Movie Keyword Analysis: Building a Vector Space Model

STAT 597A: Statistical Computing

Samuel Castillo

April 25, 2017

A Math Problem

Frozen + The Expendables =

Liam Neeson + Bruce Willis =

Slide with Bullets

How I tried to answer this question:

- Use data from IMDB.com
- Continue from the midterm with AWS
- Network plots
- Compare movies by plot keywords
- Build a vector space model
- Perform a dimensionality reduction

Plot Keywords Variable

The Data was webscraped from IMDB.com and posted on Kaggle. For each movie, there is a list of plot keywords.
Tangled (2010)

```
imdb$keywords[7]
```

```
## [[1]]
## [1] "17th century"          "based on fairy tale" "disney"
## [4] "flower"                "tower"
```

Pirates of the Caribbean At World's End (2007)

```
## [[1]]
## [1] "goddess"           "marriage ceremony" "marriage proposal"
## [4] "pirate"            "singapore"
```

How can we compare these?

Cleaning the Data

```
imdb$keywords = as.character(imdb$keywords)
imdb$keywords = strsplit(imdb$keywords, split = "|", fixed = TRUE) %>% as.l
#Reuse vectorize
n = imdb$keywords %>%
  unlist()%>%
  unique()%>%
  length()
#Create a list of all unique keywords to use for comparison
unique_keywords = imdb$keywords %>% unlist() %>% unique()

vectorize = function(keywords_list){
  #create a vector from the list
  cur_keywords = keywords_list %>% unlist()
  #initialize an empty vector
  out = c(rep(0, n))
  for(i in cur_keywords){
        index = match(i, unique_keywords)
        out[index] = 1
    }
  return(out)
```

Creating a Network

Here I take a smalller sample at first to test the system using a filter of year $>$ 2016.

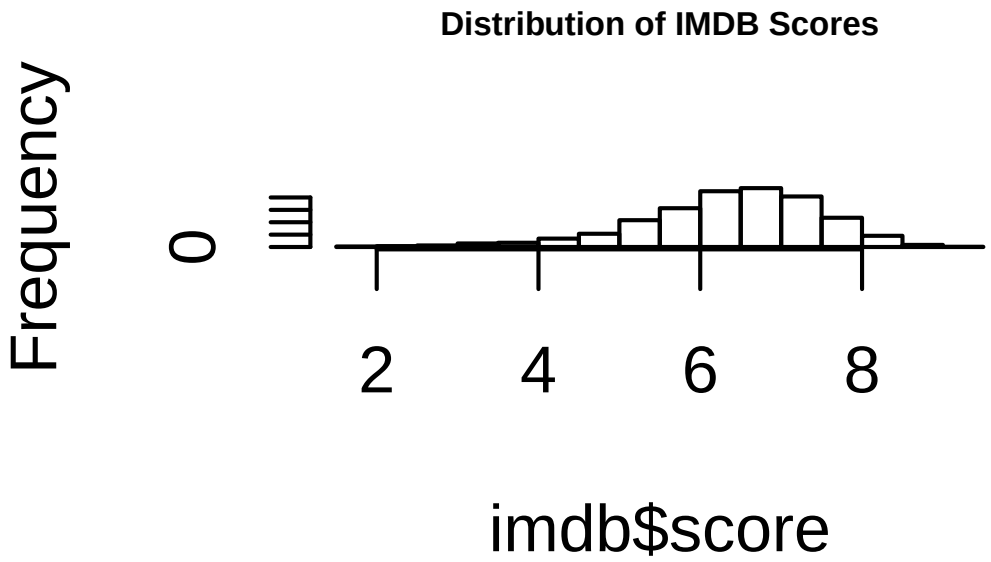
```
mydata = filter(imdb, year >= 2016)
dim(mydata)

make_network(mydata)
```

45 movies is a good starting point. This code below creates an adjacency matrix that is used for the network plot. The i,jth entry is 1 if movie i is connected to movie j and zero otherwise.

Filtering to the Top 99% By Score

```
hist(imdb$score, main = "Distribution of IMDB Scores", cex.main = 0.5)
```




```
quantile(imdb$score, probs = seq( 0.9, 1, 0.01))
```

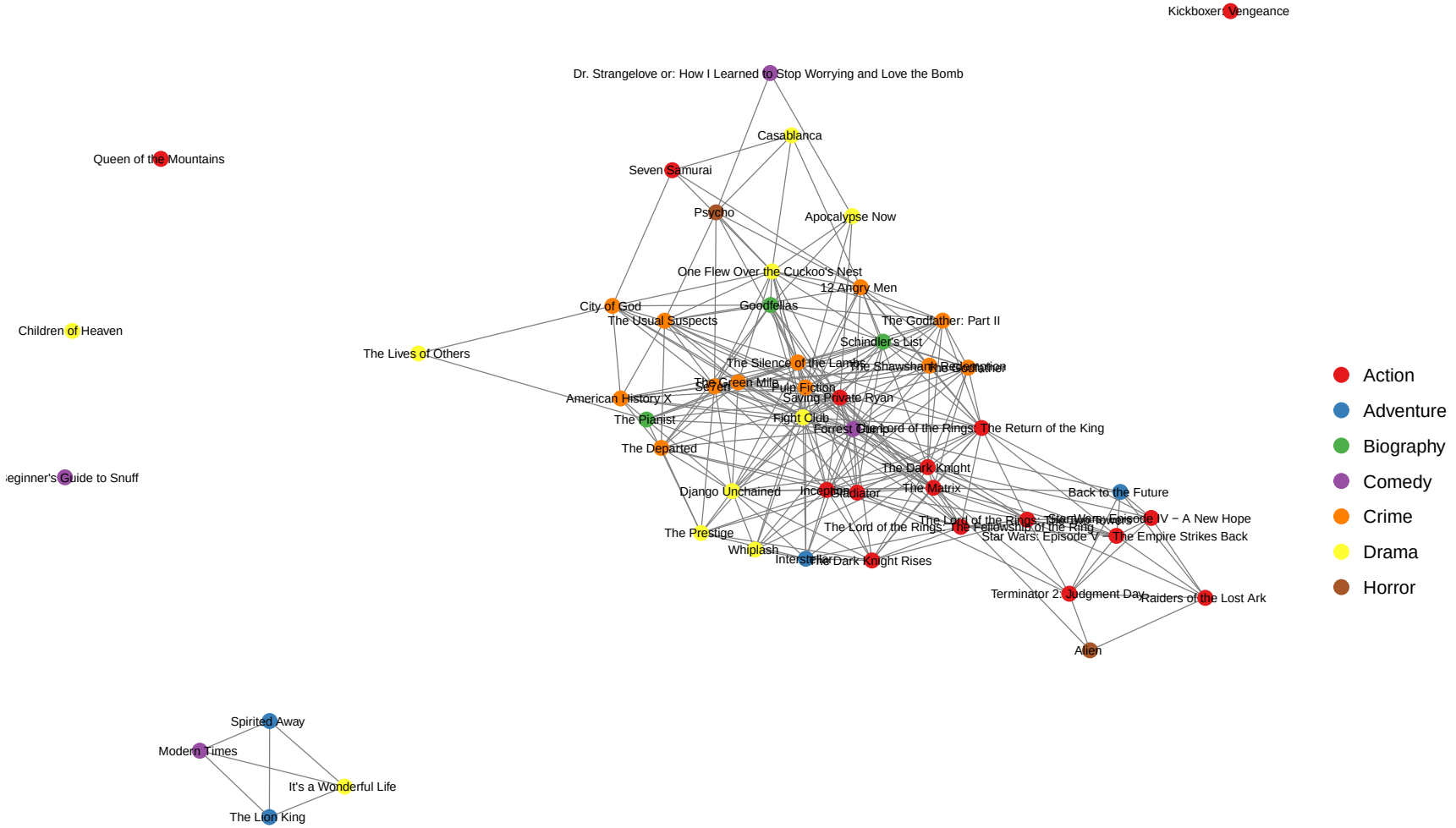
```
##  90%  91%  92%  93%  94%  95%  96%  97%  98%  99% 100%
##  7.7  7.8  7.8  7.9  7.9  8.0  8.1  8.2  8.3  8.5  9.3
```

```
mydata2 = filter( imdb, score >= 8.5)
```

Plotting the Network

Only those movies with score > 8.5. These are the top 99%

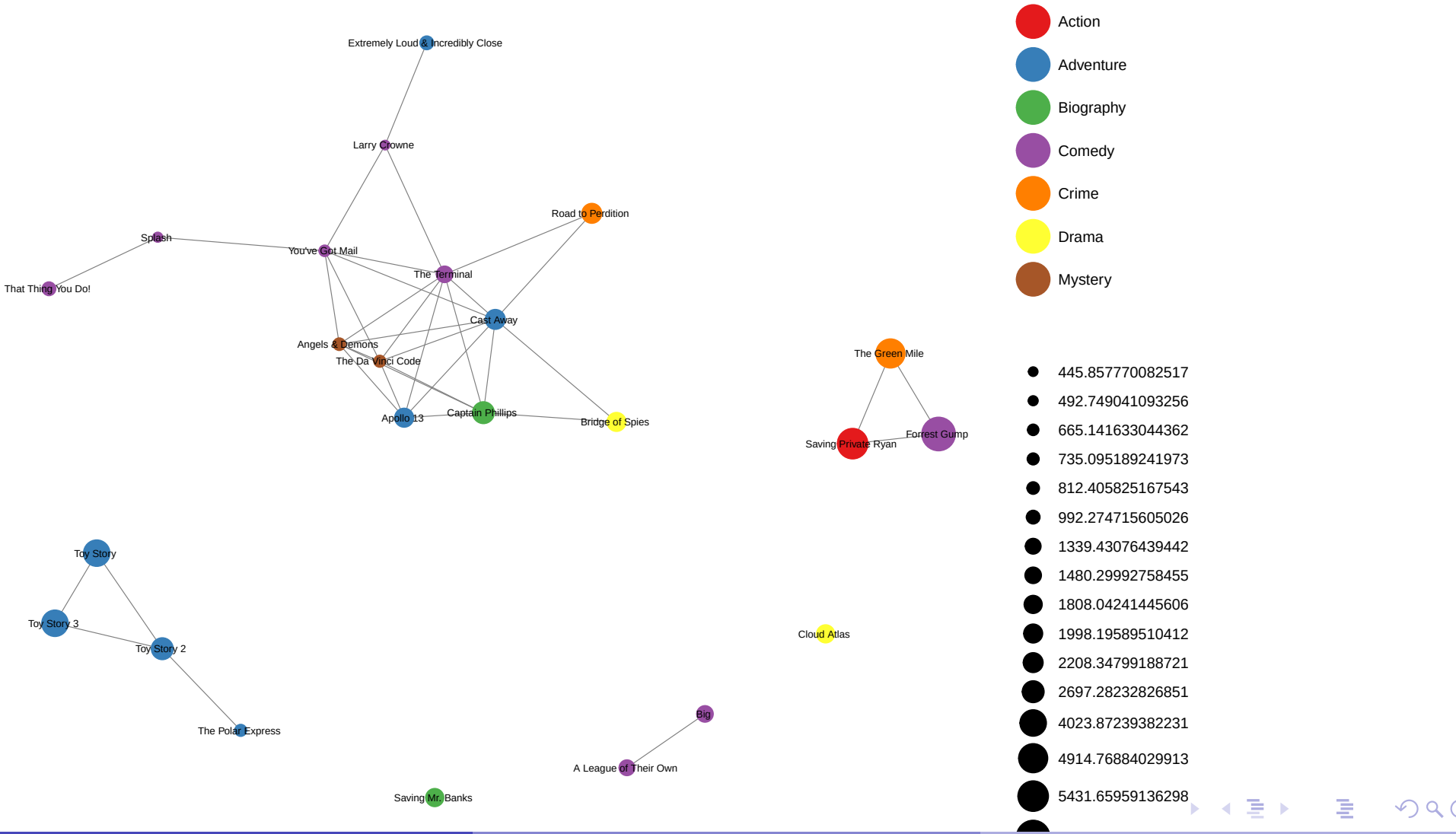
```
## Loading required package: scales
```



Tom Hanks Movies

Only movies with Tom Hanks in the leading role. Size is determined by imdb score.

```
## [1] 24 18
```



Michael Bay Movies

Only movies with Michael Bay as the director. The size attribute is the log of the adjusted budget.

The Vector Space Model

- Turn each set of keywords for a particular movie into a vector or zeros and ones
- Add movies together to create a new vector
- Assign a relational measure of how similar movies are
- Find a third movie “closest” to the first two.

Defining Similarity

Using cosine similarity:

$$Sim(A, B) = cos(\theta) = \frac{A.B}{|A||B|}$$

Using correlation:

$$Sim(A, B) = r_{AB} = \frac{n\sum a_i b_i - \sum a_i \sum b_i}{\sqrt{n\sum a_i^2 - (\sum a_i)^2}\sqrt{n\sum b_i^2 - (\sum b_i)^2}}$$

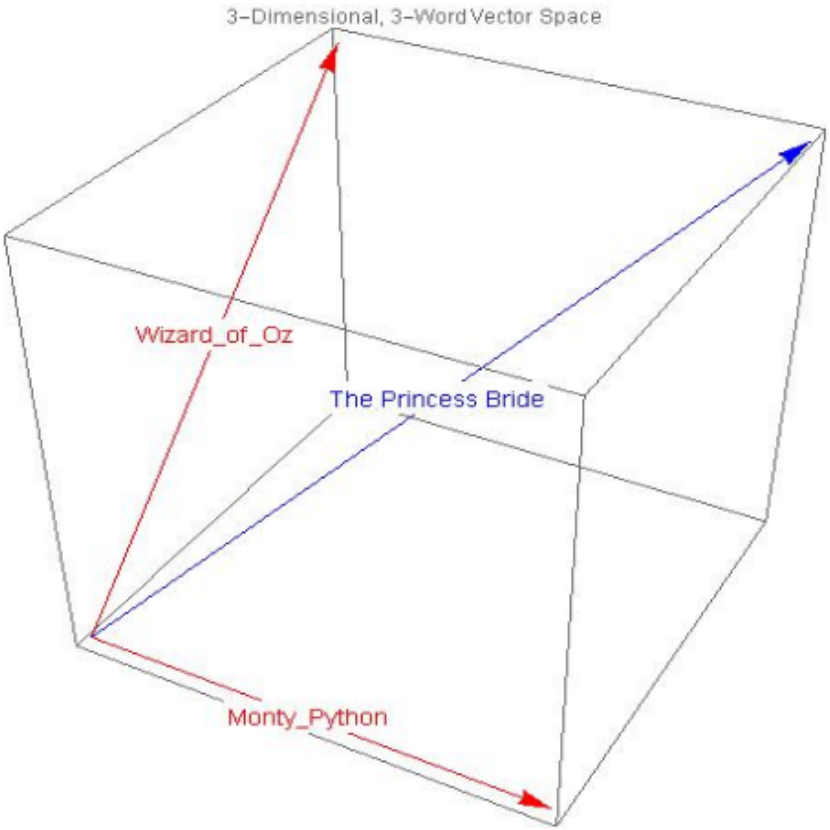
Using Euclidean Distance:

$$Sim(A, B) = \sqrt{\sum (a_i - b_i)^2}$$

Example

For each movie, there is a list of keywords. For example, for the movie 'Monte Python and the Holy Grail', the keywords might be 'knight', 'camelot', 'wizard', 'holy grail', 'castle', etc. For simplicity just assume there are two keywords, 'camelot', and 'wizard.' For a different movie, such as 'The Wizard of Oz', the keywords might be 'kansas', and 'wizard'.

Here is what this a word space might look like. In this case the dimensions have been simplified to only three words. This are 'camelot', 'wizard', and 'kansas'. The word vector for Monty Python is <1, 1, 0>, and < 0, 1, 1> for the Wizard of Oz. A potential match might be The Princess Bride.



The closest movie is determined by minimizing the angle between the sum of the two vectors and a third movie which is not one of the first two

Finding the Closest Keyword Vectors

```
#Now the "titles" are the actors names
closest_numeric_actor = function ( cur_vector, titleA = NULL, titleB = NULL
  temp = by_actor%>% filter(actor != titleA, actor!= titleB)

  my_cos = function( list_vectors){
    x = unlist( cur_vector)
    y = unlist( list_vectors)
    cosine(x,y)
  }

  euc.dist <- function(list_vectors) {
    x1 = unlist( cur_vector)
    x2 = unlist( list_vectors)
    sqrt(sum((x1 - x2) ^ 2))}

  cur_function = my_cos
  if( dist == T){cur_function = euc.dist}

  num_cores = detectCores()
  relation = mclapply( FUN = cur_function, temp$keyword_vectors, mc.cores =
```

Creating Networks with Similarity

- tiff files

Defining Addition

```
actor_addition = function( titleA = NULL, titleB = NULL, dist = F){
 actorA =  filter( by_actor, actor == titleA)
 actorB = filter( by_actor, actor == titleB)
 x = actorA$keyword_vectors %>% unlist()
 y = actorB$keyword_vectors %>% unlist()
 AplusB = x + y
 result = closest_numeric_actor(AplusB, titleA, titleB, dist)
 paste( titleA, " + ", titleB, " = " ,result)
}

actor_addition(by_actor$actor[1],by_actor$actor[1])
```

```
## [1] "50 Cent  +  50 Cent  =  Oona Laurence"

## NULL

## Time difference of 1.563632 secs
```

A Shiny App

`https://sdcastillo.shinyapps.io/imdb_addition/`

Output:

Frozen + The Expendables = The Chronicles of Narnia: The Lion, the Witch and the Wardrobe
Cosine of Angle: 0.282842712474619
Angle in Pi Radians: 0.408722554503086
Angle in Degrees: 73.5700598105554

PCA

We have a matrix with each row a keyword and each column an actor

```
n #Number of unique keywords
```

```
## [1] 7979
```

```
dim(keyword_matrix)
```

```
## [1] 7979 6120
```

```
by_actor_matrix = keyword_matrix
by_keyword = t(keyword_matrix) # rows are actors and columns are keywords
```

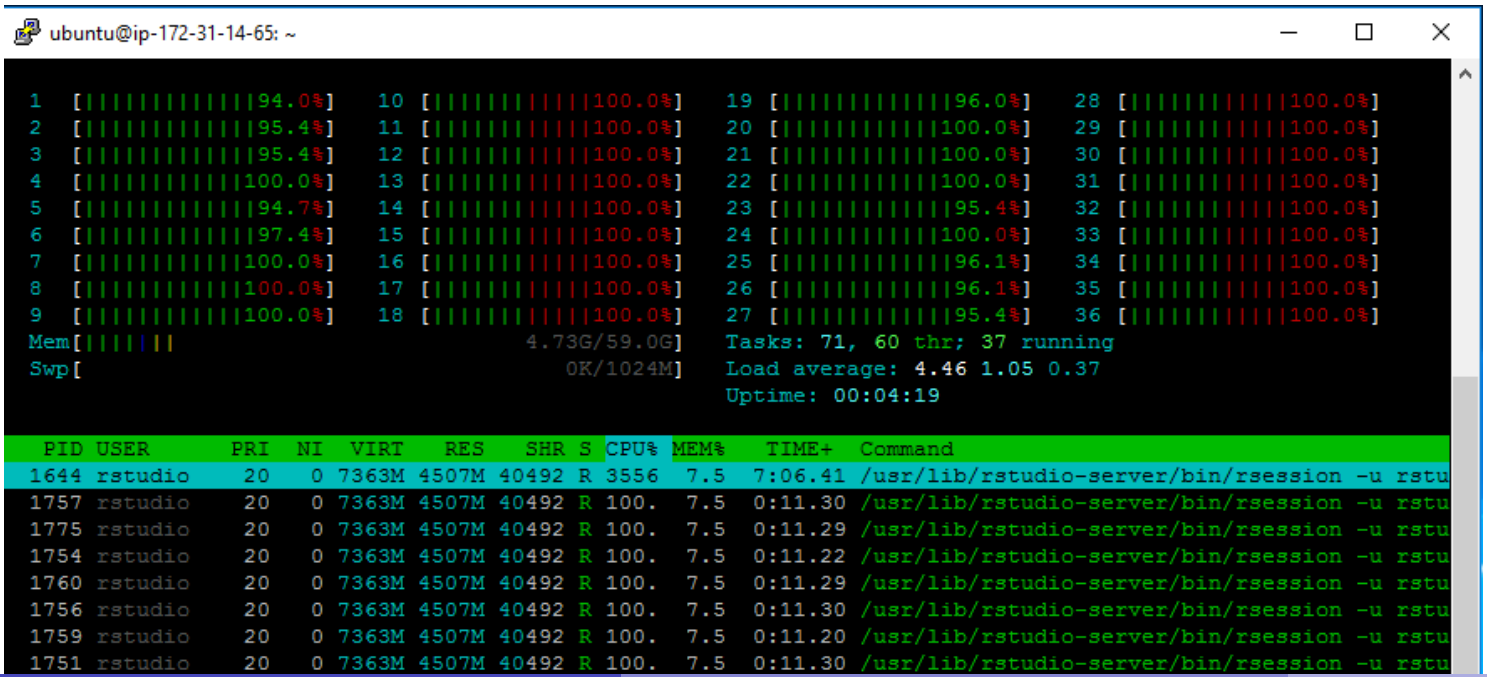
PCA

I examine both keywords and actors.

```
t11 = Sys.time()
actor.pca = prcomp( by_actor_matrix, center = T, scale. = T)
t21 = Sys.time()
keyword.pca = prcomp( by_keyword, center = T, scale. = T)
t21 - t11#Time using all 36 cores
```

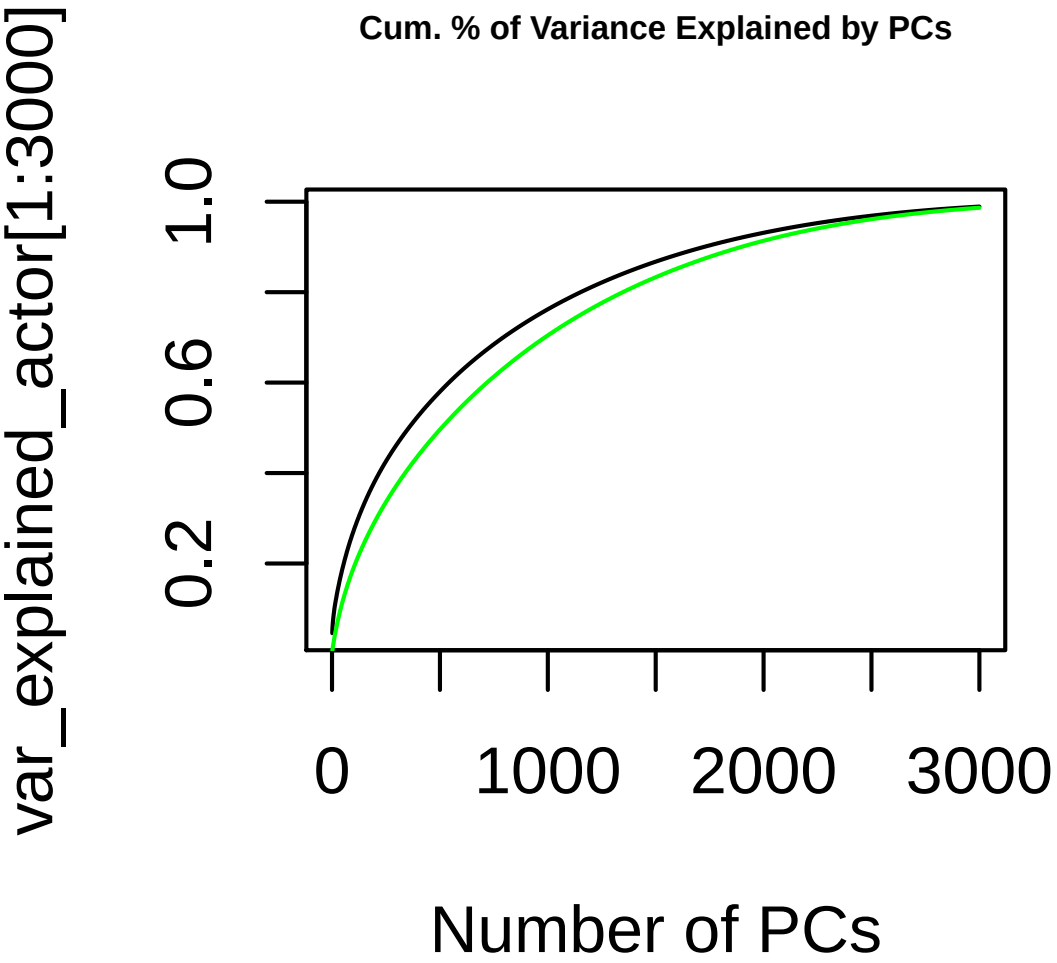
```
## Time difference of 1.73092 mins
```

Screenshot:



Interpretation of PCA

```
var_explained_actor = cumsum(actor.pca$sdev^2/sum(actor.pca$sdev^2))
var_explained_keyword = cumsum(keyword.pca$sdev^2/sum(keyword.pca$sdev^2))
```



Interpretation of PCA

```
var_explained_actor[1000]
```

```
## [1] 0.7621473
```

```
ncol(by_actor_matrix)
```

```
## [1] 6120
```

```
var_explained_keyword[1000]
```

```
## [1] 0.70441
```

Distribution of Keywords

There are 7979 total unique keywords. We can explain ~90% of the variance using linear combinations of 1000 keywords.

```
##    90%    91%    92%    93%    94%    95%    96%    97%    98%    99%
##   5.00   6.00   7.00   7.00   8.38  10.00  12.00  15.00  20.46  30.23
##   100%
## 196.00
```

```
##     all_keywords Freq
## 1           love  196
## 2         friend  161
## 3         murder  157
## 4          death  128
## 5         police  118
## 6  new york city   89
## 7    high school   86
## 8          alien   79
## 9            boy   72
## 10        school   72
```

Further Considerations

- Find correlation of principal components with actors or movies
- Include subtraction and or cross-product functions in Shiny App
- Find a way to predict the genre of a film based on keywords

Closing Thoughts

- Free $100 AWS credits. Google “AWS Educate”
- Obligatory UGrid mention (ugrid.info)