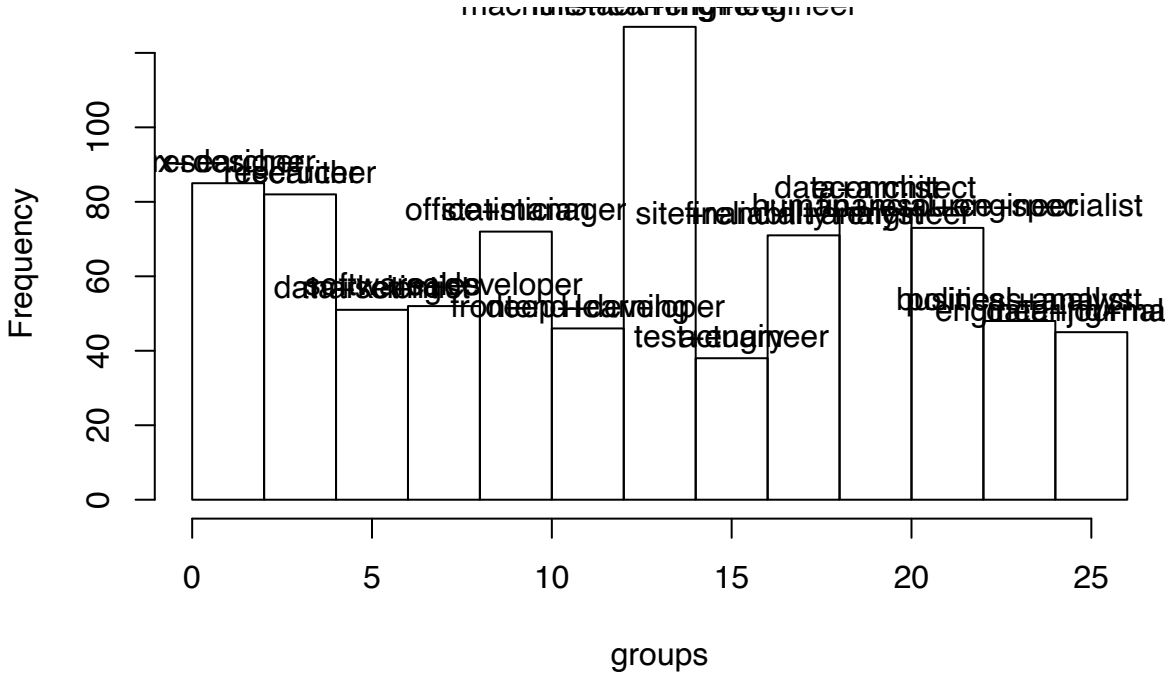
```
get_group <- function(input_num){
  grouped <- groups[input_num]
  return(job_desc$request_params$q[grouped])
}
get_group(182)
```

```
## [1] "office+manager"
```

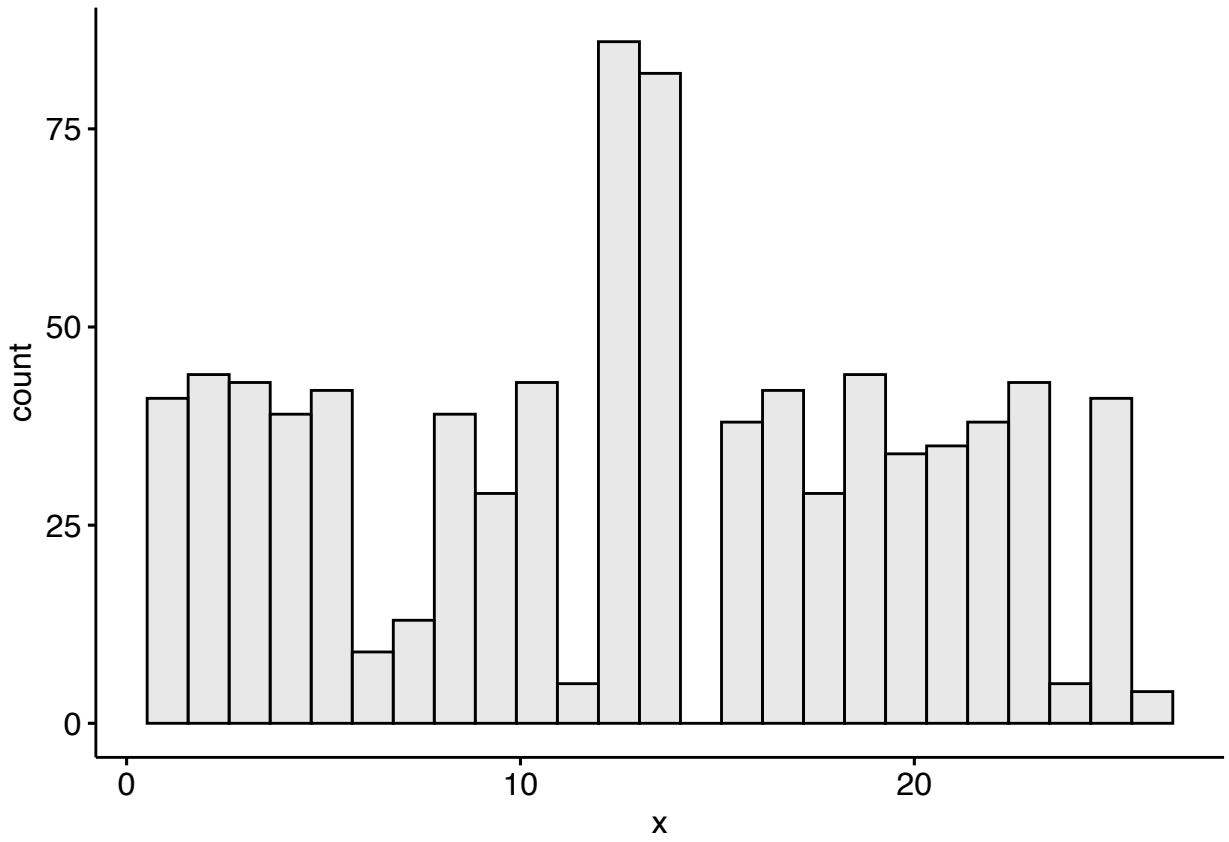
```
get_cluster <- function(input_num){
  return(clustering[[as.character(paste0("text", input_num))]])
}
get_cluster(182)
```

```
## [1] 9
```

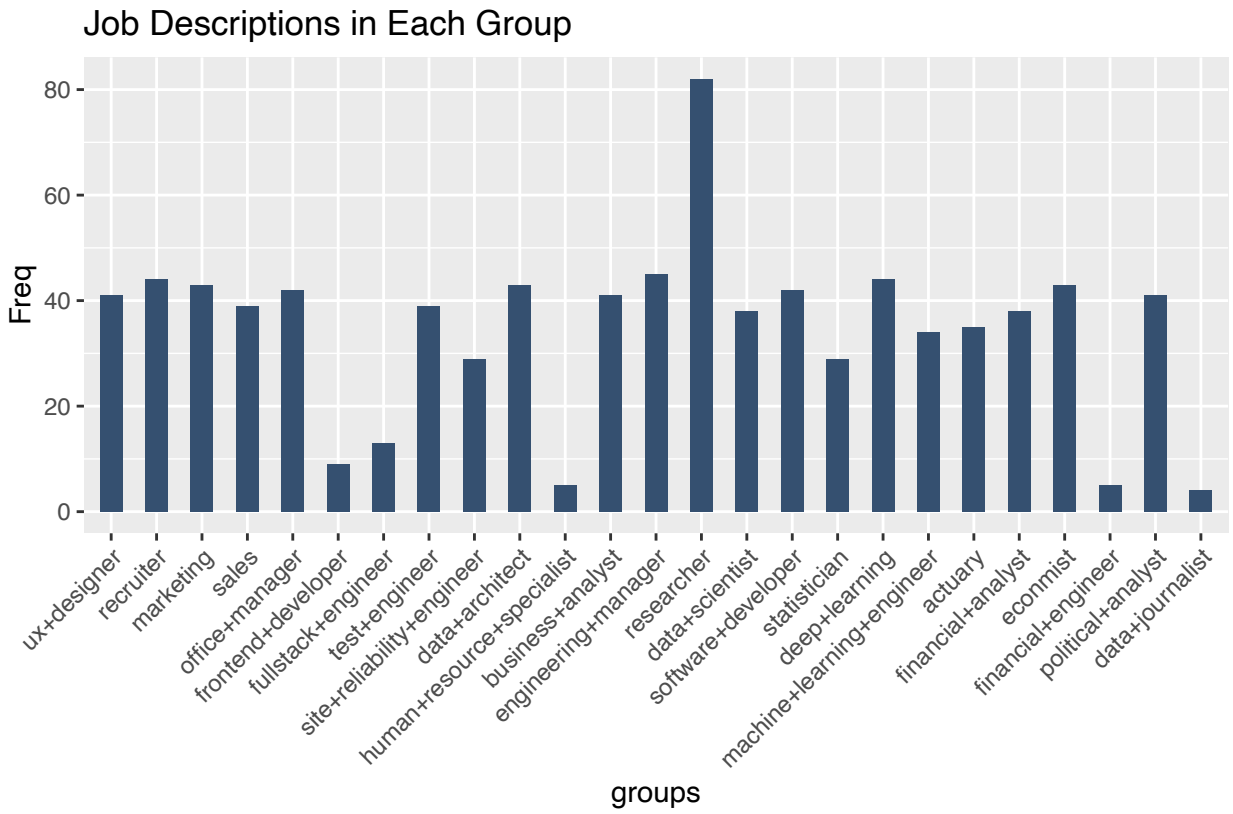
```r
hist(groups,
     main = "Histogram Showing Different Job Titles",
     labels=job_desc$request_params$q
     )
```



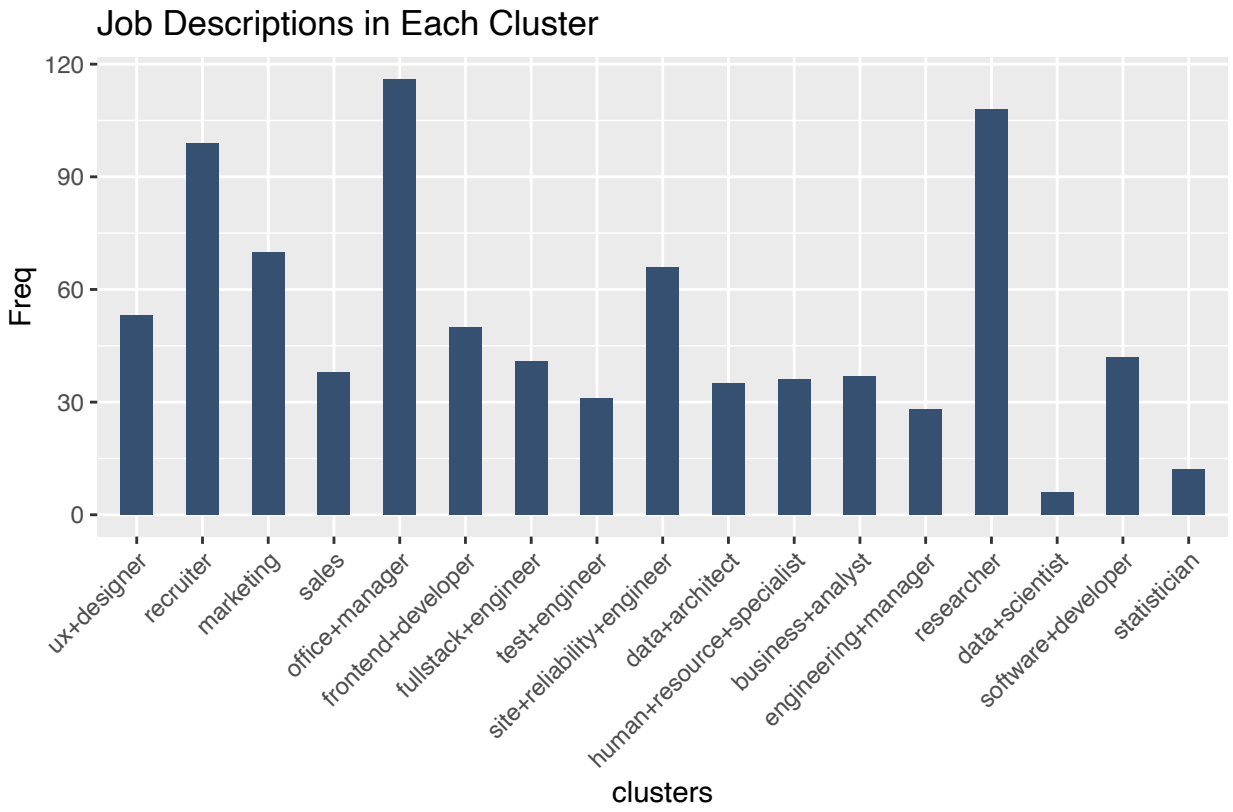

```r
#table(groups)


gghistogram(groups,
          #x = "gene", y = "citation_index",
          fill = "lightgray",
          #xlab = "Gene name", ylab = "Citation index",
          bins = 25,

          )
```




```
labels <- unique(job_desc$request_params$q)
tabled <- as.data.frame(table(groups))

# https://coolors.co/355070-6d597a-b56576-e56b6f-eaac8b
g <- ggplot(tabled, aes(groups, Freq))
g + geom_bar(stat="identity", width = 0.5, fill="#355070") +
      labs(title="Job Descriptions in Each Group",
           caption="Source: Indeed.com, Mar. 16, 2021") +
  theme(axis.text.x = element_text(angle=45, vjust=1, hjust = 1)) +
  scale_x_discrete(labels=c(labels))
```




```
clusters <- clustering
tabled <- as.data.frame(table(clusters))
g <- ggplot(tabled, aes(clusters, Freq))
g + geom_bar(stat="identity", width = 0.5, fill="#355070") +
      labs(title="Job Descriptions in Each Cluster",
           caption="Source: Indeed.com, Mar. 16, 2021") +
  theme(axis.text.x = element_text(angle=45, vjust=1, hjust = 1)) +
  scale_x_discrete(labels=c(labels))
```



Source: Indeed.com, Mar. 16, 2021

```
recommend_resume <- function(file_path = file_path, n_recommendations = n_recommendations){
  # give priority on the resume.
  # cooccurance- build up association between words
  data(stop_words) # Stop words.
  #real_resume_k <- readLines("~/Downloads/MarshResumeJan2021_Data_Mining.txt")
  #real_resume <- read_docx(file_path)
  real_resume <- readLines(file_path)
  real_resume <- str_replace_all(real_resume, pattern = '\"', replacement = "")

  real_resume <- paste(real_resume,collapse="")
  # transform it into a term document matrix
  resume <- quanteda::dfm(real_resume, verbose = FALSE)

  target_freq_resume <- as.numeric(resume)
  freqs_mat_resume <- as.matrix(resume)
  doc_freq_resume <- apply(freqs_mat_resume,2,function(x) mean(x>0))
  idf_resume <- 1/doc_freq_resume
  idf_mat_resume <- rep(idf_resume,nrow(freqs_mat_resume), byrow = TRUE, nrow = nrow(freqs_mat_resume))
  tf_idf_resume <- freqs_mat_resume * idf_mat_resume

  names(tf_idf_resume) <- colnames(tf_idf_resume)
  names(tf_idf_1) <- colnames(tf_idf_1)
  resume_desc <- as.matrix(bind_rows(tf_idf_resume,tf_idf_1))
  resume_desc[which(is.na(resume_desc))] <- 0
  names(resume_desc) <- colnames(resume_desc)
  similarities <- sim2(resume_desc, method = "cosine", norm = "l2")
  colnames(similarities) <- c('resume',1:nrow(tf_idf_1))
```

```
  get_similar_letters <- function(similarities, reference_letter, n_recommendations){
    sort(similarities[reference_letter, ], decreasing = TRUE)[1:(1 + n_recommendations)]
  }

  # how many keywords overlap

  index <-names(get_similar_letters(similarities, 1, n_recommendations))[-1]
  true_index <- as.numeric(index)-1
  similarity <- get_similar_letters(similarities, 1, n_recommendations)[-1]
  names(similarity) <- true_index

  position = c()
  for (i in 1:length(description)){
    for (j in 1:length(description[[i]])){
      if (is.na(description[[i]][j])){
        next
      }
      position <- append(position,position_names[j])
    }
  }

  recommended_position = position[true_index]
  names(recommended_position) = true_index
  cluster_name = clustering[true_index]
  names(cluster_name) = true_index

  normalize <- function(x) {
    return ((x - min(x)) / (max(x) - min(x)))
  }
  good_metric <- normalize(similarity)

  recommended_result = cbind(similarity,recommended_position,cluster_name, good_metric)

  diversity_metric <- rep(0, length(recommended_position))
  positions <- c()
  clusters <- c()
    for(i in 1:length(recommended_position)){
      if(recommended_position[i] %in% positions){
        diversity_metric[i] <- diversity_metric[i]+1}
      else {positions <- c(positions, recommended_position[i])}
      if(cluster_name[i] %in% clusters){
        diversity_metric[i] <- diversity_metric[i]+1}
      else {clusters <- c(clusters, cluster_name[i])}
    }

  recommended_result = cbind(recommended_result, diversity_metric)
  recommended_result = as.data.frame(head(recommended_result[order(recommended_result[,"diversity_metri
  return(recommended_result)
}
```

```
n_recommendations = 300
#file_path = "MarshResumeJan2021_Data_Mining.txt"
file_path = "Li_Jiaxin_resume.txt"
```

```
#file_path = "Li_Jiaxin_resume.docx"
position_names <- job_desc$request_params[[1]] #not na 1

result <- recommend_resume(file_path, n_recommendations)
```

```
#trying to text an example job desc. this didn't work

kate_example_job <- "The Sabin Center for Climate Change Law at Columbia Law School is seeking a few ex

desc_example <- c(desc, kate_example_job)
# index is 869
#similarities[869]
```

UX Designer Resume Sample https://www.monster.com/career-advice/article/sample-resume-ux-designer-experienced

Sales https://www.template.net/business/resume/sales-resume-template-download/

```
result1 <- recommend_resume("Li_Jiaxin_resume.txt", n_recommendations = 300)
result2 <- recommend_resume("MarshResumeJan2021_Data_Mining.txt", n_recommendations = 300)
result3 <- recommend_resume("Retail_Sales_Associate_Resume.txt", n_recommendations = 300)
formattable(result3)
```

similarity

recommended_position

cluster_name

good_metric

diversity_metric

160

0.132908311934115

sales

11

1

0

415

0.126747141035862

engineering+manager

3

0.940583730443776

0

270

0.124320320402496

test+engineer

1

0.917180284686814

2

517

0.0293158806106638

researcher

7

0.000989272145741958

2

68

0.0293100850925949

recruiter

10

0.000933382109661852

2

482

0.0292132980217479

researcher

5

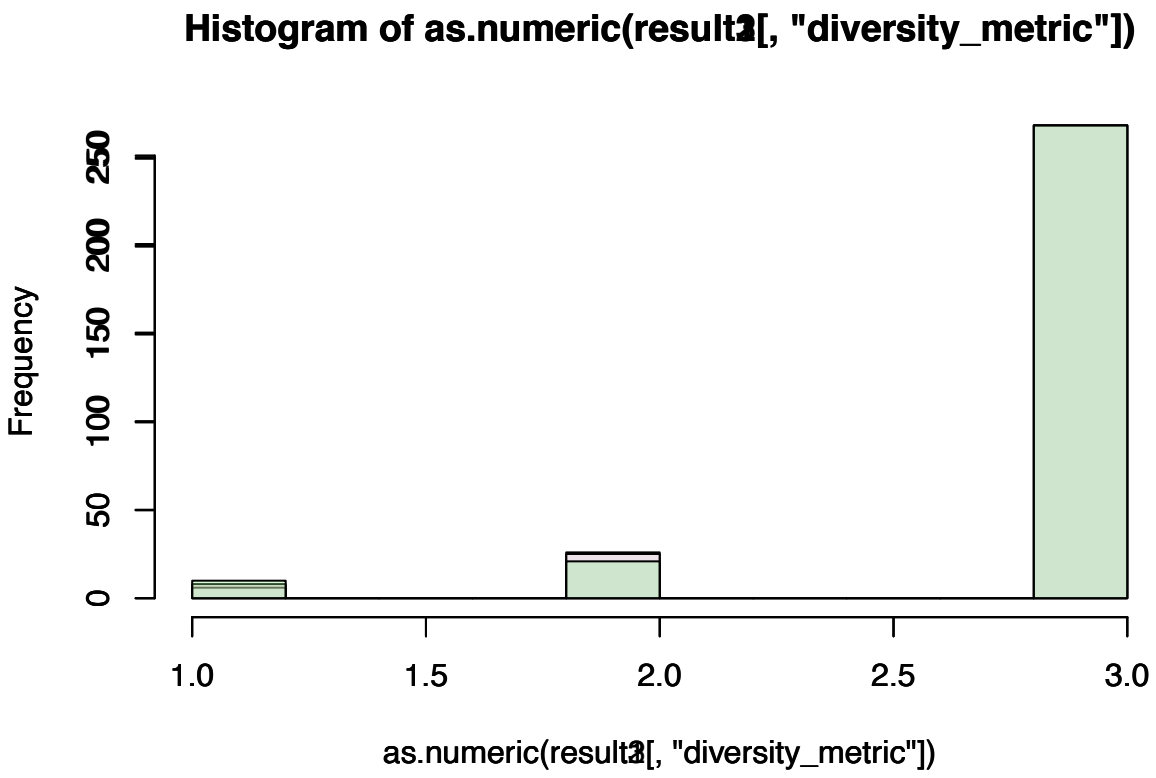
0

2

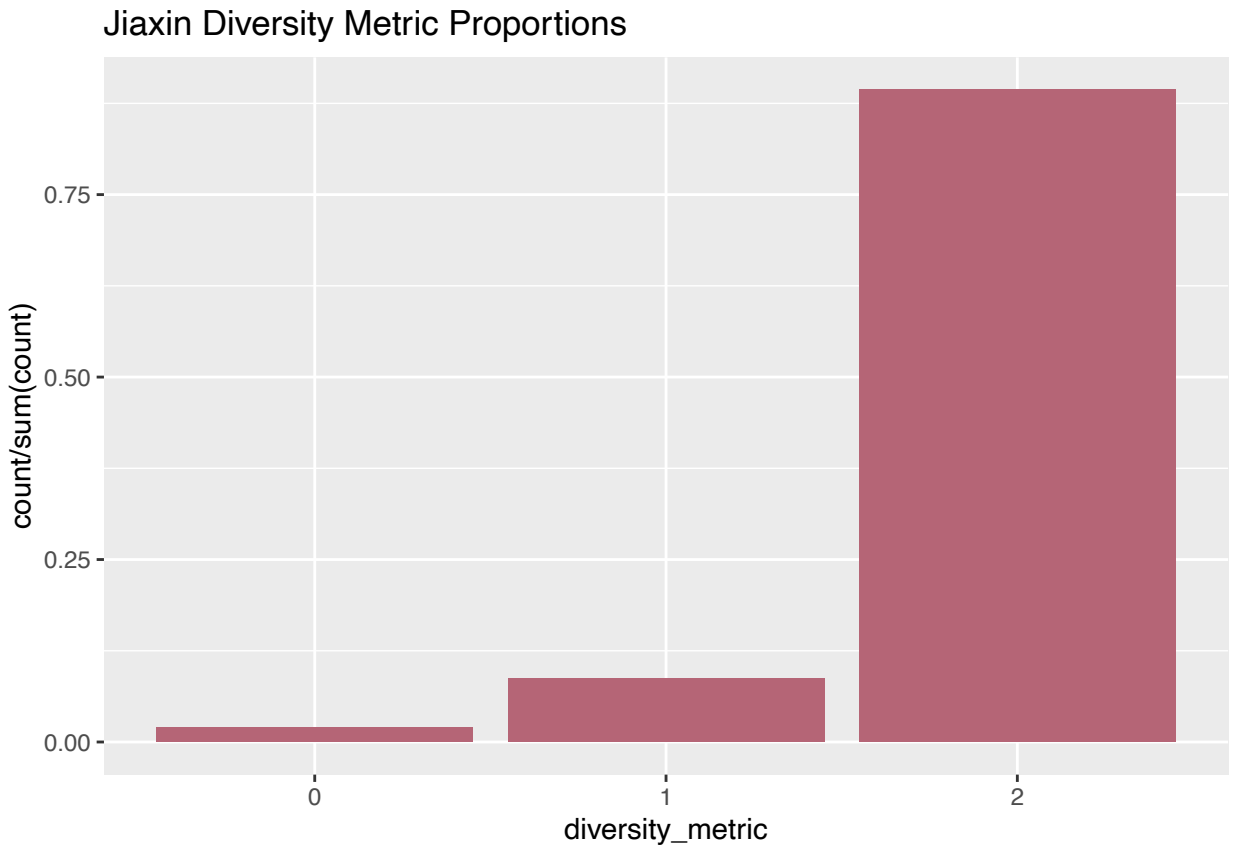
```
#plot(result2[,"similarity"], col = "red", ylim=c(0.05,.14), xlim=c(1,10),
#     main = "Top 10 Job Desciptions vs Cosine Similarity with Resume",
#     ylab= "Cosine Similarity",
#     pch=16
#     )
#par(new = TRUE)
#plot(result1[,"similarity"], type="p",col = "blue", ylim=c(0.05,.14), xlim=c(1,10), ylab=" ", pch=16)
#par(new = TRUE)
#plot(result3[,"similarity"], col = "green4", ylim=c(0.05,.14), xlim=c(1,10), ylab=" ", pch=16)
#legend("topright", legend=c("Kate", "Jiaxin", "Sales Example"), fill=c("red","blue","green4"), cex=.8)
```

Our good metric is cosine similarity of the job description and the resume. It is then normalized to a score between 0 and 1 so you can tell how "good" of a recommendation it is, regardless of the raw cosine similarity.

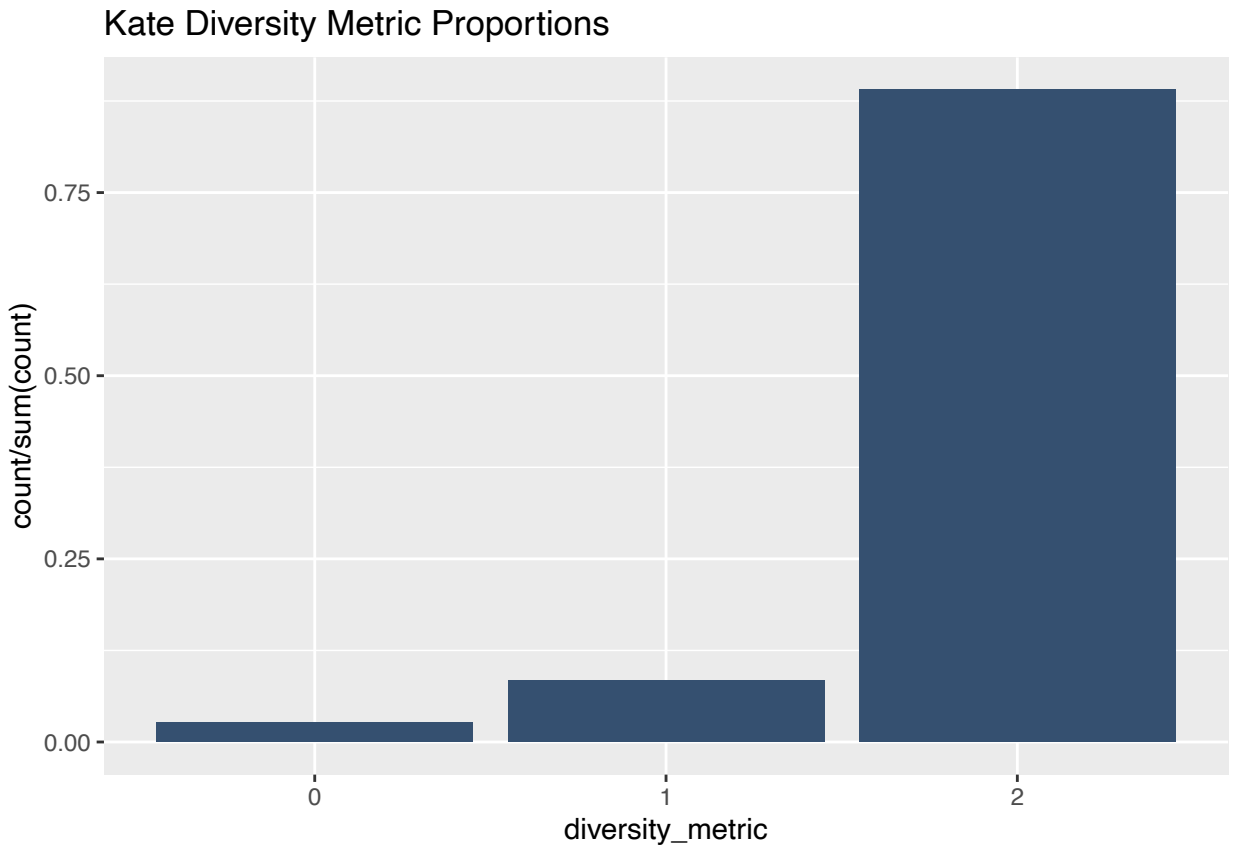
```
library(ggplot2)

c1 <- rgb(173,216,230,max = 255, alpha = 80, names = "lt.blue")
c2 <- rgb(255,192,203, max = 255, alpha = 80, names = "lt.pink")
c3 <- rgb(144,238,144, max = 255, alpha = 80, names = "lt.green")

hist(as.numeric(result1[,"diversity_metric"]),  col = c1)
par(new = TRUE)
hist(as.numeric(result2[,"diversity_metric"]), col = c2)
par(new = TRUE)
hist(as.numeric(result3[,"diversity_metric"]), col = c3)
```



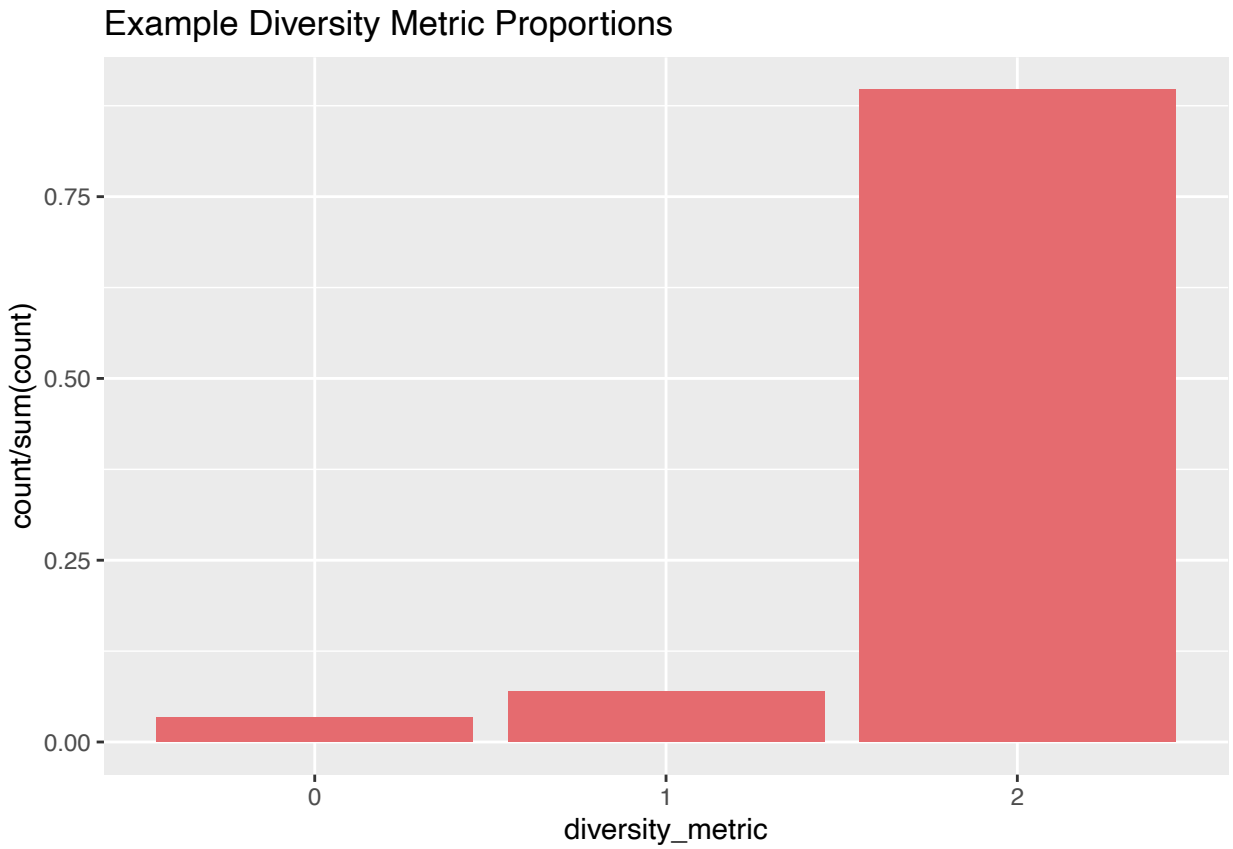

```
#data <- data.frame(Jiaxin = as.numeric(result1[,"diversity_metric"]), Kate = as.numeric(result2[,"dive
#                   Example = as.numeric(result3[,"diversity_metric"]))

par(mfrow=c(3,1))
ggplot(result1, aes(x=diversity_metric)) +
  geom_bar(fill = "#b56576", aes(y=..count../sum(..count..))) +
  ggtitle("Jiaxin Diversity Metric Proportions")
```




```
ggplot(result2, aes(x=diversity_metric)) +
  geom_bar(fill = "#355070", aes(y=..count../sum(..count..))) +
  ggtitle("Kate Diversity Metric Proportions")
```




```
ggplot(result3, aes(x=diversity_metric)) +
  geom_bar(fill = "#e56b6f", aes(y=..count../sum(..count..))) +
  ggtitle("Example Diversity Metric Proportions")
```




```
plot1 <- ggplot() +
  geom_bar(aes(diversity_metric))


#ggplot(result1, aes(x=, y=Kate)) +
#  geom_bar(position="dodge", stat="identity")

plot1 <- ggplot() +
  geom_bar(aes(x=as.numeric(result1[,"diversity_metric"]), y=..density..),  fill="lightblue") +
  geom_bar(aes(x=as.numeric(result2[,"diversity_metric"], y=..density..),  fill="darkblue"))
```