



## Projet de data mining

# L'analyse de la rubrique politique aux États-Unis

Réalisé par :

Mehdi Jebali  
Yassine Tissaoui  
Jamel Eddine Majdoub

Classe : 3<sup>ème</sup> Année MIndS

Encadré par :

M. Ayadi Walid

Année universitaire 2017/2018

# Table des matières

1	Web Scrapping	3
2	Text Mining	8

# Chapitre 1

## Web Scrapping

Avant de faire le text mining il faut avoir un support textuel sur lequel travailler. La partie suivante focalise sur l'extraction du text à partir du site web de presse électronique « The Guardian ».

Nous allons ici extraire en tant que text les articles dans la rubrique « world » (cela en titre d'exemple) du site web durant l'année 2017 ainsi que les commentaires sur ces articles. Pour cela nous utilisons les packages « rvest » et « XML »

Dans une première étapes nous allons à partir des dates générer un vecteur contenant tous les liens des articles dans cette rubrique pour l'année 2017 jusqu'à la fin de septembre.

```
> titles<-" "  
> for (j in c("jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep")) {  
+ if ( j %in% c("jan", "mar", "may", "jul", "aug", "oct", "dec")) {m<-31}  
+ else if(j %in% c("apr", "jun", "sep", "nov")) {m<-30}  
+ else {m<-28}  
+ for (i in 1:m){  
+ if (i<10) {
```

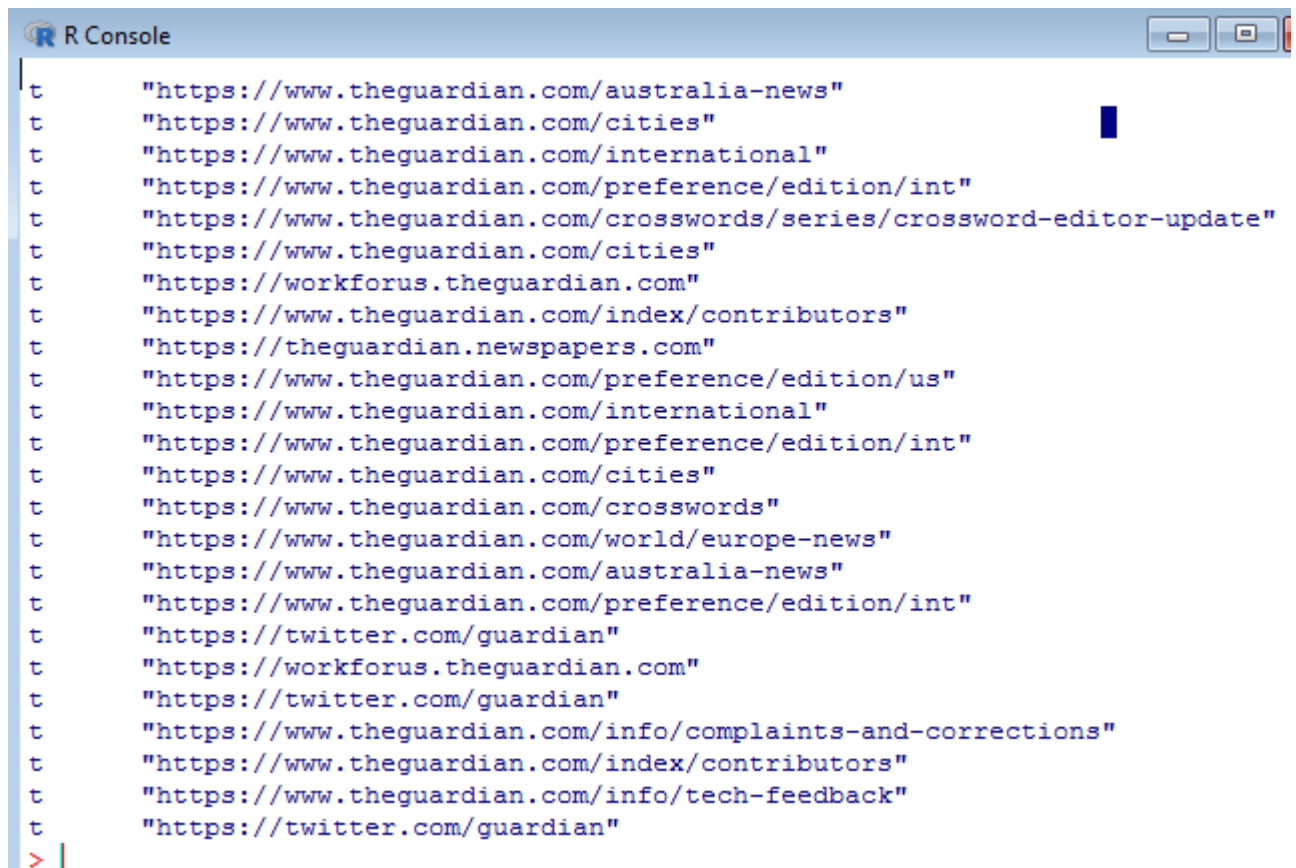
Ici il s'agit d'une boucle pour parcourir les jours de l'année jusqu'à la fin de septembre.

```
if (i<10) {  
link<-paste('https://www.theguardian.com/world/2017/', j, sep="")  
link<-paste(link, '/0', sep="")  
link<-paste(link, toString(i), sep="")  
link<-paste(link, '/all', sep="")  
}  
else {link<-paste('https://www.theguardian.com/world/2017/', j, sep="")  
link<-paste(link, '/', sep="")  
link<-paste(link, toString(i), sep="")  
link<-paste(link, '/all', sep="")  
}
```

Ici il s'agit de générer les liens contenant un index des articles de jour à partir desquels nous allons extraire les liens. La boucle conditionnelle permet de vérifier que le format des liens utilisés par le site web est respecté pour ne pas avoir des erreurs.

```
page<-read_html(link)
t<-html_attr(html_nodes(page, "a"), "href")
titles<-rbind(titles,t)
}
}
```

Finalement nous lisons le code html de la page et grâce au package XML et rvest nous sommes capable d'extraire uniquement le texte situé entre les tags `<a>` et `</a>` et ayant la commande « href » désignant le texte du lien.



```
R Console
t "https://www.theguardian.com/australia-news"
t "https://www.theguardian.com/cities"
t "https://www.theguardian.com/international"
t "https://www.theguardian.com/preference/edition/int"
t "https://www.theguardian.com/crosswords/series/crossword-editor-update"
t "https://www.theguardian.com/cities"
t "https://workforus.theguardian.com"
t "https://www.theguardian.com/index/contributors"
t "https://theguardian.newspapers.com"
t "https://www.theguardian.com/preference/edition/us"
t "https://www.theguardian.com/international"
t "https://www.theguardian.com/preference/edition/int"
t "https://www.theguardian.com/cities"
t "https://www.theguardian.com/crosswords"
t "https://www.theguardian.com/world/europe-news"
t "https://www.theguardian.com/australia-news"
t "https://www.theguardian.com/preference/edition/int"
t "https://twitter.com/guardian"
t "https://workforus.theguardian.com"
t "https://twitter.com/guardian"
t "https://www.theguardian.com/info/complaints-and-corrections"
t "https://www.theguardian.com/index/contributors"
t "https://www.theguardian.com/info/tech-feedback"
t "https://twitter.com/guardian"
> |
```

Ici nous pouvons voir le résultat du travail que nous avons fait. Cependant il reste un nombre énorme de liens complètement inutiles ainsi que des liens redondants. Pour cela nous allons traiter le vecteur des liens afin de se débarrasser de ses liens inutiles.

```
> strtail <- function(s,n=1) {
+ if(n<0)
+ substring(s,1-n)
+ else
+ substring(s,nchar(s)-n+1)
+ }
> i<-1
> while (i<=length(titles)){
+ if (is.na(titles[i])){
+ titles<-titles[-c(i)]
+ i<-i-1
+ }
+ else if (titles[i] %in% c("https://www.facebook.com/the
+ k")) {
+ titles<-titles[-c(i)]
+ i<-i-1
+ }
+ else if (titles[i] %in% c("https://www.theguardian.com/
+ titles<-titles[-c(i)]
+ i<-i-1
+ }
+ else if (strtail(titles[i],7)=="altdate"){

+ i<-i-1
+ }
+ else if (strtail(titles[i],3)=="all"){
+ titles<-titles[-c(i)]
+ i<-i-1
+ }
+
+ i<-i+1
+ }
> i<-1
> j<-2
> while (i <length(titles)){
+ j<-i+1
+ while (j <=length(titles)){
+ if (titles[i]==titles[j]){
+ titles<-titles[-c(j)]
+ j<-j-1
+ }
+ j<-j+1
+ }
+ i<-i+1
+ }
> for (i in 1:20){
+ titles<-titles[-c(i)]}
```

Le code ici permet l'enlèvement de liens interne de the « The Guardian » (liens d'inscription, liens à d'autres rubriques. . .) Il permet aussi d'éliminer les liens redondants. Cela nous permet pour la rubrique « world » de réduire le nombre de liens de plus de 100000 à seulement 7000.

```
[7136] "https://www.theguardian.com/world/2017/jan/19/south-korean-court-refuse$
[7137] "https://www.theguardian.com/world/2017/jan/20/sierra-leone-war-ebola-af$
[7138] "https://www.theguardian.com/world/2017/mar/08/mole-catchers-britain"    $
[7139] "https://www.theguardian.com/world/video/2017/aug/18/spanish-prime-minis$
[7140] "https://www.theguardian.com/world/2017/sep/11/pope-francis-injured-whil$
[7141] "https://www.theguardian.com/world/2017/sep/20/i-pray-shes-already-dead-$
[7142] "https://www.theguardian.com/world/plane-crashes"                        $
[7143] "https://www.theguardian.com/world/2017/jan/19/why-hasnt-mh370-been-foun$
[7144] "https://www.theguardian.com/global/video/2017/jan/20/youre-not-forgotte$
[7145] "https://www.theguardian.com/world/2017/mar/08/refugees-asylum-seekers-h$
[7146] "https://www.theguardian.com/world/2017/sep/11/road-crashes-cost-austral$
[7147] "https://www.theguardian.com/world/2017/sep/20/alleged-isis-sympathiser-$
[7148] "https://www.theguardian.com/world/malaysia-airlines-flight-mh370"        $
[7149] "https://www.theguardian.com/world/2017/jan/19/north-korean-icbm-test-lo$
[7150] "https://www.theguardian.com/world/2017/jan/20/deploraball-trump-lovers-$
[7151] "https://www.theguardian.com/world/2017/mar/08/british-man-on-mission-fo$
[7152] "https://www.theguardian.com/world/2017/sep/11/antisemitic-robbers-targe$
[7153] "https://www.theguardian.com/world/gallery/2017/sep/20/rescuers-fight-to$
[7154] "https://www.theguardian.com/world/2017/jan/18/british-tourists-crowd-ba$
[7155] "https://www.theguardian.com/world/2017/jan/20/tech-billionaire-opens-ne$
[7156] "https://www.theguardian.com/world/2017/mar/08/borut-pahor-slovenia-inst$
[7157] "https://www.theguardian.com/world/2017/sep/20/new-zealand-election-jet-$
[7158] "https://www.theguardian.com/world/gambia"                              $
[7159] "https://www.theguardian.com/preference/edition/int"                     $
```

Pour l'étape finale du « scrapping » nous allons lire tout liens des vecteurs afin que l'on peut générer le texte sur lequel nous allons faire le text mining. Nous parcourons le vecteur avec une boucle for, nous lisons le html de chaque page et nous prenons uniquement le texte se situant entre les tags `<p>`/`</p>`. Nous finissons cette étape par le fait de s'assurer de l'encode UTF-8 de texte et son enregistrement.

```
> Corp<-" "
> for(i in 1:length(titles)){
+ link<-titles[i]
+ page<-read_html(link)
+ pagetext<- page %>% html_nodes("p") %>% html_text()
+ Corp<-rbind(Corp,pagetext)
+ }

> Encoding(Corp)<-"UTF-8"
> write(Corp,file="text3.txt")
~ |
```

```
pagetext "Shai Masot is recorded discussing how to discredit MPs in comments de$
pagetext "30ft by 10ft hole developed after water main broke, leaving 20 homes $
pagetext "Video is first public sign since August kidnapping that US and Austra$
pagetext "Reuters in Mexico City" $
pagetext "Red Cross report highlights how unsolicited items often end up in lan$
pagetext "Transport minister says any future underwater search effort would be $
pagetext "Each day glaciologist Jason Roberts flies planes over Antarctica to m$
pagetext "Thousands took to the streets of Washington DC for peaceful protests,$
pagetext "\nEmma Graham-Harrison" $
pagetext "Leftwing outsider to face pro-business rightwinger Manuel Valls in fi$
pagetext "Stephen Schwarzman, who is expected to head the US president's busine$
pagetext "Some birds have died and remaining animals at farm in Preston would b$
pagetext "The world's largest firefighting aircraft has flown in from the US, a$
pagetext "One body found in smouldering ruins of Santa Olga, the worst-hit of s$
pagetext "Mexican billionaire praises US president as 'great negotiator' but ca$
pagetext "Arms sale in the spotlight as government faces judicial review in the$
pagetext "Hamon's trouncing of centrist Manuel Valls is damning verdict on fail$
pagetext "Australian woman tells court she didn't ask questions when her Britis$
pagetext "Police accused of planting evidence, taking cash from funeral homes a$
pagetext "Frustration over move adds to pressure over controversies surrounding$
pagetext "British PM will also urge leaders at Malta summit to spend more on de$
pagetext "Emergency government decree to decriminalise official misconduct 'not$
pagetext "French presidential candidate's chances decline after allegations fam$
pagetext "Emily Jayne Collie, a 20-year-old from Victoria, given first aid on K$
pagetext "Anton Bakov and his wife Maria want to lease three islands in Kiribat$
```

Nous pouvons voir ci-dessus une partie du résultat qui sera enregistré et puis lu pour le travail de text mining.



# Chapitre 2

## Text Mining

On passe maintenant à analyser les données textuelles extraites auparavant dont on utilise plusieurs fonctions issues de différentes librairies telles que tm, ggraph, igraph, tidytext, tidyr, wordcloud...

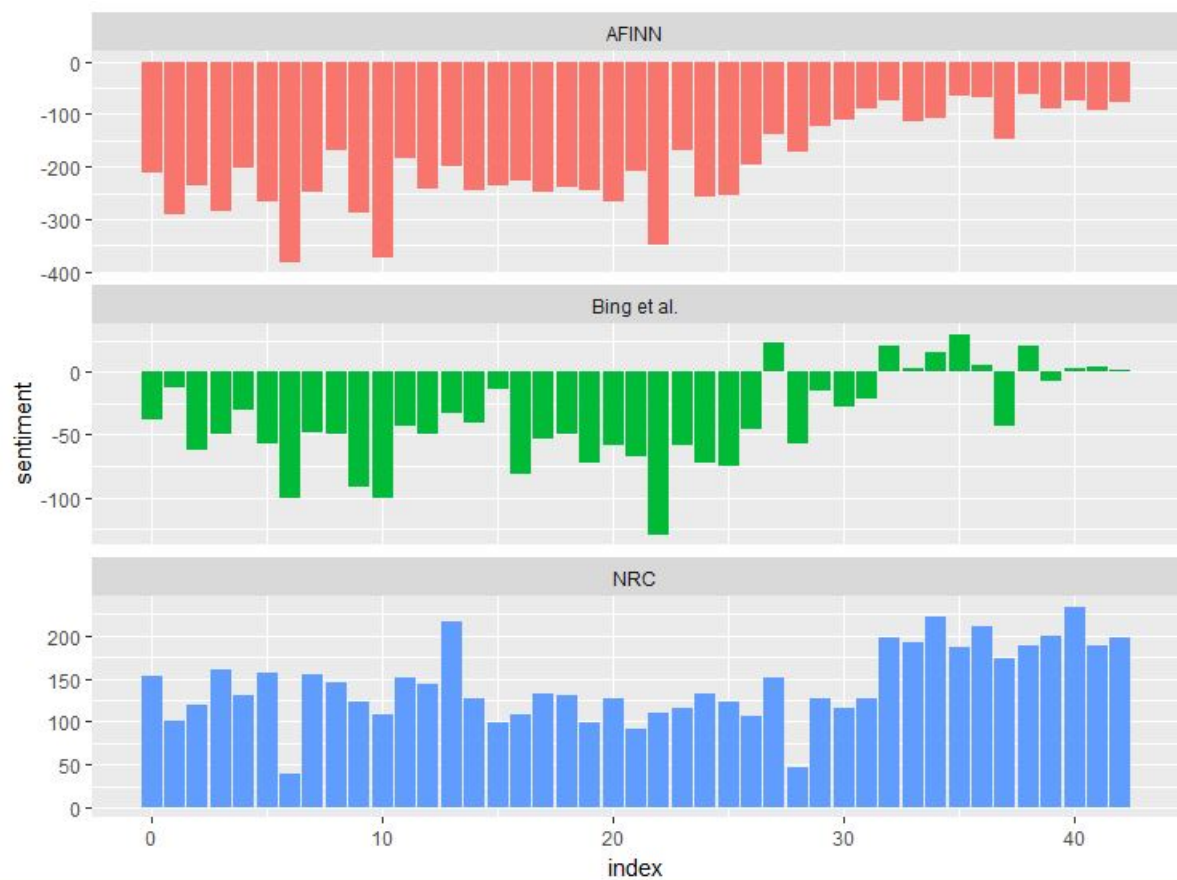
```
> setwd("C:/Users/Mehdi Jebali/Documents/Data mining/Projet data_mining")
> text<-readLines("text3.txt")
> docs<-Corpus(VectorSource(text))
> motssupprimes =c("monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday",
+ "november", "december", "gmt", "bst", "modified", "the")
> docs<-tm_map(docs, removePunctuation)
> docs<-tm_map(docs, removeNumbers)
> docs<-tm_map(docs, removeWords, motssupprimes)
```

Ensuite, nous définissons un format de texte propre comme étant une table des paragraphes structurés par ligne. Cela aide à manipuler aisément le contenu en appliquant des outils cohérents. On utilise la fonction `unnest_tokens` qui permet de découper le text en « tokens » (unité de mesure du texte qui est généralement un mot).

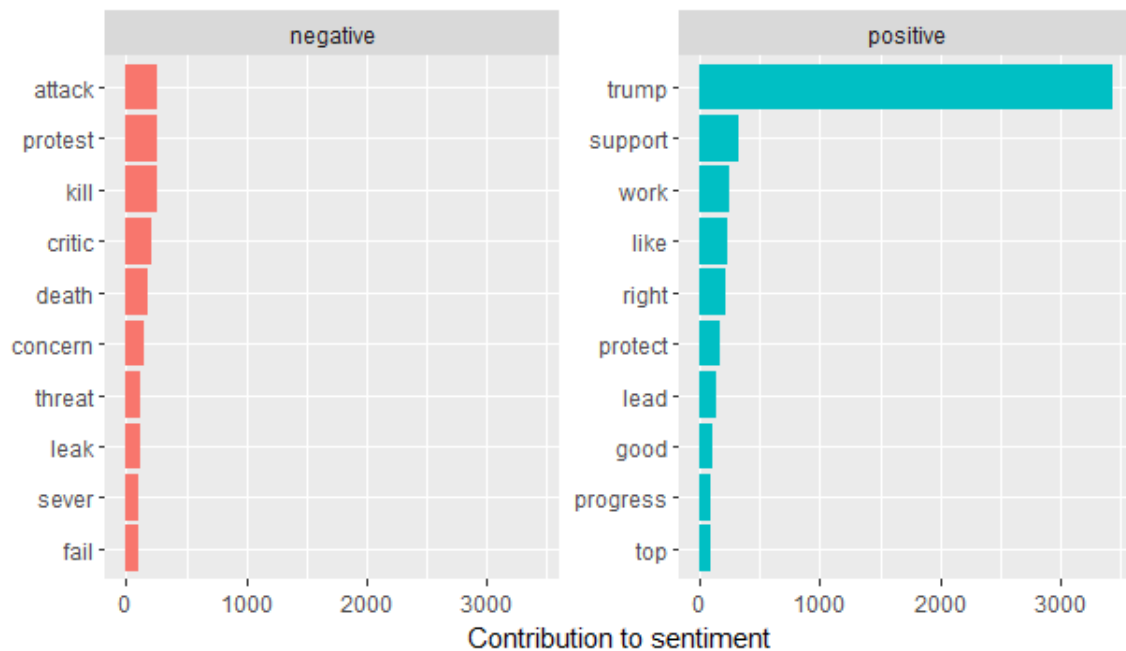
```
> dtm <- TermDocumentMatrix(docs)
> writeLines(as.character(docs), con="mycorpus.txt")
> text<-readLines("mycorpus.txt")
> trump<-data_frame(line=1:length(text), text=text)
> trump1<- trump %>% unnest_tokens(word, text)
```

Avec plusieurs options pour les lexiques de sentiment, vous pouvez vouloir plus d'informations sur lequel est approprié pour vos buts. Utilisons les trois lexiques du sentiment et examinons comment le sentiment change à travers l'arc narratif de *Pride and Prejudice*. D'abord, utilisons `filter()` pour choisir seulement les mots du roman qui nous intéressent.



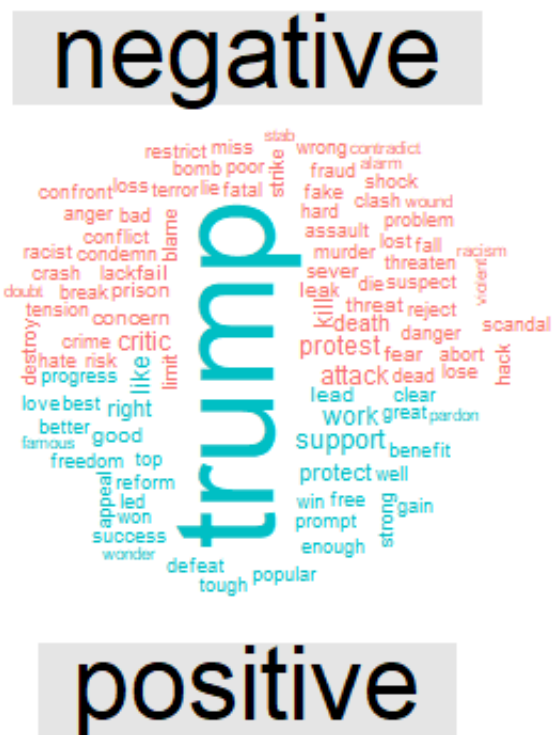


On remarque que les lexiques ont les mêmes classifications (dictionnaire) mais le lexique nrc est totalement différent. Pour éviter l'ambiguïté, on choisit le lexique « afinn ». L'un des avantages de déterminer les sentiments est d'avoir un cadre de données à la fois avec le sentiment et le mot qui le convient. En mettant `count()` ici avec des arguments à la fois le mot et le sentiment, nous découvrons à quel point chaque mot a contribué à chaque sentiment.



La figure ci-dessus montre à la fois la redondance des mots et leurs sentiments (positif, négatif) .

On peut afficher ce résultat autrement par la méthode d'affichage du nuage des points .



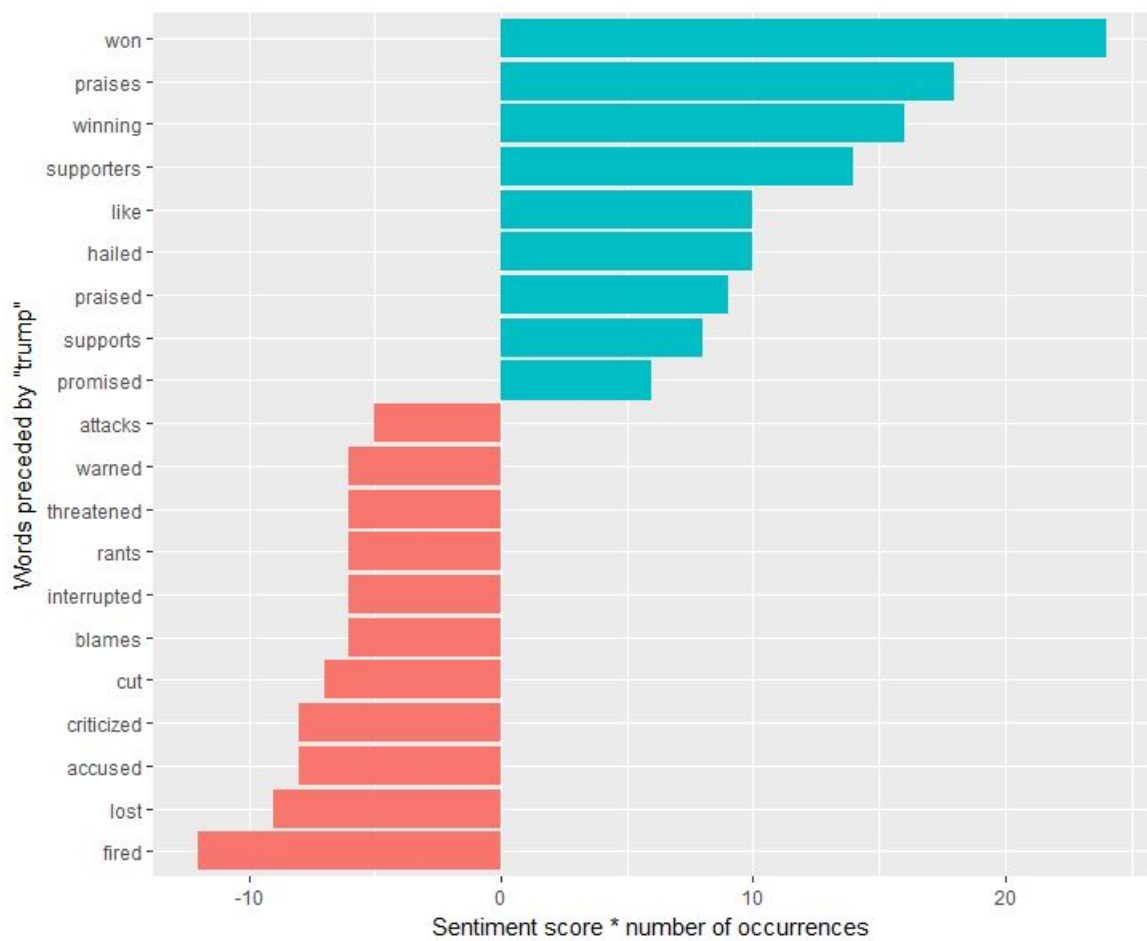
La taille d'un mot dans la figure est proportionnelle à sa fréquence dans son sentiment. Nous pouvons utiliser cette visualisation pour voir les mots positifs et négatifs les plus importants, mais les tailles des mots ne sont pas comparables entre les sentiments. Ici, le mot « trump » apparaît comme étant une valeur positive et plus fréquent car trump en anglais signifie gagner.

Finalement, nous allons explorer quelques méthodes proposées par tidytext calculer et visualiser les relations entre les mots de texte. Cela inclut l'argument `token = "ngrams"`, qui symbolise par paires de mots adjacents plutôt que par des mots individuels. Nous introduirons également deux nouveaux packages : `ggraph`, qui étend `ggplot2` pour construire des tracés de réseau, et `widyr`, qui calcule les corrélations et les distances par paires dans une trame de données propre. De plus, on va examiner quels mots ont tendance à précéder les autres immédiatement.

nous avons utilisé ces instructions pour visualiser les corrélations entre les mots des articles et les clusters formés par ces derniers.

```
> word_cors %>%  
+   filter(correlation > .15) %>%  
+   graph_from_data_frame() %>%  
+   ggraph(layout = "fr") +  
+   geom_edge_link(aes(edge_alpha = correlation), show.legend = F)  
+   geom_node_point(color = "lightblue", size = 5) +  
+   geom_node_text(aes(label = name), repel = TRUE) +  
+   theme_void()
```

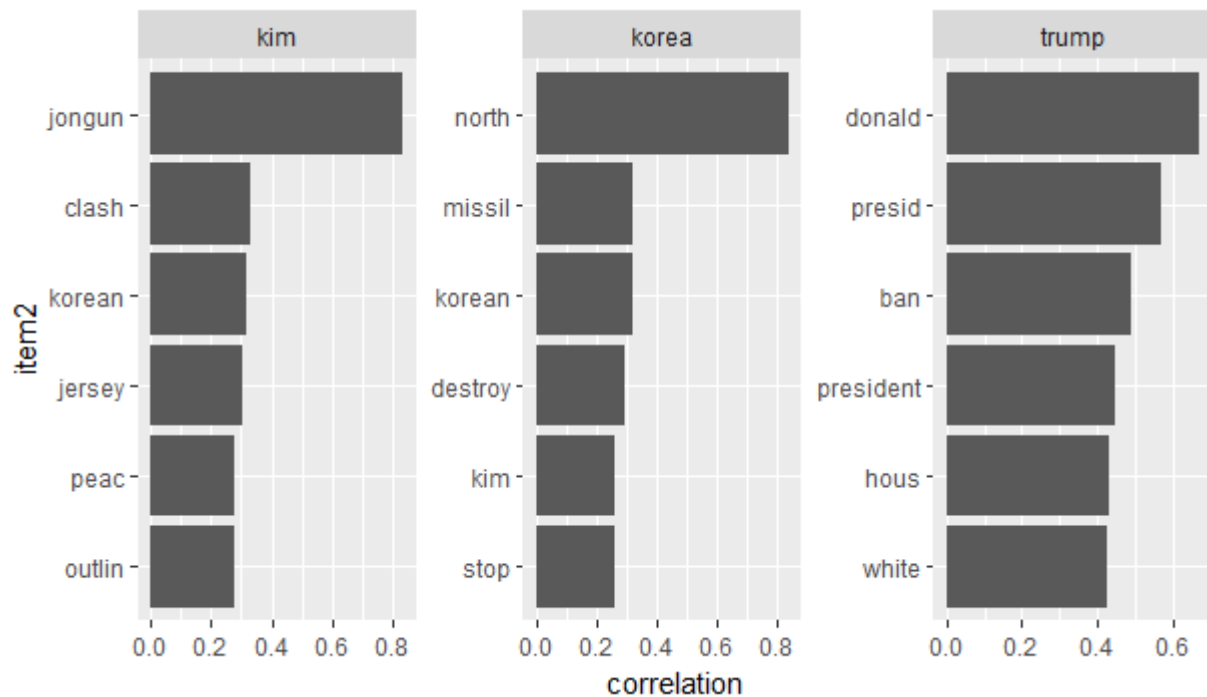




On peut voir ici la classification des mots qui précédant le mot "Trump" selon les scores de sentiment multiplié par le nombre d'occurrences, on remarque que le résultat est très logique ici parce que tous les mots positifs représentent le fait que Trump a gagné les élections et on trouve dans la partie négative aussi des mots qui caractérisent Trump comme "fired", "attacks", "blames", "warned" ...\*

## l'escalade entre Donald Trump et Kim Jong-un

On va s'intéresser maintenant à l'escalade entre trump et Kim par l'analyse des correlations des mots avec les trois mots initiaux "Kim", "Trump" et "Korea". On remarque



d'après cette figure que les mots "clash", "missil", "destroy" ont une forte corrélation avec les mots "kim" et "korea" ce qui signifie que Kim est celui qui cherche plus les problèmes avec Trump et il menace toujours de bombarder l'Amérique.