# NOTES 10

## Information Retrieval

What is the current crude oil price level?

# Data Crude20

- Package (tm) in R Feinerer & Hornik
- Adapted from data("crude")
- 19 news articles/documents from the Reuters-21578 data set. All documents belong to the topic crude dealing with crude oil
- Crude20 is a corpus, which is a collection of textual electronic documents.

```
library(NLP)
library(tm)

source <- DirSource(directory = "I:\\CrudeCorpusPackagetm", encoding =
"UTF-8") #input path for documents

crude20 <- Corpus(source, readerControl=list(reader=readPlain))

strwrap(crude20[[1]])
```

# E.g., the first document

```
> strwrap(crude20[[1]])
 [1] "Diamond Shamrock Corp said that"
 [2] "effective today it had cut its contract prices for crude oil by"
 [3] "1.50 dlrs a barrel."
 [4] "The reduction brings its posted price for West Texas"
 [5] "Intermediate to 16.00 dlrs a barrel, the copany said."
 [6] "\"The price reduction today was made in the light of falling"
 [7] "oil product prices and a weak crude oil market,\" a company"
 [8] "spokeswoman said."
 [9] "Diamond is the latest in a line of U.S. oil companies that"
[10] "have cut its contract, or posted, prices over the last two days"
[11] "citing weak oil markets."
[12] "Reuter"
```

# Data Representation

- By meaning?
- bag of words:  count the number of times that each word appears in the document. Every document corresponds to a vector of word-counts.
- Set of *x* documents and *m* terms/words
- Each document is a vector **v** in $\mathbf{R}^m$
- Document-term matrix

# Example of bag of words

- Doc1: A model represents a document as a vector of identifiers, known as term.
- Doc2: A classical model is developed on Boolean logic and classical set theory
- Query/Doc3: term vector model
- Document-Term Matrix:

| Docs | and | boolean | classical | developed | document | identifiers | known |
|---|---|---|---|---|---|---|---|
| **1** | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| **2** | 1 | 1 | 2 | 1 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Docs | logic | model | represents | set | term | theory | vector |
|---|---|---|---|---|---|---|---|
| **1** | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| **2** | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| **3** | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

# A query

- crude20[[20]] is: 'what is the current crude oil price level‘

- Now we want to find the similarity between each of the 19 documents and the new query.

```
corp = VCorpus(VectorSource(crude20))
> corp
A corpus with 20 text documents
```

# Distance Calculation

- We next build the document-term matrix for this new corpus using the function `DocumentTermMatrix`

- Then calculate the Euclidean distance between the new query and all other 19 documents

```
q= c("what","the","level","crude","oil","price") ###terms in
the 20th text##
dtm =
DocumentTermMatrix(corp,control=list(tolower=TRUE,removePunct
uation=TRUE,removeNumbers=TRUE)) ##Document Term Matrix##
newdtm = as.matrix(dtm)
dist =
sqrt(rowSums((scale(newdtm,center=newdtm[20,],scale=F)^2)))
####Distance###
mat = cbind(newdtm[,q],dist)
colnames(mat) = c(q,"dist")
```

# Results

| | what | the | level | crude | oil | price | dist |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 6 | 0 | 2 | 5 | 2 | 12.083046 |
| 2 | 1 | 19 | 0 | 0 | 12 | 1 | 39.064050 |
| 3 | 0 | 4 | 0 | 2 | 2 | 2 | 7.615773 |
| 4 | 0 | 4 | 0 | 3 | 1 | 2 | 8.602325 |
| 5 | 0 | 8 | 0 | 0 | 1 | 0 | 12.409674 |
| 6 | 0 | 15 | 0 | 2 | 7 | 2 | 34.799425 |
| 7 | 1 | 30 | 1 | 0 | 3 | 0 | 40.000000 |
| 8 | 0 | 6 | 0 | 0 | 3 | 0 | 13.601471 |
| 9 | 0 | 18 | 0 | 0 | 5 | 1 | 30.347982 |
| 10 | 0 | 27 | 0 | 0 | 9 | 0 | 37.148351 |
| 11 | 0 | 21 | 0 | 5 | 5 | 0 | 35.637059 |
| 12 | 0 | 5 | 0 | 2 | 4 | 0 | 10.908712 |
| 13 | 0 | 7 | 0 | 0 | 5 | 0 | 13.747727 |
| 14 | 0 | 4 | 0 | 2 | 4 | 0 | 11.532563 |
| 15 | 0 | 11 | 0 | 0 | 3 | 0 | 15.066519 |
| 16 | 0 | 8 | 0 | 0 | 4 | 1 | 15.524175 |
| 17 | 0 | 13 | 0 | 0 | 5 | 1 | 20.322401 |
| 18 | 0 | 5 | 0 | 2 | 3 | 1 | 11.489125 |
| 19 | 0 | 21 | 0 | 0 | 3 | 0 | 31.368774 |
| query | 1 | 1 | 1 | 1 | 1 | 1 | 0.000000 |

# Normalization

- Documents have different lengths, which can affect the amount of words

- e.g. document 1 has 91 words and document 2 has 443 words.

2 ways to normalize

1. Document length normalization: divide each vector (the word account for each term) by the total number of words in the document.

2. Euclidean length normalization: divide each vector (the word account for each term) by the Euclidean length of the vector $||\mathbf{X}||$.

# Results after document length normalization

| | what | the | level | crude | oil | price… | dist.dln |
|---|---|---|---|---|---|---|---|
| 1 | 0.0000 | 0.0800 | 0.0000 | 0.0267 | 0.0667 | 0.0267 | 0.3402 |
| 2 | 0.0028 | 0.0523 | 0.0000 | 0.0000 | 0.0331 | 0.0028 | 0.3569 |
| 3 | 0.0000 | 0.0889 | 0.0000 | 0.0444 | 0.0444 | 0.0444 | 0.3383 |
| 4 | 0.0000 | 0.0727 | 0.0000 | 0.0545 | 0.0182 | 0.0364 | 0.3459 |
| 5 | 0.0000 | 0.1096 | 0.0000 | 0.0000 | 0.0137 | 0.0000 | 0.3723 |
| 6 | 0.0000 | 0.0414 | 0.0000 | 0.0055 | 0.0193 | 0.0055 | 0.3632 |
| 7 | 0.0029 | 0.0860 | 0.0029 | 0.0000 | 0.0086 | 0.0000 | 0.3576 |
| 8 | 0.0000 | 0.0526 | 0.0000 | 0.0000 | 0.0263 | 0.0000 | 0.3647 |
| 9 | 0.0000 | 0.0714 | 0.0000 | 0.0000 | 0.0198 | 0.0040 | 0.3617 |
| 10 | 0.0000 | 0.0961 | 0.0000 | 0.0000 | 0.0320 | 0.0000 | 0.3529 |
| 11 | 0.0000 | 0.0684 | 0.0000 | 0.0163 | 0.0163 | 0.0000 | 0.3579 |
| 12 | 0.0000 | 0.0625 | 0.0000 | 0.0250 | 0.0500 | 0.0000 | 0.3529 |
| 13 | 0.0000 | 0.0795 | 0.0000 | 0.0000 | 0.0568 | 0.0000 | 0.3612 |
| 14 | 0.0000 | 0.0488 | 0.0000 | 0.0244 | 0.0488 | 0.0000 | 0.3602 |
| 15 | 0.0000 | 0.1222 | 0.0000 | 0.0000 | 0.0333 | 0.0000 | 0.3592 |
| 16 | 0.0000 | 0.0690 | 0.0000 | 0.0000 | 0.0345 | 0.0086 | 0.3608 |
| 17 | 0.0000 | 0.0823 | 0.0000 | 0.0000 | 0.0316 | 0.0063 | 0.3554 |
| 18 | 0.0000 | 0.0794 | 0.0000 | 0.0317 | 0.0476 | 0.0159 | 0.3606 |
| 19 | 0.0000 | 0.0901 | 0.0000 | 0.0000 | 0.0129 | 0.0000 | 0.3637 |
| query | 0.1429 | 0.1429 | 0.1429 | 0.1429 | 0.1429 | 0.1429 | 0.0000 |

# Results after Euclidean length normalization

| | what | the | level | crude | oil | price… | dist.l2n |
|---|---|---|---|---|---|---|---|
| 1 | 0.0000 | 0.4615 | 0.0000 | 0.1538 | 0.3846 | 0.1538 | 1.0620 |
| 2 | 0.0251 | 0.4766 | 0.0000 | 0.0000 | 0.3010 | 0.0251 | 1.1560 |
| 3 | 0.0000 | 0.4747 | 0.0000 | 0.2374 | 0.2374 | 0.2374 | 1.0502 |
| 4 | 0.0000 | 0.4288 | 0.0000 | 0.3216 | 0.1072 | 0.2144 | 1.0907 |
| 5 | 0.0000 | 0.6228 | 0.0000 | 0.0000 | 0.0778 | 0.0000 | 1.2126 |
| 6 | 0.0000 | 0.4232 | 0.0000 | 0.0564 | 0.1975 | 0.0564 | 1.2023 |
| 7 | 0.0245 | 0.7357 | 0.0245 | 0.0000 | 0.0736 | 0.0000 | 1.1624 |
| 8 | 0.0000 | 0.4264 | 0.0000 | 0.0000 | 0.2132 | 0.0000 | 1.2095 |
| 9 | 0.0000 | 0.5803 | 0.0000 | 0.0000 | 0.1612 | 0.0322 | 1.1896 |
| 10 | 0.0000 | 0.7103 | 0.0000 | 0.0000 | 0.2368 | 0.0000 | 1.1332 |
| 11 | 0.0000 | 0.5769 | 0.0000 | 0.1374 | 0.1374 | 0.0000 | 1.1646 |
| 12 | 0.0000 | 0.4319 | 0.0000 | 0.1728 | 0.3455 | 0.0000 | 1.1321 |
| 13 | 0.0000 | 0.4877 | 0.0000 | 0.0000 | 0.3484 | 0.0000 | 1.1696 |
| 14 | 0.0000 | 0.3310 | 0.0000 | 0.1655 | 0.3310 | 0.0000 | 1.1723 |
| 15 | 0.0000 | 0.6985 | 0.0000 | 0.0000 | 0.1905 | 0.0000 | 1.1524 |
| 16 | 0.0000 | 0.4961 | 0.0000 | 0.0000 | 0.2481 | 0.0620 | 1.1792 |
| 17 | 0.0000 | 0.6170 | 0.0000 | 0.0000 | 0.2373 | 0.0475 | 1.1482 |
| 18 | 0.0000 | 0.4124 | 0.0000 | 0.1650 | 0.2474 | 0.0825 | 1.1464 |
| 19 | 0.0000 | 0.6559 | 0.0000 | 0.0000 | 0.0937 | 0.0000 | 1.1972 |
| query | 0.3780 | 0.3780 | 0.3780 | 0.3780 | 0.3780 | 0.3780 | 0.0000 |

| | dist | dist.dln | dist.l2n |
|---|---|---|---|
| 1 | 12.08305 | 0.340233 | 1.061967 |
| 2 | 39.06405 | 0.356886 | 1.155974 |
| 3 | 7.615773 | 0.33827 | 1.05018 |
| 4 | 8.602325 | 0.345933 | 1.090669 |
| 5 | 12.40967 | 0.372283 | 1.212584 |
| 6 | 34.79943 | 0.363209 | 1.202259 |
| 7 | 40 | 0.357571 | 1.162416 |
| 8 | 13.60147 | 0.364733 | 1.209456 |
| 9 | 30.34798 | 0.361656 | 1.189567 |
| 10 | 37.14835 | 0.352921 | 1.133183 |
| 11 | 35.63706 | 0.357862 | 1.16457 |
| 12 | 10.90871 | 0.352858 | 1.132111 |
| 13 | 13.74773 | 0.361244 | 1.169608 |
| 14 | 11.53256 | 0.360177 | 1.172343 |
| 15 | 15.06652 | 0.359207 | 1.152379 |
| 16 | 15.52418 | 0.360777 | 1.179216 |
| 17 | 20.3224 | 0.355366 | 1.148207 |
| 18 | 11.48913 | 0.360566 | 1.146374 |
| 19 | 31.36877 | 0.363741 | 1.197218 |
| query | 0 | 0 | 0 |

# R Code

```
# Document length normalization
newdtm.dl = newdtm/rowSums(newdtm)
dist.dl =
sqrt(rowSums((scale(newdtm.dl,center=newdtm.dl[
20,],scale=F)^2)))
mat.dl = cbind(newdtm.dl[,q],dist.dl)
colnames(mat.dl) = c(q,"dist.dl")

# l2 length normalization
newdtm.l2 = newdtm/sqrt(rowSums(newdtm^2))
dist.l2 =
sqrt(rowSums((scale(newdtm.l2,center=newdtm.l2[
20,],scale=F)^2)))
mat.l2 = cbind(newdtm.l2[,q],dist.l2)
```

# Inverse document frequency (IDF)

- Special/rare words: those do not occur too often, help us locate the relevant documents

- Thus, we can assign a weight to each term in the documents. A term will be assigned a smaller weight if it occurs in more documents (not count)

- Let $n_w$ be the number of documents containing the term and D is the total number of document.

- For each vector X, multiply its $w^{th}$ component by IDF(w) = log(D/$n_w$)

- What does it mean if $n_w$=D? Every doc has this word ⇒ weight = 0

# Results after IDF

| | what | the | level | crude | oil | price… | dist.IDF |
|---|---|---|---|---|---|---|---|
| 1 | 0.0000 | 0.0000 | 0.0000 | 0.0307 | 0.0000 | 0.0267 | 0.8095 |
| 2 | 0.0075 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0028 | 0.7729 |
| 3 | 0.0000 | 0.0000 | 0.0000 | 0.0512 | 0.0000 | 0.0444 | 0.8685 |
| 4 | 0.0000 | 0.0000 | 0.0000 | 0.0628 | 0.0000 | 0.0364 | 0.8252 |
| 5 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8536 |
| 6 | 0.0000 | 0.0000 | 0.0000 | 0.0064 | 0.0000 | 0.0055 | 0.7780 |
| 7 | 0.0078 | 0.0000 | 0.0095 | 0.0000 | 0.0000 | 0.0000 | 0.7923 |
| 8 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8085 |
| 9 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0040 | 0.8234 |
| 10 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.7814 |
| 11 | 0.0000 | 0.0000 | 0.0000 | 0.0188 | 0.0000 | 0.0000 | 0.7873 |
| 12 | 0.0000 | 0.0000 | 0.0000 | 0.0288 | 0.0000 | 0.0000 | 0.8354 |
| 13 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8028 |
| 14 | 0.0000 | 0.0000 | 0.0000 | 0.0281 | 0.0000 | 0.0000 | 0.7959 |
| 15 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8735 |
| 16 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0086 | 0.7955 |
| 17 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0063 | 0.7915 |
| 18 | 0.0000 | 0.0000 | 0.0000 | 0.0366 | 0.0000 | 0.0159 | 0.8243 |
| 19 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.8382 |
| query | 0.3910 | 0.0000 | 0.4746 | 0.1646 | 0.0000 | 0.1429 | 0.0000 |

- Can perform both normalization and IDF weighting or just one of them.

# Stop words

- Common words, e.g. “for, of, is”, are not very helpful in locating the relevant information

- We call such common words found in a language as stop words. We want to remove them in counting the word frequency.

- It lists all the stop words
- `> stopwords("english")`

# After removing stop words, any changes?

```
> mat.Stop
       level crude oil price   dist
1          0     2   5     2  9.899495
2          0     0  12     1 29.103264
3          0     2   2     2  6.557439
4          0     3   1     2  7.416198
5          0     0   1     0  9.327379
6          0     2   7     2 26.172505
7          1     0   3     0 23.086793
8          0     0   3     0 10.770330
9          0     0   5     1 21.307276
10         0     0   9     0 24.289916
11         0     5   5     0 27.221315
12         0     2   4     0  9.539392
13         0     0   5     0 11.401754
14         0     2   4     0  9.433981
15         0     0   3     0 10.295630
16         0     0   4     1 11.135529
17         0     0   5     1 13.038405
18         0     2   3     1 10.000000
19         0     0   3     0 22.158520
query      1     1   1     1  0.000000
```

1
2
3

```
###Inverse document frequency###
dtm.IDF = DocumentTermMatrix(corp,
control=list(tolower=TRUE,removePunctuation=TRUE,removeNumbers=TRUE,weighting=weight
TfIdf))

newdtm.IDF = as.matrix(dtm.IDF)
dist.IDF = sqrt(rowSums((scale(newdtm.IDF,center=newdtm.IDF[20,],scale=F)^2)))
mat.IDF = cbind(newdtm.IDF[,q],dist.IDF )
colnames(mat.IDF ) = c(q,"dist")

####Stop Words###
dtm.Stop = DocumentTermMatrix(corp,control=list(tolower=TRUE,
removePunctuation=TRUE,removeNumbers=TRUE,stopwords = TRUE))
newdtm.Stop = as.matrix(dtm.Stop)

dist.Stop = sqrt(rowSums((scale(newdtm.Stop,center=newdtm.Stop[20,],scale=F)^2)))
q1 = c("level","crude","oil","price")  ###terms in the 20th text after removing stop
words##
mat.Stop= cbind(newdtm.Stop[,q1],dist.Stop)
colnames(mat.Stop) = c(q1,"dist")
```

# Information Analysis

- Information Preprocessing/Cleaning
  - Converting to lowercase
  - Remove numbers
- Information Exploration
  - Find frequent items
  - Association
  - Wordcount
  - Clustering

# Converting to lowercase

```
crude20lower <- tm_map(crude20, tolower)
```

- Example

Before:

- `Diamond Shamrock Corp said that…`

After

- `diamond shamrock corp said that…`

# Remove numbers

- Use it if numbers are irrelevant to the analysis

```
crude20removeNumbers <- tm_map(crude20, removeNumbers)
```

- Example

Before:

```
 The reduction brings its posted price for West Texas Intermediate to 16.00 dlrs a barrel, the copany said.
```

After:

```
 The reduction brings its posted price for West Texas Intermediate to . dlrs a barrel, the copany said.
```

# Find frequent items

- We can use the function findFreqTerms() to find all the terms in the corpus whose frequency is >=threshold
- We can find the popular words in the documents

```
> findFreqTerms(dtm.Stop, lowfreq=10)
 [1] "barrel"     "bpd"        "crude"      "dlrs"
"government"
 [6] "industry"   "kuwait"     "market"     "meeting"
"minister"
[11] "mln"        "official"   "oil"        "opec"
"pct"
[16] "price"      "prices"     "production" "reuter"
"saudi"
[21] "sheikh"     "world"
```

# Find item association

- We can find associations with a word, specifying a minimum correlation.
- Correlation will be 1 for two words always come together.

```
> findAssocs(dtm.Stop, c("oil"), corlimit=0.7)
     opec     named      late    prices    trying    winter   markets
     0.87      0.81      0.79      0.79      0.79      0.79      0.78
 analysts agreement emergency    buyers     fixed
     0.77      0.76      0.74      0.71      0.71
```

# Wordcount

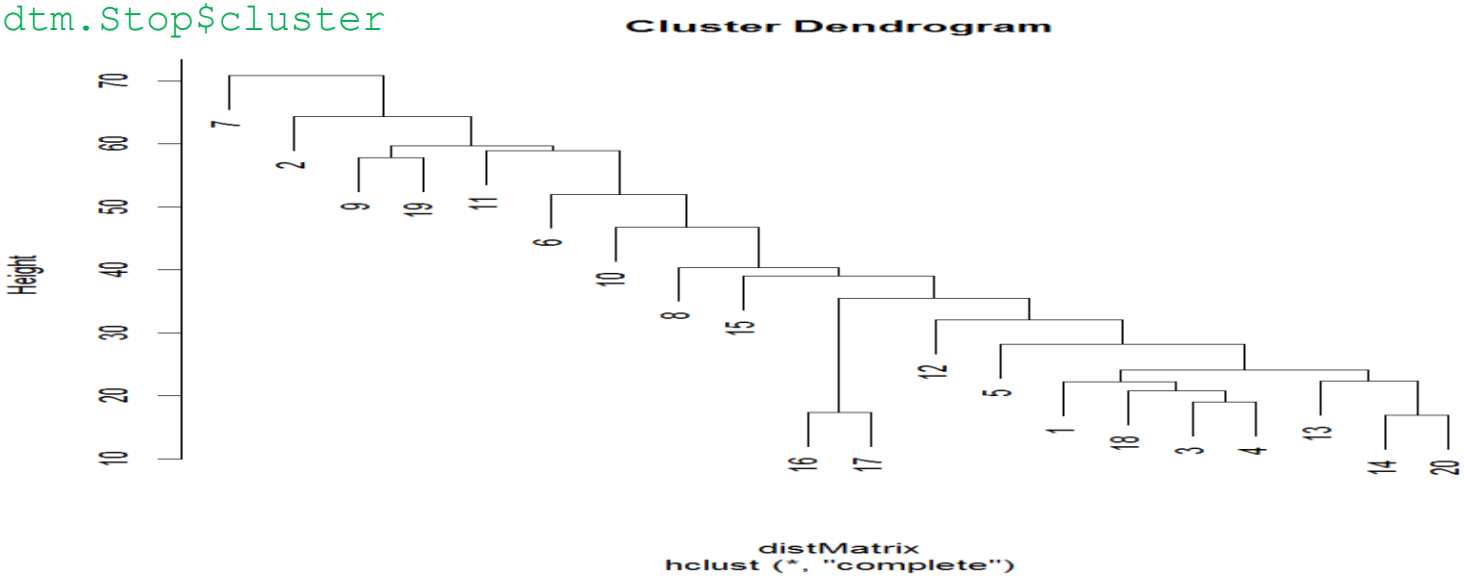
- We can use wordcloud to visually show the popularity of frequent terms in a corpus.

```
# generate wordcloud
library(wordcloud)
freq <- colSums(as.matrix(dtm.Stop[,findFreqTerms(dtm.Stop,
lowfreq=10)]))
wordcloud(names(freq), freq,
min.freq=5,random.color=TRUE,colors=rainbow(7))
```

# Clustering

- We can cluster documents based on their topic similarity.
- We can perform both hierarchical clustering and k-means clustering.

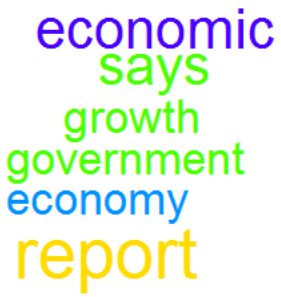
```
###Clustering###
### compute distances###
distMatrix <- dist(scale(dtm.Stop))
fit <- hclust(distMatrix)
plot(fit)
clusterdtm.Stop=kmeans(dtm.Stop,3)
clusterdtm.Stop$cluster
```




```
> clusterdtm.Stop$cluster
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 3  1  3  3  3  1  2  3  3  1  1  3  3  3  3  3  3  3  3  3
```

# Cluster Wordclouds

Used in News industry



```
layout(matrix(c(1,2,3),1,3))
for(k in 1:3){
 cl <- which(clusterdtm.Stop$cluster == k )
 tdmk <- t(dtm.Stop[cl,])
v = sort(rowSums(as.matrix(tdmk)), decreasing=TRUE)
 d = data.frame(word=names(v), freq=v)
 wordcloud(d$word, d$freq, min.freq=5,
 random.color=TRUE,colors=rainbow(7))
 }
```