USING NAIVE BAYES

To predict the genre of movies.

Daniel Alejandro Morales Castillo

ABSTRACT

This dataset it's completely original in the sense that it's a compilation of some of the movies that each of us has seen at least one time in our lifetime, the movies are divided in three main genres: Action, drama and horror. With the Naive Bayes algorithm implemented in R, we want to be able to predict a movie genre based in the description of the movie.

INDEX

BASIC DATA ANALYSIS

DATA DOMAIN

Definition

A story or event recorded by a camera as a set of moving images and shown in a theater or on television; a motion picture.

Movie genre

A film genre is a stylistic or thematic category for motion pictures based on similarities either in the narrative elements, Aesthetic approach, or the emotional response to the film.

DATA DOMAIN

Action

A type of film in which a lot of exciting things happen

Drama

Drama is a category of narrative fiction (or semi-fiction) intended to be more serious than humorous in tone.

Horror

Horror is a genre of storytelling intended to scare, shock, and thrill its audience.

VARIABLES

movieid	Id to identify the movie
name	The name of the movie
genre	The genre of the movie
description	The description of the movie
releaseyear	The release year of the movie
durationminutes	The duration of the movie in minutes

How was the data collected?

The data was collected each member of this team based on the movies that all of us has watched in our lifetime and the description pull from various sites of movie rankings.

Limitations of the study

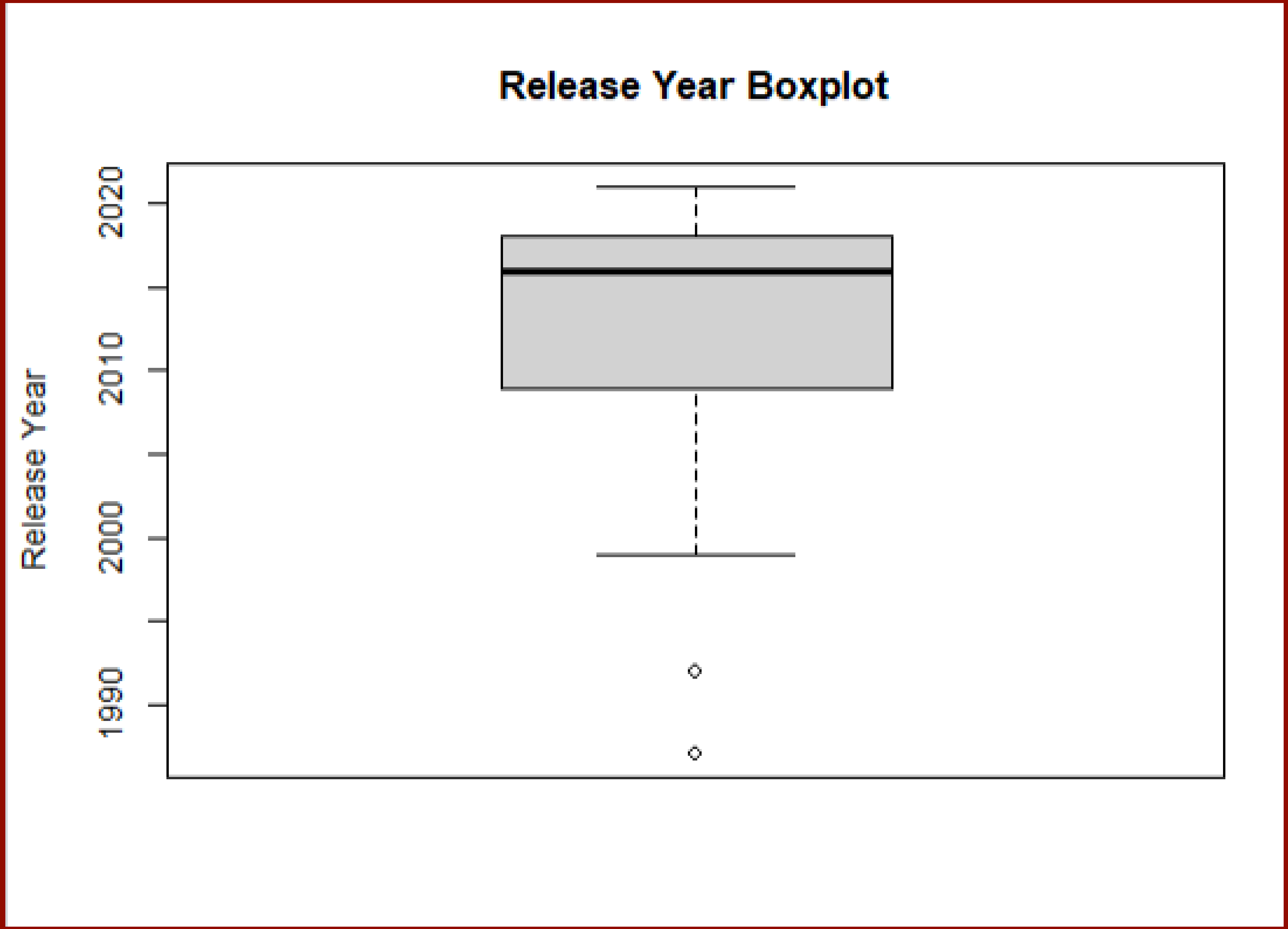
The data it's limited to 3 types of movie genres, limited to 90 observations. Our biggest limitation it's our memory to remeber which movie we have seen.

Disadvantages

The data set it's relatively small due to the limitation of our memory.

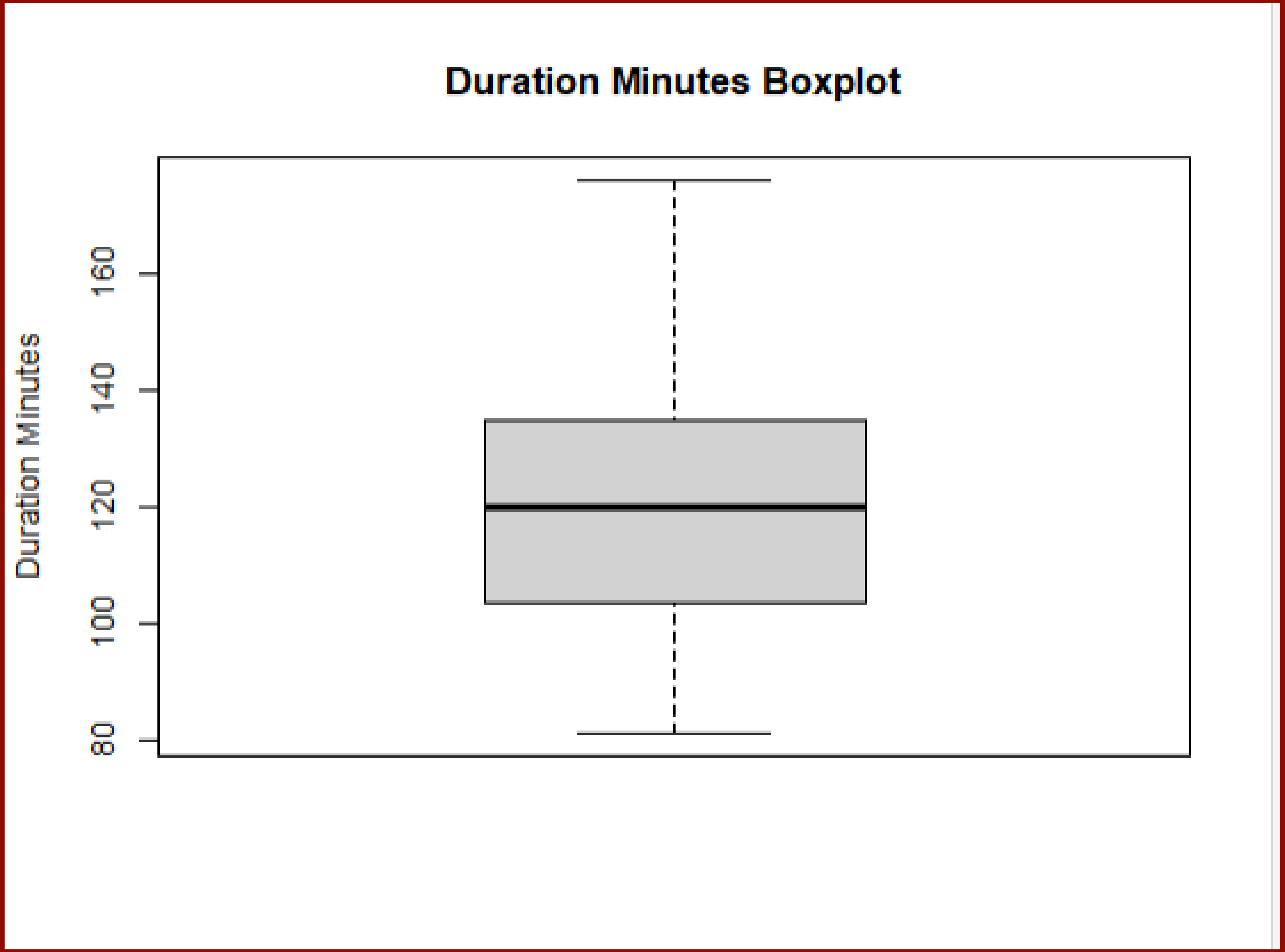
Interesting plots

Release Year Boxplot



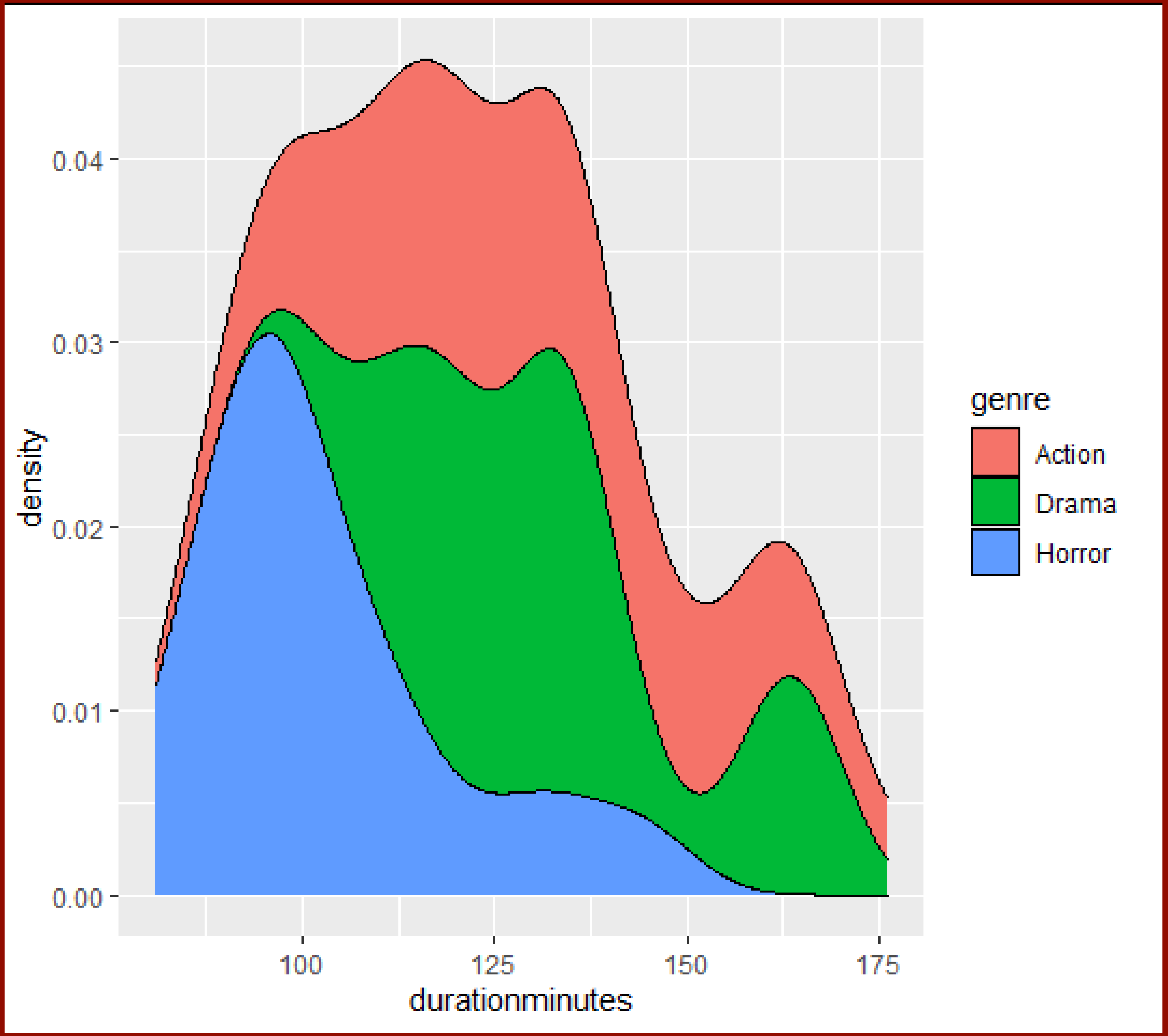
```
boxplot(movies$releaseyear, main="Release Year Boxplot", ylab="Release Year")
```

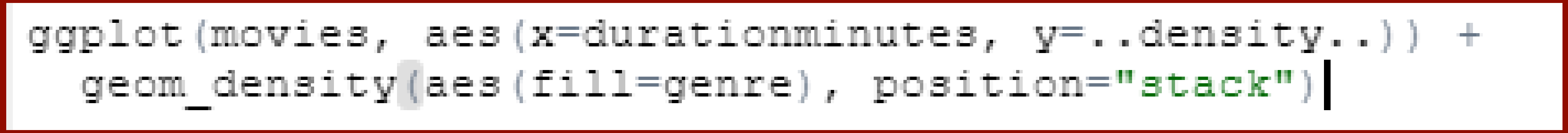
Duration Minutes Boxplot



```
boxplot(movies$durationminutes, main="Duration Minutes Boxplot", ylab="Duration Minutes")
```

Stacked density chart



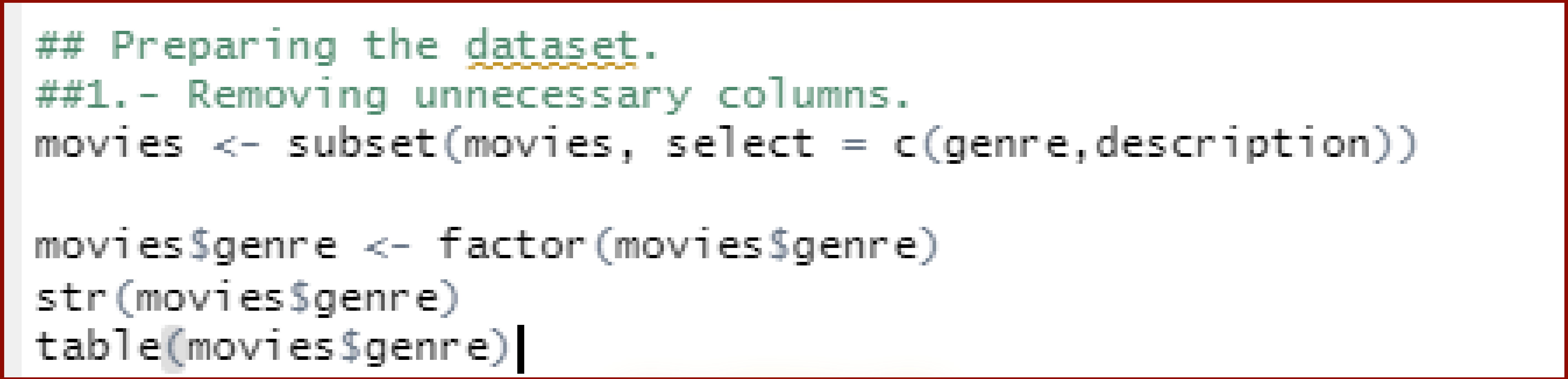


```
ggplot(movies, aes(x=durationminutes, y=..density..)) +
  geom_density(aes(fill=genre), position="stack")
```

PREPROCESSING

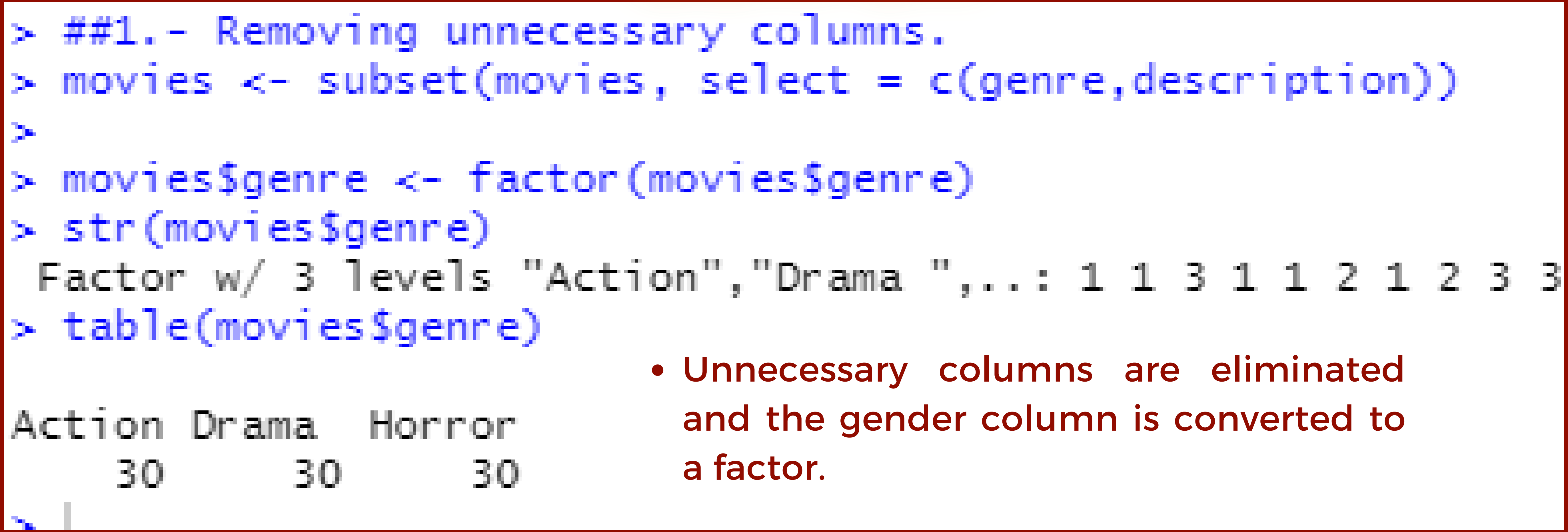
Let's transform the data into a suitable dataset

IN R:



```r
## Preparing the dataset.
##1.- Removing unnecessary columns.
movies <- subset(movies, select = c(genre,description))

movies$genre <- factor(movies$genre)
str(movies$genre)
table(movies$genre)
```

OUTPUT:



```
> ##1.- Removing unnecessary columns.
> movies <- subset(movies, select = c(genre,description))
>
> movies$genre <- factor(movies$genre)
> str(movies$genre)
 Factor w/ 3 levels "Action","Drama ",..: 1 1 3 1 1 2 1 2 3 3
> table(movies$genre)

Action Drama  Horror
    30     30     30
>
```

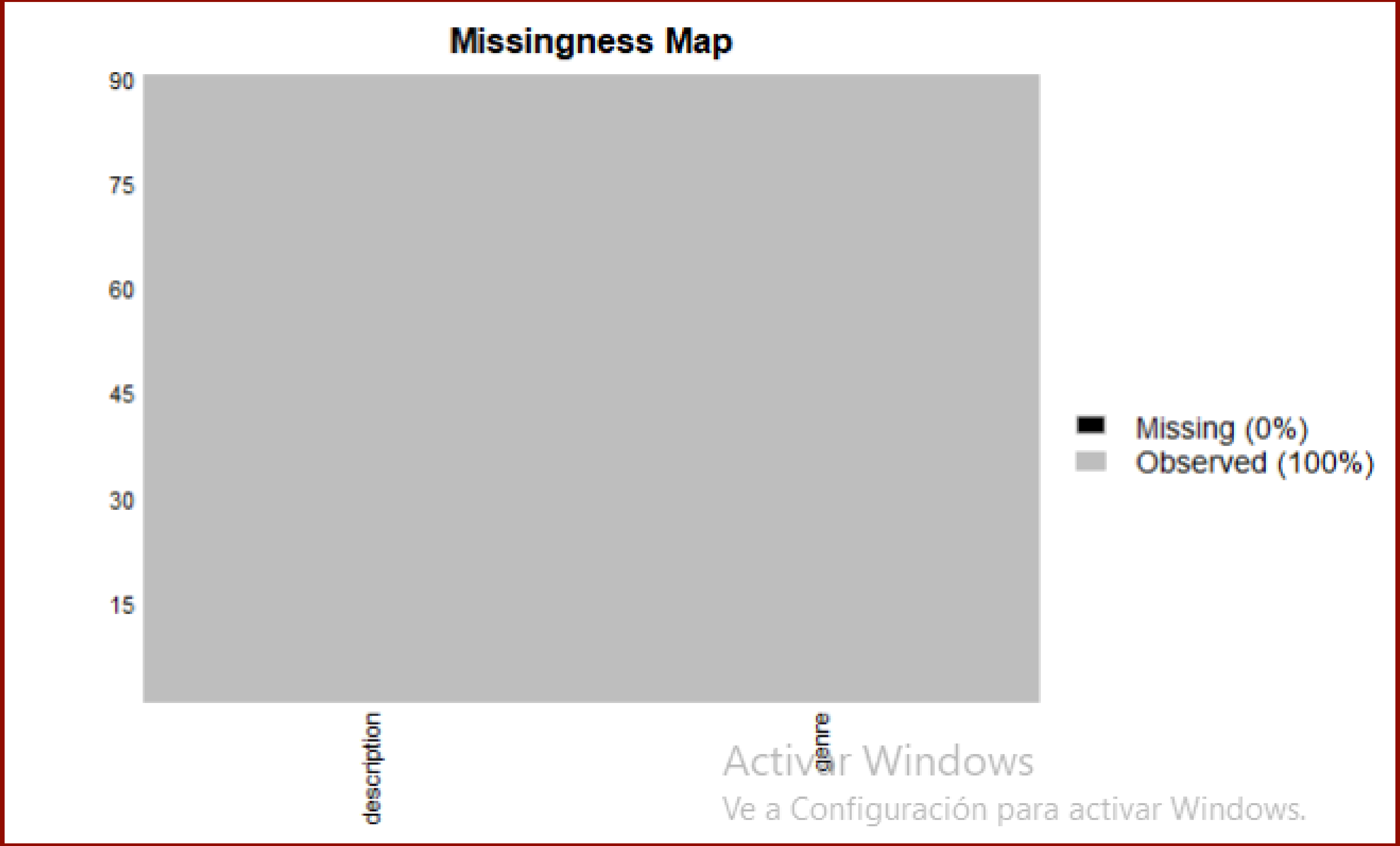
- Unnecessary columns are eliminated and the gender column is converted to a factor.

Missing values

IN R:

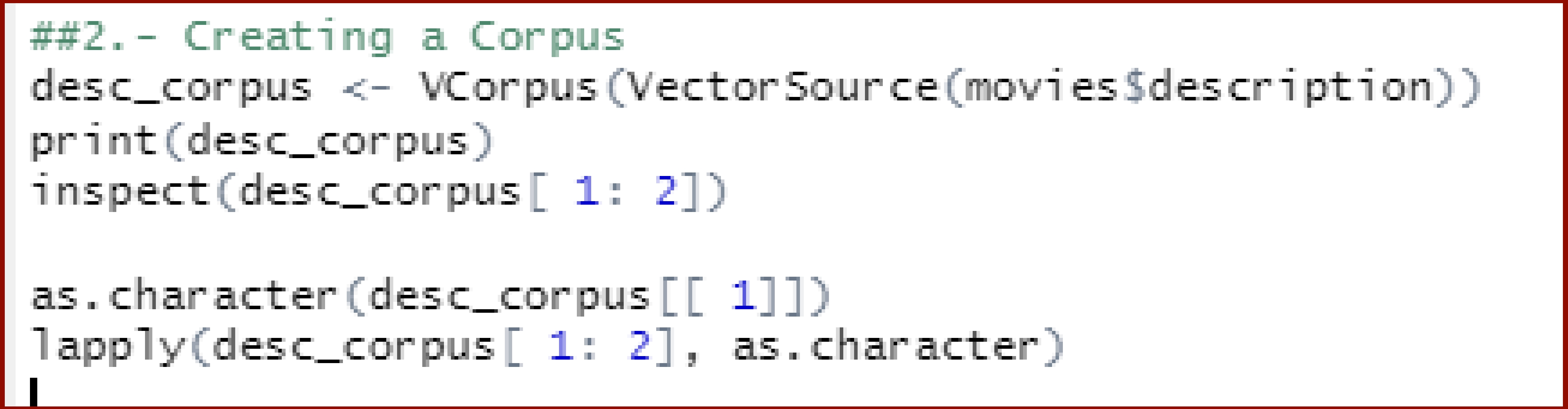
```r
library(Amelia)
missmap(movies, col=c("black", "grey"))
```

OUTPUT:



Creating a corpus

IN R:

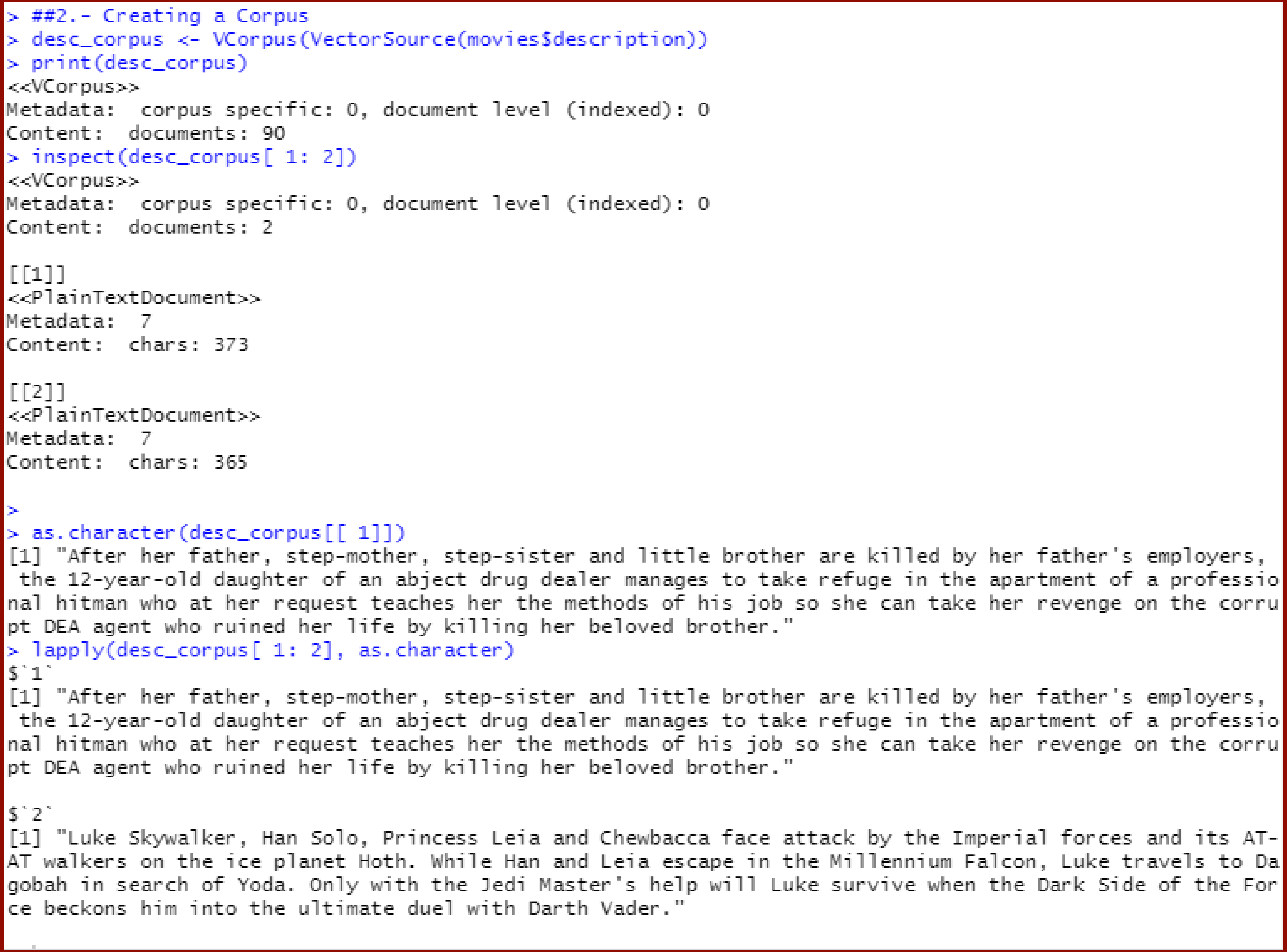


```
##2.- Creating a Corpus
desc_corpus <- VCorpus(VectorSource(movies$description))
print(desc_corpus)
inspect(desc_corpus[ 1: 2])

as.character(desc_corpus[[ 1]])
lapply(desc_corpus[ 1: 2], as.character)
```

- **A corpus is a set of text intended for research**

A corpus is created from the description variable.for each description a document will be created. The corpus was cleaned up as seen in the following slides.

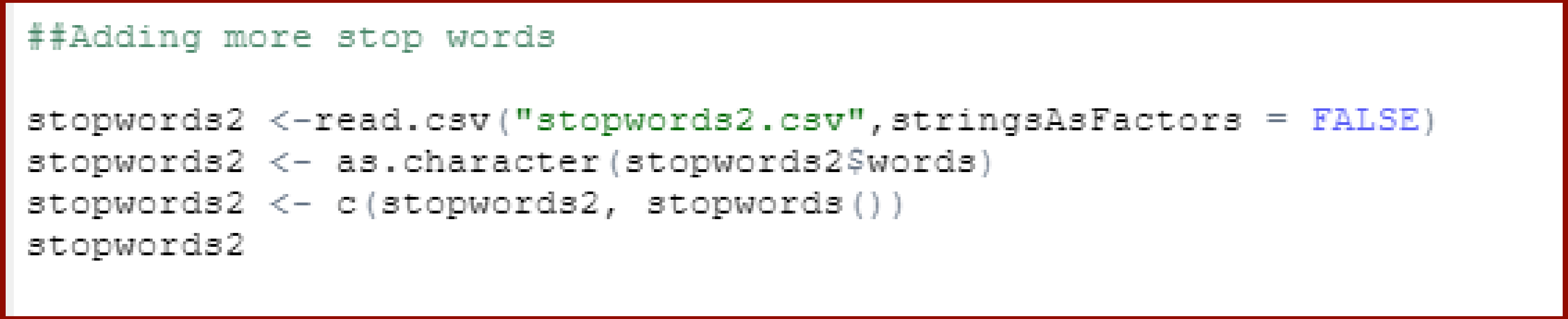
OUTPUT:



```
> ##2.- Creating a Corpus
> desc_corpus <- VCorpus(VectorSource(movies$description))
> print(desc_corpus)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 90
> inspect(desc_corpus[ 1: 2])
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 373

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 365

>
> as.character(desc_corpus[[ 1]])
[1] "After her father, step-mother, step-sister and little brother are killed by her father's employers,
 the 12-year-old daughter of an abject drug dealer manages to take refuge in the apartment of a professio
nal hitman who at her request teaches her the methods of his job so she can take her revenge on the corru
pt DEA agent who ruined her life by killing her beloved brother."
> lapply(desc_corpus[ 1: 2], as.character)
$`1`
[1] "After her father, step-mother, step-sister and little brother are killed by her father's employers,
 the 12-year-old daughter of an abject drug dealer manages to take refuge in the apartment of a professio
nal hitman who at her request teaches her the methods of his job so she can take her revenge on the corru
pt DEA agent who ruined her life by killing her beloved brother."

$`2`
[1] "Luke Skywalker, Han Solo, Princess Leia and Chewbacca face attack by the Imperial forces and its AT-
AT walkers on the ice planet Hoth. While Han and Leia escape in the Millennium Falcon, Luke travels to Da
gobah in search of Yoda. Only with the Jedi Master's help will Luke survive when the Dark Side of the For
ce beckons him into the ultimate duel with Darth Vader."
```

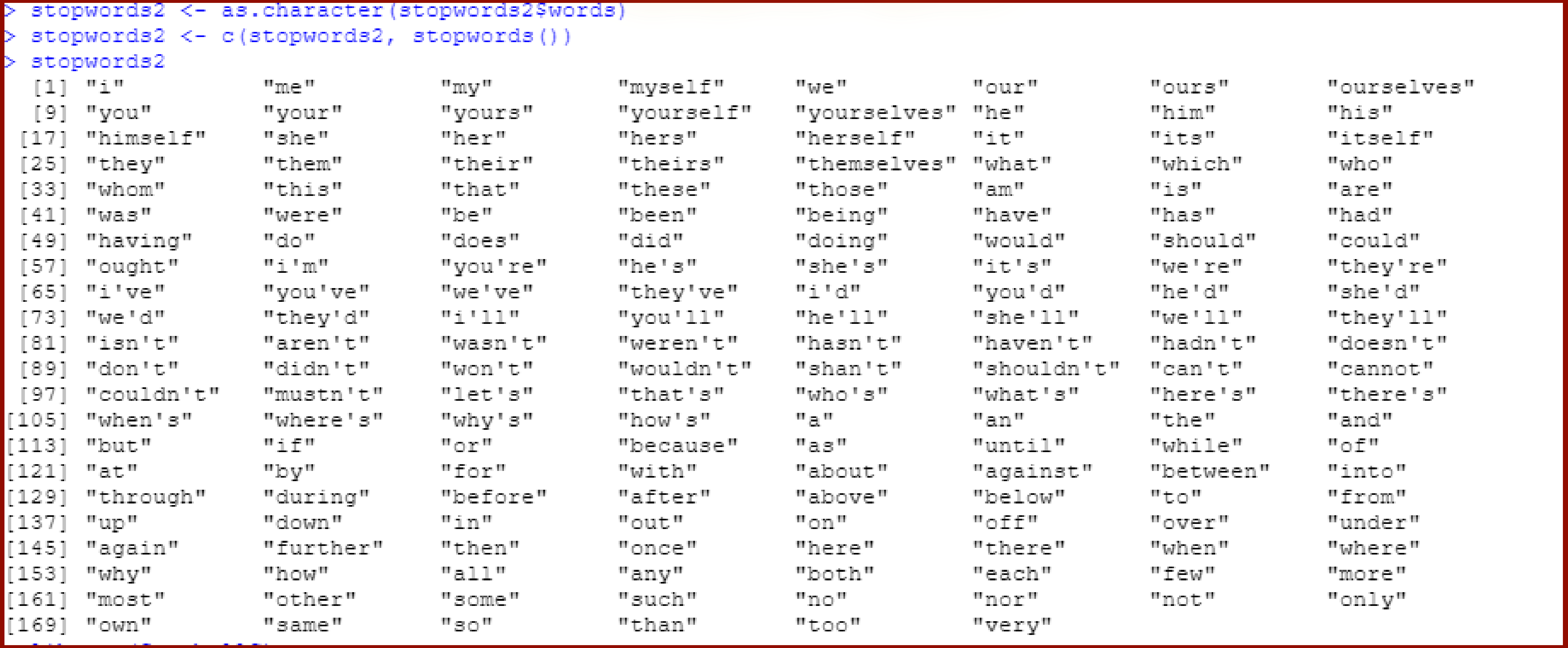
Stop words

IN R:



```r
##Adding more stop words

stopwords2 <-read.csv("stopwords2.csv",stringsAsFactors = FALSE)
stopwords2 <- as.character(stopwords2$words)
stopwords2 <- c(stopwords2, stopwords())
stopwords2
```

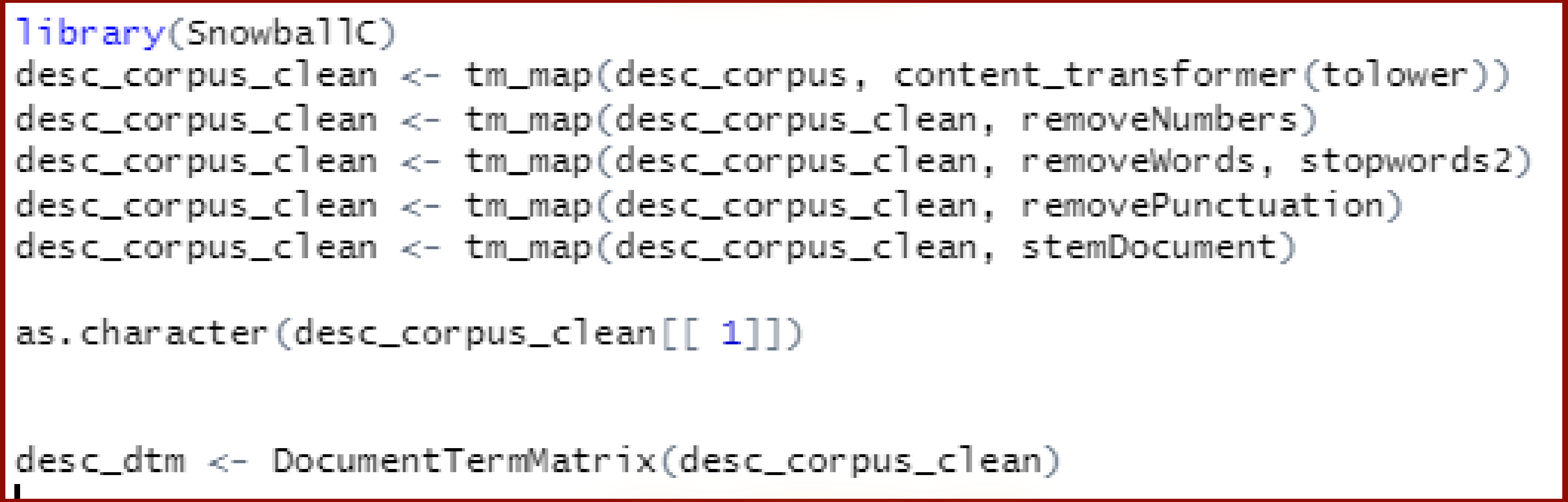
OUTPUT:



```
> stopwords2 <- as.character(stopwords2$words)
> stopwords2 <- c(stopwords2, stopwords())
> stopwords2
  [1] "i"          "me"         "my"         "myself"     "we"         "our"        "ours"       "ourselves" 
  [9] "you"        "your"       "yours"      "yourself"   "yourselves" "he"         "him"        "his"       
 [17] "himself"    "she"        "her"        "hers"       "herself"    "it"         "its"        "itself"    
 [25] "they"       "them"       "their"      "theirs"     "themselves" "what"       "which"      "who"       
 [33] "whom"       "this"       "that"       "these"      "those"      "am"         "is"         "are"       
 [41] "was"        "were"       "be"         "been"       "being"      "have"       "has"        "had"       
 [49] "having"     "do"         "does"       "did"        "doing"      "would"      "should"     "could"     
 [57] "ought"      "i'm"        "you're"     "he's"       "she's"      "it's"       "we're"      "they're"   
 [65] "i've"       "you've"     "we've"      "they've"    "i'd"        "you'd"      "he'd"       "she'd"     
 [73] "we'd"       "they'd"     "i'll"       "you'll"     "he'll"      "she'll"     "we'll"      "they'll"   
 [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"     "haven't"    "hadn't"     "doesn't"   
 [89] "don't"      "didn't"     "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"      "cannot"    
 [97] "couldn't"   "mustn't"    "let's"      "that's"     "who's"      "what's"     "here's"     "there's"   
[105] "when's"     "where's"    "why's"      "how's"      "a"          "an"         "the"        "and"       
[113] "but"        "if"         "or"         "because"    "as"         "until"      "while"      "of"        
[121] "at"         "by"         "for"        "with"       "about"      "against"    "between"    "into"      
[129] "through"    "during"     "before"     "after"      "above"      "below"      "to"         "from"      
[137] "up"         "down"       "in"         "out"        "on"         "off"        "over"       "under"     
[145] "again"      "further"    "then"       "once"       "here"       "there"      "when"       "where"     
[153] "why"        "how"        "all"        "any"        "both"       "each"       "few"        "more"      
[161] "most"       "other"      "some"       "such"       "no"         "nor"        "not"        "only"      
[169] "own"        "same"       "so"         "than"       "too"        "very"      
```

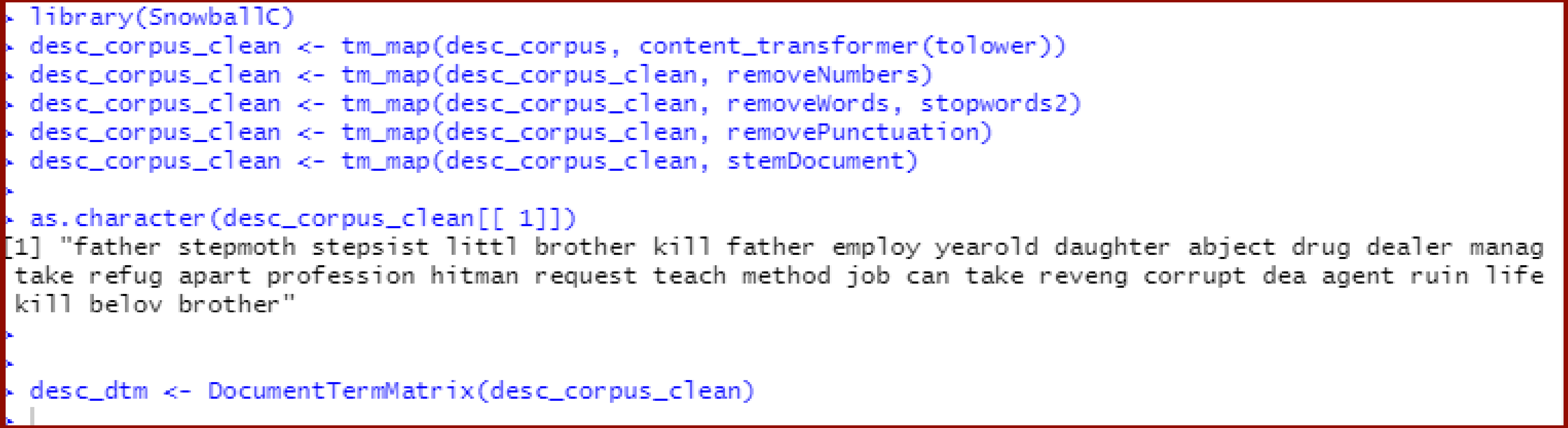
Stemming

IN R:



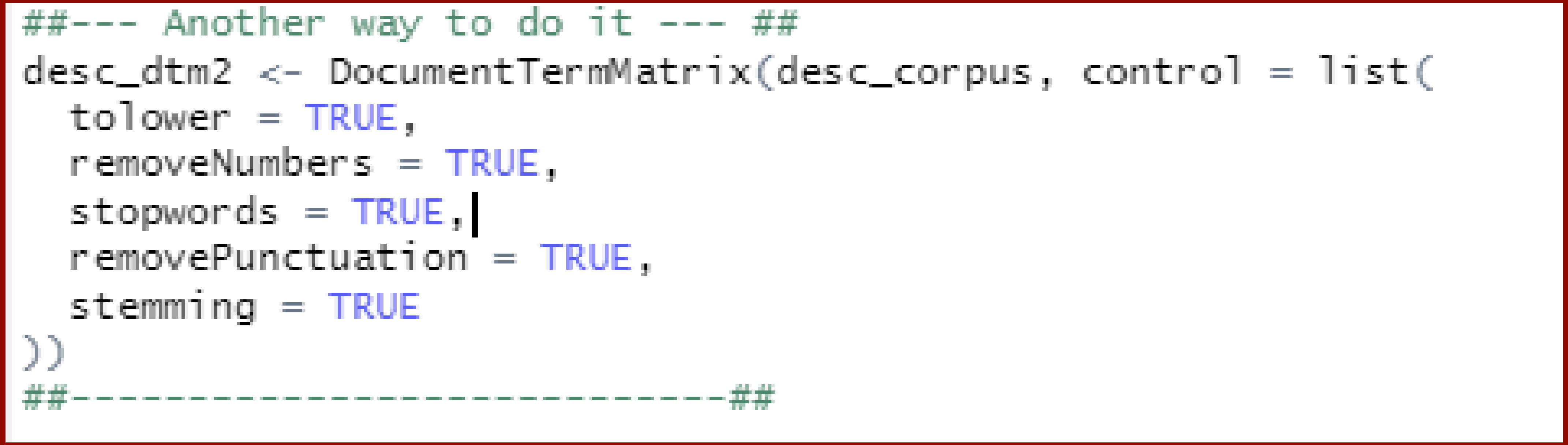
```
library(SnowballC)
desc_corpus_clean <- tm_map(desc_corpus, content_transformer(tolower))
desc_corpus_clean <- tm_map(desc_corpus_clean, removeNumbers)
desc_corpus_clean <- tm_map(desc_corpus_clean, removeWords, stopwords2)
desc_corpus_clean <- tm_map(desc_corpus_clean, removePunctuation)
desc_corpus_clean <- tm_map(desc_corpus_clean, stemDocument)

as.character(desc_corpus_clean[[ 1]])


desc_dtm <- DocumentTermMatrix(desc_corpus_clean)
```

OUTPUT:



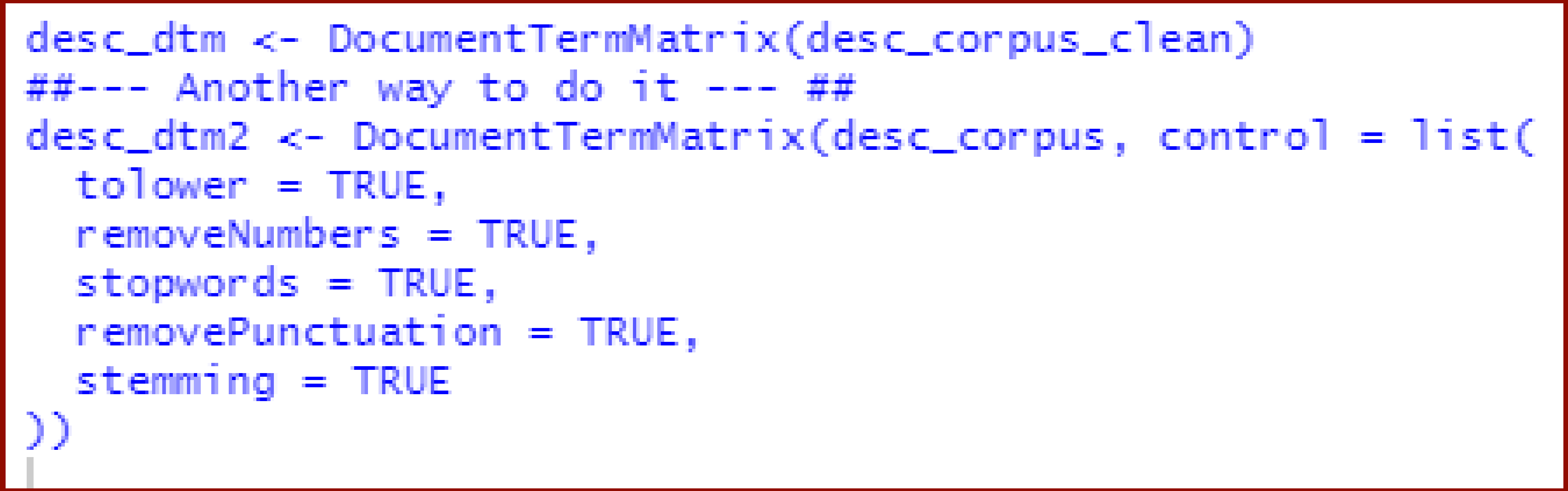
```
> library(SnowballC)
> desc_corpus_clean <- tm_map(desc_corpus, content_transformer(tolower))
> desc_corpus_clean <- tm_map(desc_corpus_clean, removeNumbers)
> desc_corpus_clean <- tm_map(desc_corpus_clean, removeWords, stopwords2)
> desc_corpus_clean <- tm_map(desc_corpus_clean, removePunctuation)
> desc_corpus_clean <- tm_map(desc_corpus_clean, stemDocument)
>
> as.character(desc_corpus_clean[[ 1]])
[1] "father stepmoth stepsist littl brother kill father employ yearold daughter abject drug dealer manag
 take refug apart profession hitman request teach method job can take reveng corrupt dea agent ruin life
 kill belov brother"
>
>
> desc_dtm <- DocumentTermMatrix(desc_corpus_clean)
>
```

IN R:



```
##--- Another way to do it --- ##
desc_dtm2 <- DocumentTermMatrix(desc_corpus, control = list(
  tolower = TRUE,
  removeNumbers = TRUE,
  stopwords = TRUE,
  removePunctuation = TRUE,
  stemming = TRUE
))
##----------------------------##
```

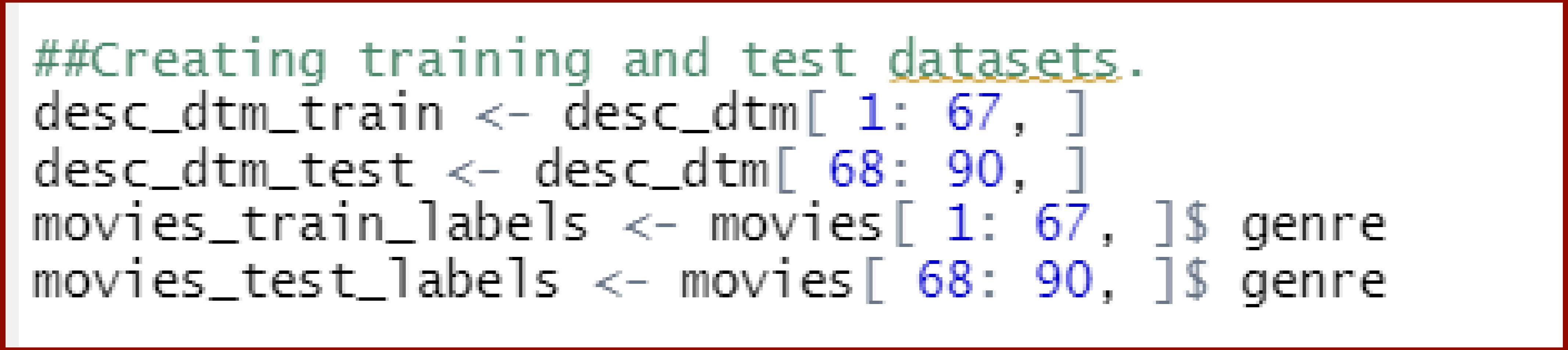
OUTPUT:



```
desc_dtm <- DocumentTermMatrix(desc_corpus_clean)
##--- Another way to do it --- ##
desc_dtm2 <- DocumentTermMatrix(desc_corpus, control = list(
  tolower = TRUE,
  removeNumbers = TRUE,
  stopwords = TRUE,
  removePunctuation = TRUE,
  stemming = TRUE
))
```

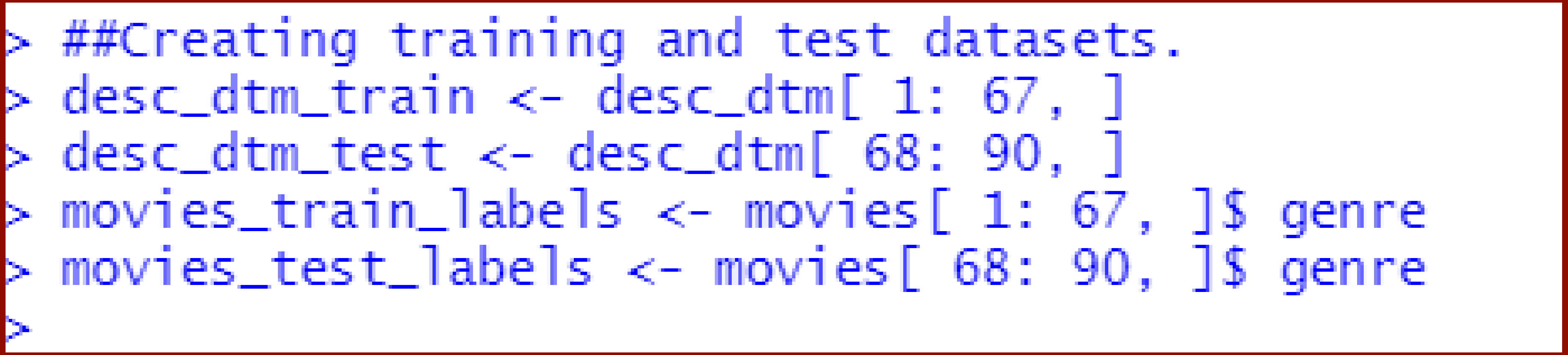
Training and testing sets

IN R:



```
##Creating training and test datasets.
desc_dtm_train <- desc_dtm[ 1: 67, ]
desc_dtm_test <- desc_dtm[ 68: 90, ]
movies_train_labels <- movies[ 1: 67, ]$ genre
movies_test_labels <- movies[ 68: 90, ]$ genre
```

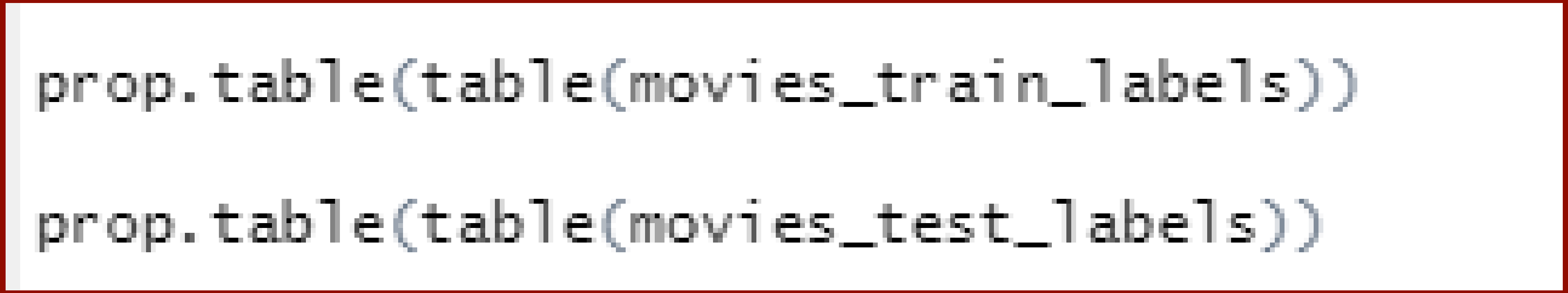
OUTPUT:



```
> ##Creating training and test datasets.
> desc_dtm_train <- desc_dtm[ 1: 67, ]
> desc_dtm_test <- desc_dtm[ 68: 90, ]
> movies_train_labels <- movies[ 1: 67, ]$ genre
> movies_test_labels <- movies[ 68: 90, ]$ genre
>
```

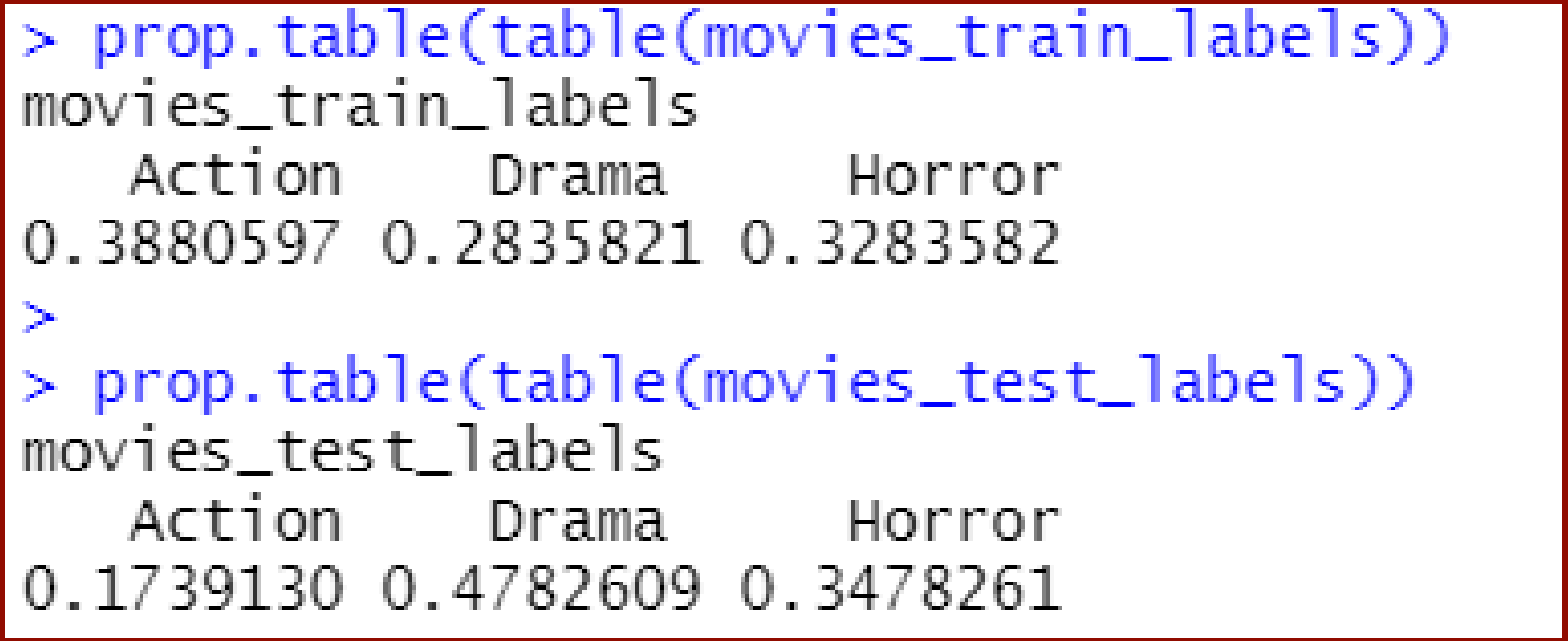
Table of proportions

IN R:



```r
prop.table(table(movies_train_labels))

prop.table(table(movies_test_labels))
```

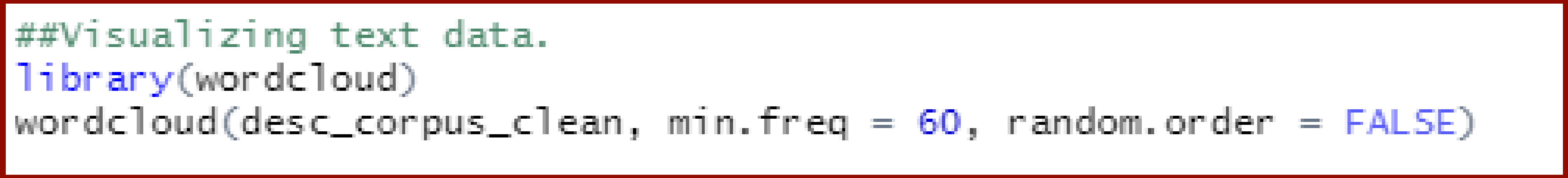
OUTPUT:



```
> prop.table(table(movies_train_labels))
movies_train_labels
   Action     Drama    Horror 
0.3880597 0.2835821 0.3283582 
> 
> prop.table(table(movies_test_labels))
movies_test_labels
   Action     Drama    Horror 
0.1739130 0.4782609 0.3478261 
```

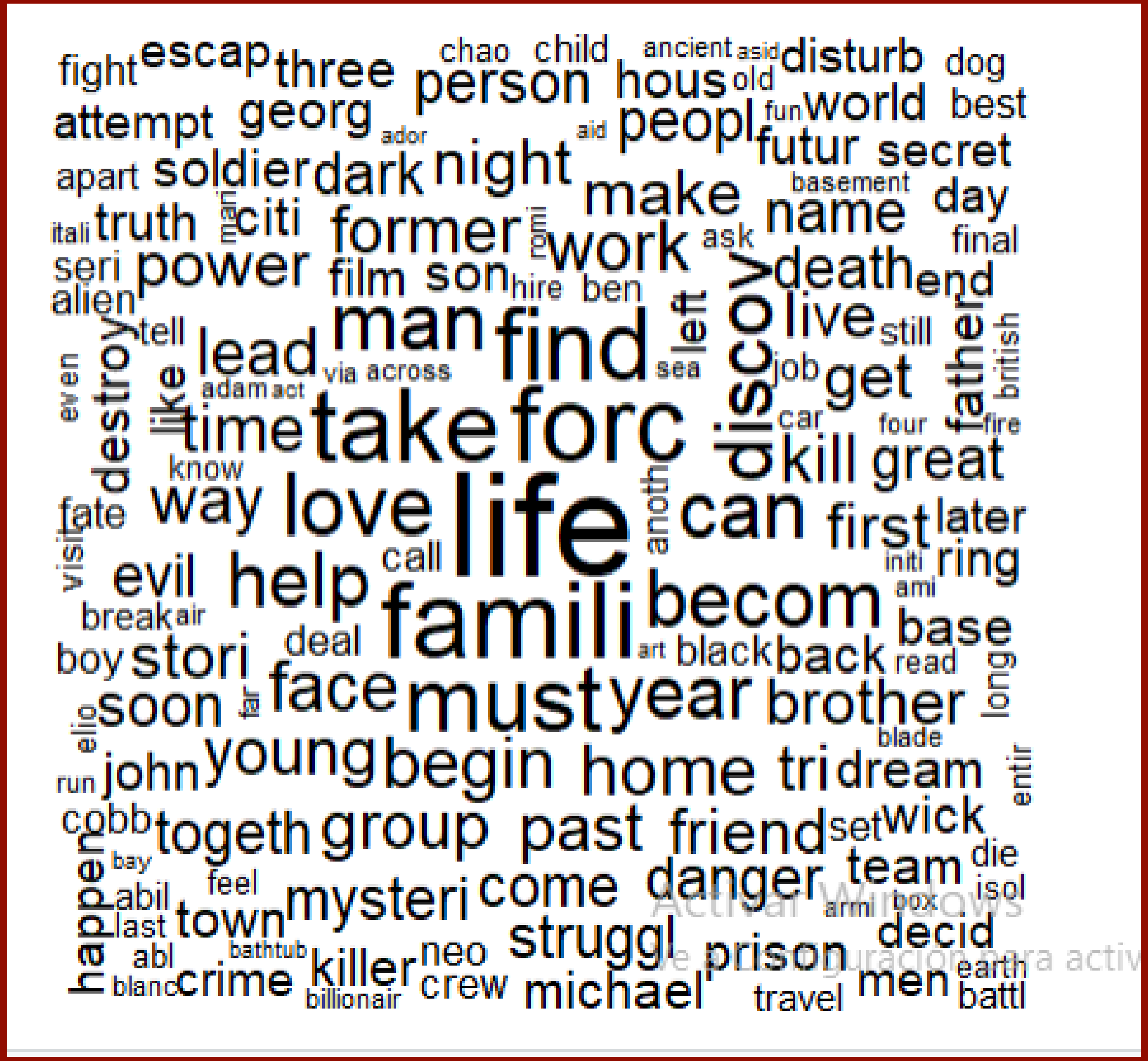
Word clouds

IN R:



```
##Visualizing text data.
library(wordcloud)
wordcloud(desc_corpus_clean, min.freq = 60, random.order = FALSE)
```

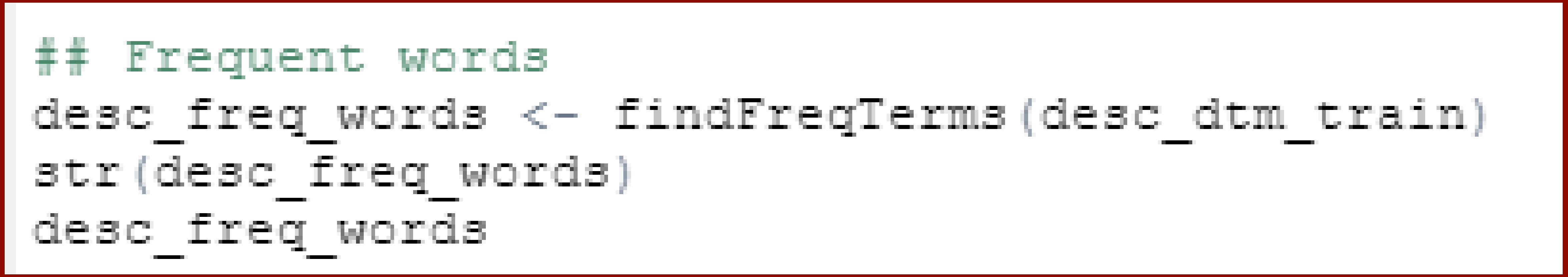
OUTPUT:



You can see the most common words represented by the largest words and the less common words, represented by the smallest.

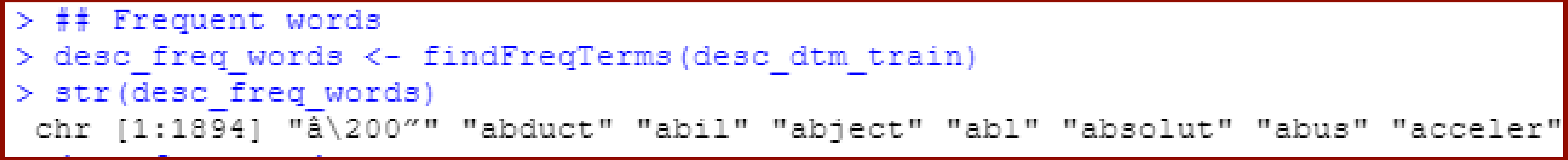
Frequent terms

IN R:



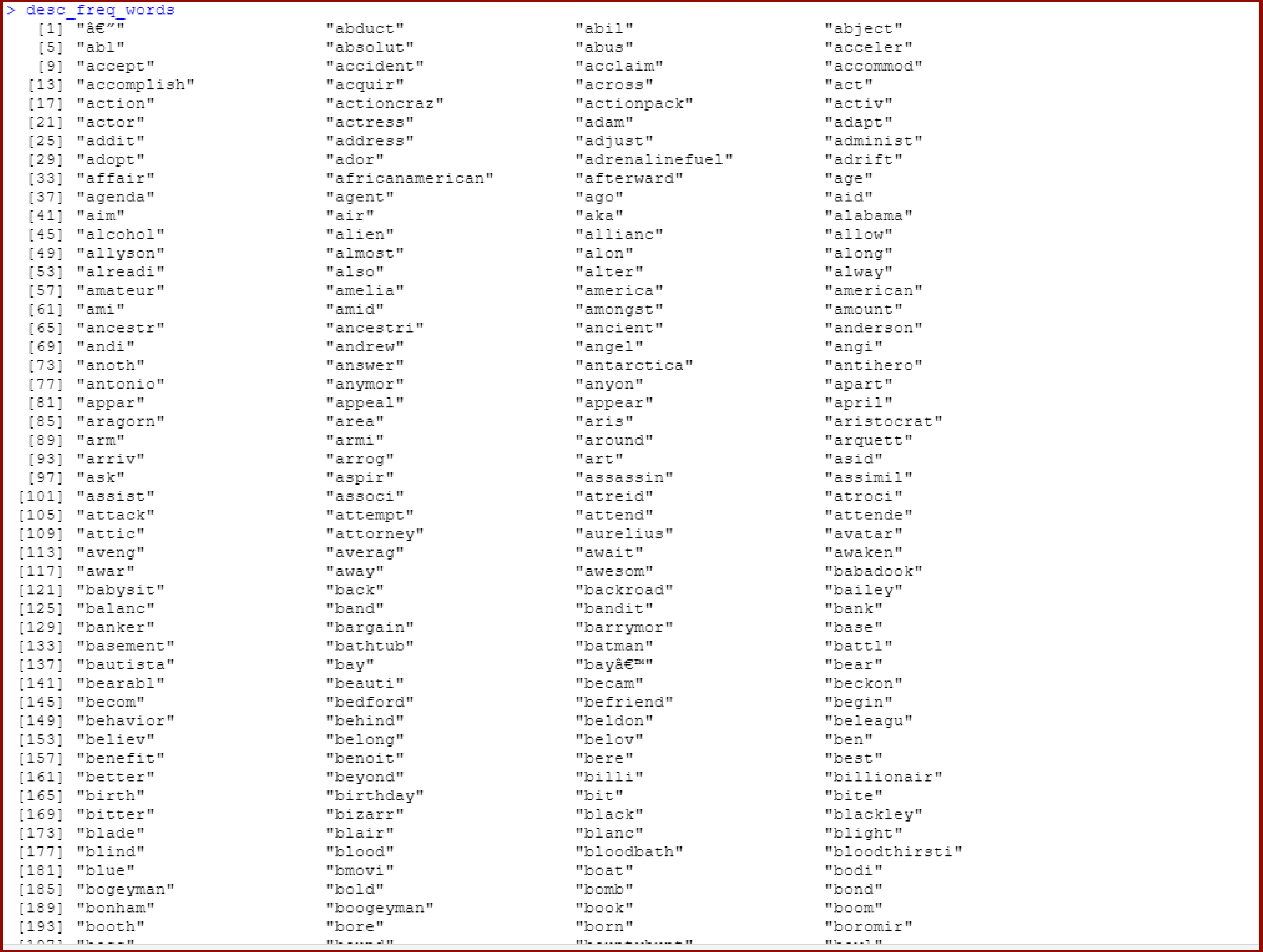
```
## Frequent words
desc_freq_words <- findFreqTerms(desc_dtm_train)
str(desc_freq_words)
desc_freq_words
```

OUTPUT:



```
> ## Frequent words
> desc_freq_words <- findFreqTerms(desc_dtm_train)
> str(desc_freq_words)
 chr [1:1894] "â\200“" "abduct" "abil" "abject" "abl" "absolut" "abus" "acceler"
```

OUTPUT:



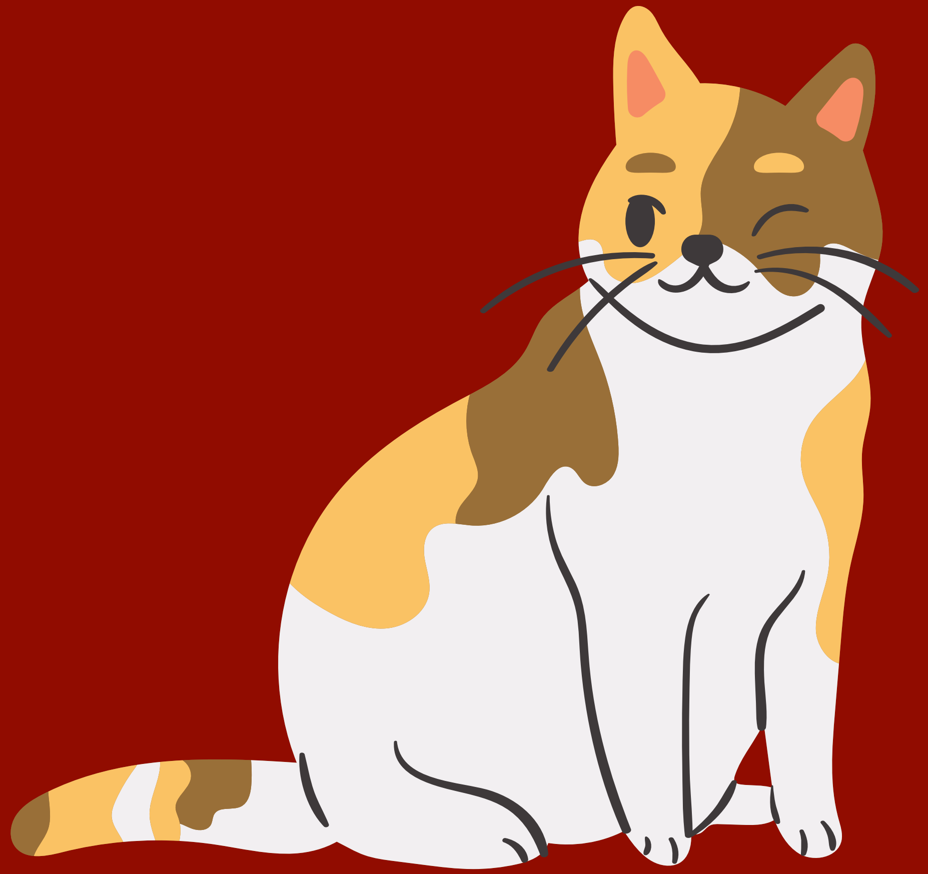
```
> desc_freq_words
  [1] "â€”"                    "abduct"                 "abil"                   "abject"
  [5] "abl"                    "absolut"                "abus"                   "acceler"
  [9] "accept"                 "accident"               "acclaim"                "accommod"
 [13] "accomplish"             "acquir"                 "across"                 "act"
 [17] "action"                 "actioncraz"             "actionpack"             "activ"
 [21] "actor"                  "actress"                "adam"                   "adapt"
 [25] "addit"                  "address"                "adjust"                 "administ"
 [29] "adopt"                  "ador"                   "adrenalinefuel"         "adrift"
 [33] "affair"                 "africanamerican"        "afterward"              "age"
 [37] "agenda"                 "agent"                  "ago"                    "aid"
 [41] "aim"                    "air"                    "aka"                    "alabama"
 [45] "alcohol"                "alien"                  "allianc"                "allow"
 [49] "allyson"                "almost"                 "alon"                   "along"
 [53] "alreadi"                "also"                   "alter"                  "alway"
 [57] "amateur"                "amelia"                 "america"                "american"
 [61] "ami"                    "amid"                   "amongst"                "amount"
 [65] "ancestr"                "ancestri"               "ancient"                "anderson"
 [69] "andi"                   "andrew"                 "angel"                  "angi"
 [73] "anoth"                  "answer"                 "antarctica"             "antihero"
 [77] "antonio"                "anymor"                 "anyon"                  "apart"
 [81] "appar"                  "appeal"                 "appear"                 "april"
 [85] "aragorn"                "area"                   "aris"                   "aristocrat"
 [89] "arm"                    "armi"                   "around"                 "arquett"
 [93] "arriv"                  "arrog"                  "art"                    "asid"
 [97] "ask"                    "aspir"                  "assassin"               "assimil"
[101] "assist"                 "associ"                 "atreid"                 "atroci"
[105] "attack"                 "attempt"                "attend"                 "attende"
[109] "attic"                  "attorney"               "aurelius"               "avatar"
[113] "aveng"                  "averag"                 "await"                  "awaken"
[117] "awar"                   "away"                   "awesom"                 "babadook"
[121] "babysit"                "back"                   "backroad"               "bailey"
[125] "balanc"                 "band"                   "bandit"                 "bank"
[129] "banker"                 "bargain"                "barrymor"               "base"
[133] "basement"               "bathtub"                "batman"                 "battl"
[137] "bautista"               "bay"                    "bayâ€™"                 "bear"
[141] "bearabl"                "beauti"                 "becam"                  "beckon"
[145] "becom"                  "bedford"                "befriend"               "begin"
[149] "behavior"               "behind"                 "beldon"                 "beleagu"
[153] "believ"                 "belong"                 "belov"                  "ben"
[157] "benefit"                "benoit"                 "bere"                   "best"
[161] "better"                 "beyond"                 "billi"                  "billionair"
[165] "birth"                  "birthday"               "bit"                    "bite"
[169] "bitter"                 "bizarr"                 "black"                  "blackley"
[173] "blade"                  "blair"                  "blanc"                  "blight"
[177] "blind"                  "blood"                  "bloodbath"              "bloodthirsti"
[181] "blue"                   "bmovi"                  "boat"                   "bodi"
[185] "bogeyman"               "bold"                   "bomb"                   "bond"
[189] "bonham"                 "boogeyman"              "book"                   "boom"
[193] "booth"                  "bore"                   "born"                   "boromir"
```

PROCESSING AND RESULT

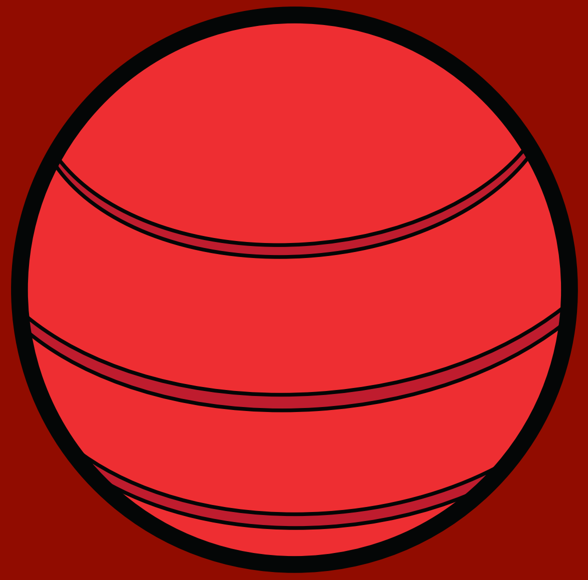
Naive Bayes

Let's put it simple.

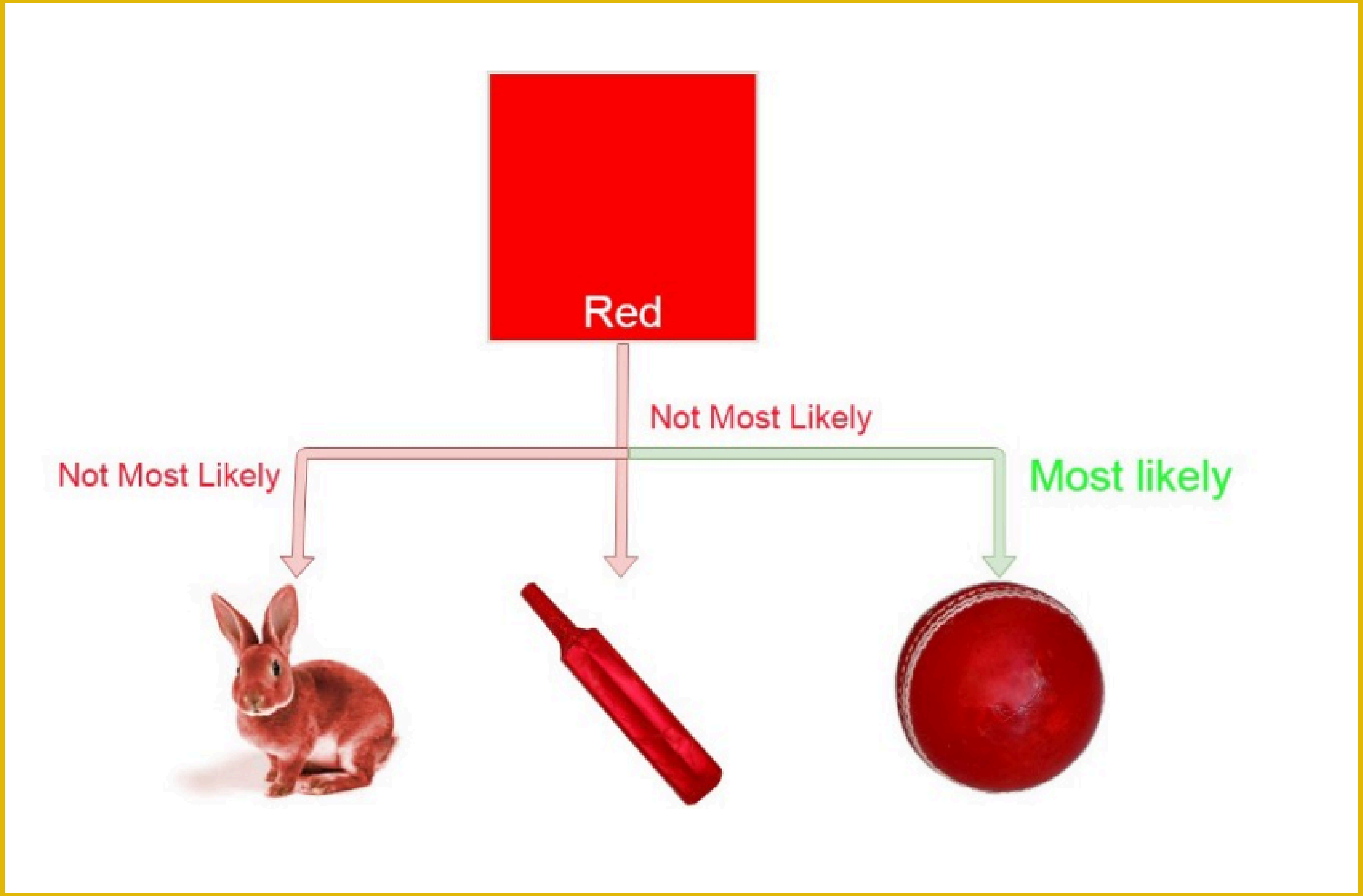
Let’s assume that you are walking on the playground. Now you see some red object in front of you. **This red object can be a bat or a cat or a ball. You will definitely assume that it will be a ball.** But why so?



Let’s us think you are making a machine and you have given the task as above to classify an object in between bat, ball and a cat. At first you will think of creating a machine that will identify the characters of the object and then map it with your classification objects such that if an object is a circle then it will be a ball or if the object is living-being then it will be a cat or in our case, if our object is red then it is most probable that it will be a ball. Why so? because from our childhood we have seen a red ball but a red cat or a red bat is very unlikely to our eyes.



So in our case, we can classify an object by mapping its features with our classifier individually. As in our case, this red color was mapped with a bat, a cat, and a ball, but eventually, we get the most probability of red object with a ball and therefore we classified that object with a ball.



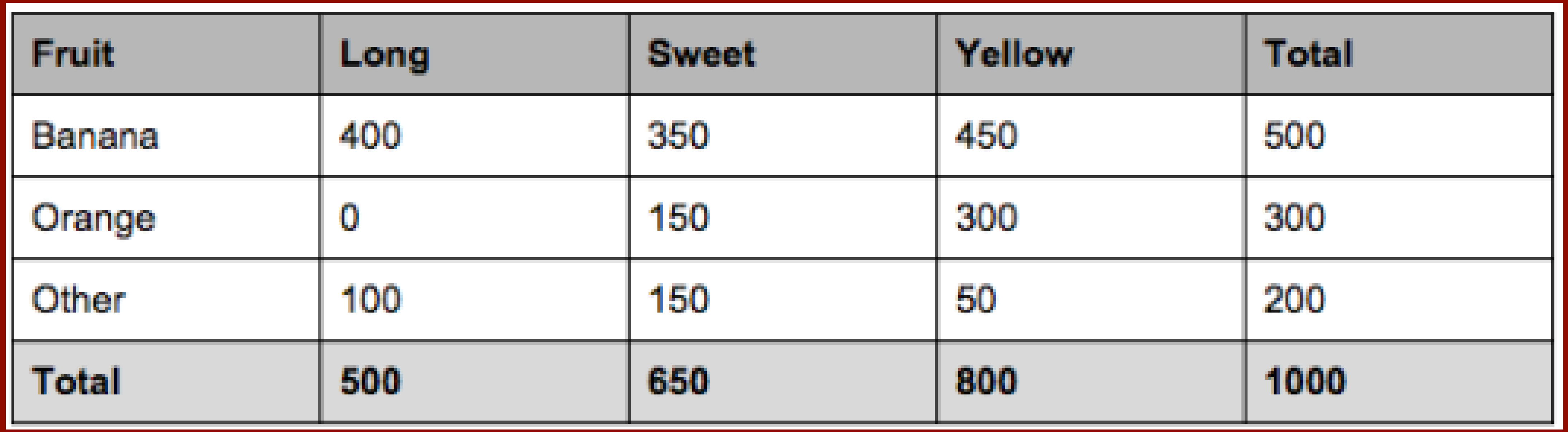
FORMULA

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

For example

We have data on 1000 pieces of fruit. The fruit being a Banana, Orange or some other fruit and imagine we know 3 features of each fruit, whether it's long or not, sweet or not and yellow or not.



Fruit	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	**500**	**650**	**800**	**1000**

So from the table what do we already know?

- 50% of the fruits are bananas
- 30% are oranges
- 20% are other fruits

Based on our training set we can also say the following:

- From 500 bananas 400 (0.8) are Long, 350 (0.7) are Sweet and 450 (0.9) are Yellow
- Out of 300 oranges, 0 are Long, 150 (0.5) are Sweet and 300 (1) are Yellow
- From the remaining 200 fruits, 100 (0.5) are Long, 150 (0.75) are Sweet and 50 (0.25) are Yellow

Which should provide enough evidence to **predict the class** of another fruit as it's introduced.

If we're told that the additional fruit is Long, Sweet and Yellow, we can classify it using the following formula and subbing in the values for each outcome, whether it's a Banana, an Orange or Other Fruit. The one with the highest probability (score) being the winner

Based on our training set we can also say the following:

- From 500 bananas 400 (0.8) are Long, 350 (0.7) are Sweet and 450 (0.9) are Yellow
- Out of 300 oranges, 0 are Long, 150 (0.5) are Sweet and 300 (1) are Yellow
- From the remaining 200 fruits, 100 (0.5) are Long, 150 (0.75) are Sweet and 50 (0.25) are Yellow

Which should provide enough evidence to **predict the class** of another fruit as it's introduced.

If we're told that the additional fruit is Long, Sweet and Yellow, we can classify it using the following formula and subbing in the values for each outcome, whether it's a Banana, an Orange or Other Fruit. The one with the highest probability (score) being the winner

Banana:

$$P\left(\frac{Banana}{Long, Sweet, Yellow}\right) = \frac{P\left(\frac{Long}{Banana}\right) \times P\left(\frac{Sweet}{Banana}\right) \times P\left(\frac{Yellow}{Banana}\right) \times P(Banana)}{P(Long)\,P(Sweet)\,P(Yellow)}$$

$$P\left(\frac{Banana}{Long, Sweet, Yellow}\right) = \frac{(0.8) \times (0.7) \times (0.9) \times (0.5)}{0.25 \times 0.33 \times 0.41}$$

$$P\left(\frac{Banana}{Long, Sweet, Yellow}\right) = 0.252$$

Orange:

$$P\left(\frac{Orange}{Long, Sweet, Yellow}\right) = 0$$

Other Fruit:

$$P\left(\frac{Other}{Long, Sweet, Yellow}\right) = \frac{P\left(\frac{Long}{Other}\right) \times P\left(\frac{Sweet}{Other}\right) \times P\left(\frac{Yellow}{Other}\right) \times P(Other)}{P(Long)\, P(Sweet)\, P(Yellow)}$$

$$P\left(\frac{Other}{Long, Sweet, Yellow}\right) = \frac{(0.5) \times (0.75) \times (0.25) \times (0.2)}{0.25 \times 0.33 \times 0.41}$$

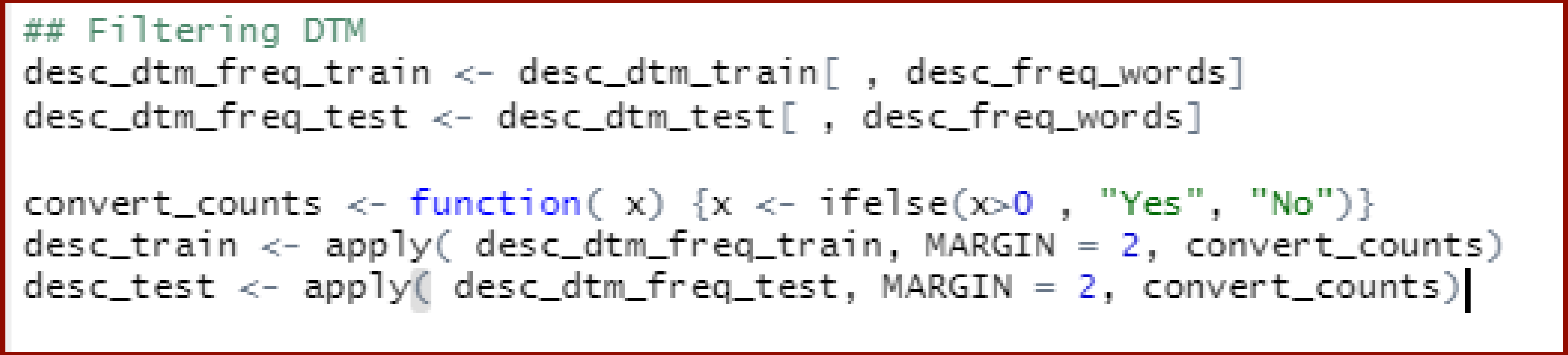
$$P\left(\frac{Other}{Long, Sweet, Yellow}\right) = 0.01875$$

The result:

In this case, based on the higher score (0.252 for banana) we can assume this Long, Sweet and Yellow fruit is in fact, a Banana.

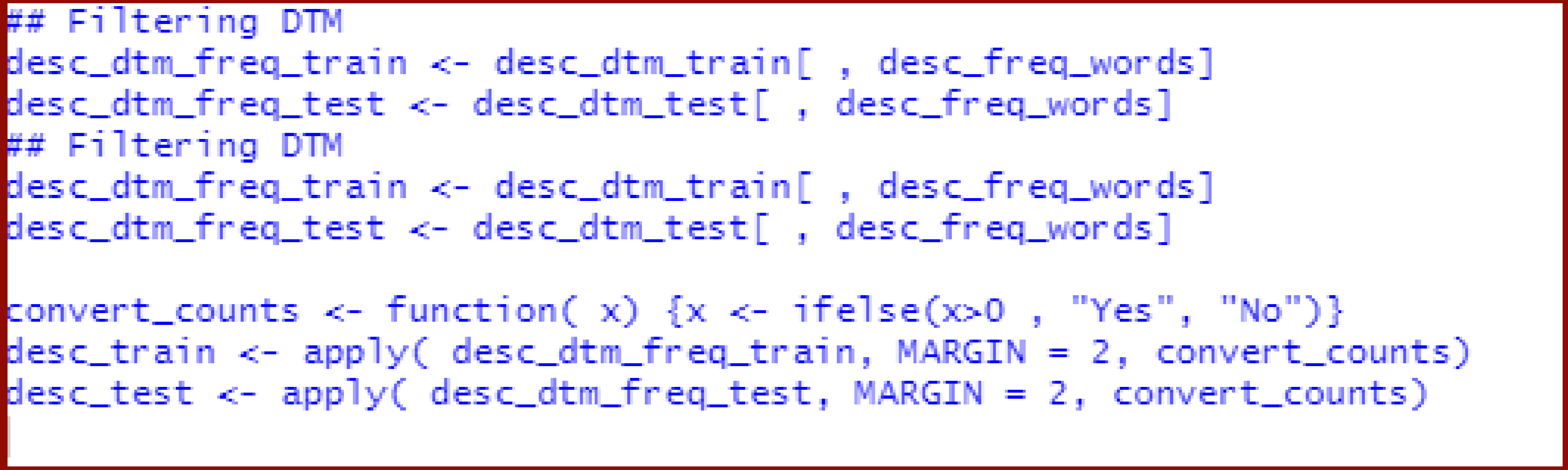
Applying Naive Bayes in our dataset with R

IN R:



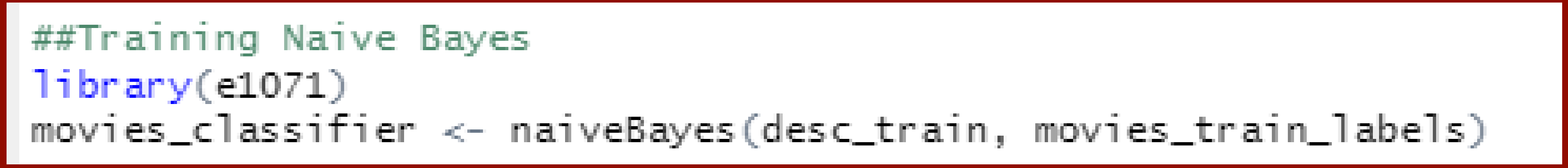
```
## Filtering DTM
desc_dtm_freq_train <- desc_dtm_train[ , desc_freq_words]
desc_dtm_freq_test <- desc_dtm_test[ , desc_freq_words]

convert_counts <- function( x) {x <- ifelse(x>0 , "Yes", "No")}
desc_train <- apply( desc_dtm_freq_train, MARGIN = 2, convert_counts)
desc_test <- apply( desc_dtm_freq_test, MARGIN = 2, convert_counts)
```

OUTPUT:



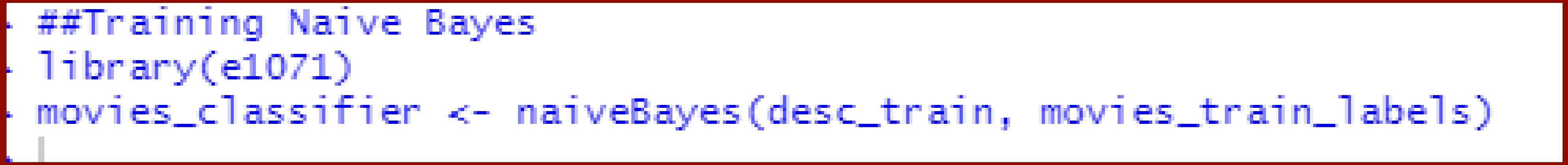
```
## Filtering DTM
desc_dtm_freq_train <- desc_dtm_train[ , desc_freq_words]
desc_dtm_freq_test <- desc_dtm_test[ , desc_freq_words]
## Filtering DTM
desc_dtm_freq_train <- desc_dtm_train[ , desc_freq_words]
desc_dtm_freq_test <- desc_dtm_test[ , desc_freq_words]

convert_counts <- function( x) {x <- ifelse(x>0 , "Yes", "No")}
desc_train <- apply( desc_dtm_freq_train, MARGIN = 2, convert_counts)
desc_test <- apply( desc_dtm_freq_test, MARGIN = 2, convert_counts)
```

IN R:



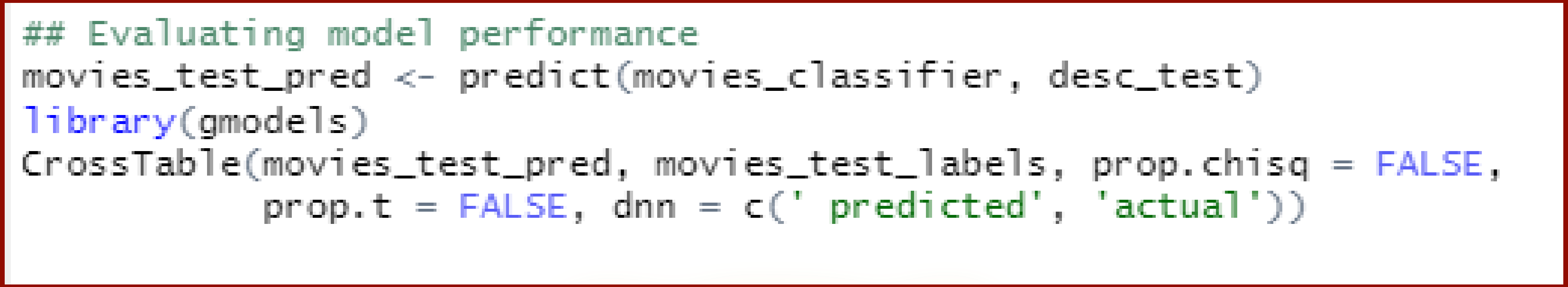
```
##Training Naive Bayes
library(e1071)
movies_classifier <- naiveBayes(desc_train, movies_train_labels)
```

OUTPUT:



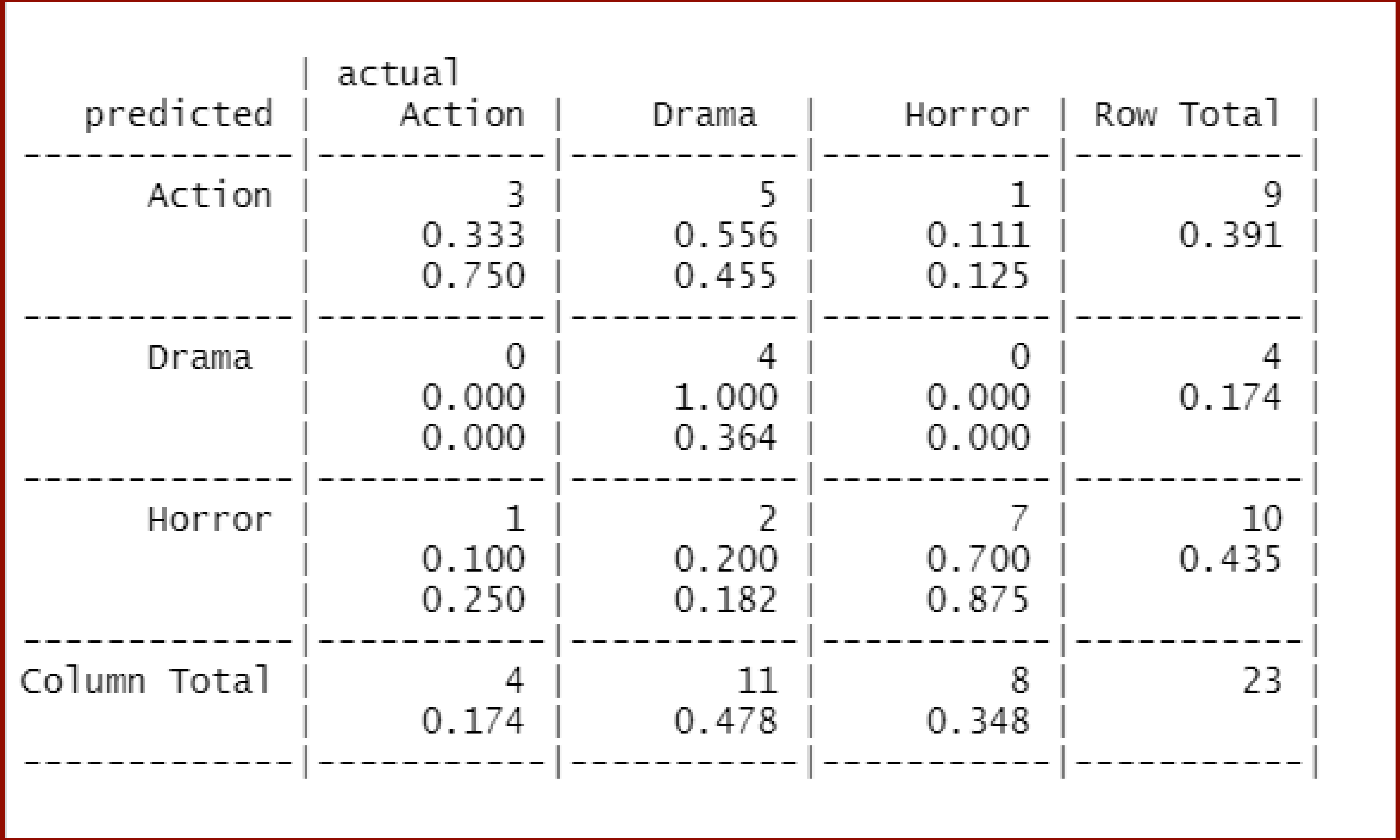
```
> ##Training Naive Bayes
> library(e1071)
> movies_classifier <- naiveBayes(desc_train, movies_train_labels)
> 
```

IN R:



```r
## Evaluating model performance
movies_test_pred <- predict(movies_classifier, desc_test)
library(gmodels)
CrossTable(movies_test_pred, movies_test_labels, prop.chisq = FALSE,
           prop.t = FALSE, dnn = c(' predicted', 'actual'))
```

OUTPUT:



```
             | actual
   predicted |    Action |     Drama |    Horror | Row Total |
-------------|-----------|-----------|-----------|-----------|
      Action |         3 |         5 |         1 |         9 |
             |     0.333 |     0.556 |     0.111 |     0.391 |
             |     0.750 |     0.455 |     0.125 |           |
-------------|-----------|-----------|-----------|-----------|
       Drama |         0 |         4 |         0 |         4 |
             |     0.000 |     1.000 |     0.000 |     0.174 |
             |     0.000 |     0.364 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|
      Horror |         1 |         2 |         7 |        10 |
             |     0.100 |     0.200 |     0.700 |     0.435 |
             |     0.250 |     0.182 |     0.875 |           |
-------------|-----------|-----------|-----------|-----------|
Column Total |         4 |        11 |         8 |        23 |
             |     0.174 |     0.478 |     0.348 |           |
-------------|-----------|-----------|-----------|-----------|
```

Classification outputs

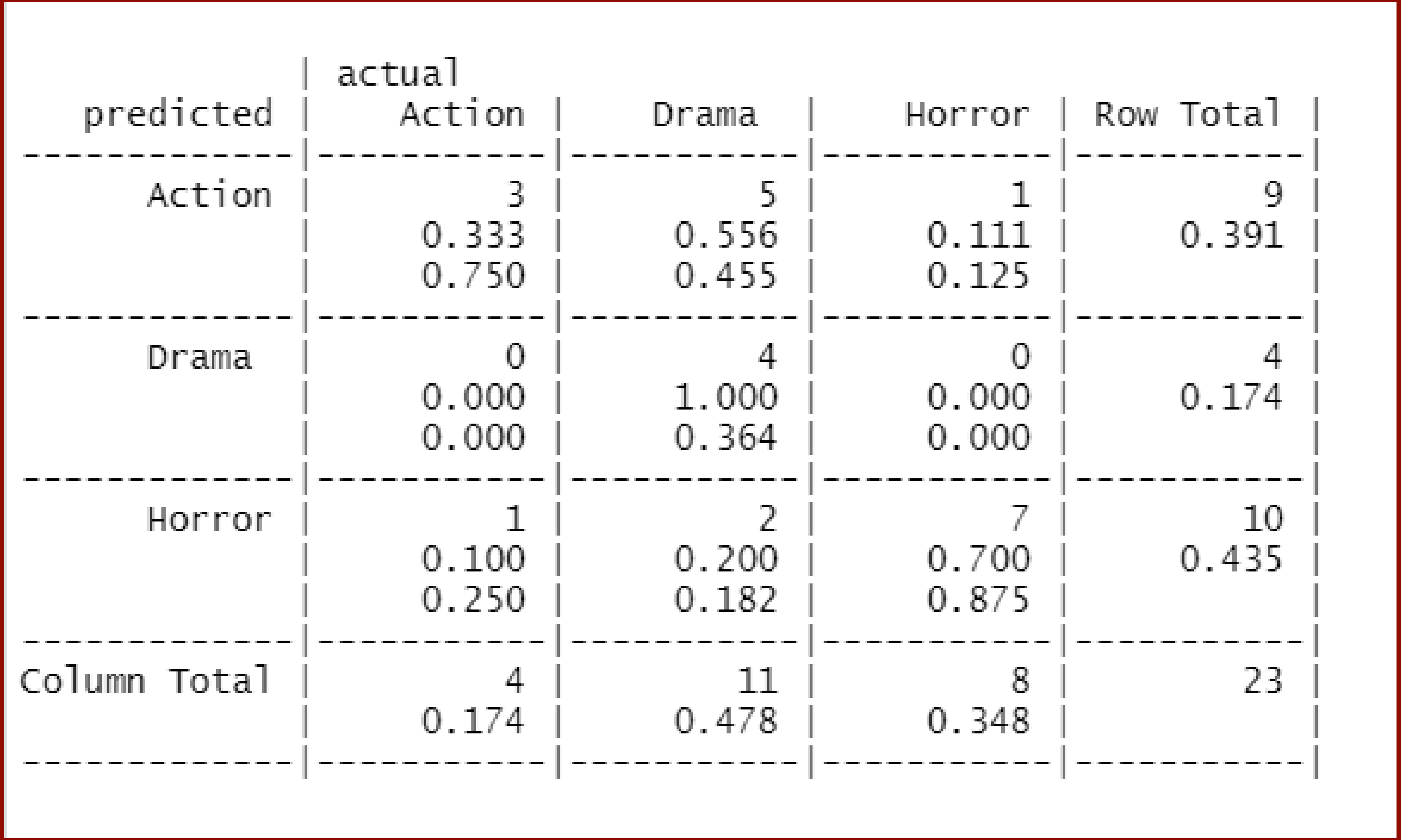
Multi-Class Classification

Multi-class classification refers to those classification tasks that have more than two class labels.

Unlike binary classification, multi-class classification does not have the notion of normal and abnormal outcomes. Instead, examples are classified as belonging to one among a range of known classes.

In this case as we want to predict 3 types of genres, the classification output of our dataset with naive bayes it's Multi-Class Classification

Frequency tables and interpretation



```
             | actual 
   predicted |    Action |     Drama |    Horror | Row Total | 
-------------|-----------|-----------|-----------|-----------|
      Action |         3 |         5 |         1 |         9 | 
             |     0.333 |     0.556 |     0.111 |     0.391 | 
             |     0.750 |     0.455 |     0.125 |           | 
-------------|-----------|-----------|-----------|-----------|
      Drama  |         0 |         4 |         0 |         4 | 
             |     0.000 |     1.000 |     0.000 |     0.174 | 
             |     0.000 |     0.364 |     0.000 |           | 
-------------|-----------|-----------|-----------|-----------|
      Horror |         1 |         2 |         7 |        10 | 
             |     0.100 |     0.200 |     0.700 |     0.435 | 
             |     0.250 |     0.182 |     0.875 |           | 
-------------|-----------|-----------|-----------|-----------|
Column Total |         4 |        11 |         8 |        23 | 
             |     0.174 |     0.478 |     0.348 |           | 
-------------|-----------|-----------|-----------|-----------|
```

CONCLUSIONS AND LIMITATIONS

Does the study generalize to other domains?

Of course! As like KNN, we can use Naive Bayes to predict a lot of other things.
The best times to use it is when we want to use text classification, when the dataset is big and when the training sets are small.

Limitations

Naive bayes it's not as great estimator as can be, so we don't think it's reliable in certain cases as predict which disease someone has based on symptoms and so.

Advantages

Good with muticlass prediction, just like this case. Amazing results in text classification. And so easy to understand.

What would you do to improve your analysis?

Make sure that the movie description has more specific words, because we found that drama it's pretty much difficult to predict, because drama can be easily confused with the other two genres, some drama movies fit action movies as well, like a war movie that its drama driven, if the algorithm takes the word 'war' as action instead of drama, then the algorithm effectivity gets lower. And if possible, write our own descriptions to reduce the ambiguous descriptions that some movies have.

What is the main weakness of your project?

The weakness of our project is that the plots of the movies are difficult to classify, it is possible, but there are times when people cannot guess the genre of a movie just by reading the plot or a synopsis of a movie, so for the algorithm is not so easy to classify them.

Bibliography

- https://www.rottentomatoes.com/
- https://www.imdb.com/
- https://movies.fandom.com/wiki/Moviepedia
- https://stackoverflow.com/questions/12980081/create-a-stacked-density-graph-in-ggplot2
- https://machinelearningmastery.com/types-of-classification-in-machine-learning/
- Gaurav Chauhan, https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf