Rating and sentiment analysis for Friends

2019-05-28

The other day Netflix suggested me to watch the TV show Friends. As I really enjoyed watching it as kid, I figured it would be nice to watch it again. However, this time, first I wanted to do some text and rating analyses to the show :)

We can start by loading the required packages:

library(tidyverse)  
library(tidytext)  
library(skimr)

I also used functions from these packages:

packages <- c("ggrepel", "stringi", "grid", "varhandle",  
 "plotly", "scales", "viridis")   
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {  
 install.packages(setdiff(packages, rownames(installed.packages())))   
}

In addition, in order to get the IMDB ratings we need to also install/load [this](https://github.com/RMHogervorst/imdb) package. However, if you’re only interested in the data, I uploaded it to my GitHub repository [here](https://github.com/AmirDJV/AmirDJV.github.io/blob/master/content/post/Rating_and_sentiment_analysis_for_Friends/imdb.rds).

#Load imdb package, install if not loaded  
if (require(imdb) == 0) {devtools::install\_github("rmhogervorst/imdb")}

## Loading required package: imdb

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,  
## logical.return = TRUE, : there is no package called 'imdb'

## Downloading GitHub repo rmhogervorst/imdb@HEAD

##   
## checking for file 'C:\Users\Amir\AppData\Local\Temp\RtmpycHbcW\remotes2e3034ec5852\RMHogervorst-imdb-c8cc757/DESCRIPTION' ... v checking for file 'C:\Users\Amir\AppData\Local\Temp\RtmpycHbcW\remotes2e3034ec5852\RMHogervorst-imdb-c8cc757/DESCRIPTION'  
## - preparing 'imdb': (419ms)  
## checking DESCRIPTION meta-information ... checking DESCRIPTION meta-information ... v checking DESCRIPTION meta-information  
## - checking for LF line-endings in source and make files and shell scripts  
## - checking for empty or unneeded directories  
## - building 'imdb\_0.2.0.tar.gz'  
##   
##

## Installing package into 'C:/Users/Amir/Documents/R/win-library/4.0'  
## (as 'lib' is unspecified)

We’ll start by getting the IMDB data for the series. This can be done nicely using the imdb wrapper package for open IMDB.

add\_key\_to\_renviron("you API key here")  
  
imdb <- tbl\_df(c(1:10)) %>%  
 mutate(data = map(value, function(x) {  
 imdbSeries("friends", seasons = x)  
 }))  
  
# We'll save the data, so we won't abuse the website  
saveRDS(imdb, "imdb.rds")

Now, let’s load the IMDB data and do a quick look of what we have here.

imdb <- readRDS("Rating\_and\_sentiment\_analysis\_for\_Friends/imdb.rds") %>%   
 unnest(data) %>%   
 select(title = Title,   
 season = Season,   
 episode = Episode,   
 rating = imdbRating)   
  
skimr::skim(imdb)

Data summary

|  |  |
| --- | --- |
| Name | imdb |
| Number of rows | 238 |
| Number of columns | 4 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 1 |
| numeric | 3 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| title | 0 | 1 | 17 | 50 | 0 | 238 | 0 |

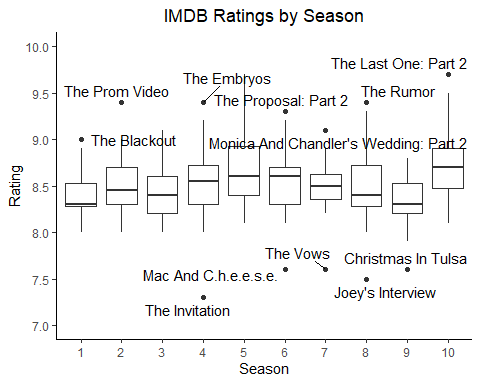
**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| season | 0 | 1 | 5.42 | 2.83 | 1.0 | 3.00 | 5.0 | 8.0 | 10.0 | ▇▇▇▇▇ |
| episode | 0 | 1 | 12.42 | 6.92 | 1.0 | 6.25 | 12.0 | 18.0 | 25.0 | ▇▇▇▇▆ |
| rating | 0 | 1 | 8.52 | 0.37 | 7.3 | 8.30 | 8.5 | 8.7 | 9.7 | ▁▃▇▃▁ |

We can see that we have 238 episodes, the average rating was 8.52 with a standard deviation of 0.37, and the median was 8.5.

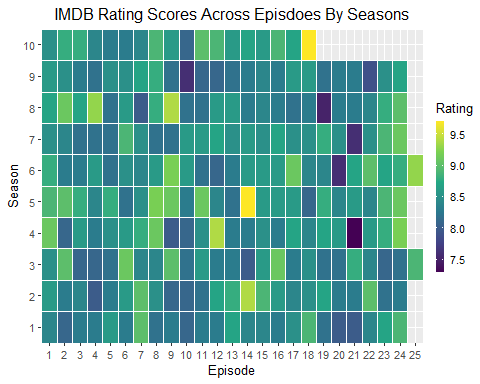
We can also do a quick box-plot to see what were the ratings for each season. We can also mark which episodes were outliers.

imdb %>%   
 group\_by(season) %>%  
 mutate(outlier = ifelse(  
 rating < quantile(rating, 0.25) - 1.5 \* IQR(rating) |  
 rating > quantile(rating, 0.75) + 1.5 \* IQR(rating),   
 stringi::stri\_trans\_totitle(  
 gsub("The One with|The One Where", "", title)),  
 NA)) %>%   
 ggplot(aes(factor(season), rating)) +   
 geom\_boxplot() +  
 scale\_y\_continuous(breaks = seq(7, 10, 0.5),  
 limits = c(7, 10)) +   
 labs(title = "IMDB Ratings by Season",   
 x = "Season",   
 y = "Rating") +  
 guides(color=FALSE) +  
 theme\_classic() +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 ggrepel::geom\_text\_repel(aes(label = outlier),  
 na.rm = TRUE)

 It seems that overall, the series kept for most of its episodes the same rating. If we’ll look at the median (the black line inside the boxes) rating, it seems somewhat stable. However, we can see that some episodes had really high and low ratings. Being familiar with show, some of these extreme ratings makes sense to me. For example, the episode were Ross wanted to take Rachel to her and Monica’s prom (The prom video, season 2), or the episode “Christmas In Tulsa” (season 9) which was basically flashbacks from old episodes.

We can also use heat-map to look for patterns between the episodes’ rating across the different seasons.

imdb %>%   
 ggplot(aes(episode, season, fill = rating)) +   
 geom\_tile(colour = "white", size = 0.2)+  
 labs(x = "Episode",   
 y = "Season",  
 fill = "Rating",  
 title = "IMDB Rating Scores Across Episdoes By Seasons") +  
 scale\_x\_continuous(expand = c(0,0), breaks = seq(1, 25, 1)) +  
 scale\_y\_continuous(expand = c(0,0), breaks = seq(1, 10, 1)) +  
 scale\_fill\_continuous(type = "viridis") +  
 theme\_grey(base\_size = 10) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 legend.margin=margin(grid::unit(0,"cm")),  
 legend.key.height=grid::unit(0.8,"cm"),  
 legend.key.width=grid::unit(0.2,"cm"))



Now, let’s load the transcript, filter for the main characters, do some cleaning, make it tidy and remove some stop words.

You can find the transcripts file [here](https://github.com/AmirDJV/AmirDJV.github.io/blob/master/content/post/Rating_and_sentiment_analysis_for_Friends/friends.csv).

scripts <- read.csv("Rating\_and\_sentiment\_analysis\_for\_Friends/friends.csv")  
  
scripts <- scripts %>%   
 purrr::set\_names(tolower(names(scripts))) %>%  
 select(season, episode, title,  
 actor = author,  
 text = quote) %>%   
 varhandle::unfactor() %>%   
 filter(actor == "Rachel" |   
 actor == "Monica" |  
 actor == "Joey" |   
 actor == "Chandler" |  
 actor == "Ross" |  
 actor == "Phoebe") %>%   
 mutate(  
 episode = as.numeric(episode),  
 text = gsub("\\s\*\\([^\\)]+\\)|\\[.\*?\\]","",  
 text),   
 text = gsub("\\written.\*", "", text)) %>%   
 tidytext::unnest\_tokens(word, text) %>%  
 anti\_join(stop\_words, by = "word")

We can take a quick pick at the scripts:

head(scripts)

## season episode title actor word  
## 1 1 1 The One Where Monica Gets A Roommate Monica guyi  
## 2 1 1 The One Where Monica Gets A Roommate Joey guy  
## 3 1 1 The One Where Monica Gets A Roommate Joey there'sgotta  
## 4 1 1 The One Where Monica Gets A Roommate Joey wrong  
## 5 1 1 The One Where Monica Gets A Roommate Chandler joey  
## 6 1 1 The One Where Monica Gets A Roommate Chandler benice

summary(scripts)

## season episode title actor   
## Min. : 1.000 Min. : 1.00 Length:155457 Length:155457   
## 1st Qu.: 3.000 1st Qu.: 6.00 Class :character Class :character   
## Median : 6.000 Median :12.00 Mode :character Mode :character   
## Mean : 5.501 Mean :12.27   
## 3rd Qu.: 8.000 3rd Qu.:18.00   
## Max. :10.000 Max. :25.00   
## word   
## Length:155457   
## Class :character   
## Mode :character   
##   
##   
##

Let’s start with small things, like who spoke the most and what was the most occurring word for each character.

scripts %>%   
 group\_by(actor) %>%   
 summarise(numberOfWords = sum(!is.na(word)))

## # A tibble: 6 x 2  
## actor numberOfWords  
## <chr> <int>  
## 1 Chandler 25385  
## 2 Joey 26758  
## 3 Monica 23895  
## 4 Phoebe 23825  
## 5 Rachel 27359  
## 6 Ross 28235

scripts %>%   
 group\_by(actor) %>%   
 count(word) %>%   
 top\_n(3)

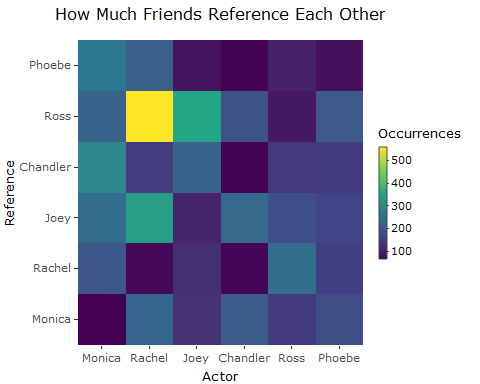
## Selecting by n

## # A tibble: 18 x 3  
## # Groups: actor [6]  
## actor word n  
## <chr> <chr> <int>  
## 1 Chandler gonna 380  
## 2 Chandler hey 519  
## 3 Chandler yeah 558  
## 4 Joey hey 1095  
## 5 Joey uh 509  
## 6 Joey yeah 967  
## 7 Monica gonna 423  
## 8 Monica hey 414  
## 9 Monica yeah 407  
## 10 Phoebe gonna 337  
## 11 Phoebe hey 464  
## 12 Phoebe yeah 789  
## 13 Rachel gonna 526  
## 14 Rachel ross 525  
## 15 Rachel yeah 809  
## 16 Ross hey 769  
## 17 Ross uh 758  
## 18 Ross yeah 842

Interestingly, one of the things that Rachel says the most is “Ross”. That made me wonder how much do they reference each other.

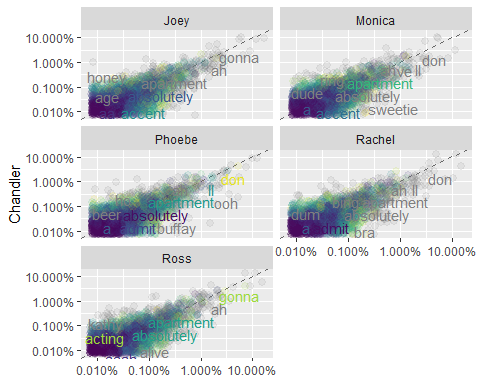
plotly::ggplotly(  
 scripts %>%   
 filter(grepl("monica|rachel|joey|chandler|ross|phoebe",  
 tolower(word))) %>%   
 mutate(reference = str\_extract(  
 pattern = "monica|rachel|joey|chandler|ross|phoebe",  
 string = tolower(word))) %>%  
 group\_by(actor, reference) %>%   
 summarise(Occurrence = sum(!is.na(reference))) %>%   
 ungroup() %>%   
 mutate(Actor = factor(  
 x = stringi::stri\_trans\_totitle(actor),   
 levels = c("Monica", "Rachel", "Joey",  
 "Chandler", "Ross", "Phoebe")),   
 Reference = factor(  
 x = stringi::stri\_trans\_totitle(reference),   
 levels = c("Monica", "Rachel", "Joey",  
 "Chandler", "Ross", "Phoebe"))) %>%  
 ggplot(aes(Actor, Reference, fill = Occurrence)) +   
 geom\_tile(colour = "white", size = 0.2) +  
 labs(x = "Actor",   
 y = "Reference",  
 fill = "Occurrences",  
 title = "How Much Friends Reference Each Other") +  
 scale\_x\_discrete(expand = c(0,0)) +  
 scale\_y\_discrete(expand = c(0,0)) +  
 scale\_fill\_continuous(type = "viridis") +  
 theme\_grey(base\_size = 10) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 legend.margin=margin(grid::unit(0,"cm")),  
 legend.key.height=grid::unit(0.8,"cm"),  
 legend.key.width=grid::unit(0.2,"cm"))  
)

## `summarise()` has grouped output by 'actor'. You can override using the `.groups` argument.

 Not surprisingly, they don’t tend to reference themselves as often. However, as we’ve seen before, Rachel references Ross even more than any other character reference the others. Even Ross doesn’t reference Rachel that often.

We can also examine how different is the vocabulary each characters use:

scripts %>%   
 mutate(word = str\_extract(word, "[a-z']+")) %>%  
 count(actor, word) %>%  
 group\_by(actor) %>%  
 mutate(proportion = n / sum(!is.na(word))) %>%   
 select(-n) %>%   
 spread(actor, proportion) %>%   
 gather(actor, proportion, -Chandler, -word) %>%  
 na.omit() %>%  
 ggplot(aes(x = proportion, y = Chandler,   
 color = abs(Chandler - proportion))) +  
 geom\_abline(color = "gray40", lty = 2) +  
 geom\_jitter(alpha = 0.1, size = 2.5,  
 width = 0.3, height = 0.3) +  
 geom\_text(aes(label = word),   
 check\_overlap = TRUE, vjust = 1.5) +  
 scale\_x\_log10(labels = scales::percent\_format()) +  
 scale\_y\_log10(labels = scales::percent\_format()) +  
 viridis::scale\_color\_viridis(limits = c(0, 0.001)) +  
 facet\_wrap(~actor, ncol = 2) +  
 theme(legend.position="none") +  
 labs(y = "Chandler", x = NULL)



Let’s continue to some sentiment analyses

sentimentScript <- scripts %>%   
 inner\_join(tidytext::get\_sentiments("bing"), by = "word") %>%  
 group\_by(actor, season, episode, sentiment) %>%  
 count(word) %>%   
 spread(sentiment, n, fill = 0) %>%  
 summarise(sentiment = sum(positive) - sum(negative)) %>%  
 ungroup()

## `summarise()` has grouped output by 'actor', 'season'. You can override using the `.groups` argument.

We can look for the most negative season:

sentimentScript %>%   
 group\_by(season) %>%  
 summarise(avg = mean(sentiment)) %>%   
 ungroup() %>%  
 arrange(avg)

## # A tibble: 10 x 2  
## season avg  
## <int> <dbl>  
## 1 6 -1.65   
## 2 9 -1.18   
## 3 1 -1.09   
## 4 3 -0.993   
## 5 2 -0.964   
## 6 5 -0.928   
## 7 8 -0.457   
## 8 4 -0.232   
## 9 10 0.0784  
## 10 7 0.297

Or the most negative character for each season:

sentimentScript %>%   
 group\_by(season, actor) %>%  
 summarise(avg = mean(sentiment)) %>%   
 top\_n(-1) %>%  
 ungroup() %>%  
 arrange(avg)

## `summarise()` has grouped output by 'season'. You can override using the `.groups` argument.

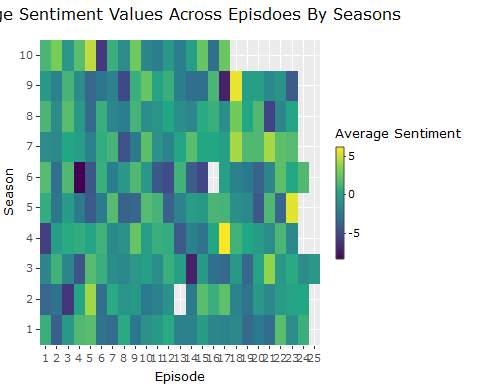
## Selecting by avg

## # A tibble: 10 x 3  
## season actor avg  
## <int> <chr> <dbl>  
## 1 2 Chandler -3.43  
## 2 9 Chandler -3.35  
## 3 6 Monica -2.77  
## 4 3 Rachel -2.48  
## 5 1 Chandler -1.96  
## 6 5 Rachel -1.83  
## 7 8 Chandler -1.83  
## 8 10 Chandler -1.71  
## 9 4 Monica -1.52  
## 10 7 Phoebe -1.52

We can also create a heat-map to identify the most negative and positive episode over the seasons

plotly::ggplotly(  
 sentimentScript %>%  
 group\_by(episode, season) %>%   
 summarise(avg = mean(sentiment)) %>%   
 ggplot(aes(episode, season, fill = avg)) +   
 geom\_tile(colour = "white", size = 0.2) +  
 labs(x = "Episode",   
 y = "Season",  
 fill = "Average Sentiment",  
 title = "Avrage Sentiment Values Across Episdoes By Seasons") +  
 scale\_x\_continuous(expand = c(0,0), breaks = seq(1, 25, 1)) +  
 scale\_y\_continuous(expand = c(0,0), breaks = seq(1, 10, 1)) +  
 scale\_fill\_continuous(type = "viridis") +  
 theme\_grey(base\_size = 10) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 legend.margin=margin(grid::unit(0,"cm")),  
 legend.key.height=grid::unit(0.8,"cm"),  
 legend.key.width=grid::unit(0.2,"cm"))  
 )

## `summarise()` has grouped output by 'episode'. You can override using the `.groups` argument.



We can also animate the movement of sentiments over episodes and season for each character

library(gganimate)

## No renderer backend detected. gganimate will default to writing frames to separate files  
## Consider installing:  
## - the `gifski` package for gif output  
## - the `av` package for video output  
## and restarting the R session

plot1 <- sentimentScript %>%  
 ggplot(aes(episode, sentiment, color = actor)) +   
 geom\_point() +   
 facet\_wrap(~actor) +  
 scale\_x\_continuous(expand = c(0,0), breaks = seq(1, 25, 2)) +  
 labs(title = "Season: {frame\_time}",  
 x = "Episode",  
 y = "Average Sentiment For Episode",   
 color = "Actor") +  
 transition\_time(season) +  
 ease\_aes("linear") +  
 theme(plot.title = element\_text(hjust = 0.5))  
  
animate(plot1, fps=5, end\_pause = 30)

## Warning: No renderer available. Please install the gifski, av, or magick package  
## to create animated output

Thank you for reading!