Comparing the Readability of All Seven Harry Potter Books

true

2022-02-27

First, load in libraries and text files:

```
#install.packages("devtools")
#install.packages("tidytext")
#install.packages("plyr")
#install.packages("tidyverse")
#install.packages("quanteda")
#install.packages("quanteda.textstats")

# load libraries
library(devtools)
```

```
## Loading required package: usethis
```

```
#devtools::install_github("bradleyboehmke/harrypotter")
library(harrypotter)
library(tidytext)
library(plyr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
```

```
library(quanteda)
```

```
## Package version: 3.2.0
## Unicode version: 13.0
## ICU version: 69.1
```

```
## Parallel computing: 4 of 4 threads used.
```

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

```
library(quanteda.textstats)
library(ggplot2)
```

As a reminder, we have seven books — each stored as a character vector where each chapter is an element in that vector — now available in our workspace. These are:

1. `philosophers_stone`: Harry Potter and the Philosophers Stone (1997)
2. `chamber_of_secrets`: Harry Potter and the Chamber of Secrets (1998)
3. `prisoner_of_azkaban`: Harry Potter and the Prisoner of Azkaban (1999)
4. `goblet_of_fire`: Harry Potter and the Goblet of Fire (2000)
5. `order_of_the_phoenix`: Harry Potter and the Order of the Phoenix
6. `half_blood_price`: Harry Potter and the Half-Blood Prince (2005)
7. `deathly_hallows`: Harry Potter and the Deathly Hallows (2007)

As you'll recall, we want to convert these to corpus objects that are easier to work with.

Convert to corpus objects [note, pulling code from Week 2 tutorial]:

```
# list out the object (book) names that we need
myBooks <- c("philosophers_stone", "chamber_of_secrets", "prisoner_of_azkaban", "goblet_of_fire", "orde
  "half_blood_prince", "deathly_hallows")

# create loop.
for (i in 1:length(myBooks)){

  # create corpora
  corpusCall <- paste(myBooks[i],"_corpus <- corpus(",myBooks[i],")", sep = "")
  eval(parse(text=corpusCall))

  # change document names for each chapter to include the book title. If you don't do this, the documen
  namesCall <- paste("tmpNames <- docnames(",myBooks[i],"_corpus)", sep = "")
  eval(parse(text=namesCall))
  bindCall <- paste("docnames(",myBooks[i],"_corpus) <- paste(\"",myBooks[i],"\", tmpNames, sep = \"-\"
  eval(parse(text=bindCall))

  # create summary data
  summaryCall <- paste(myBooks[i],"_summary <- summary(",myBooks[i],"_corpus)", sep = "")
  eval(parse(text=summaryCall))

  # add indicator
  bookCall <- paste(myBooks[i],"_summary$book <- \"",myBooks[i],"\"", sep = "")
```

```
  eval(parse(text=bookCall))

  # add chapter indicator
  chapterCall <- paste(myBooks[i],"_summary$chapter <- as.numeric(str_extract(",myBooks[i],"_summary$Te
  eval(parse(text=chapterCall))

  # add meta data to each corpus
  metaCall <- paste("docvars(",myBooks[i],"_corpus) <- ",myBooks[i],"_summary", sep = "")
  eval(parse(text=metaCall))

}

# once the loop finishes up, check to make sure you've created what you want
#docvars(deathly_hallows_corpus)
```

Now that we have a corpus for each book, we'll move to tokenization (back to Week 5 Tutorial code)

```
# the default breaks on white space
#philosophers_stone_tokens <- tokens(philosophers_stone_corpus)
#print(philosophers_stone_tokens)

#Going to attempt to do it as a loop!

# create loop.
for (i in 1:length(myBooks)){

  # create tokens
  tokensCall <- paste(myBooks[i],"_tokens <- tokens(",myBooks[i],")", sep = "")
  eval(parse(text=tokensCall))

}
```

Next we're going to try and do some preprocessing of our data:

```
#We're going to drop punctuation and numbers, namely because I do not think that including them in the

#I'm not sure how you'd write the loop for this, so I'm just going to do it manually. I'm also realizin

# you can also drop punctuation as well as numbers

philosophers_stone_tokens <- tokens(philosophers_stone_corpus,
    remove_punct = T,
    remove_numbers = T)

chamber_of_secrets_tokens <- tokens(chamber_of_secrets_corpus,
    remove_punct = T,
    remove_numbers = T)

prisoner_of_azkaban_tokens <- tokens(prisoner_of_azkaban_corpus,
    remove_punct = T,
    remove_numbers = T)

goblet_of_fire_tokens <- tokens(goblet_of_fire_corpus,
```

```
    remove_punct = T,
    remove_numbers = T)

order_of_the_phoenix_tokens <- tokens(order_of_the_phoenix_corpus,
    remove_punct = T,
    remove_numbers = T)

half_blood_prince_tokens <- tokens(half_blood_prince_corpus,
    remove_punct = T,
    remove_numbers = T)

deathly_hallows_tokens <- tokens(deathly_hallows_corpus,
    remove_punct = T,
    remove_numbers = T)
```

Now we're going to remove stopwords:

```
# remove stopwords from our tokens object
# again there's probably a way to code this to loop, but I don't know how!

philosophers_stone_tokens <- tokens_select(philosophers_stone_tokens,
                                           pattern = stopwords("en"),
                                           selection = "remove")


chamber_of_secrets_tokens <- tokens_select(chamber_of_secrets_tokens,
                                           pattern = stopwords("en"),
                                           selection = "remove")

prisoner_of_azkaban_tokens <- tokens_select(prisoner_of_azkaban_tokens,
                                           pattern = stopwords("en"),
                                           selection = "remove")

goblet_of_fire_tokens <- tokens_select(goblet_of_fire_tokens,
                                           pattern = stopwords("en"),
                                           selection = "remove")

order_of_the_phoenix_tokens <- tokens_select(order_of_the_phoenix_tokens,
                                           pattern = stopwords("en"),
                                           selection = "remove")

half_blood_prince_tokens <- tokens_select(half_blood_prince_tokens,
                                           pattern = stopwords("en"),
                                           selection = "remove")

deathly_hallows_tokens <- tokens_select(deathly_hallows_tokens,
                                           pattern = stopwords("en"),
                                           selection = "remove")
```

As I'm looking at the code, I now realize that I didn't actually need to do any of this, because this is all done on the tokens, and readability is calculated on the corpus! Oh well. If at some point I want to do anything with this, I've got it coded now (hopefully correctly!)

So, one of the things I'm wondering is, can I create one full HP corpus and calculate readability on that, by title?

Note: this was code from Week 2 Tutorial

```
# list out the object (book) names that we need
myBooks <- c("philosophers_stone", "chamber_of_secrets", "prisoner_of_azkaban", "goblet_of_fire", "orde
  "half_blood_prince", "deathly_hallows")

# create loop.
for (i in 1:length(myBooks)){

  # create corpora
  corpusCall <- paste(myBooks[i],"_corpus <- corpus(",myBooks[i],")", sep = "")
  eval(parse(text=corpusCall))

  # change document names for each chapter to include the book title. If you don't do this, the documen
  namesCall <- paste("tmpNames <- docnames(",myBooks[i],"_corpus)", sep = "")
  eval(parse(text=namesCall))
  bindCall <- paste("docnames(",myBooks[i],"_corpus) <- paste(\"",myBooks[i],"\", tmpNames, sep = \"-\"
  eval(parse(text=bindCall))

  # create summary data
  summaryCall <- paste(myBooks[i],"_summary <- summary(",myBooks[i],"_corpus)", sep = "")
  eval(parse(text=summaryCall))

  # add indicator
  bookCall <- paste(myBooks[i],"_summary$book <- \"",myBooks[i],"\"", sep = "")
  eval(parse(text=bookCall))

  # add chapter indicator
  chapterCall <- paste(myBooks[i],"_summary$chapter <- as.numeric(str_extract(",myBooks[i],"_summary$Te
  eval(parse(text=chapterCall))

  # add meta data to each corpus
  metaCall <- paste("docvars(",myBooks[i],"_corpus) <- ",myBooks[i],"_summary", sep = "")
  eval(parse(text=metaCall))

}

# once the loop finishes up, check to make sure you've created what you want
#docvars(deathly_hallows_corpus)
```

Now to create one overall HP corpus:

```
# create combined corpora of the first 7 Harry Potter books.
harry_potter_corpus <- c(philosophers_stone_corpus, chamber_of_secrets_corpus, prisoner_of_azkaban_corpu
  half_blood_prince_corpus, deathly_hallows_corpus)
#summary(harry_potter_corpus)
#docvars(harry_potter_corpus)
```

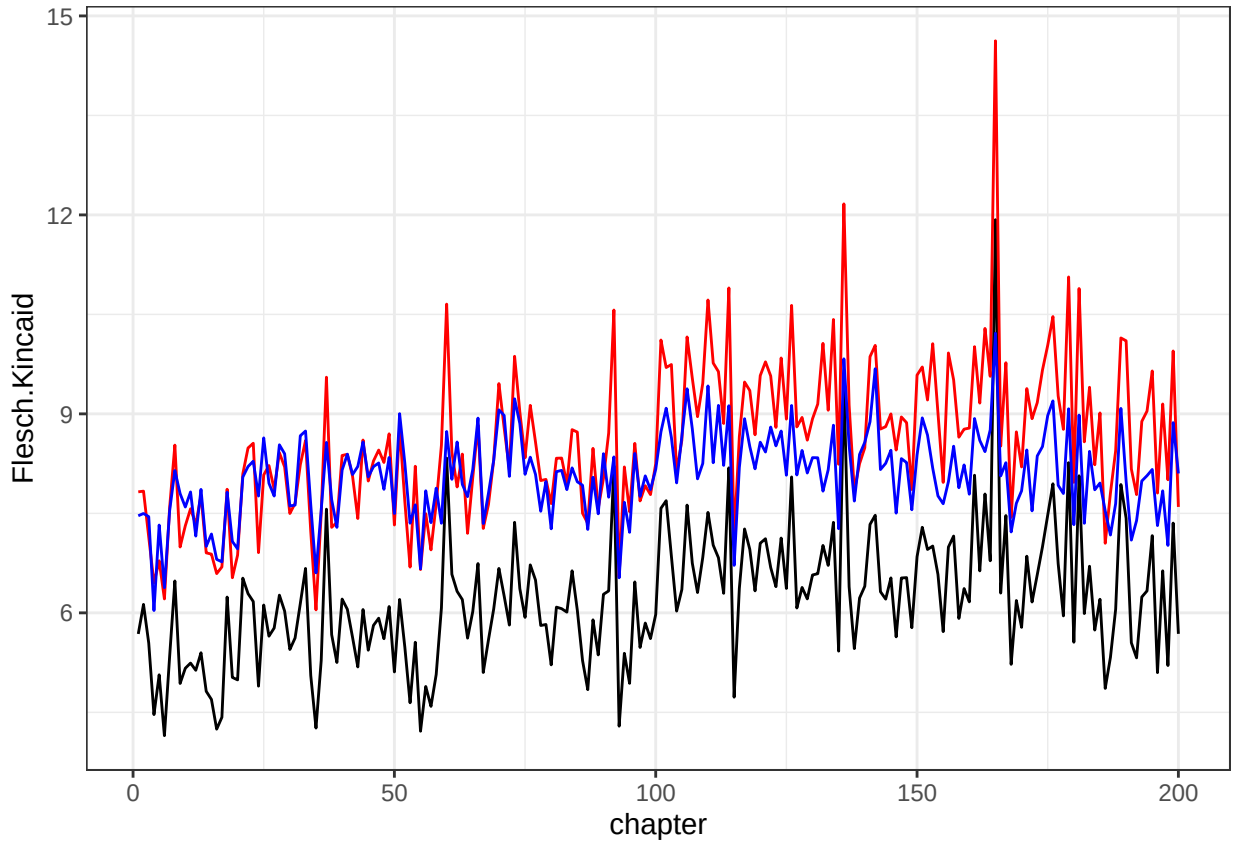
From here, we'll calculate readability by book (Week 5 Tutorial code)

```r
#readbaility for the whole corpis
readability <- textstat_readability(harry_potter_corpus,
                                    measure = c("Flesch.Kincaid", "FOG", "Coleman.Liau.grade"))
# add in a chapter number
readability$chapter <- c(1:nrow(readability))

# add in a chapter number
readability$book <- harry_potter_corpus$book


# look at the dataset
#head(readability)
#tail(readability)

# plot results
ggplot(readability, aes(x = chapter)) +
  geom_line(aes(y = Flesch.Kincaid), color = "black") +
  geom_line(aes(y = FOG), color = "red") +
  geom_line(aes(y = Coleman.Liau.grade), color = "blue") +
  theme_bw()
```



Okay, while this is interesting, I'm actually more interested in between-book comparisons, so I'm going to look at means.

```
readability_means <-   readability %>%                                          # Specify data frame
     group_by(book) %>%                                                      # Specify group indicator
    summarise_at(vars(Flesch.Kincaid, FOG, Coleman.Liau.grade, chapter),        # Specify column
                 list(name = mean))                                      # Specify function
#readability_means

#This gives us the means for comparison, but it's bugging me that the books aren't in order. One way to

#also I have no idea why it appended names to my variables?

#readability_means$chapter_name <- as.numeric(readability_means$chapter_name) - turns out, don't have t

#rearrange the data by chapter name which is the proxy for book order
readability_means <- arrange(readability_means, chapter_name)
#readability_means

#subset to leave the order column out so it looks nice
print_means <- select(readability_means, book, Flesch.Kincaid_name, FOG_name, Coleman.Liau.grade_name)
print_means
```

```
## # A tibble: 7 x 4
##   book                 Flesch.Kincaid_name FOG_name Coleman.Liau.grade_name
##   <chr>                              <dbl>    <dbl>                   <dbl>
## 1 philosophers_stone                  5.11     7.20                    7.28
## 2 chamber_of_secrets                  5.71     7.73                    7.89
## 3 prisoner_of_azkaban                 5.56     7.92                    7.96
## 4 goblet_of_fire                      6.08     8.34                    8.04
## 5 order_of_the_phoenix                6.66     9.17                    8.42
## 6 half_blood_prince                   6.69     9.19                    8.34
## 7 deathly_hallows                     6.57     9.11                    8.08
```

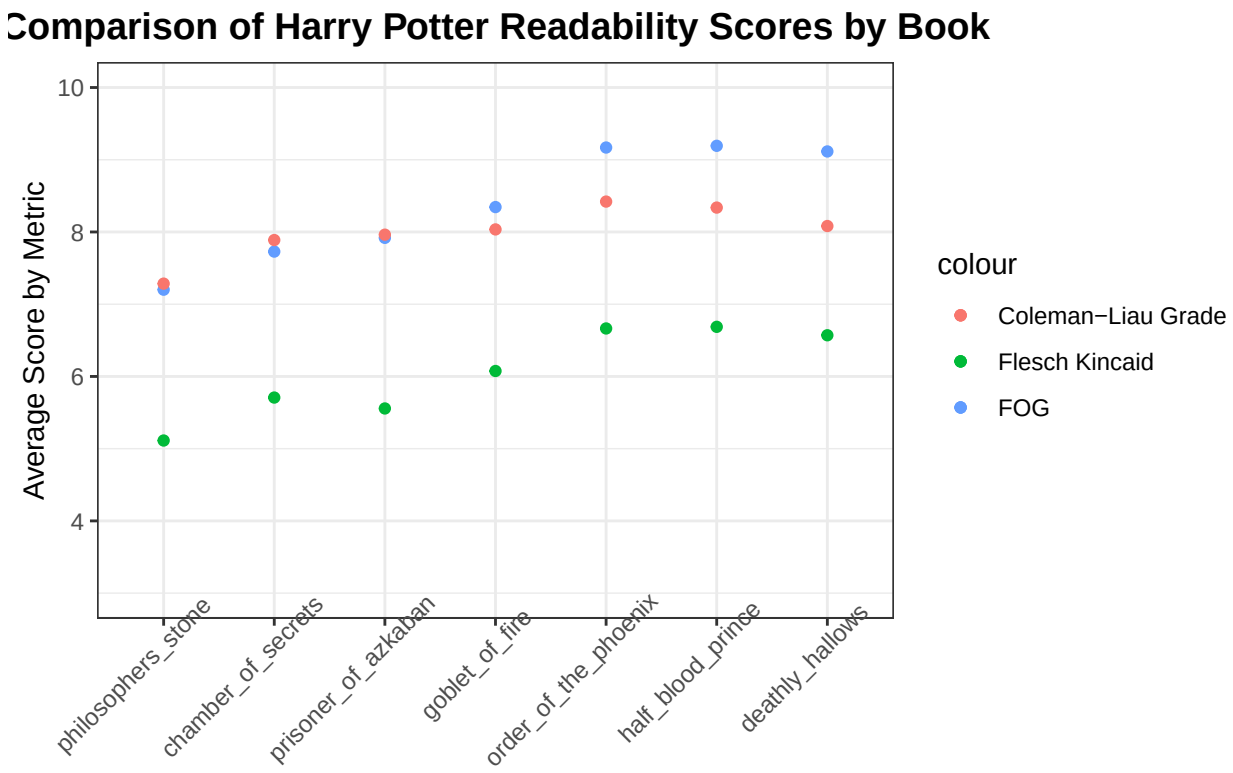
Now we can plot the comparison!

```
#level_order <- c("philosophers_stone", "chamber_of_secrets", "prisoner_of_azkaban", "goblet_of_fire",

# plot results
#it is plotting the x axis alphabetically so I'm going to try something here

readability_means$book <- factor(readability_means$book, levels = readability_means$book)

ggplot(readability_means, aes(x = book)) +
  geom_point(aes(x = book, y = Flesch.Kincaid_name, colour = 'Flesch Kincaid')) +
  geom_point(aes(x = book, y = FOG_name, colour = 'FOG')) +
  geom_point(aes(x = book, y = Coleman.Liau.grade_name, colour = 'Coleman-Liau Grade'),readability_means
  theme_bw() +
  ggtitle("Comparison of Harry Potter Readability Scores by Book") +
  theme(plot.title = element_text(hjust = 0.5, face="bold",
                            size=14)) +
  xlab(" ") + theme(axis.text.x= element_text(size=9, angle=45, hjust = .75)) +
  ylab("Average Score by Metric") + ylim(3, 10)
```



Hooray! I think I've done what I wanted to do!! Compare avaerage scores by metric over the seven books of the series!