Bios 611 HW5

Li Jiang

10/29/2024

Q1: Explain functions

'md5_hash': return a MD5 hash from the given URL

'cache_path': return the cache path from the given URL

'fetch_raw': retrieve the raw HTML content from the given URL without caching and adding a random delay (6 to 12 sec) to avoid excessive requests. Also, logs the time of each request.

'fetch': First, check if there's local cache file for the given URL, if not, it fetches it and saves it to the cache and returns a parsed HTML object.

'episode_lists_urls': get each episode's url from the webpage

'tokenize_and_count': Process the input text by removing punctuation, converting to lowercase, tokenizing, filtering out stopwords and count the frequency of each word

'get_text_of_episodes': extract text content from the episode URL

'get_word_counts_for_episodes': calculate the word frequency for the text of each episode

'get_total_word_count': sum word counts frequency from all episodes

'convert_to_word_count_vectors': convert the word counts into vector

'write_word_counts_to_csv': write the vector for each url into a csv file

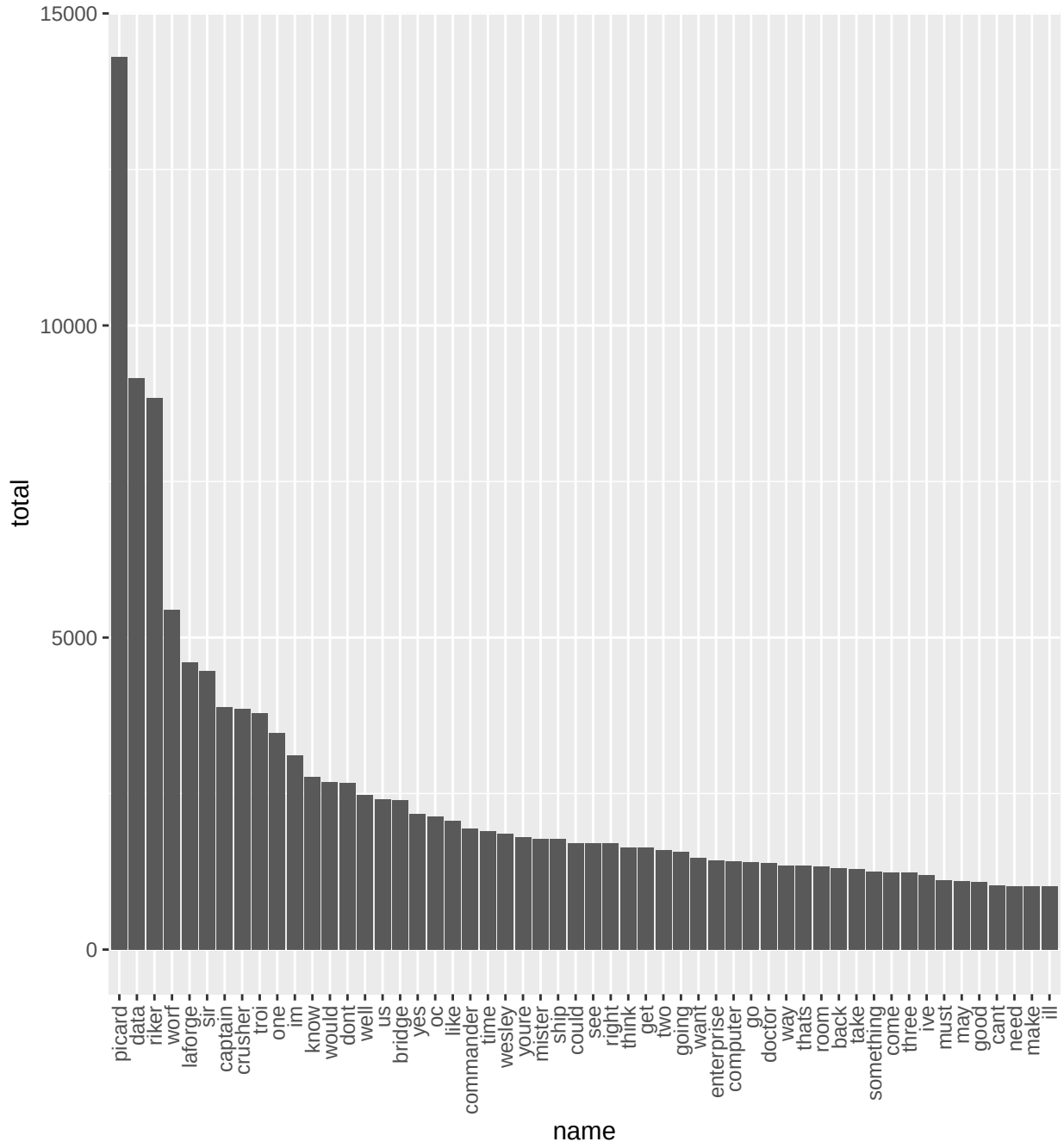
Q2: Visualization

Part a

```
d <- read_csv("episode_word_counts.csv")
```

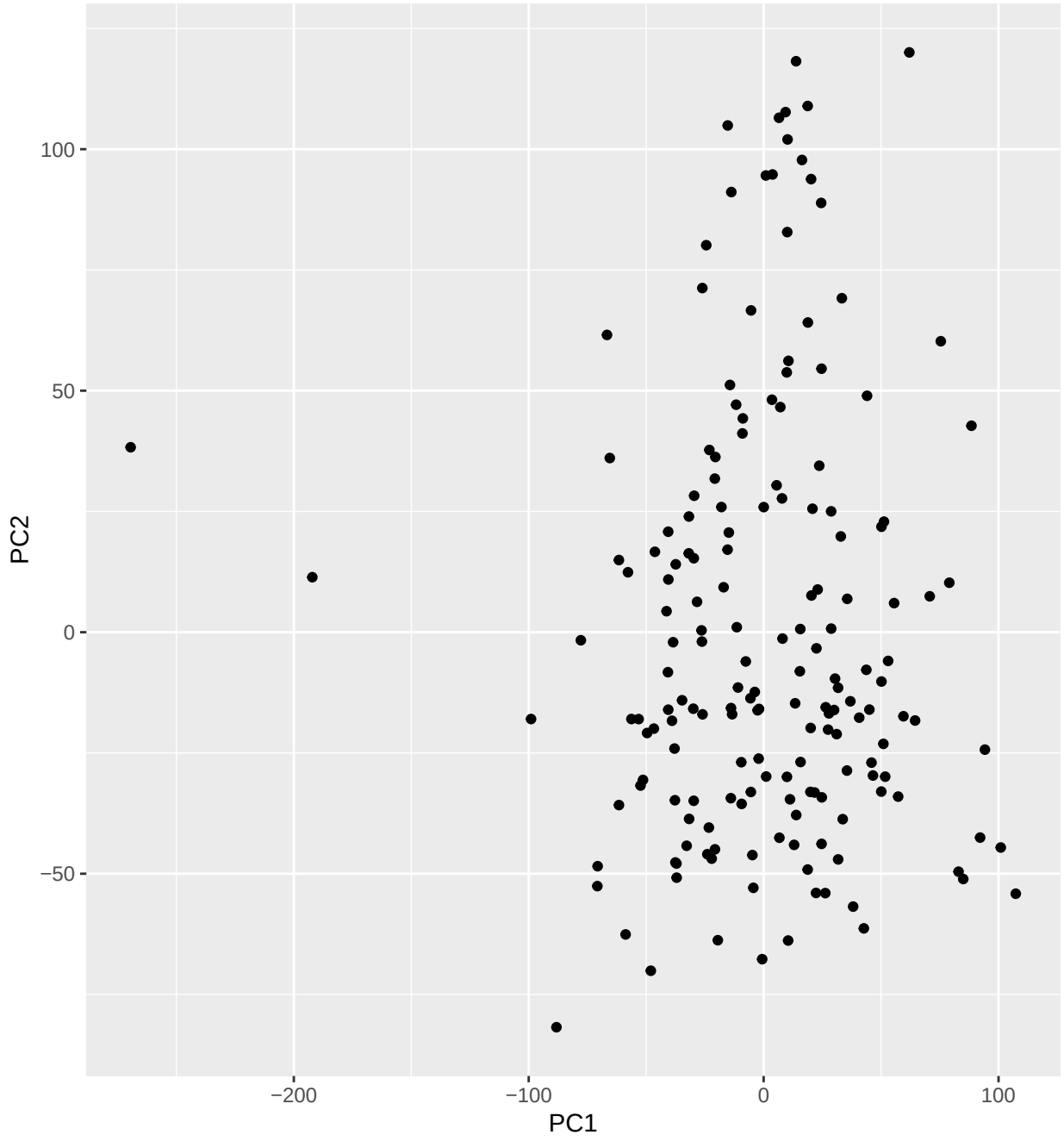
```
## Rows: 176 Columns: 3149
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (1): Episode URL
## dbl (3148): captains, log, stardate, destination, planet, four, beyond, lies...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
m <- d %>% select(-`Episode URL`) %>% colSums() %>% as_tibble()
d <- d %>% select(-`Episode URL`) %>% mutate(`Episode URL` = d %>% pull(`Episode URL`))

l <- d %>% pivot_longer(cols=captains:devron) %>%
    group_by(name) %>%
    summarise(total=sum(value))

l <- l %>% mutate(name = factor(name, l %>% arrange(desc(total)) %>% pull(name)))

ggplot(l %>% filter(total > 1000), aes(name, total)) + geom_col() + theme(axis.text.x = element_text(ang
```



Part b

```
pca <- prcomp(d %>% select(-`Episode URL`) %>% as.matrix())
pcs <- pca$x %>% as_tibble()
pcs1 <- pcs[,1:2] %>% as.data.frame()
pcs %>% ggplot(aes(PC1,PC2))+geom_point()
```

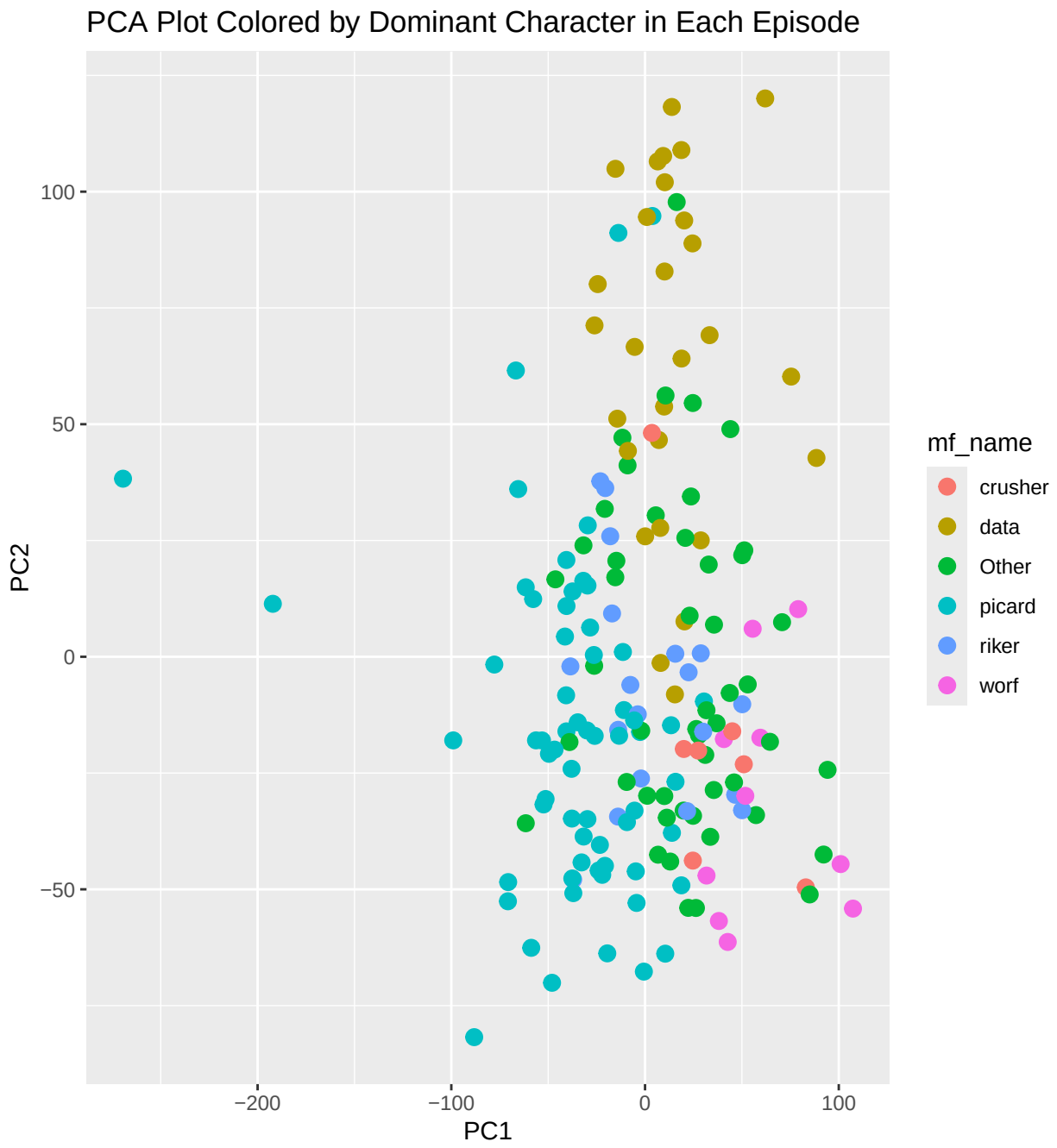


Part c

```
mf_chr <- d %>%
    pivot_longer(captains:devron) %>%
    group_by(`Episode URL`) %>%
    arrange(desc(value)) %>%
    slice_head(n = 1) %>%
    mutate(rank = row_number()) %>%
    select(mf_name = name, mf_value = value)
```

```
## Adding missing grouping variables: `Episode URL`
```

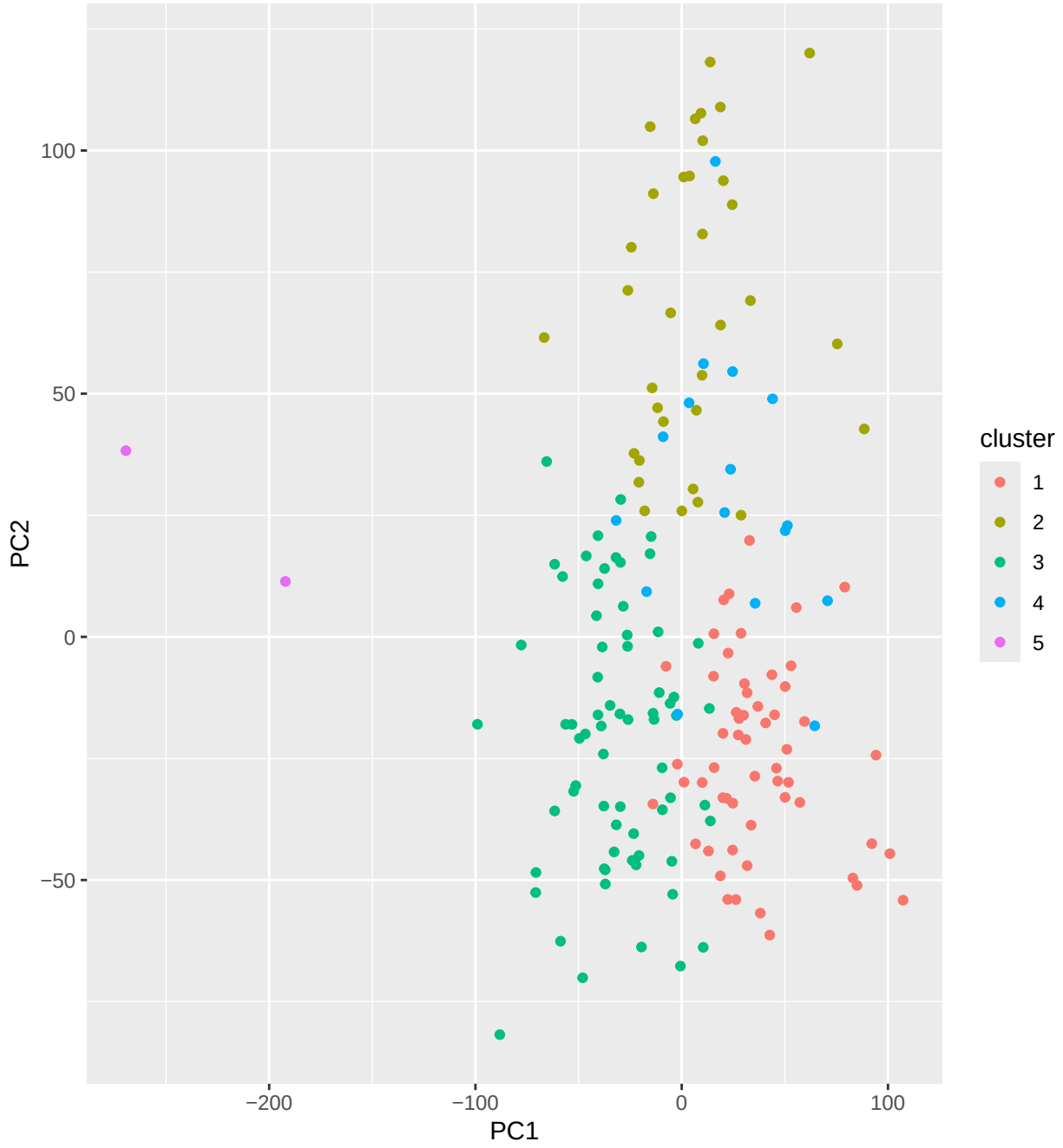
```
mf_chr<- mf_chr [,-1]


chr_count <- mf_chr %>% group_by(mf_name) %>% tally() %>% arrange(desc(n))

pca_mf <- cbind(mf_chr, pcs1)

top5_name <- as.character(chr_count[1:5, 1][[1]])

pca_mf <- pca_mf %>%
    mutate(mf_name = ifelse(mf_name %in% top5_name, mf_name, "Other"))

ggplot(pca_mf, aes(x = PC1, y = PC2, color = mf_name)) +
    geom_point(size = 3) +
    labs(title = "PCA Plot Colored by Dominant Character in Each Episode",
         x = "PC1", y = "PC2")
```



The top 5 most often names are picard, data, riker, worf, crusher. Picard-dominated episodes appear across PC1, indicating his consistent presence across various themes and episode types. And data-dominated episodes appear in specific area.

Q3: Clustering

```
pca_mf$cluster <- factor(kmeans(d %>% select(-`Episode URL`), centers = 5)$cluster)
pca_mf %>% ggplot(aes(PC1,PC2,color=cluster)) + geom_point()
```



It is similar to 2c.

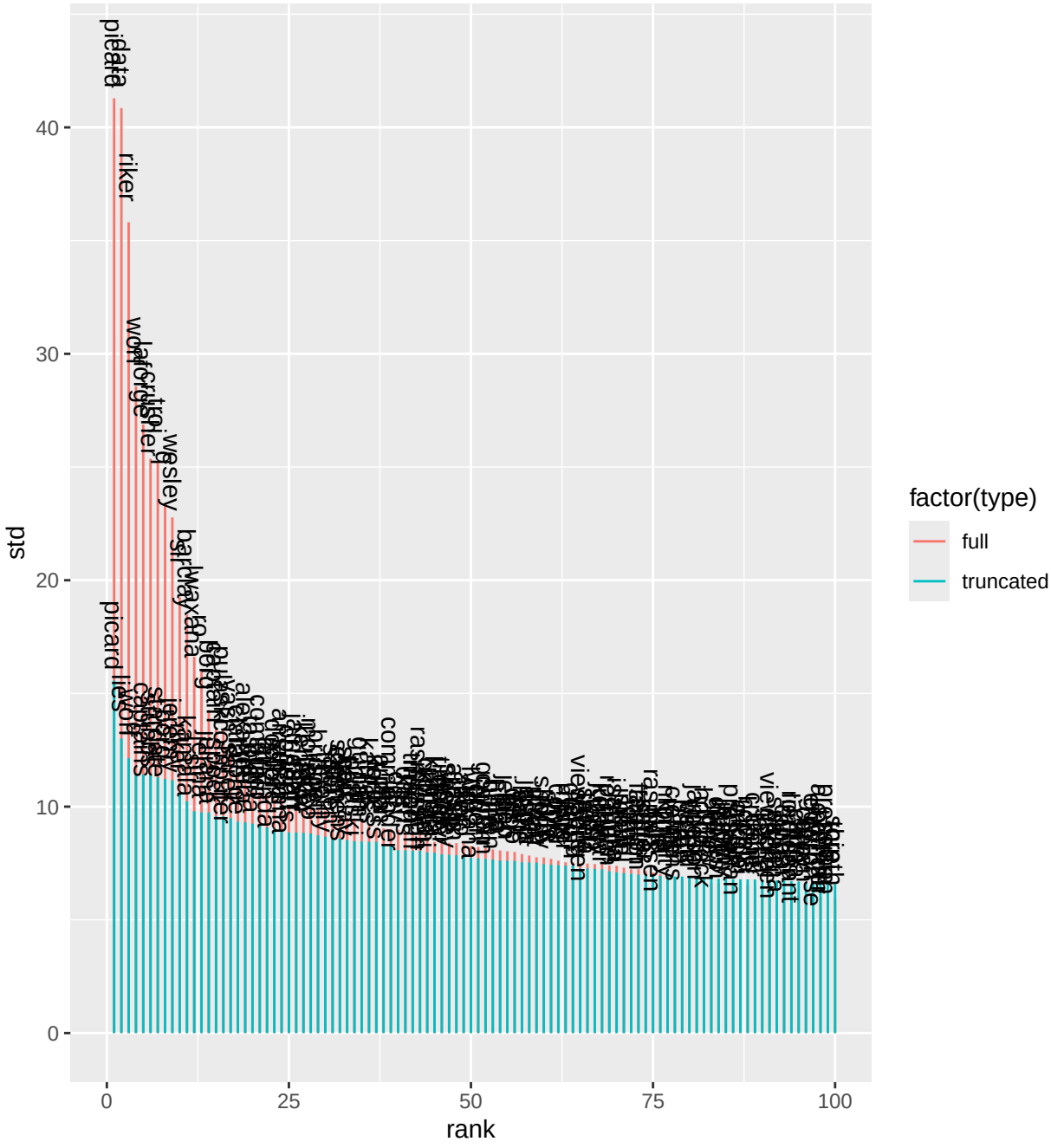
Q4: Classifier

```
stds <- d %>%
    summarise(across(captains:devron, sd)) %>%
    pivot_longer(captains:devron) %>%
    rename(std=value) %>%
    arrange(desc(std)) %>%
```

```
    mutate(name = factor(name, name)) %>%
    mutate(rank=1:nrow(.))

pcs <- pca$x %>% as_tibble() %>% mutate(across(PC71:PC176, ~ 0)) %>% as.matrix()

truncated_stds <- pca$rotation %*% pcs %>% t() %>% as_tibble() %>%
    summarise(across(captains:devron, sd)) %>%
    pivot_longer(captains:devron) %>%
    rename(std=value) %>%
    arrange(desc(std)) %>%
    mutate(name = factor(name, name)) %>%
    mutate(rank=1:nrow(.))

std_df <- bind_rows(stds %>% mutate(type="full"),
                    truncated_stds %>% mutate(type="truncated"))

ggplot(std_df, aes(rank, std)) +
    geom_segment(aes(x=rank, xend=rank, y=0, yend=std, color=factor(type))) + xlim(0, 100) +
    geom_text(aes(label=name, y=std+2), angle=-90)
```

```
## Warning: Removed 6096 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```

```
## Warning: Removed 6096 rows containing missing values or values outside the scale range
## (`geom_text()`).
```




```
d_shrunk <- d %>%
    select(all_of(std_df %>% filter(type=="truncated" & rank <= 70) %>% pull(name))) %>%
    mutate(first_half=(row_number() < max(row_number())/2)*1)

library(gbm)
```

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.co
```

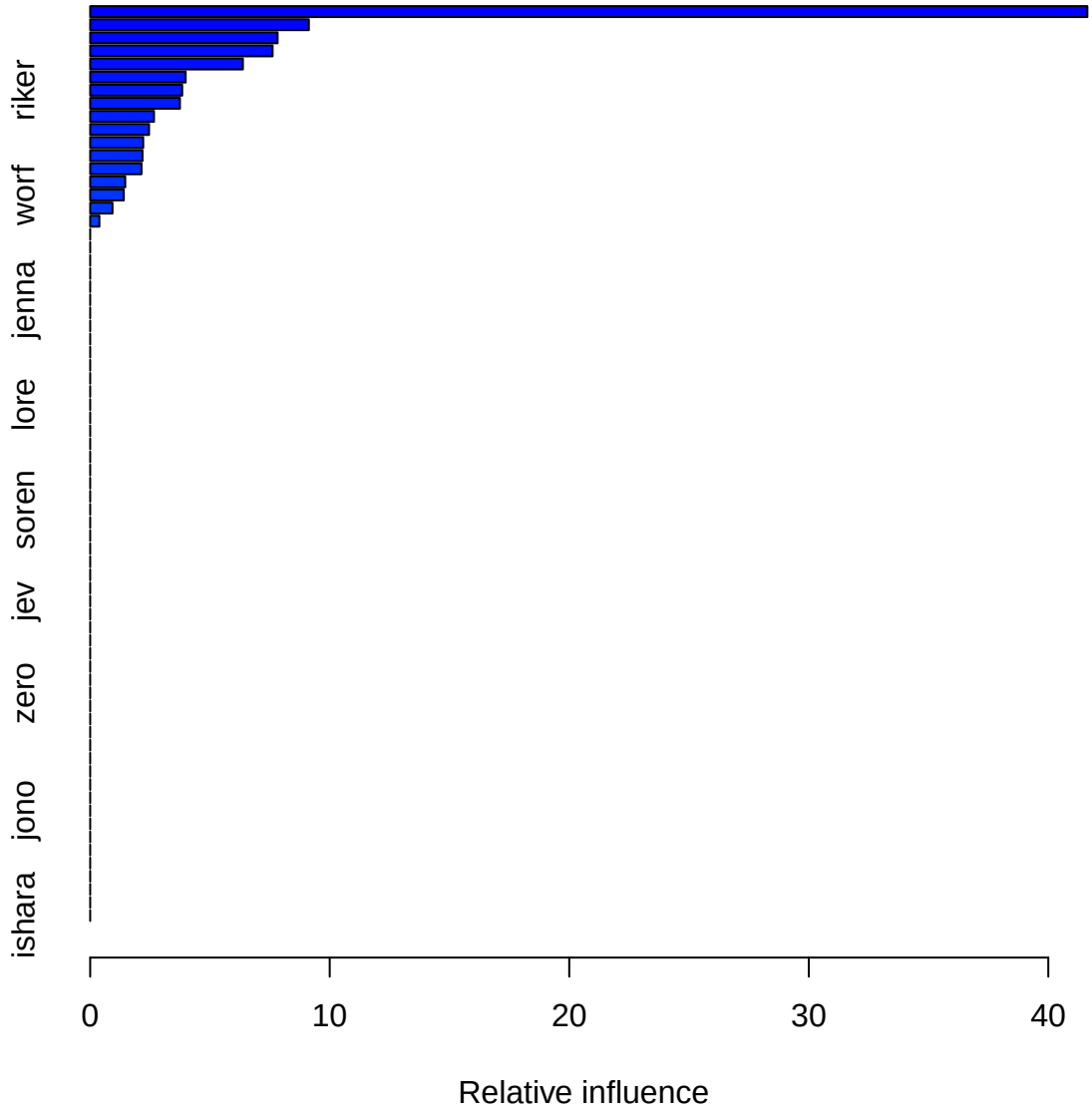
```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
train_i <- runif(nrow(d_shrunk)) < 0.75;

train <- d_shrunk %>% filter(train_i);
test <- d_shrunk %>% filter(!train_i);

model <- gbm(first_half ~ ., data=train)
```

```
## Distribution not specified, assuming bernoulli ...

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 9: jeremy has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 24: susanna has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 26: marr has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 35: tam has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 40: jev has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 54: liko has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 58: nuria has no variation.
```

```
summary(model)
```




```
##                var    rel.inf
## wesley     wesley 41.6312337
## captain   captain  9.1300516
## picard     picard  7.8255167
## obrien     obrien  7.6153515
## tasha       tasha  6.3745707
## sir           sir  3.9865261
## riker       riker  3.8412927
## stardate stardate  3.7471870
## doctor     doctor  2.6644346
## laforge   laforge  2.4579258
```

```
## computer computer  2.2140904
## captains captains  2.1842807
## pulaski   pulaski  2.1444385
## oc             oc  1.4636689
## worf         worf  1.4031468
## four         four  0.9317518
## lies         lies  0.3845324
## q               q  0.0000000
## stubbs     stubbs  0.0000000
## borg         borg  0.0000000
## jeremy     jeremy  0.0000000
## kamala     kamala  0.0000000
## jenna       jenna  0.0000000
## juliana   juliana  0.0000000
## spock       spock  0.0000000
## sarek       sarek  0.0000000
## nella       nella  0.0000000
## okona       okona  0.0000000
## fajo         fajo  0.0000000
## amanda     amanda  0.0000000
## lore         lore  0.0000000
## susanna   susanna  0.0000000
## armus       armus  0.0000000
## marr         marr  0.0000000
## alkar       alkar  0.0000000
## jellico   jellico  0.0000000
## scott       scott  0.0000000
## timothy   timothy  0.0000000
## soren       soren  0.0000000
## clemens   clemens  0.0000000
## barclay   barclay  0.0000000
## tam           tam  0.0000000
## jr             jr  0.0000000
## kahless   kahless  0.0000000
## john         john  0.0000000
## jev           jev  0.0000000
## korris     korris  0.0000000
## macduff   macduff  0.0000000
## russell   russell  0.0000000
## kolrami   kolrami  0.0000000
## ardra       ardra  0.0000000
## jason       jason  0.0000000
## zero         zero  0.0000000
## lwaxana   lwaxana  0.0000000
## odan         odan  0.0000000
## liko         liko  0.0000000
## leah         leah  0.0000000
## dirgo       dirgo  0.0000000
## satie       satie  0.0000000
## nuria       nuria  0.0000000
## wyatt       wyatt  0.0000000
## jono         jono  0.0000000
## kyle         kyle  0.0000000
## riva         riva  0.0000000
```

```
## finn         finn  0.0000000
## guinan     guinan  0.0000000
## soong       soong  0.0000000
## vash         vash  0.0000000
## kehleyr   kehleyr  0.0000000
## ishara     ishara  0.0000000
```

```
predict_part <- predict(model, newdata = test, type = "response")
```

```
## Using 100 trees...
```

```
predict_part
```

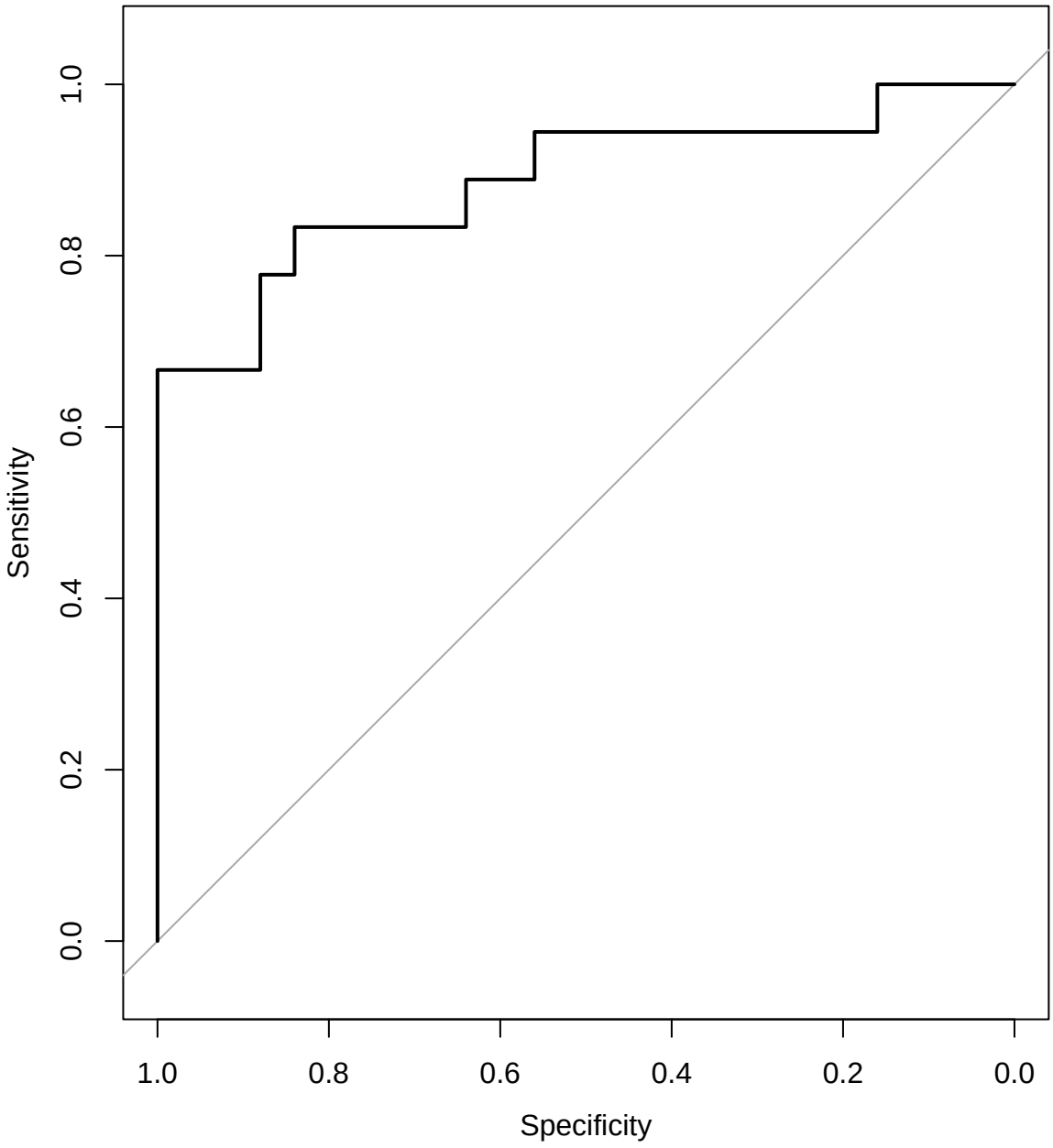
```
##  [1] 0.60170070 0.99573002 0.82005315 0.43216239 0.95104593 0.66202207
##  [7] 0.97852929 0.14890888 0.98249392 0.91134607 0.09735571 0.42808708
## [13] 0.98550141 0.79692446 0.84102386 0.91637967 0.35422121 0.03560656
## [19] 0.47726716 0.04638075 0.26657591 0.45037571 0.09976602 0.07372857
## [25] 0.05532362 0.06995522 0.07770646 0.25106809 0.17453177 0.02450838
## [31] 0.15287921 0.07648308 0.59330870 0.38122921 0.05957594 0.03344363
## [37] 0.06462630 0.05342155 0.10371848 0.23464450 0.01354863 0.02686260
## [43] 0.03581457
```

```
roc <- roc(test$first_half,predict_part)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc)
```



The AUC is

```
roc$auc
```

```
## Area under the curve: 0.8867
```

Q5: load into python

```
import pandas as pd
df = pd.read_csv("episode_word_counts.csv")
print(f"the data contains {len(df)} rows")
```

```
## the data contains 176 rows
```

```
df_cleaned = df[df.iloc[:, 1:].select_dtypes(include='number').sum(axis=1) >= 100]
df_cleaned.to_csv("episode_word_counts_cleaned.csv", index=False)
```