

Étape 1 : Collecte et Préparation des Données

1.1. Collecte des données textuelles : Extraire des données textuelles à partir d'Internet, de fichiers texte ou d'autres sources.

- `install.packages("readxl")`
- `library(readxl)`
- `data <- readxl::read_excel("C:/Users/KAYA/Desktop/y.xlsx")`

1.1.1. Examen des données

- `data`



	Ville	Population	Popdensite	Popactive	Partipolitique	Revenumoyen	SmartCity	CO2tonnes	pol
1	Paris	2145906	20437	54.9	Parti Socialiste	28570	Non	5566781	Dès 2004, la Ville de Paris s'est engagée dans la lutte contre ...
2	Lyon	522228	10880	51.4	Europe Écologie Les Verts	24150	Non	1414698	la Métropole de Lyon agit dans le cadre de son Plan climat a...
3	Marseille	870321	3611	41.9	Parti Socialiste	19370	Non	1885563	Économiser l'énergie, les carburants L'ensemble des travaux ...
4	Lille	236234	6750	49.1	Parti Socialiste	19580	Non	856091	Le diagnostic territorial de la MEL (Métropole Européenne d...
5	Bordeaux	259809	5302	50.9	Europe Écologie Les Verts	23360	Oui	643606	Quelle Métropole souhaitons-nous pour 2030 ? Bordeaux M...
6	Toulouse	498003	4220	50.9	Les Républicains	21440	Oui	1743562	Principales cibles ? Le transport routier et le chauffage (au g...
7	Nice	343477	4771	42.9	Horizons	20530	Oui	1084312	Concernant la trajectoire carbone, un des objectifs majeurs ...
8	Strasbourg	290576	3725	46.2	Europe Écologie Les Verts	19220	Oui	1032624	S'appuyant sur un cadre stratégique approuvé en février 20...

- `str(data)`

```
> str(data)
tibble [8 × 9] (S3: tbl_df/tbl/data.frame)
 $ ville      : chr [1:8] "Paris" "Lyon" "Marseille" "Lille" ...
 $ Population : num [1:8] 2145906 522228 870321 236234 259809 ...
 $ Popdensite : num [1:8] 20437 10880 3611 6750 5302 ...
 $ Popactive  : chr [1:8] "54.9" "51.4" "41.9" "49.1" ...
 $ Partipolitique: chr [1:8] "Parti Socialiste" "Europe Écologie Les Verts"
"Parti Socialiste" "Parti Socialiste" ...
 $ Revenumoyen : num [1:8] 28570 24150 19370 19580 23360 ...
 $ SmartCity   : chr [1:8] "Non" "Non" "Non" "Non" ...
 $ CO2tonnes   : num [1:8] 5566781 1414698 1885563 856091 643606 ...
 $ pol         : chr [1:8] "Dès 2004, la Ville de Paris s'est engagée dan
```

- Data comporte 8 lignes et 9 colonnes. Il y a 5 colonnes de type 'chr' (chaîne de caractères) et 4 colonnes de type 'num' (numérique).

- Conversion en facteur des lignes contenant du texte.
 - `data$Ville <- as.factor(data$Ville)`
 - `data$Partipolitique <- as.factor(data$Partipolitique)`
 - `data$SmartCity <- as.factor(data$SmartCity)`
- Les valeurs numériques semblant être des caractères (chr) sont converties en valeurs numériques
- `data$Popactive <- as.numeric(data$Popactive)`
- `pol_env <- data$pol`

```
> str(data)
tibble [8 × 9] (S3: tbl_df/tbl/data.frame)
 $ Ville      : Factor w/ 8 levels "Bordeaux","Lille",...: 6 3 4 2 1 8 5 7
 $ Population : num [1:8] 2145906 522228 870321 236234 259809 ...
 $ Popdensite : num [1:8] 20437 10880 3611 6750 5302 ...
 $ Popactive  : num [1:8] 54.9 51.4 41.9 49.1 50.9 50.9 42.9 46.2
 $ Partipolitique: Factor w/ 4 levels "Europe Écologie Les Verts",...: 4 1 4 4 1 3 2 1
 $ Revenumoyen : num [1:8] 28570 24150 19370 19580 23360 ...
 $ SmartCity   : Factor w/ 2 levels "Non","Oui": 1 1 1 1 2 2 2 2
 $ CO2tonnes   : num [1:8] 5566781 1414698 1885563 856091 643606 ...
 $ pol         : chr [1:8] "Dès 2004, la ville de Paris s'est engagée dans la lutte contre
```

```
for (i in 1:length(pol_env)) {

  nombre_lettres <- nchar(pol_env[i])

  cat(paste("Ligne", i, "nombre de lettres :", nombre_lettres), "\n")

}
```

Avant de nettoyer les données, calculer le nombre de lettres pour chaque ligne dans la colonne 'pol_env'

```
Ligne 1 nombre de lettres : 1966
Ligne 2 nombre de lettres : 1951
Ligne 3 nombre de lettres : 2527
Ligne 4 nombre de lettres : 1840
Ligne 5 nombre de lettres : 2259
Ligne 6 nombre de lettres : 2270
Ligne 7 nombre de lettres : 2496
Ligne 8 nombre de lettres : 1722
```

```
for (i in 1:length(pol_env)) {

  nombre_mots <- length(strsplit(pol_env[i], "\\s")[[1]])

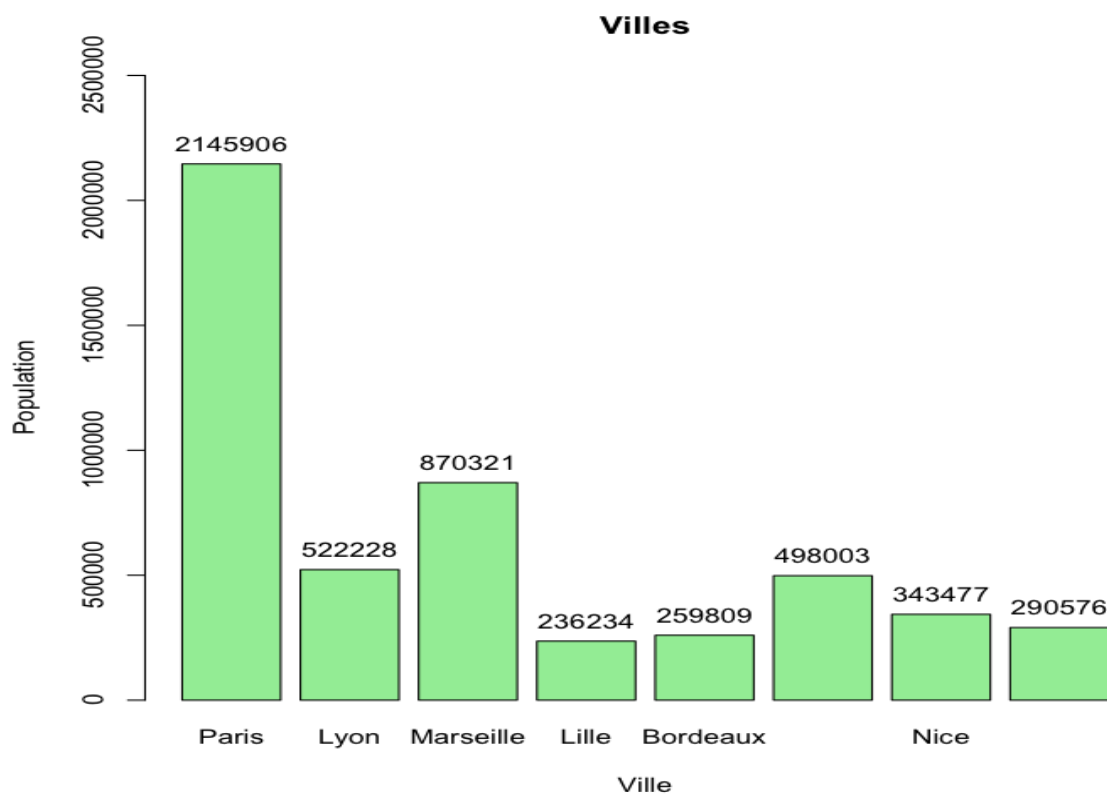
  cat(paste("Ligne", i, "nombre de mots:", nombre_mots), "\n")

}
```

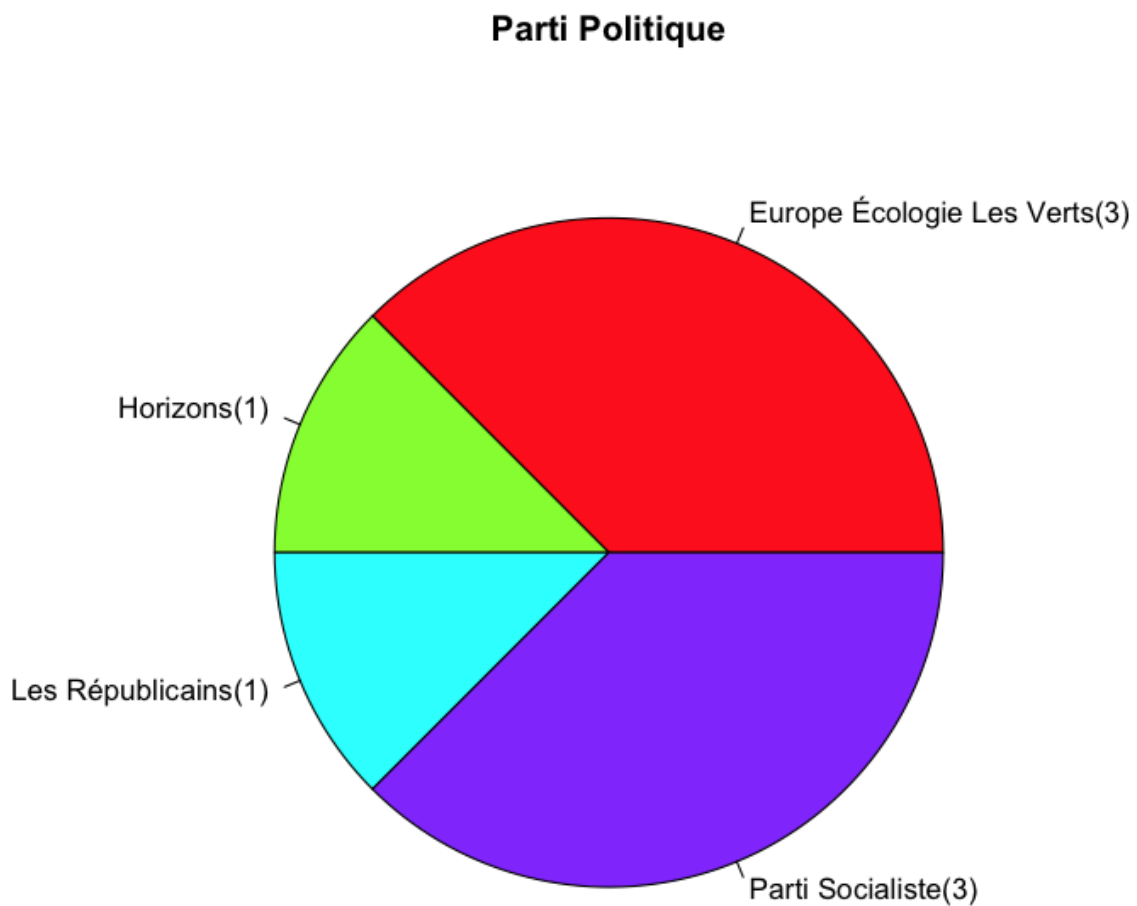
Avant de nettoyer les données, calculer le nombre de mots pour chaque ligne dans la colonne 'pol_env'.

```
Ligne 1 nombre de mots: 336
Ligne 2 nombre de mots: 315
Ligne 3 nombre de mots: 423
Ligne 4 nombre de mots: 281
Ligne 5 nombre de mots: 354
Ligne 6 nombre de mots: 361
Ligne 7 nombre de mots: 377
Ligne 8 nombre de mots: 253
```

```
bp <- barplot(data$Population, names.arg = data$Ville, col = "lightgreen",
xlab = "Ville", ylab = "Population",
main = "Villes",
ylim = c(0, 2500000))
text(x = bp, y = data$Population, labels = data$Population, pos = 3, col = "black")
```



```
pie(party_counts, labels = paste(names(party_counts), "(", party_counts, ")", sep = "" ),  
col = rainbow(length(party_counts)),  
main = "Parti Dağılımı")
```



2.1 Nettoyage des données : Élimination des caractères inutiles, des chiffres et des espaces, normalisation de la casse, prise en compte des caractères spéciaux.

- `install.packages("tm")`
- `library(tm)`
- Téléchargement de la bibliothèque 'text mining (tm)' pour le nettoyage des données.

- `vecteur_texte <- data$pol`

"Vecteur_texte," représente un vecteur de texte contenant la colonne "pol" du data frame "data".

- `source_texte <- VectorSource(vecteur_texte)`

- `texte_corpus <- Corpus(source_texte)`

➤ "Texte_corpus" représente un corpus de texte créé à partir de la source de texte "source_texte".

- `texte_corpus <- tm_map(texte_corpus, content_transformer(tolower))`
- `texte_corpus <- tm_map(texte_corpus, removePunctuation)`
- `texte_corpus <- tm_map(texte_corpus, removeNumbers)`
- `texte_corpus <- tm_map(texte_corpus, content_transformer(function(x) gsub("•", "", x)))`
- `inspect(texte_corpus)`

```
> inspect(texte_corpus)
<<SimpleCorpus>>
Metadata: corpus specific: 1, document level (indexed): 0
Content: documents: 8

[1] dès la ville de paris s'est engagée dans la lutte contre le changement climatique en réalisant un premier bilan des consommations d'énergie et des émissions de gaz à effet de serre du territoire elle a ensuite adopté en à l'unanimité un plan climat ambitieux avec des objectifs forts\r\n réduire de les émissions de gaz à effet de serre et les consommations d'énergie\r\n porter à la part des énergies renouvelables et de récupération entre et \r\nl'ambition de paris pour le climat a pris une nouvelle dimension lors de la e conférence internationale pour le climat cop accueillie par la france en aboutissant à l'adoption de l'accord de paris par la communauté internationale\r\nentre le décembre et le février le projet de nouveau plan climat a fait l'objet d'une consultation publique permettant de recueillir les avis des citoyens ainsi que ceux de la métropole de la région et de l'état plus de participations ont été recueillies dont la grande majorité souligne que les actions du nouveau plan répondent de manière très satisfaisante à l'urgence climatique les participants ont identifié trois thématiques prioritaires alimentation durable sobriété énergétique
```

➤ Tous les caractères du corpus sont mis en minuscules, les symboles sont supprimés et les numéros sont supprimés

- `texte_corpus <- tm_map(texte_corpus, content_transformer(function(x) gsub(" ", "", x)))`
- `texte_corpus <- tm_map(texte_corpus, content_transformer(function(x) gsub("\r\n", " ", x)))`
- `texte_corpus <- tm_map(texte_corpus, content_transformer(function(x) gsub("« | » |' ", "", x)))`


```
> inspect (texte_corpus)
<<SimpleCorpus>>
Metadata: corpus specific: 1, document level (indexed): 0
Content: documents: 8

[1] dès la ville de paris sest engagée dans la lutte contre le changement climatique en réalisant un premier bilan des consommations d'énergie et des émissions de gaz à effet de serre du territoire elle a ensuite adopté en à l'unanimité un plan climat ambitieux avec des objectifs forts réduire de les émissions de gaz à effet de serre et les consommations d'énergie porter à la part des énergies renouvelables et de récupération entre et l'ambition de paris pour le climat a pris une nouvelle dimension lors de la conférence internationale pour le climat cop accueillie par la france en aboutissant à l'adoption de l'accord de paris par la communauté internationale entre le décembre et le février le projet de nouveau plan climat a fait l'objet d'une consultation publique permettant de recueillir les avis des citoyens ainsi que ceux de la métropole de la région et de l'état plus de participations ont été recueillies dont la grande majorité souligne que les actions du nouveau plan répondent de manière très satisfaisante à l'urgence climatique les participants ont identifié trois thématiques prioritaires alimentation durable sobriété énergétique et adaptation du territoire les actions mises en avant ont été recycler et valoriser des déchets disposer de transports publics propres en atteindre l'alimentation durable rénover les équipements publics faire de paris une capitale cyclable atteindre de surfaces perméables et végétales parmi les propositions reçues certaines ont été ajoutées au plan climat un contrôle renforcé de la limitation de l'éclairage nocturne des bâtiments tertiaires l'étude de la mise en place d'une plateforme de dons alimentaires des particuliers aux personnes les plus fragiles l'étendue des possibilités de correspondance du ticket t entre tram bus métro et rer
[2] la métropole de lyon agit dans le cadre de son plan climat air énergie territorial pcaet pour lutter à l'échelle individuelle et collective contre le dérèglement climatique une baisse de émissions de gaz à effet de serre par rapport à l'an une baisse de des consommations d'énergie par rapport à le doublement de la
```

Suppression des caractères spéciaux non supprimés par des commandes à l'aide d'une fonction

2.1.1 Stop-Words

Remarque : 'Stop-words' avant 'wordcloud'

```
install.packages("wordcloud")
```

```
library(wordcloud)
```

- `wordcloud(texte_corpus [[1]], min.freq = 2, main = "Paris word cloud")`



- wordcloud(texte_corpus [[2]], min.freq = 2, main = "Lyon word cloud")



- wordcloud(texte_corpus [[3]], min.freq = 2, main = "Marseille word cloud")



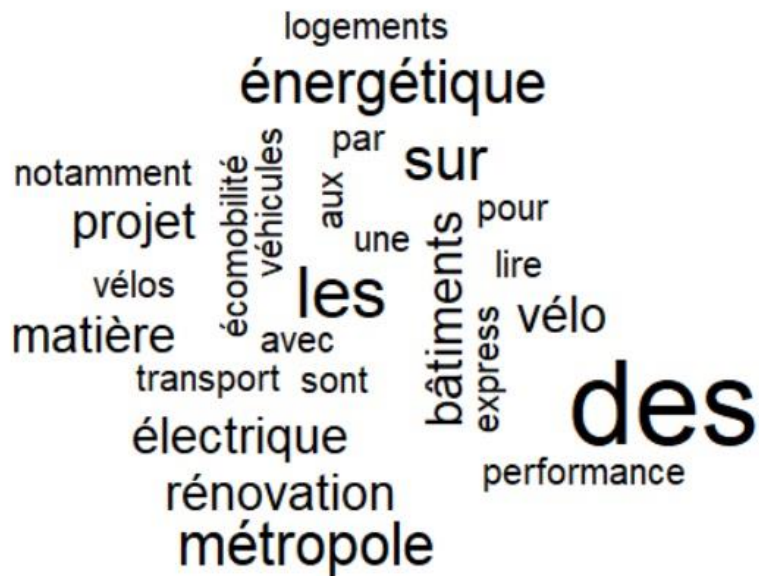
- wordcloud(texte_corpus[[4]], min.freq = 2, main = "Lille word cloud")



- wordcloud(texte_corpus[[5]], min.freq = 2, main = "Bordeaux word cloud")



- wordcloud(texte_corpus [[6]], min.freq = 2, main = "Toulouse word cloud")



- wordcloud(texte_corpus [[7]], min.freq = 2, main = "Nice word cloud")



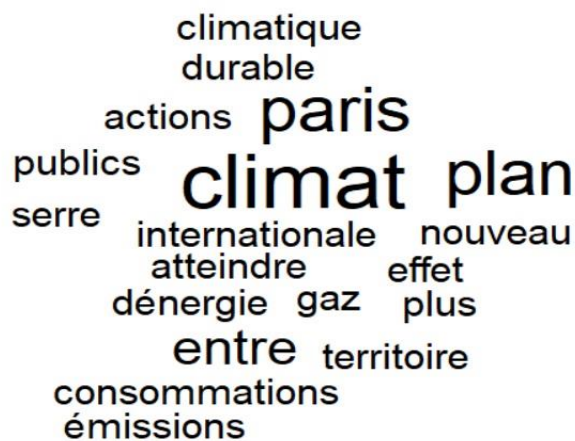
- `wordcloud(texte_corpus [[8]], min.freq = 2, main = "Strasbourg word cloud")`



2.1.2 Stop-Words

- `texte_corpus <- tm_map(texte_corpus, removeWords, stopwords("fr"))`
- `texte_corpus <- tm_map(texte_corpus, content_transformer(function(x) gsub("\\bdune\\b", "", x)))`
- `texte_corpus <- tm_map(texte_corpus, content_transformer(function(x) gsub("\\bcest\\b", "", x)))`
- `texte_corpus <- tm_map(texte_corpus, content_transformer(function(x) gsub("\\bdici\\b", "", x)))`

Paris:



Lyon:



A word cloud for Lyon. The words are arranged in a roughly circular shape. The most prominent words are 'consommations' at the bottom, 'dénergie' in the middle-left, 'permet' in the middle-right, and 'métropole' at the top. Other words include 'climat', 'rapport', 'baisse', 'vont', 'transports', 'lyon', 'bâtiments', 'vélo', 'grâce', 'part', 'plan', 'commun', 'déplacements', 'baisser', 'faire', 'dont', 'plus', 'logements', 'territoire', 'énergétique', 'aussi', and 'rapport'.

consommations

Marseille:



A word cloud for Marseille. The words are arranged in a roughly circular shape. The most prominent words are 'l'énergie' at the top, 'ville' in the middle-left, 'économie' in the middle-right, and 'inscriptions' at the bottom. Other words include 'mondial', 'faire', 'plan', 'solaire', 'publics', 'ligne', 'marseille', 'sensibiliser', 'vélos', 'forum', 'arceaux', 'titre', 'eau', 'services', 'public', 'ressource', 'reçue', 'équipement', and 'consommation'.

l'énergie

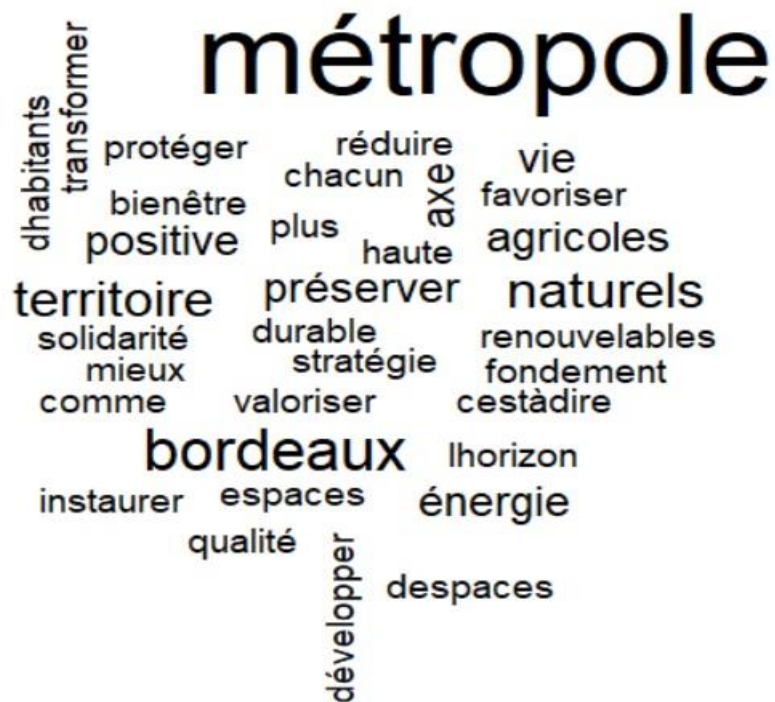
Lille:



A word cloud for Lille, with the most prominent words being 'logements', 'climatique', 'plan', 'nouveau', 'renovation', 'transition', and 'énergétique'. Other visible words include 'dénergie', 'développement', 'notamment', 'lancement', 'dactions', 'sociaux', 'métropole', 'territoire', 'bilan', 'œuvre', 'mise', 'réseaux', 'chaleur', and 'transition'.

logements
dénergie
développement
notamment
plan
lancement
climatique
dactions
sociaux
nouveau
renovation
métropole
territoire
bilan
œuvre
mise
réseaux
chaleur
transition
énergétique

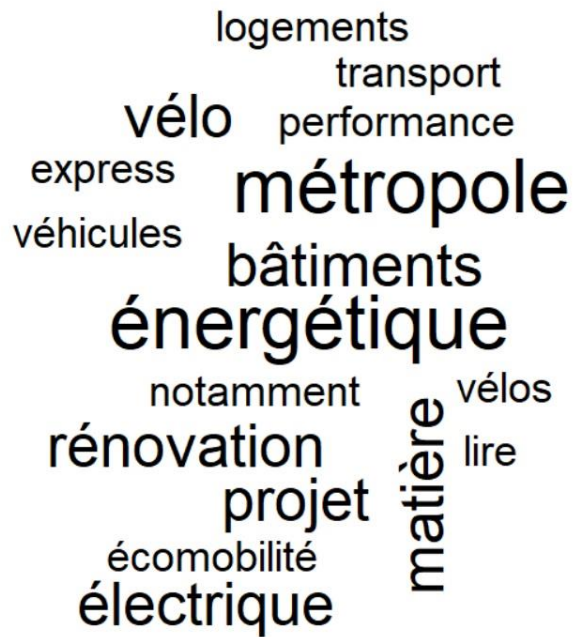
Bordeaux



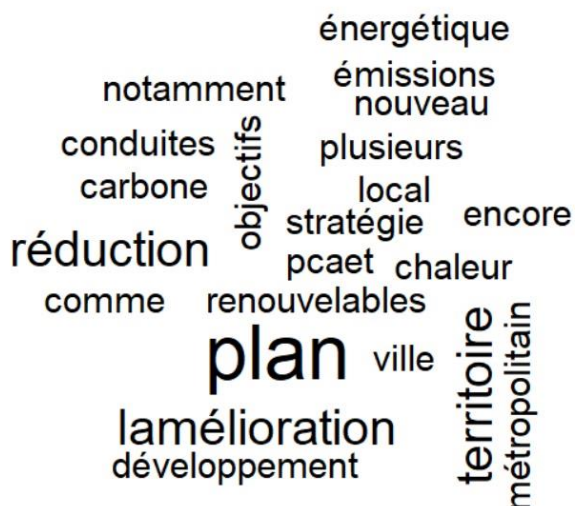
A word cloud for Bordeaux, with the most prominent words being 'métropole', 'bordeaux', 'territoire', 'préservé', 'naturels', 'énergie', and 'développer'. Other visible words include 'habitants', 'transformer', 'protéger', 'bien-être', 'positive', 'plus', 'haute', 'réduire', 'chacun', 'axe', 'vie', 'favoriser', 'agricoles', 'solidaire', 'durable', 'stratégie', 'renouvelables', 'fondement', 'comme', 'valoriser', 'cestàdire', 'horizon', 'instaurer', 'espaces', 'qualité', and 'despaces'.

habitants
transformer
protéger
bien-être
positive
plus
haute
réduire
chacun
axe
vie
favoriser
agricoles
solidaire
durable
stratégie
renouvelables
fondement
comme
valoriser
cestàdire
horizon
instaurer
espaces
qualité
développer
despaces
métropole
bordeaux
territoire
préservé
naturels
énergie

Toulouse:



Nice:



Strasbourg:

strasbourg
énergétique
objectifs
plusieurs
ville
bâti
économie
développement
écologique
transformation

Nombre de lettres et de mots après les stop-words

- `chiffres_lettres <- sapply(texte_corpus, function(x) nchar(as.character(x)))`
 - `chiffres_mots<- sapply(texte_corpus, function(x) length(unlist(strsplit(as.character(x), "\\s"))))`
 - `resultats <- data.frame(nombre_lettre = chiffres_lettres, nombre_mot = chiffres_mots)`
 - `print(resultats)`
- ```
> chiffres_lettres <- sapply(texte_corpus, function(x) nchar(as.character(x)))
> chiffres_mots<- sapply(texte_corpus, function(x) length(unlist(strsplit(as.character(x), "\\s"))))
> resultats <- data.frame(nombre_lettre = chiffres_lettres, nombre_mot = chiffres_mots)
> print(resultats)
 nombre_lettre nombre_mot
1 1558 320
2 1531 303
3 2014 399
4 1519 267
5 1843 336
6 1837 354
7 2052 368
8 1440 250
```

Après le traitement des Stop Words, nous remarquons que les mots courants et généralement dépourvus de signification ont été supprimés des textes. Par conséquent, le nombre de lettres et de mots dans les textes a diminué après le traitement des Stop Words.

### Stemming

- `texte_corpus <- tm_map(texte_corpus, stemDocument, language = "french")`
- `chiffres_lettres <- sapply(texte_corpus, function(x) nchar(as.character(x)))`
- `chiffres_mots <- sapply(texte_corpus, function(x) length(unlist(strsplit(as.character(x), "\\s"))))`
- `resultats2 <- data.frame(nombre_lettre = chiffres_lettres, nombre_mot = chiffres_mots)`
- `print(resultats2)`

```
> chiffres_lettres <- sapply(texte_corpus, function(x) nchar(as.character(x)))
> chiffres_mots <- sapply(texte_corpus, function(x) length(unlist(strsplit(as.character(x), "\\s"))))
> resultats2 <- data.frame(nombre_lettre = chiffres_lettres, nombre_mot = chiffres_mots)
> print(resultats2)
 nombre_lettre nombre_mot
1 1135 167
2 1088 156
3 1460 219
4 1091 155
5 1372 190
6 1338 193
7 1482 204
8 1056 146
```

Après la Stemming, nous regroupons les informations sur les formes racinées des textes ainsi que le nombre de lettres et de mots dans un cadre de données.

### Completion

- `sauvegarde= texte_corpus`
- `completion=function(texte_corpus,dico){PlainTextDocument(stripWhitespace(paste(stemCompletion(unlist(strsplit(as.character(texte_corpus), " ")),dictionary=dico),sep=" ",collapse=" ")))}`
- `texte_corpus =lapply(texte_corpus,completion,dico=sauvegarde)`
- `texte_corpus =sapply(texte_corpus,"[[",1)`
- `texte_corpus =Corpus(VectorSource(texte_corpus))`
- `inspect(texte_corpus)`

```
> sauvegarde= texte_corpus
> completion=function(texte_corpus,dico)
{PlainTextDocument(stripWhitespace(paste(stemCompletion(unlist(strsplit(as.character(texte_corpus), "
")),dictionary=dico),sep=" ",collapse=" ")))}
> texte_corpus =lapply(texte_corpus,completion,dico=sauvegarde)
> texte_corpus =sapply(texte_corpus,"[[",1)
> texte_corpus =Corpus(VectorSource(texte_corpus))
> inspect(texte_corpus)
<<SimpleCorpus>>
Metadata: corpus specific: 1, document level (indexed): 0
Content: documents: 8
```

```
[1] despac vill paris sest engag lutt contr chang climat réalis premi bilan consomm dénerg émiss gaz effet serr
territoire a ensuit adopt lunanim plan climat ambiti object fort réduire émiss gaz effet serr consomm dénerg port
part énerget renouvel récupérer entre lambit paris climat a pris nouvel dimens lor effet confèrent international
climat cop accueil franc about ladopt laccord paris communaut international entre décembr févri projet nouveau
```

### \* Calcul du nombre de lettres et de mots après « Completion »

- `chiffres_lettres <- sapply(texte_corpus, function(x) nchar(as.character(x)))`

- `chiffres_mots <- sapply(texte_corpus, function(x) length(unlist(strsplit(as.character(x), "\\s"))))`
- `resultats3 <- data.frame(nombre_lettre = chiffres_lettres, nombre_mot = chiffres_mots)`
- `print(resultats3)`

```

> chiffres_lettres <- sapply(texte_corpus, function(x) nchar(as.character(x)))
> chiffres_mots <- sapply(texte_corpus, function(x) length(unlist(strsplit(as.character(x), "\\s"))))
> resultats3 <- data.frame(nombre_lettre = chiffres_lettres, nombre_mot = chiffres_mots)
> print(resultats3)
 nombre_lettre nombre_mot
1 1150 167
2 1098 156
3 1476 219
4 1100 155
5 1400 190
6 1356 193
7 1490 204
8 1060 146

```

Dans le but de trouver les formes racines (ou "stems") des mots dans les données textuelles et de compléter les mots manquants, cette opération est réalisée.

## Analise des Donnés

```
install.packages("FactoMineR")
```

```
library(FactoMineR)
```

```
analyse1=DocumentTermMatrix(texte_corpus)
```

```
analyse1
```

```
<DocumentTermMatrix (documents: 8, terms: 677)>>
```

```
Non-/sparse entries: 1074/4342
```

```
Sparsity : 80%
```

```
Maximal term length: 15
```

```
Weighting : term frequency (tf)
```

- Documents: 8, terms: 677: Votre collection de textes contient 8 documents différents, comprenant un total de 677 termes uniques. Cela indique que vos données textuelles se composent de 8 documents distincts et contiennent un total de 608 termes différents.

- Non-/sparse entries: 945/3919: Dans la matrice document-termes (DTM), il y a un total de 1074 entrées de termes, dont 4342 sont vides (c'est-à-dire de valeur zéro). Cela montre que la plupart des termes dans vos textes sont rares, et de nombreux termes ne se trouvent pas dans chaque document.

- Sparsity: 80%: Le taux de sparsité de la DTM est de 80 %. Cela indique que la DTM est principalement clairsemée, et que la plupart des termes sont rarement présents dans les documents.

- **Maximal term length: 15:** La longueur maximale des termes dans la DTM est de 15 caractères. En d'autres termes, la DTM n'inclut pas de termes ayant plus de 15 caractères.
- **Weighting: term frequency (tf):** Le type de pondération des termes utilisé dans la DTM est "term frequency (tf)". Cela signifie que les termes sont pondérés en fonction de leur fréquence d'apparition dans les documents. La pondération des termes reflète la fréquence des termes dans les documents.

analyse1bis=removeSparseTerms(analyse1,0.4)

analyse1bis

**<<DocumentTermMatrix (documents: 8, terms: 23)>>**

**Non-/sparse entries: 137/47**

**Sparsity : 26%**

**Maximal term length: 9**

**Weighting : term frequency (tf)**

- **Documents: 8, terms: 23:** La nouvelle matrice document-termes (DTM) contient 8 documents et un total de 23 termes différents. Le nombre de termes a considérablement diminué par rapport à la DTM précédente.
- **Non-/sparse entries: 137/47:** Dans la DTM, il y a un total de 137 entrées de termes, dont 47 sont vides (c'est-à-dire de valeur zéro). Cela indique que la nouvelle DTM est plus dense.
- **Sparsity: 26%:** Le taux de sparsité de la DTM est de 26 %. Cela montre que la nouvelle DTM est plus dense, bien que certains termes soient encore rarement présents.
- **Maximal term length 9:** La longueur maximale des termes dans la DTM est de 9 caractères.
- **Weighting: term frequency (tf):** Le type de pondération des termes utilisé dans la DTM est "term frequency (tf)". Cela signifie que les termes sont pondérés en fonction de leur fréquence d'apparition dans les documents. La pondération des termes reflète la fréquence des termes dans les documents.

Cette nouvelle matrice document-termes (DTM) contient moins de termes et ces termes sont représentés de manière plus dense. Cela nous permet d'utiliser une DTM plus significative et efficace pour l'analyse. Une DTM plus dense peut contribuer à une analyse plus efficace des données textuelles.

```
analyse1bis=as.matrix(analyse1bis)
```

analyse1bis

```
> analyse1bis
Terms
Docs bât climat consom dénerg effet gaz métropol nouveau objet plan public renouvel rénov serr territoire
1 1 7 2 2 3 2 2 2 1 4 2 1 1 2 2
2 2 4 5 4 1 1 3 0 0 3 1 1 3 1 2
3 0 0 4 1 0 1 0 1 2 2 4 0 0 0 0
4 1 5 0 2 1 1 3 3 1 3 0 2 4 1 2
5 0 0 2 1 0 0 11 0 1 0 0 2 0 0 5
6 4 0 1 1 0 1 4 1 1 1 2 2 4 0 1
7 1 2 1 2 1 1 0 3 2 5 0 2 1 1 3
8 2 2 1 0 1 1 0 0 2 1 1 0 1 1 1

Terms
Docs valoris vill énerget commun développ habit moyen not
1 1 1 2 0 0 0 0 0
2 1 0 4 2 1 2 1 1
3 1 5 2 0 1 0 1 0
4 1 0 7 1 2 1 1 3
5 2 4 5 2 3 2 1 1
6 0 0 7 2 1 1 0 2
7 1 2 3 2 3 1 0 2
8 1 3 2 0 2 0 2 0
```

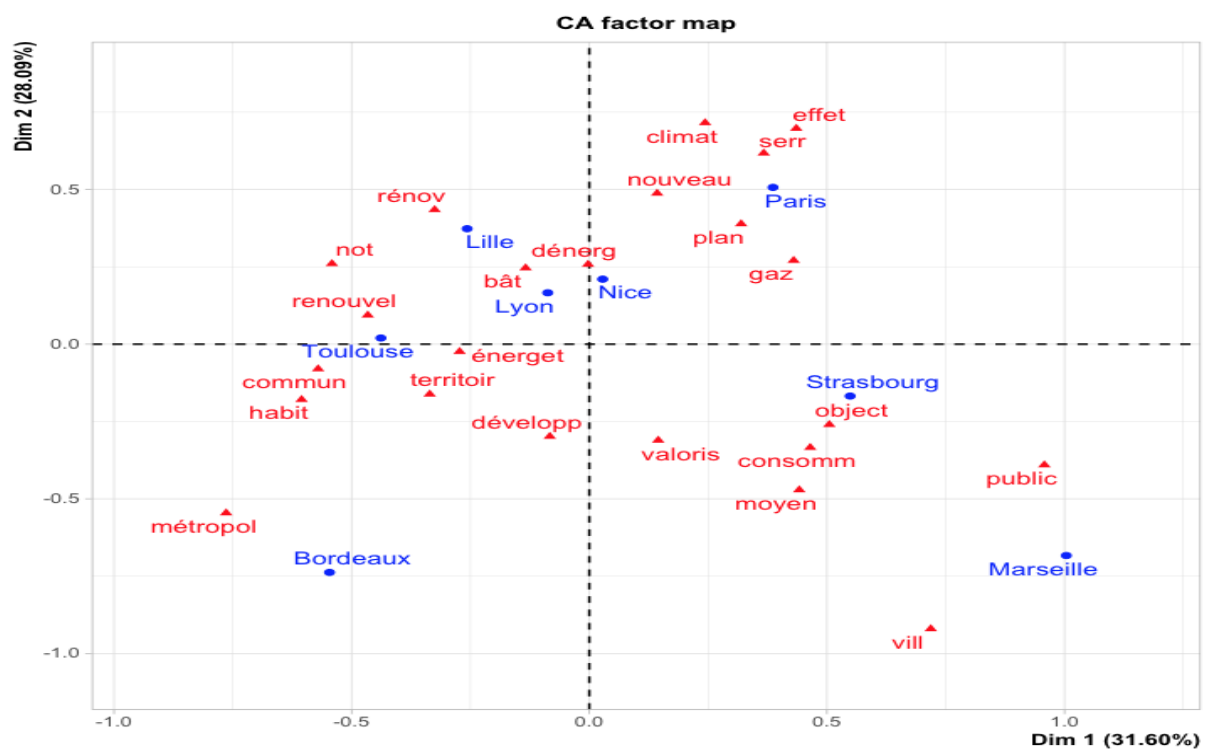
Ce graphique montre les fréquences des termes spécifiques présents dans 8 documents différents (Docs). Chaque ligne représente un document, tandis que chaque colonne représente un terme. Les valeurs dans le graphique indiquent combien de fois le terme correspondant apparaît dans ce document particulier. Par exemple, le terme "bât" apparaît une fois dans le document "Docs 1".

```
rownames(analyse1bis)=data$Ville
```

```
plot(CA(as.matrix(analyse1bis)))
```

```
analyse1bis=as.matrix(analyse1bis)
```

```
plot(CA(analyse1bis))
```





## Analyse TF-idf

```
analyse2=DocumentTermMatrix(texte_corpus,control=list(weighing=weightTfIdf))
```

analyse2

```
> analyse2
<<DocumentTermMatrix (documents: 8, terms: 677)>>
Non-/sparse entries: 1066/4350
Sparsity : 80%
Maximal term length: 15
Weighting : term frequency - inverse document frequency (normalized) (tf-idf)
>
```

- "analyse2" est une matrice document-termes (DTM) qui comprend 8 documents différents et un total de 677 termes différents.
- Il y a un total de 1066 entrées de termes dans la DTM, dont 4350 sont des entrées vides (c'est-à-dire de valeur zéro), indiquant que la matrice est principalement clairsemée, avec une sparsité de 80 %.
- La longueur maximale des termes dans la DTM est de 15 caractères

```
analyse2bis=removeSparseTerms(analyse2,0.4)
```

analyse2bis

```
> analyse2bis=removeSparseTerms(analyse2,0.4)
> analyse2bis
<<DocumentTermMatrix (documents: 8, terms: 22)>>
Non-/sparse entries: 129/47
Sparsity : 27%
Maximal term length: 9
Weighting : term frequency - inverse document frequency (normalized) (tf-idf)
```

- Cette matrice est une matrice document-termes (DTM) qui comprend 8 documents différents et un total de 22 termes différents.
- Il y a un total de 129 entrées de termes dans la DTM, dont 47 sont des entrées vides (c'est-à-dire de valeur zéro), ce qui indique que la matrice est principalement dense, avec une sparsité de 26 %.
- La longueur maximale des termes dans la DTM est de 9 caractères.

```
analyse2bis=as.matrix(analyse2bis)
```

```
analyse2bis
```

```
> analyse2bis=as.matrix(analyse2bis)
> analyse2bis
```

| Docs | bât         | climat      | consomm      | dénerg       | effet       | gaz          | métropol    | nouveau     | object      | plan        | public      | renouvel    |
|------|-------------|-------------|--------------|--------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 1    | 0.002530716 | 0.028942094 | 0.0023493302 | 0.0023493302 | 0.012403754 | 0.0023493302 | 0.00826917  | 0.008269170 | 0.001174665 | 0.004698660 | 0.008269170 | 0.002530716 |
| 2    | 0.005320994 | 0.017386459 | 0.0061745217 | 0.0049396174 | 0.004346615 | 0.0012349043 | 0.01303984  | 0.000000000 | 0.000000000 | 0.003704713 | 0.004346615 | 0.002660497 |
| 3    | 0.000000000 | 0.000000000 | 0.0035840945 | 0.0008960236 | 0.000000000 | 0.0008960236 | 0.000000000 | 0.003153823 | 0.001792047 | 0.001792047 | 0.012615291 | 0.000000000 |
| 4    | 0.002695049 | 0.022015322 | 0.0000000000 | 0.0025018841 | 0.004403064 | 0.0012509421 | 0.01320919  | 0.013209193 | 0.001250942 | 0.003752826 | 0.000000000 | 0.005390097 |
| 5    | 0.000000000 | 0.000000000 | 0.0020278429 | 0.0010139215 | 0.000000000 | 0.000000000  | 0.03925679  | 0.000000000 | 0.001013921 | 0.000000000 | 0.000000000 | 0.004368816 |
| 6    | 0.008783862 | 0.000000000 | 0.0010192861 | 0.0010192861 | 0.000000000 | 0.0010192861 | 0.01435073  | 0.003587682 | 0.001019286 | 0.001019286 | 0.007175364 | 0.004391931 |
| 7    | 0.002054641 | 0.006713583 | 0.0009536885 | 0.0019073770 | 0.003356792 | 0.0009536885 | 0.000000000 | 0.010070375 | 0.001907377 | 0.004768443 | 0.000000000 | 0.004109282 |
| 8    | 0.005724655 | 0.009352716 | 0.0013285867 | 0.0000000000 | 0.004676358 | 0.0013285867 | 0.000000000 | 0.000000000 | 0.002657173 | 0.001328587 | 0.004676358 | 0.000000000 |

| Docs | renouv      | serr        | territoir    | valoris      | vill        | commun       | développ     | habit       | moyen       | not         |
|------|-------------|-------------|--------------|--------------|-------------|--------------|--------------|-------------|-------------|-------------|
| 1    | 0.002530716 | 0.008269170 | 0.002349330  | 0.0011746651 | 0.004134585 | 0.000000000  | 0.000000000  | 0.000000000 | 0.000000000 | 0.000000000 |
| 2    | 0.007981490 | 0.004346615 | 0.0012349043 | 0.000000000  | 0.008693230 | 0.0012349043 | 0.008693230  | 0.004346615 | 0.004346615 | 0.004346615 |
| 3    | 0.000000000 | 0.000000000 | 0.000000000  | 0.0008960236 | 0.015769114 | 0.000000000  | 0.0008960236 | 0.000000000 | 0.003153823 | 0.000000000 |
| 4    | 0.010780195 | 0.004403064 | 0.002501884  | 0.0012509421 | 0.000000000 | 0.004403064  | 0.0025018841 | 0.004403064 | 0.004403064 | 0.013209193 |
| 5    | 0.000000000 | 0.000000000 | 0.005069607  | 0.0020278429 | 0.014275198 | 0.007137599  | 0.0030417644 | 0.007137599 | 0.003568800 | 0.003568800 |
| 6    | 0.008783862 | 0.000000000 | 0.001019286  | 0.000000000  | 0.000000000 | 0.007175364  | 0.0010192861 | 0.003587682 | 0.000000000 | 0.007175364 |
| 7    | 0.002054641 | 0.003356792 | 0.002861066  | 0.0009536885 | 0.006713583 | 0.006713583  | 0.0028610655 | 0.003356792 | 0.000000000 | 0.006713583 |
| 8    | 0.002862328 | 0.004676358 | 0.001328587  | 0.0013285867 | 0.014029074 | 0.000000000  | 0.0026571735 | 0.000000000 | 0.009352716 | 0.000000000 |

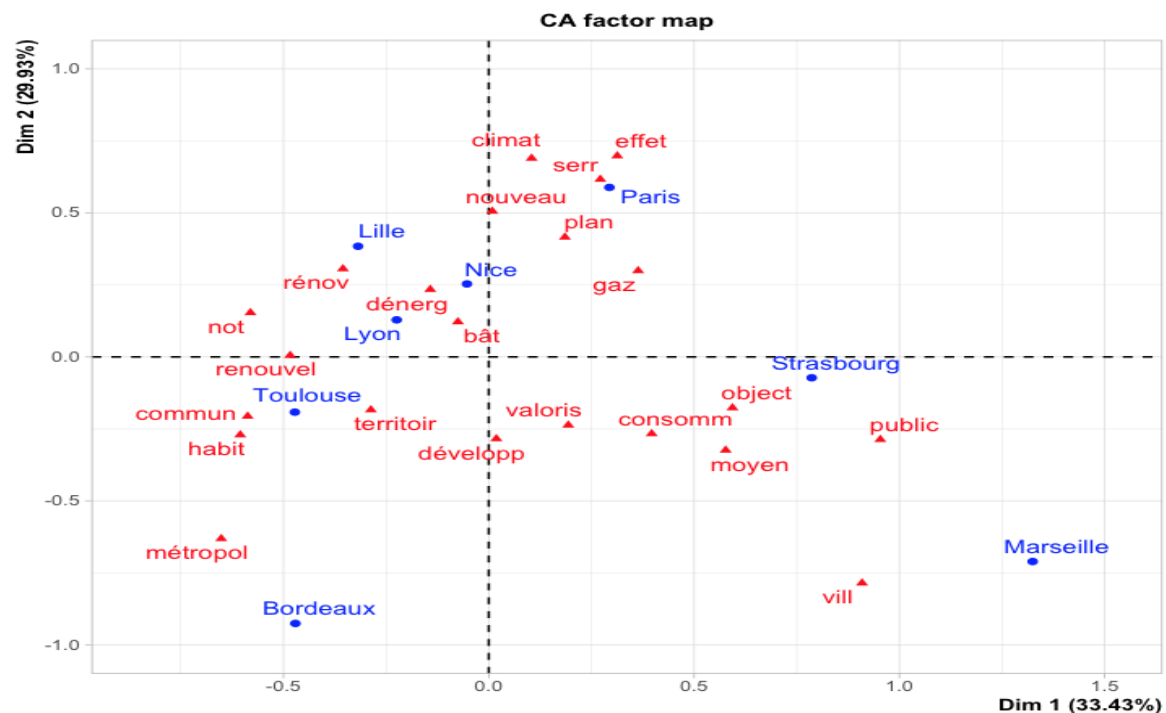
```
rownames(analyse2bis)=data$Ville
```

```
plot(CA(analyse2bis))
```

```
plot(CA(analyse2bis),invisible="col")
```

```
plot(CA(analyse2bis),invisible="row")
```

```
plot(CA(analyse2bis),ylim=c(-1,1))
```



```
termes_avec_le_plus_haut_TF_IDF <- apply(analyse2bis, 1, which.max)
```

```
termes_avec_le_plus_haut_TF_IDF
```

```
> termes_avec_le_plus_haut_TF_IDF <- apply(analyse2bis, 1, which.max)
> termes_avec_le_plus_haut_TF_IDF
 Paris Lyon Marseille Lille Bordeaux Toulouse Nice
 2 2 17 2 7 7 8
Strasbourg
 17
> |
```

```
villes <- data$Ville
```

**# Identifier les noms des termes ayant le poids TF-IDF le plus élevé dans chaque document**

```
noms_des_termes_plus_élevés_TF_IDF <-
colnames(analyse2bis)[termes_avec_le_plus_haut_TF_IDF]
```

**# Création du cadre de données**

```
donnees_des_plus_élevés_TF_IDF <- data.frame(Ville = villes,
Nom_des_Termes_le_plus_élevé_TF_IDF = noms_des_termes_plus_élevés_TF_IDF)
```

**# Afficher les résultats**

```
donnees_des_plus_élevés_TF_IDF
```

```
> villes <- data$Ville
> noms_des_termes_plus_élevés_TF_IDF <- colnames(analyse2bis)[termes_avec_le_plus_haut_TF_IDF]
> donnees_des_plus_élevés_TF_IDF <- data.frame(Ville = villes, Nom_des_Termes_le_plus_élevé_TF_IDF = noms_des_termes_plus_élevés_TF_IDF)
> donnees_des_plus_élevés_TF_IDF
 Ville Nom_des_Termes_le_plus_élevé_TF_IDF
1 Paris climat
2 Lyon climat
3 Marseille vill
4 Lille climat
5 Bordeaux métropol
6 Toulouse métropol
7 Nice nouveau
8 Strasbourg vill
> |
```

## ANALYSE DE SIMILITUDE DES TEXTES.

```
tfidf_matrix <- weightTfidf(analyse2)
```

**# Extraction des vecteurs TF-IDF des villes**

```
vecteur_paris <- as.numeric(as.matrix(tfidf_matrix)[1,])
```

```

vecteur_lyon <- as.numeric(as.matrix(tfidf_matrix)[2,])
vecteur_marseille <- as.numeric(as.matrix(tfidf_matrix)[3,])
vecteur_lille <- as.numeric(as.matrix(tfidf_matrix)[4,])
vecteur_bordeaux <- as.numeric(as.matrix(tfidf_matrix)[5,])
vecteur_toulouse <- as.numeric(as.matrix(tfidf_matrix)[6,])
vecteur_nice <- as.numeric(as.matrix(tfidf_matrix)[7,])
vecteur_strasbourg <- as.numeric(as.matrix(tfidf_matrix)[8,])

```

### **# Calculer les similitudes entre les villes**

```

similarity_lyon_paris <- sum(vecteur_lyon * vecteur_paris) / (sqrt(sum(vecteur_lyon^2)) *
sqrt(sum(vecteur_paris^2)))

similarity_marseille_paris <- sum(vecteur_marseille * vecteur_paris) /
(sqrt(sum(vecteur_marseille^2)) * sqrt(sum(vecteur_paris^2)))

similarity_lille_paris <- sum(vecteur_lille * vecteur_paris) / (sqrt(sum(vecteur_lille^2)) *
sqrt(sum(vecteur_paris^2)))

similarity_bordeaux_paris <- sum(vecteur_bordeaux * vecteur_paris) /
(sqrt(sum(vecteur_bordeaux^2)) * sqrt(sum(vecteur_paris^2)))

similarity_toulouse_paris <- sum(vecteur_toulouse * vecteur_paris) /
(sqrt(sum(vecteur_toulouse^2)) * sqrt(sum(vecteur_paris^2)))

similarity_nice_paris <- sum(vecteur_nice * vecteur_paris) / (sqrt(sum(vecteur_nice^2)) *
sqrt(sum(vecteur_paris^2)))

similarity_strasbourg_paris <- sum(vecteur_strasbourg * vecteur_paris) /
(sqrt(sum(vecteur_strasbourg^2)) * sqrt(sum(vecteur_paris^2)))

cat("Score de Similarité Paris-Lyon:", similarity_lyon_paris, "\n")
cat("Score de Similarité Paris-Marseille:", similarity_marseille_paris, "\n")
cat("Score de Similarité Paris-Lille:", similarity_lille_paris, "\n")
cat("Score de Similarité Paris-Bordeaux:", similarity_bordeaux_paris, "\n")
cat("Score de Similarité Paris-Toulouse:", similarity_toulouse_paris, "\n")
cat("Score de Similarité Paris-Nice:", similarity_nice_paris, "\n")

```

```
cat("Score de Similarité Paris-Strasbourg:", similarity_strasbourg_paris, "\n")
```

```
> cat("Score de Similarité Paris-Lyon:", similarity_lyon_paris, "\n")
Score de Similarité Paris-Lyon: 0.01537617
> cat("Score de Similarité Paris-Marseille:", similarity_marseille_paris, "\n")
Score de Similarité Paris-Marseille: 0.02643783
> cat("Score de Similarité Paris-Lille:", similarity_lille_paris, "\n")
Score de Similarité Paris-Lille: 0.02690809
> cat("Score de Similarité Paris-Bordeaux:", similarity_bordeaux_paris, "\n")
Score de Similarité Paris-Bordeaux: 0.02146351
> cat("Score de Similarité Paris-Toulouse:", similarity_toulouse_paris, "\n")
Score de Similarité Paris-Toulouse: 0.003896712
> cat("Score de Similarité Paris-Nice:", similarity_nice_paris, "\n")
Score de Similarité Paris-Nice: 0.02157327
> cat("Score de Similarité Paris-Strasbourg:", similarity_strasbourg_paris, "\n")
Score de Similarité Paris-Strasbourg: 0.02268408
```

## Paris, Lyon, Lille - "climat"

Nous mesurons les similarités entre les trois villes (Paris, Lyon et Lille ) où le terme "climat" a la plus grande pondération de TF-IDF.

```
similarity_lille_lyon <- sum(vecteur_lille * vecteur_lyon) / (sqrt(sum(vecteur_lille^2)) *
sqrt(sum(vecteur_lyon^2)))
```

```
similarity_lyon_paris <- sum(vecteur_lyon * vecteur_paris) / (sqrt(sum(vecteur_lyon^2)) *
sqrt(sum(vecteur_paris^2)))
```

```
similarity_lille_paris <- sum(vecteur_lille * vecteur_paris) / (sqrt(sum(vecteur_lille^2)) *
sqrt(sum(vecteur_paris^2)))
```

```
cat("Score de Similarité Lille-Lyon:", similarity_lille_lyon, "\n")
```

```
cat("Score de Similarité Lyon-Paris:", similarity_lyon_paris, "\n")
```

```
cat("Score de Similarité Lille-Paris:", similarity_lille_paris, "\n")
```

```
> cat("Score de Similarité Lille-Lyon:", similarity_lille_lyon, "\n")
Score de Similarité Lille-Lyon: 0.03192819
> cat("Score de Similarité Lyon-Paris:", similarity_lyon_paris, "\n")
Score de Similarité Lyon-Paris: 0.01537617
> cat("Score de Similarité Lille-Paris:", similarity_lille_paris, "\n")
Score de Similarité Lille-Paris: 0.02690809
```

Score de Similarité Lille-Lyon : Le score de similarité entre Lille et Lyon est calculé à 0,0319. Ce score indique que les textes de Lille et de Lyon sont plus similaires, bien que le degré de similarité soit encore relativement faible.



Score de Similarité Lyon-Paris : Le score de similarité entre Lyon et Paris est calculé à 0,0153. Ce score suggère que les textes entre Lyon et Paris sont relativement peu similaires, indiquant des différences notables.

Score de Similarité Lille-Paris : Le score de similarité entre Lille et Paris est calculé à 0,0269. Ce score indique que les textes entre Lille et Paris sont plus similaires, bien que le degré de similarité reste relativement bas.

## **Bordeaux et Toulouse-"métropol"**

Nous mesurons les similarités entre les deux villes (Bordeaux et Toulouse) où le mot "métropol" a le poids TF-IDF le plus élevé.

```
similarity_bordeaux_toulouse <- sum(vecteur_bordeaux * vecteur_toulouse) /
(sqrt(sum(vecteur_bordeaux^2)) * sqrt(sum(vecteur_toulouse^2)))

cat("Score de Similarité Bordeaux-Toulouse:", similarity_bordeaux_toulouse, "\n")
```

```
> cat("Score de Similarité Bordeaux-Toulouse:", similarity_bordeaux_toulouse, "\n")
Score de Similarité Bordeaux-Toulouse: 0.007138049
```

Score de Similarité Bordeaux-Toulouse : Le score de similarité entre Bordeaux et Toulouse est calculé à 0,0071. Ce score indique que les textes entre Bordeaux et Toulouse sont relativement peu similaires, suggérant des différences significatives.

## **ANALYSE DE SIMILARITE DE TEXTE BASEE SUR LA PRESENCE OU L'ABSENCE DE 'SMART' PAR VILLE**

1) Similarité entre les villes intelligentes(Smart city) et les villes non intelligentes.

### **# "Villes non intelligentes"**

```
vecteur_non_intelligent <- rbind(vecteur_paris, vecteur_lyon, vecteur_marseille, vecteur_lille)
```

### **# "Villes intelligentes"**

```
vecteur_intelligent <- rbind(vecteur_bordeaux, vecteur_toulouse, vecteur_nice, vecteur_strasbourg)
```

### **# Similarité entre les "villes intelligentes" et les "villes non intelligentes"**

```
similarité_villes_intelligentes_non_intelligentes <- sum(vecteur_intelligent %*%
t(vecteur_non_intelligent)) / (sqrt(sum(vecteur_intelligent^2)) *
sqrt(sum(t(vecteur_non_intelligent)^2)))
```

```
cat("Similarité entre les Villes Intelligentes et les Villes Non Intelligentes :",
similarité_villes_intelligentes_non_intelligentes, "\n")
```

```
> cat("Similarité entre les Villes Intelligentes et les Villes Non Intelligentes :", similarité_villes_intelligentes_non_intelligentes, "\n")
Similarité entre les Villes Intelligentes et les Villes Non Intelligentes : 0.0761063
>
```

Similarité entre les Villes Intelligentes et les Villes Non Intelligentes : Le score de similarité donné, qui est de "0,0761", indique que les textes entre les "Villes Intelligentes" (villes intelligentes) et les "Villes Non Intelligentes" (villes non intelligentes) ont une similarité de niveau moyen. Cette similarité mesure les ressemblances entre les textes de deux types de villes différents.

Ce résultat suggère qu'il existe des similitudes entre les villes intelligentes et les villes non intelligentes, mais qu'il subsiste encore des différences notables.

## 2) Calcul de la similarité entre les villes intelligentes

### # Créez les vecteurs pour les villes intelligentes

```
vecteur_intelligent <- rbind(vecteur_bordeaux, vecteur_toulouse, vecteur_nice, vecteur_strasbourg)
```

### # Calcul de la similarité entre les villes intelligentes (internes)

```
similarité_villes_intelligentes <- sum(vecteur_intelligent %*% t(vecteur_intelligent)) /
(sqrt(sum(vecteur_intelligent^2)) * sqrt(sum(t(vecteur_intelligent)^2)))
```

```
cat("Similarité entre les Villes Intelligentes (internes) :", similarité_villes_intelligentes, "\n")
```

```
> cat("Similarité entre les Villes Intelligentes (internes) :", similarité_villes_intelligentes, "\n")
Similarité entre les Villes Intelligentes (internes) : 1.055845
.
```

Similarité entre les "Villes Intelligentes" (Smart Cities) : La valeur de similarité est de 1.055845, indiquant que les textes des villes intelligentes sont plus similaires entre eux.

## 3) Calcul de la similarité entre les villes non intelligentes

### # Créez les vecteurs pour les villes non intelligentes

```
vecteur_non_intelligent <- rbind(vecteur_paris, vecteur_lyon, vecteur_marseille, vecteur_lille)
```

### # Calcul de la similarité entre les Villes Non Intelligentes (internes)

```
similarité_villes_non_intelligentes <- sum(vecteur_non_intelligent %*% t(vecteur_non_intelligent)) /
(sqrt(sum(vecteur_non_intelligent^2)) * sqrt(sum(t(vecteur_non_intelligent)^2)))
```

```
cat("Similarité entre les Villes Non Intelligentes (internes) :", similarité_villes_non_intelligentes, "\n")
```

```
> cat("Similarité entre les Villes Non Intelligentes (internes) :", similarité_villes_non_intelligentes, "\n")
Similarité entre les Villes Non Intelligentes (internes) : 1.07142
>
```

Similarité entre les "Villes Non Intelligentes" (Non-Smart Cities) : La valeur de similarité est de 1.07142, montrant que les textes des villes non intelligentes sont également assez similaires entre eux.

En résumé, bien qu'il existe une certaine similarité entre les textes des villes intelligentes et non intelligentes, les textes des villes intelligentes présentent une similarité plus marquée, tout comme les textes des villes non intelligentes qui sont également similaires entre eux.