Textual Analysis of Expert Reports to increase knowledge of Technological Risks

Julie Seguela

https://gitlab.com/jseguela/r_in_insurance_2017

R in Insurance - 8 june 2017

My background

- Master degree in Statistics and Econometrics - *Toulouse School of Economics*
- PhD in Applied Statistics and Machine Learning - *Conservatoire National des Arts et Métiers*
- Statistician who has evolved to a data scientist
- Interests: R, web and textual data, machine learning, high performance computing
- 8 years working with R and a growing interest
- **Covea**: French mutualist group formed with 3 insurance companies (GMF, MAAF, MMA) http://www.covea.eu/

Text Mining in Insurance

- Insurers hold several sources of textual data:
  - *Comments from advisers during telephone exchanges*
  - *Customers' comments from satisfaction surveys*
  - *Claims narratives (-> second talk in Big Data session)*
  - *Expert reports:*
    - about individual claims
    - about business claims (-> this talk)
- Several projects are in progress at Covea
- This talk is about expert reports which describe circumstances, causes and consequences of technological accidents in several industries

Purpose of this work/PoC

- Extract useful information from expert reports in order to increase our knowledge of technological accidents
- Identify frequent types of events and their causes if they are known
- Highlight good R packages for text analysis and visualization
- Expert reports comes from **open source data**, why?
  - *Projects at Covea are still in progress and results not yet communicable…*
  - *Working with open data : anyone can access, use or share*
  - *Useful if no or few historical claims data available internally*
- This code can be shared to interested people
- Open data + open code insure the reproducibility of the study

Data source description

- **ARIA** database (*Analysis, Reasearch and Information on Accidents*)
  http://www.aria.developpement-durable.gouv.fr/?lang=en
- Published by BARPI (analysis office at the Ministry of sustainable development)
- Database:
  - *30,000 accidents which have occurred mainly in France since 1995*
  - *Susceptible to damage public health or safety, or environment*
  - *Reports written by civil protection or environment inspection*
  - *Inventory is not exhaustive but significant, and almost all accidents relative to major risks are registered*

Example of an expert report

```
In a sawmill subject to declaration, a fire ignites at 20:30 at the
level of 2 sheds of wood storage of 500 m². Local residents give the alarm.
Firemen protect neighboring buildings. They extinguish the fire around 3:45.
The 2 hangars and 1 000 m³ of finished products are destroyed. The damage is
estimated at EUR 600 000. 5 employees are unemployed. However, the production
tool is spared. The site was shut down for 3 days after the fire broke out.
An act of malevolence could be the cause of the fire.
```

- Pros:
  - *no ambiguity, only facts are detailed*
  - *reports are well written, with very few spelling mistakes*
  - *normalised wording*
- Cons:
  - *texts can be very long as well as very short*
  - *technical vocabulary*

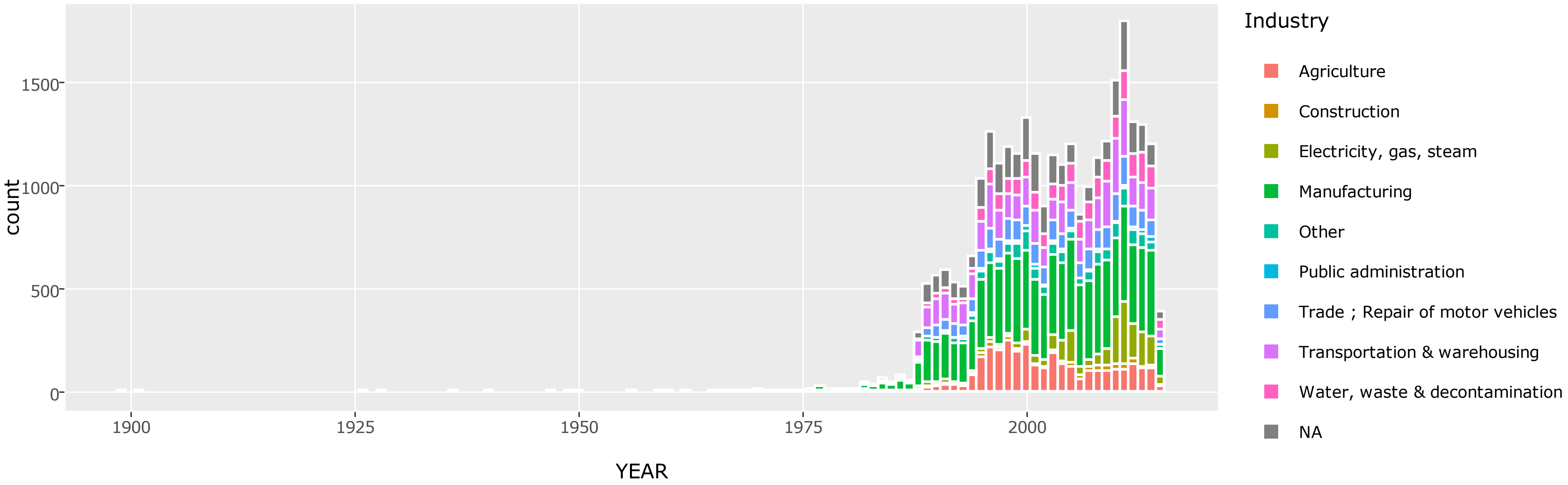
Dataset description: list of variables

- 28,260 accidents
- Variables:
  - *identifier of accidents*
  - *date of accident*
  - *french department*
  - *french city*
  - *industry type*
  - *text of the report*

N°47930 - 18/04/2016 - FRANCE - 24 - COULOUNIEIX-CHAMIERS
D35.22 - Distribution de combustibles gazeux par conduites
Vers 9h50, une fuite de gaz naturel se produit suite à l'arrachement d'une conduite de distribution à 4 bars lors de travaux de voirie. La fuite se produit au niveau de voies ferrées. Le trafic ferroviaire est interrompu sur deux voies pendant 1h20. Un périmètre de sécurité de 100 m est mis en place ; 45 personnes sont évacuées. Les services du gaz stoppent la fuite par écrasement de la conduite. La circulation est rétablie après que les mesures d'explosimétrie se révèlent négatives.

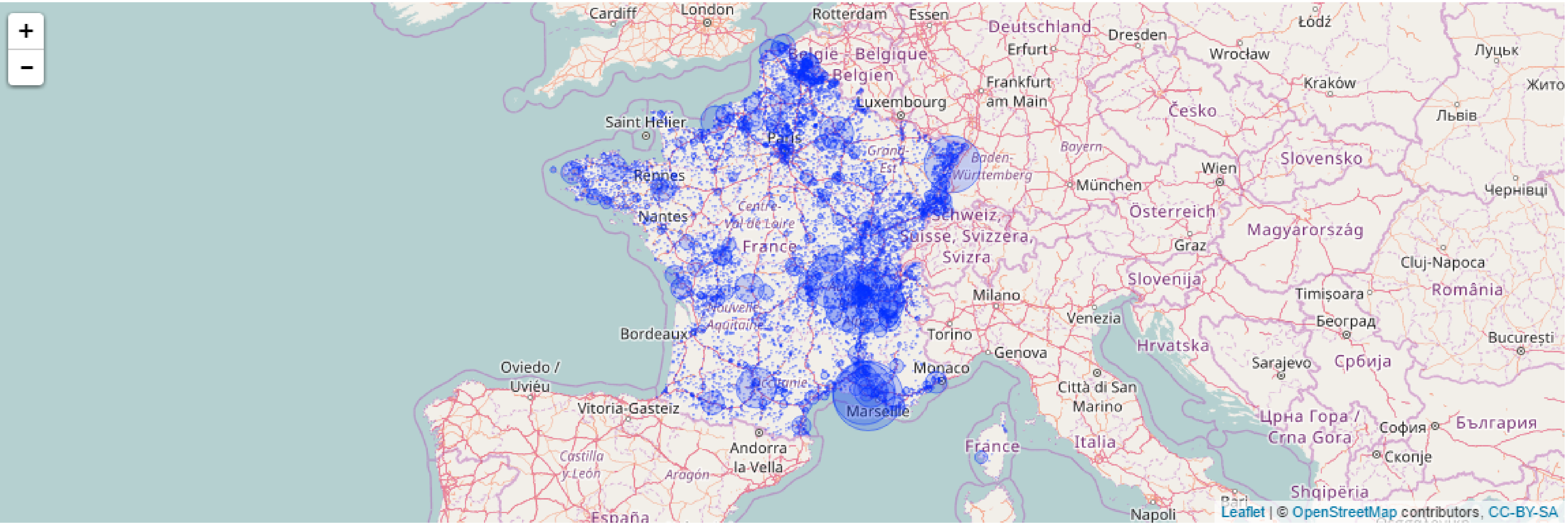
Dataset description: accidents over time

```r
acc <- readRDS("data/acc.rds")
g <- ggplot(aes(x = YEAR), data = acc) + geom_bar(aes(fill = Industry), color = "white") +
  theme(legend.position = "right")
ggplotly(g)
```



Dataset description: accidents over French territory

```
library(leaflet)
city.freq <- acc %>% filter(!is.na(CITY)) %>% group_by(CITY_CODE, CITY, long, lat) %>% summarize(freq = n())
m <- leaflet() %>% addTiles() %>%  addCircleMarkers(lng=city.freq$long, lat=city.freq$lat,
                                          popup = paste(city.freq$CITY, ": ", city.freq$freq),
                                          radius = city.freq$freq/8, weight = 1)
m  # Print the map
```



What is Text Mining?

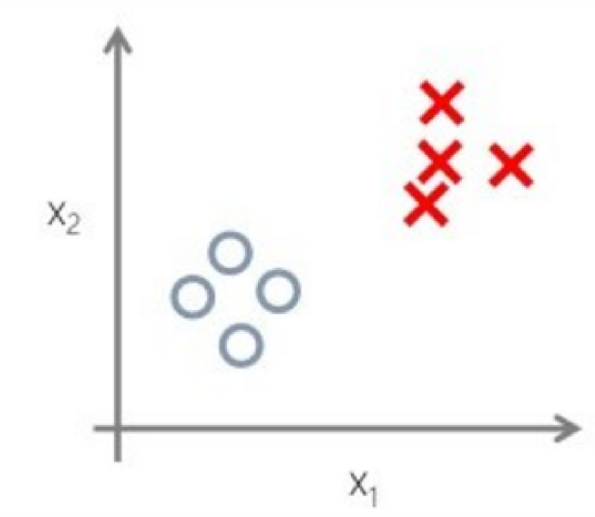
- Textual data are in essence unstructured. It is not possible to analyze them without applying some processing.
- Text mining techniques allows to extract useful information from these unstructured data.
- It relies on:
  - *Bag-of-words model, or representation in vectors of features created according to some defined rules*
  - *Data mining techniques applied to these now structured data*

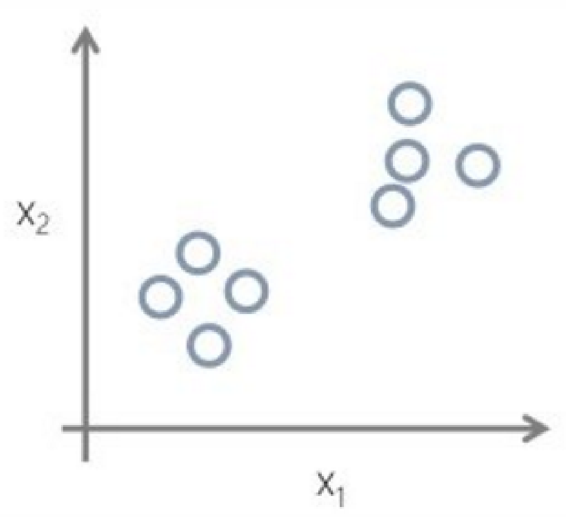
A bit of vocabulary

- **Document:** a generic term to name the analysed unit. In our case, the analysed unit is a text, but it could also be images or videos.
- **Corpus:** a set of documents grouped together for analysis. These documents must be sufficiently numerous, written in the same language and deal with the same field.
- **Terms:** all different words found in the corpus.
- **Tokens:** lexical units obtained after splitting texts according to a defined rule (in general, the words of the text)

Supervised vs Unsupervised

- We must distinguish *supervised* learning from *unsupervised* learning
  - *Document categorization with categories known a priori is a supervised task (classification)*
  - *Document clustering with no a priori information is an unsupervised task*





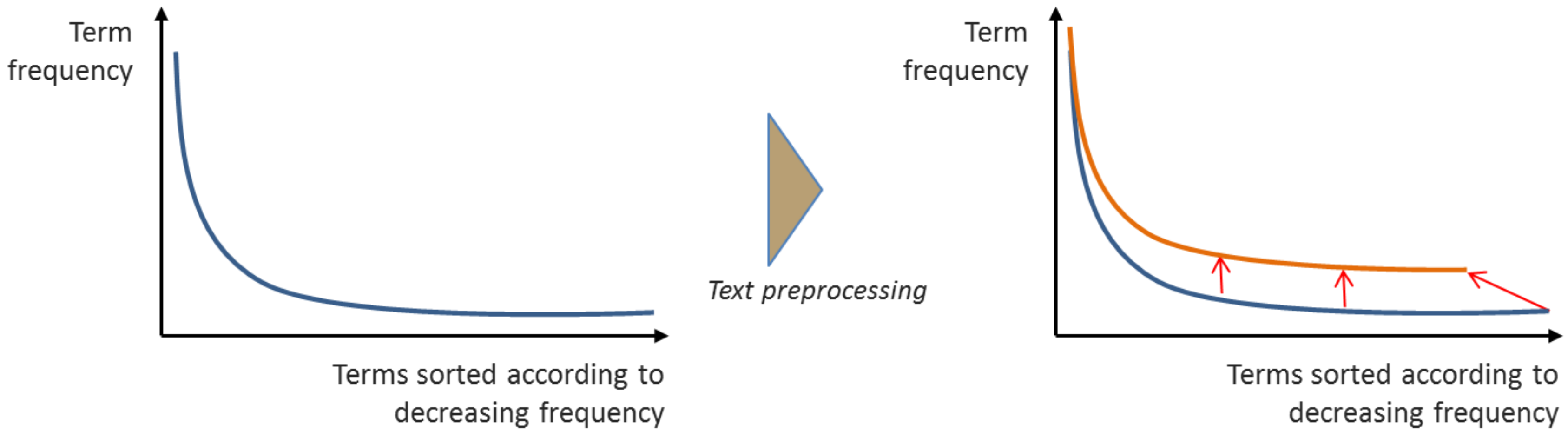
- **Our work corresponds to the second case**

Keep in mind limitations

- Except in some particular cases, text mining still requires significant human intervention
  - *text cleaning according to the business field*
  - *parameters to be adjusted according to the studied corpus*
  - *analysis and interpretation of results*
- For all unsupervised text mining applications, result validation is mainly subjective (when truth is not known *a priori*, no quantitative measure to assess performance)
- With *bag-of-words* representation, the order of words in a sentence is lost (but co-occurrences are kept)
- Since text mining relies on statistical techniques, at least hundreds of texts are needed to get interpretable results
- The more heterogeneous the texts in the Corpus, the more difficult the analysis

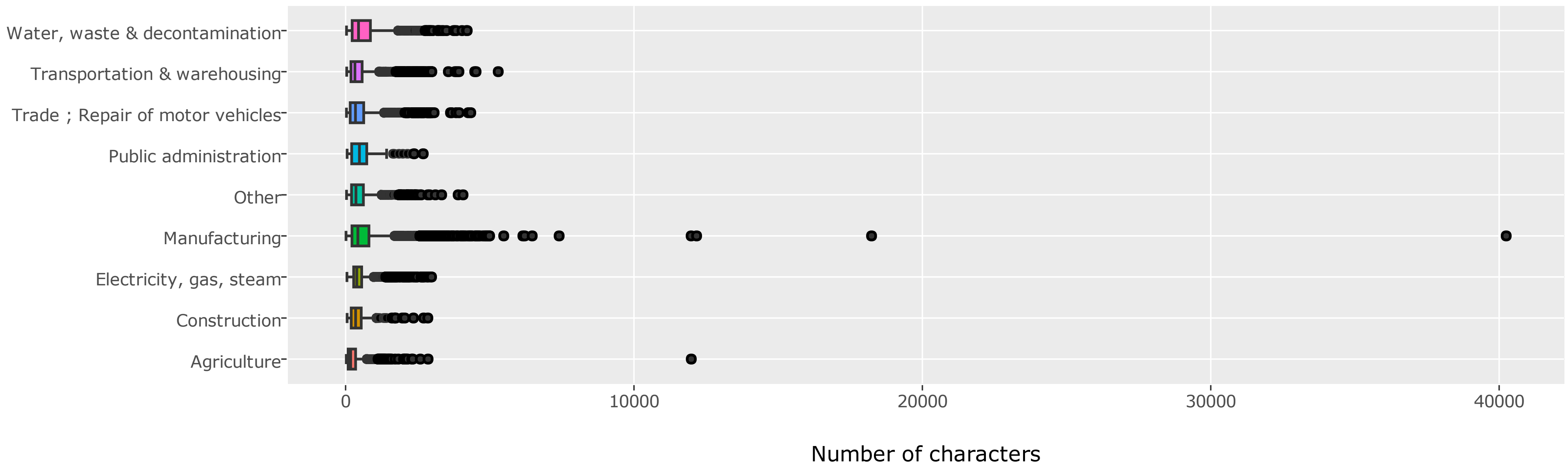
Concept of the Long Tail

- Once document texts are cut into words, the distribution of term frequency is similar to the famous "long tail"
- A minority of terms (the most common in the language) represent a big part of total occurences, while a very large number of terms will have few or only one occurrence
- Purpose of preprocessing (in addition to cleaning and removing noise) is to:
  - *reduce the total number of distinct terms*
  - *increase frequency of each term (by grouping) to increase analysis robustness and detection of rare themes*



Document lengths according to industry

```r
acc$doc_length <- nchar(acc$doc)
p <- ggplot(aes(x=Industry, y=doc_length, fill=Industry), data=acc) + geom_boxplot() +
  coord_flip() + theme(legend.position="none", plot.margin=unit(c(1,1,1,1), "cm")) +
  xlab("") + ylab("Number of characters")
ggplotly(p)
```



Corpus creation

With **tm** package, the first step is to create a Corpus object:

```
library(tm)
MyCorpus <- VCorpus(VectorSource(acc$doc), readerControl = list(language = "fr"))
names(MyCorpus) <- acc$ID

# First document
content(MyCorpus[[1]])
```

[1] "Un incendie se déclare dans un bâtiment d'une entreprise spécialisée dans la récupération de vieilles palettes. Une heure de lutte est nécessaire pour maîtriser le sinistre s'étendant sur 500 m² détruisant un stock de palettes, des machines, 1 500 l de fioul et des bouteilles de gaz. La grande quantité de palettes entourant le bâtiment a pu être préservée. Une reprise de feu 7 heures après, pendant le nettoyage du hangar par les employés de l'entreprise, est rapidement maîtrisée."

Text preprocessing

Then, we apply some basic transformations to normalize texts, for example:

```
# to lowercase
MyCorpus <- tm_map(MyCorpus, content_transformer(tolower))

# remove accents
MyCorpus <- tm_map(MyCorpus, content_transformer(function(x) chartr("àâéèêëîïôöûùü","aaeeeeiioouuu", x)))

# remove punctuation
MyCorpus <- tm_map(MyCorpus, content_transformer(
                   function(x) gsub("[^[:alnum:][:space:]%€]", " ", x)
                   ))

# recode costs
MyCorpus <- tm_map(MyCorpus, content_transformer(
                   function(x) gsub("([[:digit:]]+[[:space:]]*)+(€|(euro)|(euros))", " _euro_", x)
                   ))
```

We also uniformise times, areas, volumes, distances, digits…

Term-Document Matrix creation (1/2)

```
TDM <- TermDocumentMatrix(MyCorpus, control = list(removePunctuation = F, stopwords = F,
                                                    wordLengths = c(1, Inf)))
TDM
```

```
## <<TermDocumentMatrix (terms: 29475, documents: 28260)>>
## Non-/sparse entries: 1556868/831406632
## Sparsity           : 100%
## Maximal term length: 35
## Weighting          : term frequency (tf)
```

```
# The longest term
Terms(TDM)[which.max(nchar(Terms(TDM)))]
```

[1] “paratrichloromethylphenylisocyanate”

```
grep("paratrichlorométhylphénylisocyanate", acc$doc)
```

[1] 23966

Term-Document Matrix creation (2/2)

Little extract of TDM before removing terms:

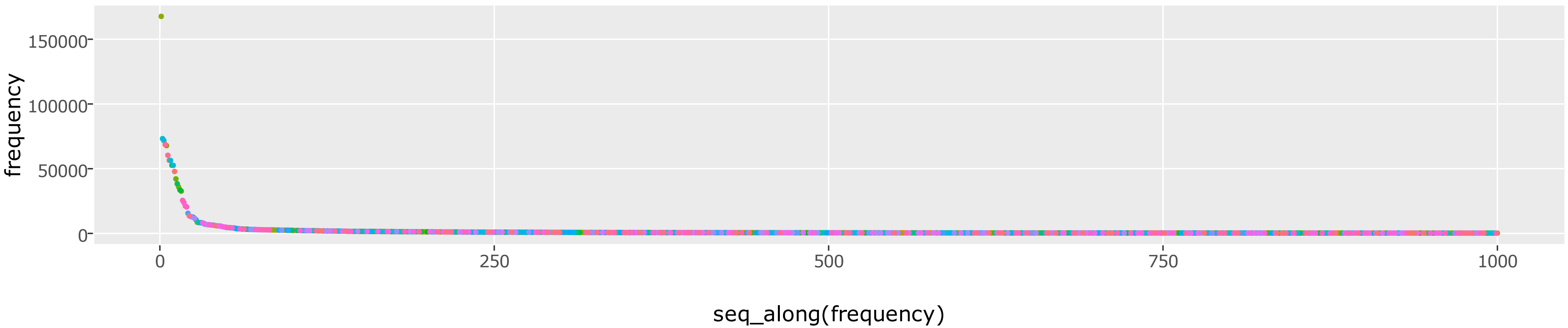
```
as.matrix(TDM[which(apply(TDM[,1:5],1,sum)>0), 1:5][1:15,])
```

```
##                Docs
## Terms          339076 31878 679195  1 480690
##   _area_            1     0      0  0      0
##   _digit_           1     0      1 16      1
##   _dist_            0     0      0  0      1
##   _time_            0     1      0  2      0
##   _volume_          1     0      1  0      0
##   4eme              0     1      0  0      0
##   a                 1     0      1  9      0
##   absence           0     1      0  0      0
##   acceptable        0     0      0  1      0
##   active            0     0      0  1      0
##   aerent            0     0      0  0      1
##   agissait          0     1      0  0      0
##   agrochimique      0     0      0  1      0
##   air               0     0      0  3      0
##   alerte            0     0      0  1      0
```

Some statistics about our corpus

Some statistics before we do significant transformations on the corpus:

```
# Term frequencies
term.freq <- data.frame(term = rownames(TDM), frequency = slam::row_sums(TDM)) %>% arrange(desc(frequency))
p <- ggplot(term.freq[1:1000, ], aes(x=seq_along(frequency), y=frequency, colour=term)) +
  geom_point(size = 0.5) + theme(legend.position="none", plot.margin=unit(c(1,1,1,1), "cm"))
ggplotly(p, tooltip = c("colour", "y"))
```




```
round(sum(term.freq$frequency[1:30])/sum(term.freq$frequency), 2)
```

[1] 0.48

Removing stopwords (1/2)

- We remove the most common words in french language, these are named *stopwords*
- We add a personal list of frequent and useless words in this corpus

```
head(stopwords("en"), 10) # illustration with english stopwords
```

```
##  [1] "i"         "me"        "my"        "myself"    "we"
##  [6] "our"       "ours"      "ourselves" "you"       "your"
```

```
# additional words to remove
to_add <- c("vers", "celui", "celle", "ceux", "celles", "ci", "ce", "quand", "quant", "egalement",
            "aussi", "ainsi", "tous", "tout", "toutes", "toute", "min", "elles", "elle", "ils", "selon",
            "afin", "dont", "tres", "deja", "enfin", "jusqu", "ni", "ne", "autour", "avant", "apres",
            "fois", "ans", "autres", "autre", "puis", "h", "lieu", "lieux", "puis", "quelque",
            "quelques", "encore", "matin", "non", "alors", "peu", "pu", "cependant", "peut", "donc",
            "faire", "etre", "situe", "pendant", "_digit_", "_time_", "_euro_", "suite", "lors", "sous")
# my stopword list
mystopwords <- union(stopwords("fr"), to_add)
# same transformations as applied on documents
mystopwords <- chartr("àâéèêëîïôöûùü","aaeeeeiioouuu", tolower(mystopwords))
MyCorpus <- tm_map(MyCorpus, stripWhitespace)
```

Removing stopwords (2/2)

```
TDM <- TermDocumentMatrix(MyCorpus, control = list(removePunctuation = F, stopwords = mystopwords,
                                                   wordLengths = c(1, Inf)))

as.matrix(TDM[which(apply(TDM[,1:5],1,sum)>0), 1:5][1:15,])
```

```
##                  Docs
## Terms            339076 31878 679195 1 480690
##   _area_              1     0      0 0      0
##   _dist_              0     0      0 0      1
##   _volume_            1     0      1 0      0
##   4eme                0     1      0 0      0
##   absence             0     1      0 0      0
##   acceptable          0     0      0 1      0
##   active              0     0      0 1      0
##   aerent              0     0      0 0      1
##   agissait            0     1      0 0      0
##   agrochimique        0     0      0 1      0
##   air                 0     0      0 3      0
##   alerte              0     0      0 1      0
##   alimentations       0     1      0 0      0
##   ambiant             0     0      0 1      0
##   analyses            0     0      0 1      0
```

Lemmatization (1/2)

- **tm** package provides access to stemming but it is not appropriate to french language
- We prefer to apply *lemmatization*, which is more relevant:
  - *conjugated verbs put to infinitive*
  - *plural form put to singular form*
  - *feminine form put to masculine form*
  - *use of **TreeTagger***
  - *efficient because reports are written carefully by experts*
  - *package **koRpus** and **treetag()** function*

After install of *Perl* and *TreeTagger*:

```
tagged.text <- treetag(file.path, treetagger="manual", lang="fr", encoding = "UTF-8",
                       TT.options=list(path="C:/TreeTagger", preset="fr-utf8"))
```

Lemmatization (2/2)

Lemmatization is a long process, so we have built a fixed dictionary to transform each term into its lemma:

```
dic <- readRDS("data/dic_lemm.rds")
tdm_words <- data.frame(token = rownames(TDM), order = 1:nrow(TDM), stringsAsFactors = F)
tdm_words <- merge(tdm_words, dic, by = "token", all.x = T, sort = F)
tdm_words <- tdm_words[order(tdm_words$order),]
tdm_words$lemma <- ifelse(is.na(tdm_words$lemma), tdm_words$token, tdm_words$lemma)

# Transforming to Lemmas
TDMlm <- slam::rollup(TDM, 1, tdm_words$lemma)
```

The number of distinct terms has now decreased a lot:

```
TDMlm
```

```
## <<TermDocumentMatrix (terms: 18706, documents: 28260)>>
## Non-/sparse entries: 999195/527632365
## Sparsity           : 100%
## Maximal term length: 35
```

Removing sparse terms

- At this time, TDM is still very sparse…
- Since we will use statistical techniques, we choose to remove terms with less than 20 occurrences:

```
freq.min <- 20
sparsity <- 1-freq.min/ncol(TDMlm)
TDMlm <- removeSparseTerms(TDMlm, sparse = sparsity) ; TDMlm
```

```
## <<TermDocumentMatrix (terms: 3791, documents: 28260)>>
## Non-/sparse entries: 949490/106184170
## Sparsity           : 99%
## Maximal term length: 17
```

- Sparsity is still very high… but the number of distinct terms has drastically decreased! (in accordance with the principle of the long tail)

```
term.freq <- data.frame(term = rownames(TDMlm), frequency = slam::row_sums(TDMlm)) %>% arrange(desc(frequency))
round(sum(term.freq$frequency[1:30])/sum(term.freq$frequency), 2)
```

[1] 0.18

First wordcloud

After translation to english, we obtain this first wordcloud:

```
library(wordcloud)
wordcloud(words = term.freq$term.en, freq = term.freq$frequency, max.words = 100,
          random.order = F, random.color = T, colors = brewer.pal(8,"Dark2"), scale = c(4, 0.5))
```

Specific terms

- When an illustrative variable is available to describe documents, we can use it to identify *specific terms* of each category.
- These terms are often more relevant than high-frequency based terms to caracterize the corpus
- *Specific terms* are terms which are *over-represented* on a category compared to what could be expected with a uniform distribution
- Two options:
  - *specificTerms() function in the* ***RcmdrPlugin.temis*** *package, relying on a statistical test based on hypergeometric distribution*
  - *comparison.cloud() function in the* ***wordcloud*** *package (next slide)*
- In our case, we will extract specific terms of each **industry**

Comparison Cloud (1/2)

- *comparison.cloud()* function in the **worcloud** package compares the frequencies of words across documents:
  - *Let $p_{i,j}$ be the rate at which word i occurs in document j, and p_i be the average across documents: $\sum_j p_{i,j}/ndocs$*
  - *The size of each word is mapped to its maximum deviation: $max_j(p_{i,j} - p_i)$*
  - *Its angular position is determined by the document where that maximum occurs.*
- For each industry, we will paste all documents into a single one in order to compare industries:

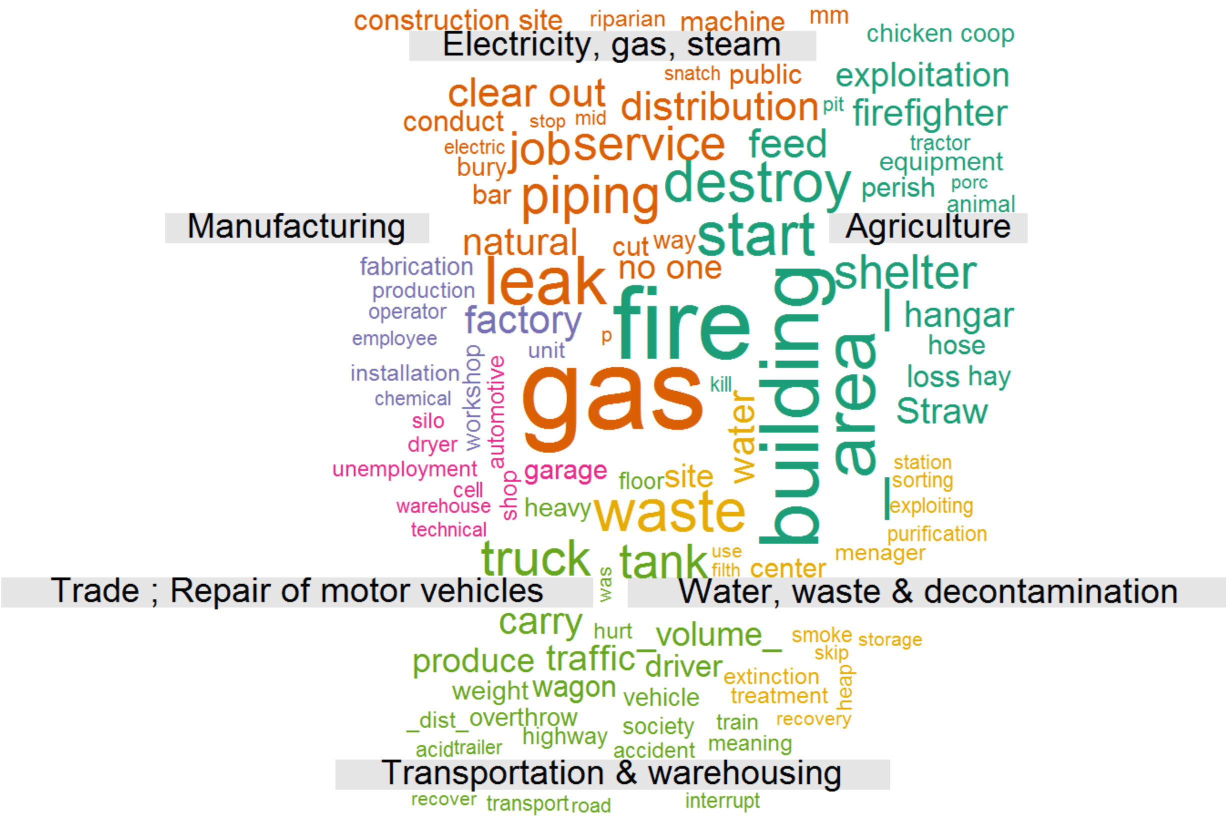
```
freqInd
```

```
## [1] "Manufacturing"                    "Transportation & warehousing"
## [3] "Agriculture"                      "Trade ; Repair of motor vehicles"
## [5] "Water, waste & decontamination"   "Electricity, gas, steam"
```

```
# Matrix agregated by category
comp.matrix <- slam::rollup(TDMlm[, which(acc$Industry %in% freqInd)], 2,
                            acc$Industry[which(acc$Industry %in% freqInd)])
comp.matrix <- as.matrix(comp.matrix)
```

Comparison Cloud (2/2)

```
comparison.cloud(comp.matrix, max.words=150, random.order=FALSE, title.size = 1.2, scale = c(4, 0.5))
```



Commonality Cloud

We can also look at the cloud of words shared across industries:

```
commonality.cloud(comp.matrix, max.words=100, random.order=FALSE, scale = c(4, 0.5),
                  colors = colorRampPalette(c("grey","black"))(40))
```

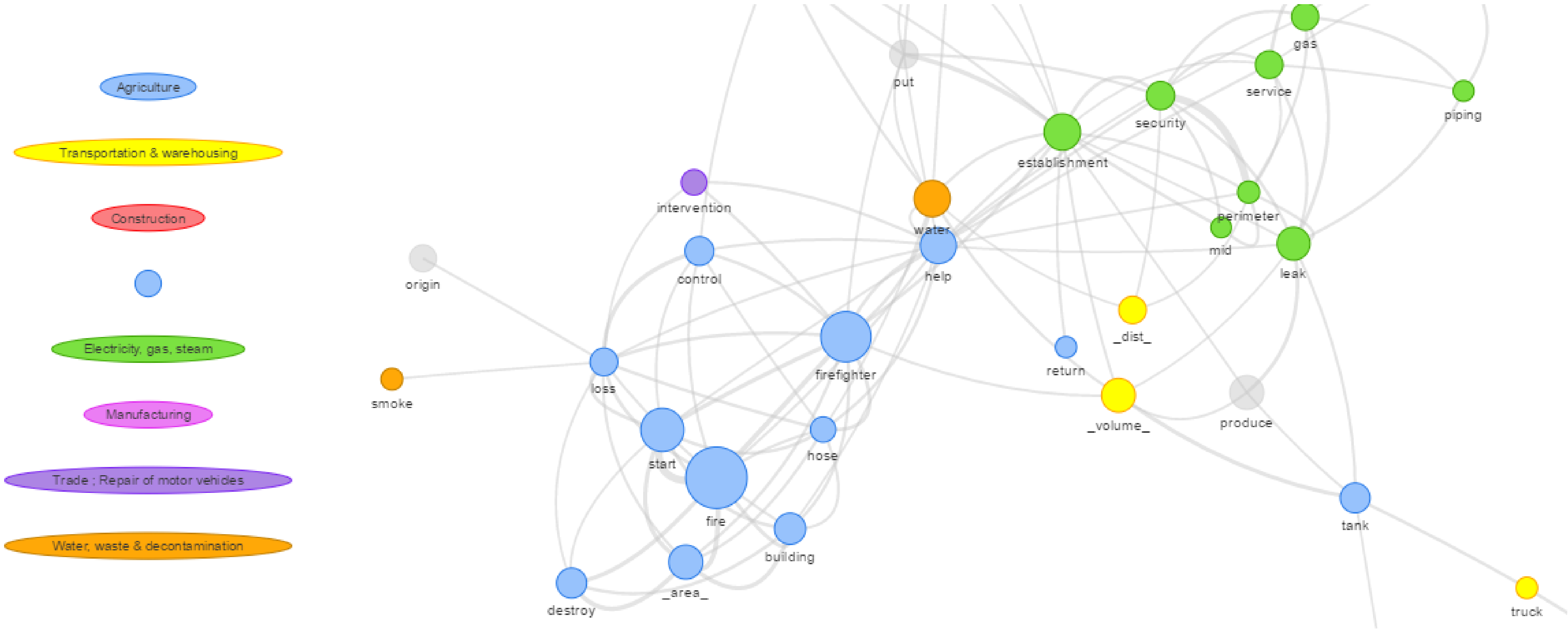

```
# size is correlated to the minimum frequency across industries
```

Co-occurrence network (1/2)

- **visNetwork** package
- Terms are represented by **nodes**:
  - *size is proportional to the number of documents where the term occurs*
  - *a term does not appear under a fixed threshold*
  - *term is affected with a color to the industry where it has the highest specificity score*
- Relations between terms are represented by **edges**:
  - *width is proportional to the Jaccard similarity between the two terms*
  - *an edge does not appear under a fixed threshold*

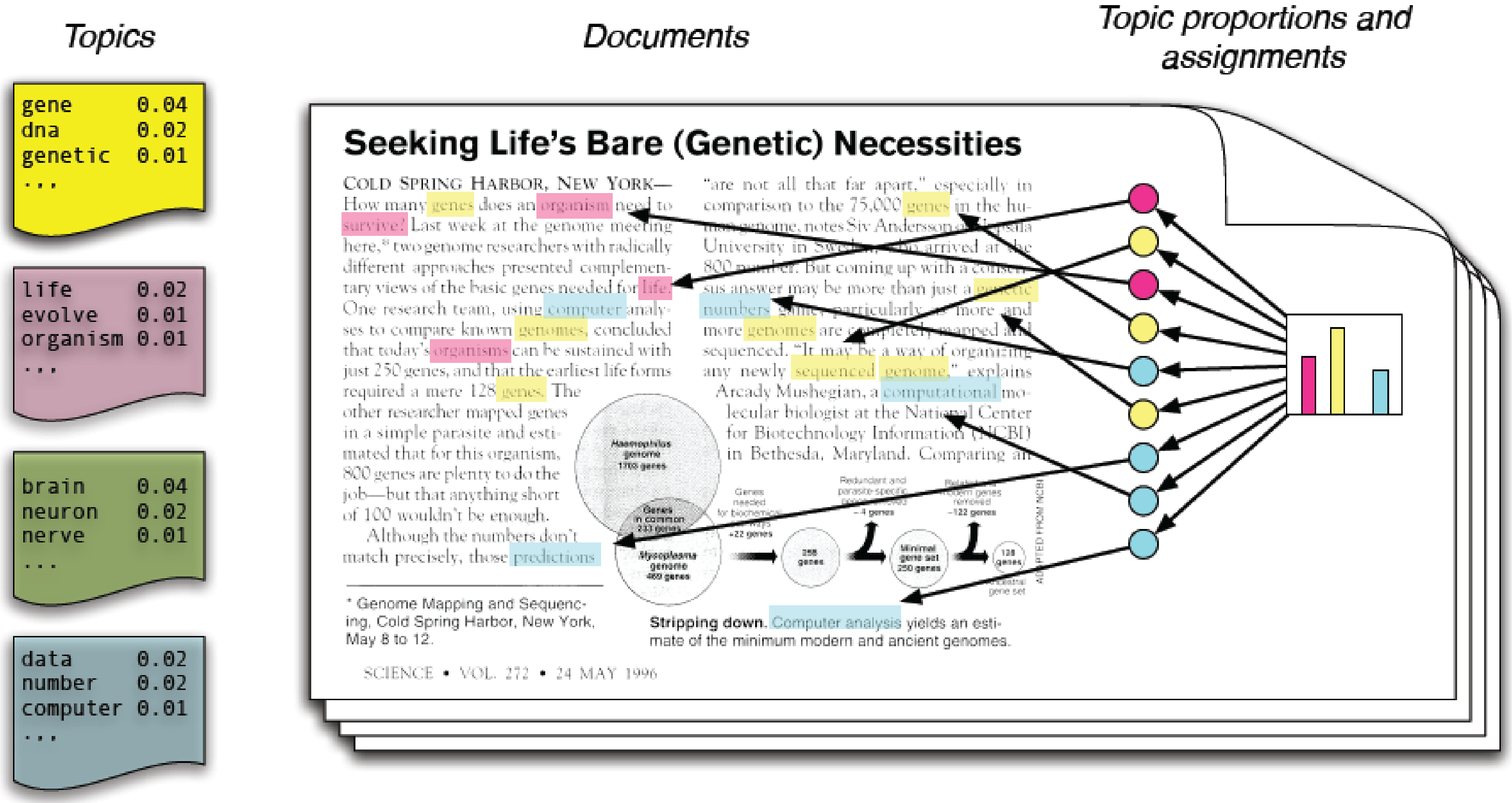
Co-occurrence network (2/2)

```r
source("D:/dt/Documents/R projects/rfunctions/textmining/plot_words.R")
plot.words(TDMlmt, nodeMinFreq = 2400, edgeMinSim = 0.18, df.group = word.ind)
```



Topic Modeling

- Topic modeling refers to algorithms which allow to discover the main “topics” (themes) in a large collection of documents
- It provides a quick way to perform *unsupervised classification* on documents
- Key assumptions: *bag of words* concept and documents are not ordered
- The *Latent Dirichlet Allocation* model (2) is a Bayesian mixture model for discrete data where topics are assumed to be uncorrelated
- The *Correlated Topics Model* (3) is an extension of the original LDA model where correlations between topics are allowed: that we will use


Topics
Documents
Topic proportions and assignments
gene 0.04
dna 0.02
genetic 0.01
...
life 0.02
evolve 0.01
organism 0.01
...
brain 0.04
neuron 0.02
nerve 0.01
...
data 0.02
number 0.02
computer 0.01
...
Seeking Life's Bare (Genetic) Necessities
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.
SCIENCE • VOL. 272 • 24 MAY 1996

Topic Modeling: perplexity score

- Perplexity is often used to evaluate the models on held-out data
- Perplexity for a test set of documents d_t is given by (4):

$$Perplexity(d_t) = \exp(-\frac{\log(p(d_t))}{\text{count of tokens}})$$

where $\log(p(d_t))$ is the likelihood of unseen documents

- The lower the perplexity, the “better” the model

Topic Modeling from a practical point of view

- **k**, the number of topics that the algorithm use to classify documents has to be fixed *a priori*: the main difficulty of these algorithms
- One option is to minimise *perplexity* by cross validation, but it does not systematically give a semantically meaningful choice of **k**
- From a practical point of view, we can simply run the algorithm for different values of **k** and make a choice by inspecting the results
- Topic models have successfully been applied to article databases to identify similar articles and group articles by theme as part of search engine queries
- However, the topic model fit does not return an actual *topic* (term/phrase) on the basis of documents that are clustered together: it must be determined **subjectively** by the analyst

CTM: first attempt (1/2)

We will choose the number of topics *k* according to *perplexity* score on a test sample:

```
DTMlm <- weightTf(as.DocumentTermMatrix(TDMlm))
# Removing empty documents from DTM
DTMnz <- DTMlm[which(slam::row_sums(DTMlm) > 0), ]

# for the choice of k relatively to perplexity score
set.seed(1110)
test.ind <- sample(1:nrow(DTMnz), round(0.2*nrow(DTMnz)))
DTMlearn <- DTMnz[-test.ind, ]
DTMtest <- DTMnz[test.ind, ]

library(topicmodels)
SEED <- 1110
df.perp.1 <- data.frame()
for (K in c(2,5,10)){
  assign(paste0("CTM_", K), CTM(DTMlearn, k = K,
                                control = list(seed = SEED, var = list(tol = 10^-4), em = list(tol = 10^-3))))
  CTM_temp <- eval(parse(text = paste0("CTM_", K)))

  # Perplexity
  perp <- perplexity(CTM_temp, DTMtest)
  df.perp.1 <- rbind(df.perp.1, data.frame(k = K, perplexity = perp))

  # Ten most frequent terms for each topic
  assign(paste0("CTM_", K, ".terms"), terms(CTM_temp, 5))
}
```

CTM: first attempt (2/2)

```r
# Topics keywords
CTM_2.terms
```

```
##      Topic 1      Topic 2    
## [1,] "eau"        "incendie" 
## [2,] "feu"        "pompier"  
## [3,] "exploitant" "_volume_" 
## [4,] "securite"   "fuite"    
## [5,] "site"       "declarer" 
```

```r
CTM_5.terms
```

```
##      Topic 1      Topic 2    Topic 3        Topic 4    Topic 5   
## [1,] "exploitant" "declarer" "eau"          "feu"      "_volume_"
## [2,] "effectuer"  "pompier"  "incendie"     "batiment" "pompier" 
## [3,] "place"      "incendie" "feu"          "gaz"      "incendie"
## [4,] "feu"        "fuite"    "pompier"      "fuite"    "evacuer" 
## [5,] "eau"        "_area_"   "intervention" "site"     "gaz"     
```

```r
CTM_10.terms
```

```
##      Topic 1      Topic 2    Topic 3   Topic 4        Topic 5       
## [1,] "exploitant" "pompier"  "eau"     "site"         "unite"       
## [2,] "vanne"      "incendie" "silo"    "installation" "accident"    
## [3,] "operateur"  "eteindre" "site"    "usine"        "usine"       
## [4,] "pression"   "fumee"    "cellule" "exploitant"   "declencher"  
## [5,] "incident"   "secours"  "dechet"  "local"        "installation"
##      Topic 6    Topic 7        Topic 8     Topic 9     Topic 10  
## [1,] "incendie" "exploitant"   "eau"       "fuite"     "produire"
## [2,] "feu"      "electrique"   "_volume_"  "gaz"       "pompier" 
## [3,] "declarer" "pompier"      "pollution" "citerne"   "origine" 
## [4,] "_area_"   "installation" "polluer"   "securite"  "_dist_"  
## [5,] "batiment" "entreprise"   "_dist_"    "perimetre" "explosion"
```

- The same keywords appear in almost all the topics -> these are very frequent words

CTM: TF-IDF filtering (1/2)

We have previously filtered sparse words, but we will also filter words with a low TF-IDF:

```
term.tfidf <- tapply(DTMnz$v/row_sums(DTMnz)[DTMnz$i], DTMnz$j, mean) * log2(nDocs(DTMnz)/col_sums(DTMnz > 0))

# TF-IDF distribution
quantile(term.tfidf, probs = seq(0,1,0.1))
```

```
##         0%        10%        20%        30%        40%        50%
## 0.04844971 0.10241991 0.11880383 0.13563971 0.15248231 0.17181121
##        60%        70%        80%        90%       100%
## 0.19219844 0.21726000 0.25672846 0.33215346 1.93718632
```

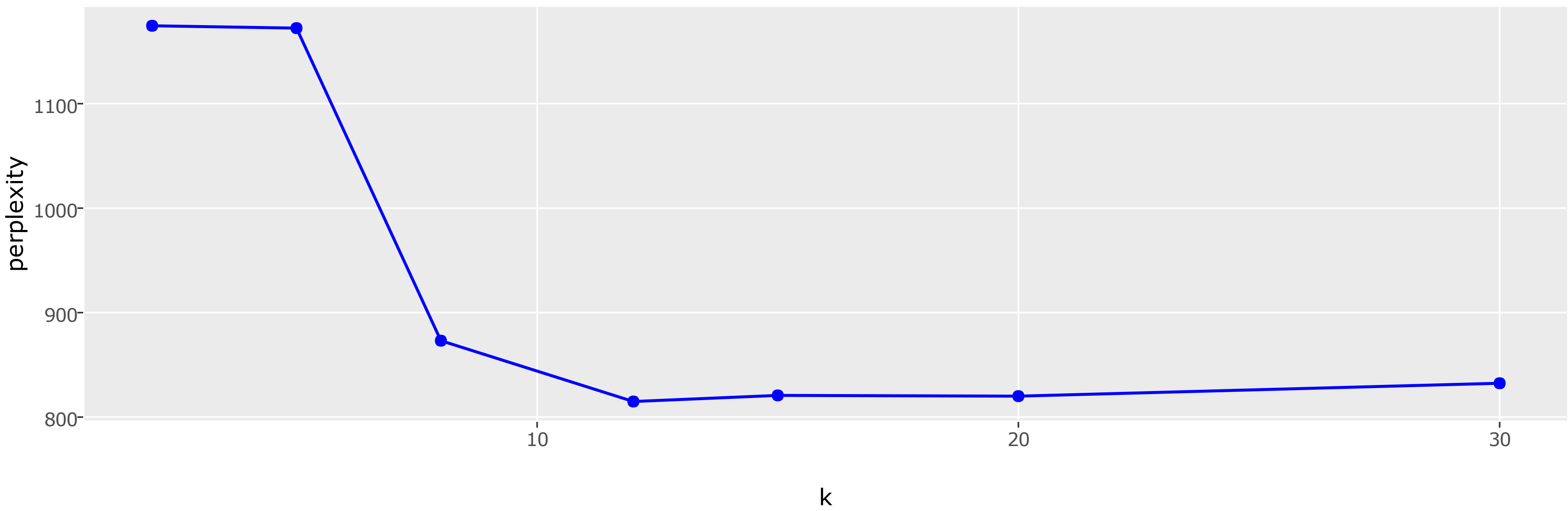
```
# TF-IDF cutting
DTMtfidf <- DTMnz[, term.tfidf >= 0.13]
DTMtfidf <- DTMtfidf[which(slam::row_sums(DTMtfidf) > 0), ]

# for the choice of k relatively to perplexity score
set.seed(1110)
test.ind <- sample(1:nrow(DTMtfidf), round(0.2*nrow(DTMtfidf)))
DTMtfidf_learn <- DTMtfidf[-test.ind, ]
DTMtfidf_test <- DTMtfidf[test.ind, ]
```

CTM: TF-IDF filtering (2/2)

```r
df.perp <- data.frame()
for (K in c(2, 5, 8, 12, 15, 20, 30)){
  assign(paste0("CTM_tfidf_", K), CTM(DTMtfidf_learn, k = K,
                                     control = list(seed = SEED, var = list(tol = 10^-4), em = list(tol = 10^-3))))

  CTM_temp <- eval(parse(text = paste0("CTM_tfidf_", K)))

  # Probability of assigment to the most likely topic
  assign(paste0("CTM_tfidf_", K, ".probs"), apply(posterior(CTM_temp)$topics, 1, function(x) x[which.max(x)]))

  # Perplexity
  perp <- perplexity(CTM_temp, DTMtfidf_test)
  df.perp <- rbind(df.perp, data.frame(k = K, perplexity = perp))
}
```

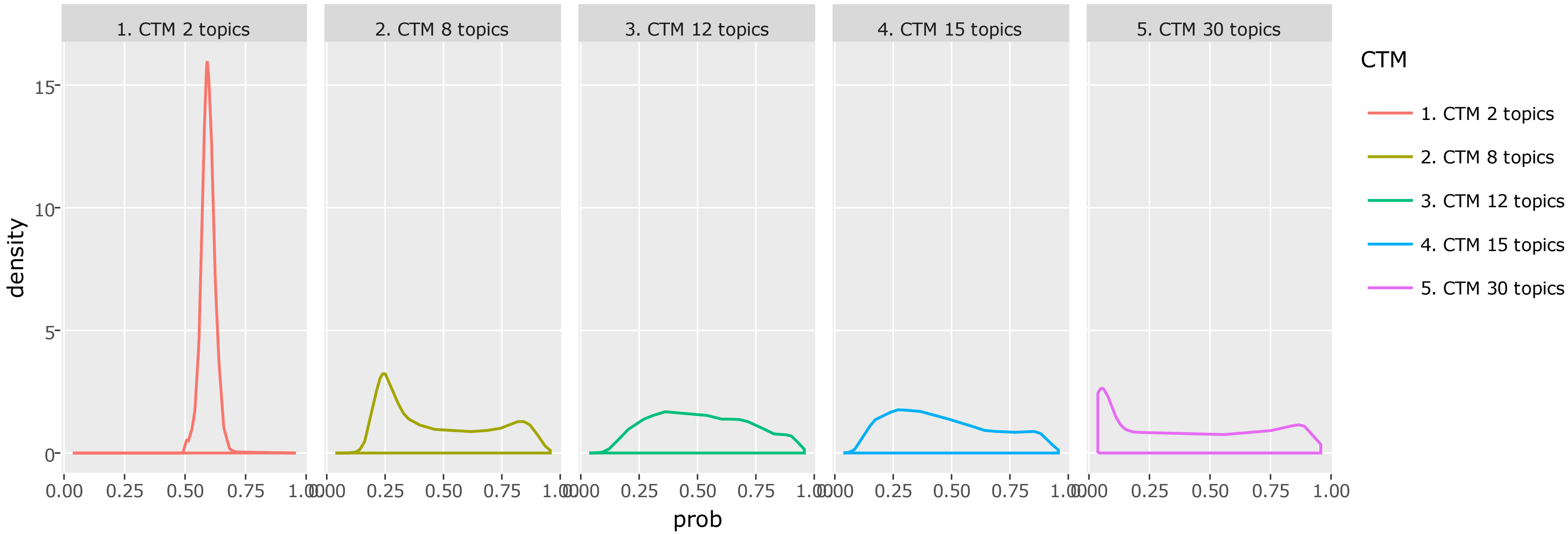
CTM: perplexity according to the number of topics

```
p <- ggplot(df.perp, aes(x = k, y = perplexity)) + geom_point(colour = I("blue")) +
  geom_line(colour = I("blue")) + theme(plot.margin = unit(c(1,1,1,1), "cm"))
ggplotly(p)
```



CTM: proba of assignment to the most likely topic

```r
ndocs <- length(CTM_tfidf_2.probs)
probs <- data.frame(CTM = rep(c("1. CTM 2 topics", "2. CTM 8 topics", "3. CTM 12 topics",  "4. CTM 15 topics",
                               "5. CTM 30 topics"), each = ndocs),
                    prob = c(CTM_tfidf_2.probs, CTM_tfidf_8.probs, CTM_tfidf_12.probs, CTM_tfidf_15.probs, CTM_tfidf_30.probs))
# comparing densities according to the number of topics
g <- qplot(x = prob, data = probs, geom = "density", col = CTM, facets = . ~ CTM)
ggplotly(g)
```



CTM: proba distribution according to the topic (1/2)

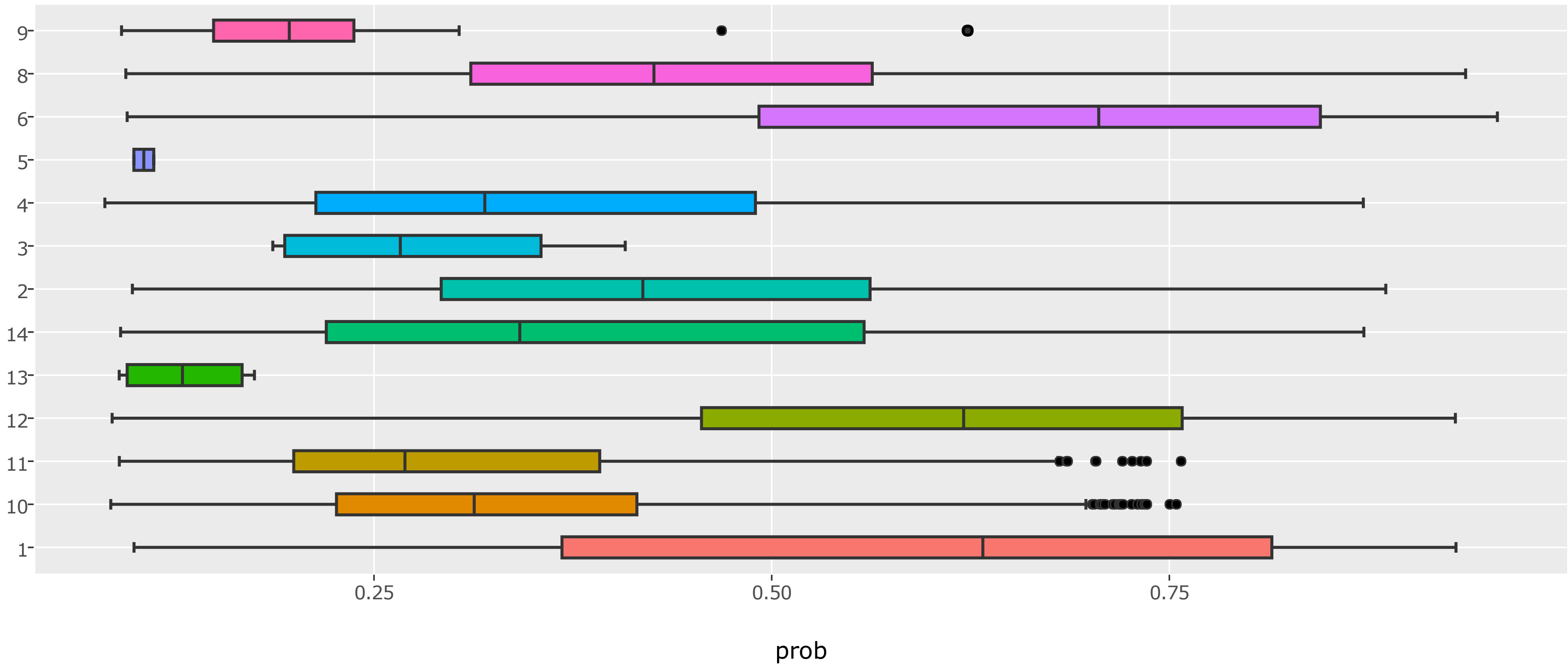
```
# Choice of the number of topics
CTM_final <- CTM(DTMtfidf, k = 15, control = list(seed = SEED, var = list(tol = 10^-4), em = list(tol = 10^-3)))
CTM_final.topics <- topics(CTM_final, 1)
table(CTM_final.topics)
```

```
## CTM_final.topics
##    1    2    3    4    5    6    8    9   10   11   12   13   14
## 3144 2243    6  756    2 3073 2374   39 9241 1342 4809    4 1209
```

```
CTM_final.probs <- apply(posterior(CTM_final)$topics, 1, function(x) x[which.max(x)])

df.topic <- data.frame(topic = as.character(CTM_final.topics),
                       ID = as.numeric(names(CTM_final.topics)),
                       prob = CTM_final.probs)
```

CTM: proba distribution according to the topic (2/2)

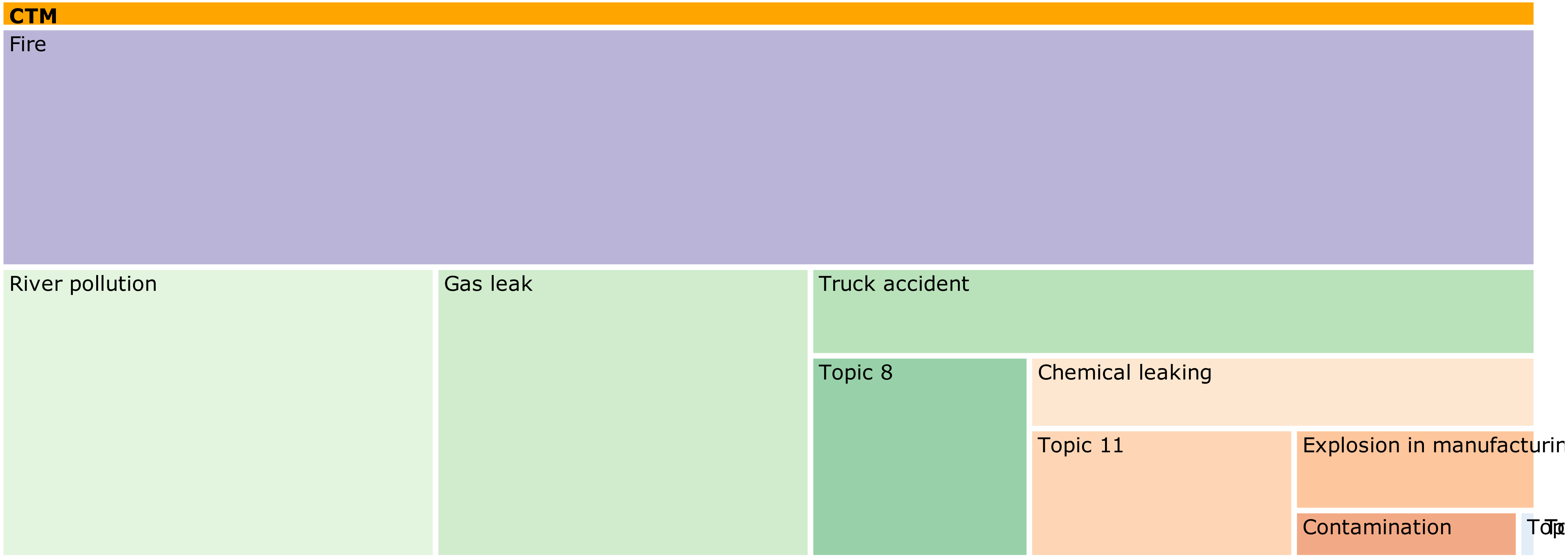
```
# compare proba distribution between topics
p <- ggplot(aes(x=topic, y=prob, fill=topic), data=df.topic) + geom_boxplot(width = 0.5) +
  coord_flip() + theme(legend.position = "none", plot.margin = unit(c(1,1,1,1), "cm")) + xlab("")
ggplotly(p)
```



CTM: topic description (1/2)

```
library(D3partitionR)
D3partitionR(data = treemap_list, type="treeMap", width = 1100,
             labelStyle = "font-size: 11pt",
             tooltipOptions = list(showAbsolutePercent=FALSE,showRelativePercent=FALSE))
```

[1] TRUE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE [1] FALSE



CTM: topic description (2/2)

```r
# most representative documents for selected topics
library(DT)
datatable(doc.repr[, c("lab", "prob", "doc.en")], options = list(lengthMenu = c(2, 4, 6)))
```

Show 2 entries Search:

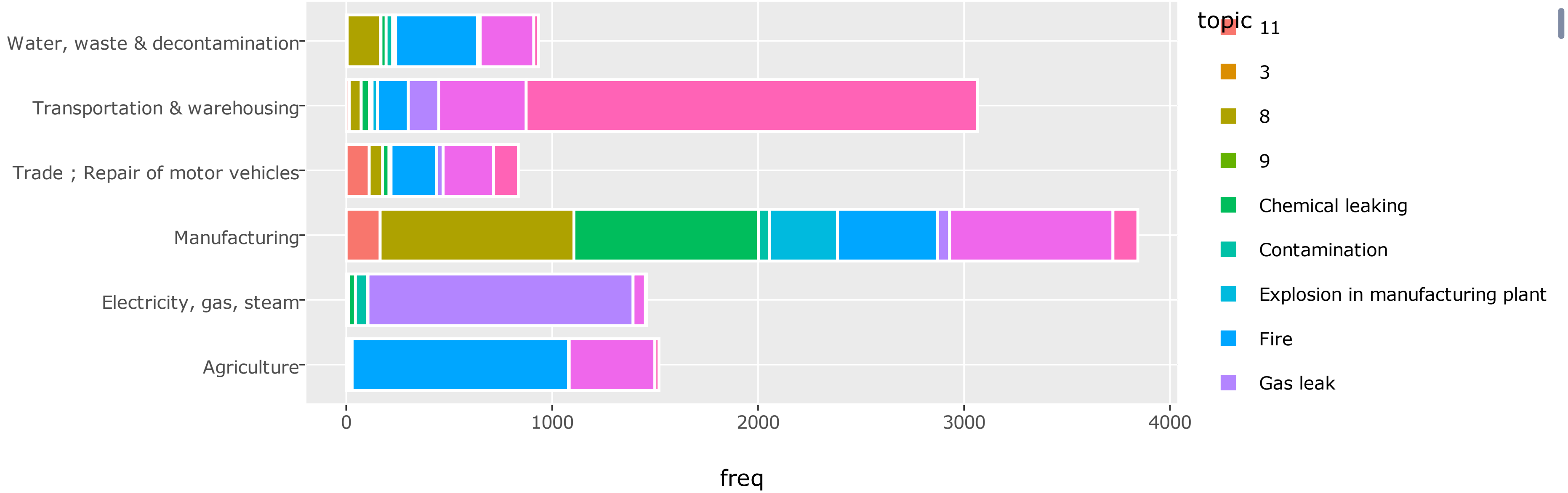
	lab	prob	doc.en
1	Truck accident	0.96	At about 1700 hours, a Belgian tanker carrying 23,000 liters of cobalt chloride spilled at PK 214 on the A28 in the direction of Le Havre-Bassens, terminating its course on the emergency stopband and the ditch. The unscrupulous driver calls for help. The tank has a slight leak at a manhole. The gendarmes, 19 firemen and 2 employees of the motorway operator intervene. Traffic on the North-South pavement is deflected, a safety perimeter is set up and absorbent is spread to recover the product. The accident vehicle is deposited in a tank of the carrier from Belgium. The truck is then raised. Traffic is restored on one lane at 7:15 am and totally at 10:30 am. The inattention of the driver is at the origin of the accident internal formation of his drivers.
2	Truck accident	0.95	A tanker transporting 6 t of liquefied propane reverses itself at 9 am on its delivery route in a ditch on the D 21 road. A slight leak is observed on a bridle. At 09:15, the driver temporarily clogs the leak with water and a rag. A safety perimeter of 150 m is established and the gendarmes stop traffic at 10:10 am for 4 hours. Relief clogs the leak with a plug of ice. The truck is picked up at 2:20 pm with 2 cranes and, after explosive measurements, is allowed to return to its loading site where it will be degassed. The driver was traveling at 10 km / h on a sloping road with an icy roadway. The vehicle slipped on an ice sheet and departed towards the edge of the road.

Showing 1 to 2 of 6 entries

Previous 1 2 3 Next

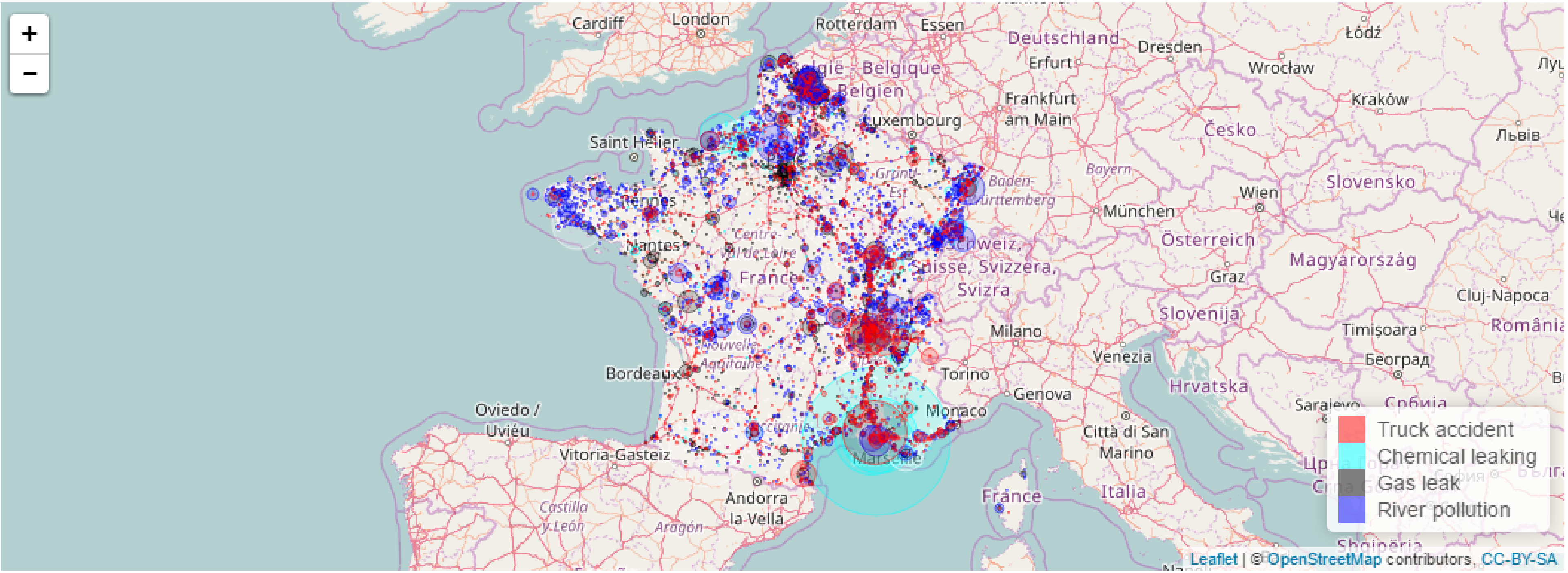
CTM: topic distribution per industry

```
ind.topic.freq <- doc.topic %>% filter(Industry %in% freqInd) %>%
  group_by(topic, Industry) %>% summarize(freq = n()) %>%  as.data.frame()

g <- ggplot(aes(x = Industry, y = freq), data = ind.topic.freq) +
  geom_bar(aes(fill = topic), color = "white", stat = "Identity") +
  coord_flip() + theme(plot.margin = unit(c(1,1,1,1), "cm")) + xlab("")
ggplotly(g)
```



CTM: topics over French territory

```
m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addCircleMarkers(lng=city.topic.freq$long, lat=city.topic.freq$lat,
                   popup = paste(city.topic.freq$CITY, ": ", city.topic.freq$freq),
                   radius = city.topic.freq$freq/2, weight = 1, color = city.topic.freq$color) %>%
  addLegend("bottomright", colors = c("red", "cyan", "black", "blue"),
            labels = c('Truck accident', 'Chemical leaking', 'Gas leak', 'River pollution'))
m  # Print the map
```



Causes analysis: preprocessing

- To analyse precisely the events that have been the causes of these accidents, we will create a new corpus:
  - *For the sake of clarity, we reduce the perimeter to **Manufacturing** industry*
  - *We cut reports into sentences thanks to annonate() function in **NLP** package*
  - *After examining the reports, we extract sentences relative to causes thanks to regular expressions:*
    - sentence contains *cause*
    - sentence contains *origin*
    - sentence contains *due to*
    - etc.
- We then apply same preprocessing as applied on full reports corpus (except that we lower min frequency threshold from 20 to 10)

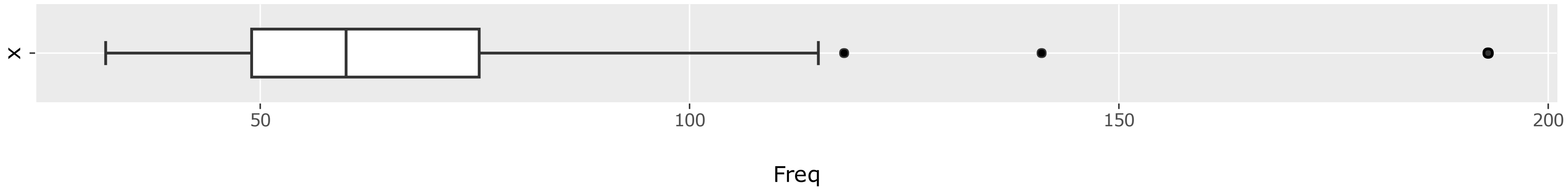
Causes analysis: co-occurrence network

```
plot.words(TDMclmt, nodeMinFreq = 180, edgeMinSim = 0.07)
```

Causes analysis: clustering

- We use *spherical k-means* (5) to perform clustering on documents:

```
library(skmeans)
set.seed(1110)
SKMEANS <- skmeans(coord.doc.fr, k = 100)

# sizes of clusters
p <- ggplot(as.data.frame(table(SKMEANS$cluster)), aes(x="", y=Freq)) + geom_boxplot() + coord_flip()
ggplotly(p)
```

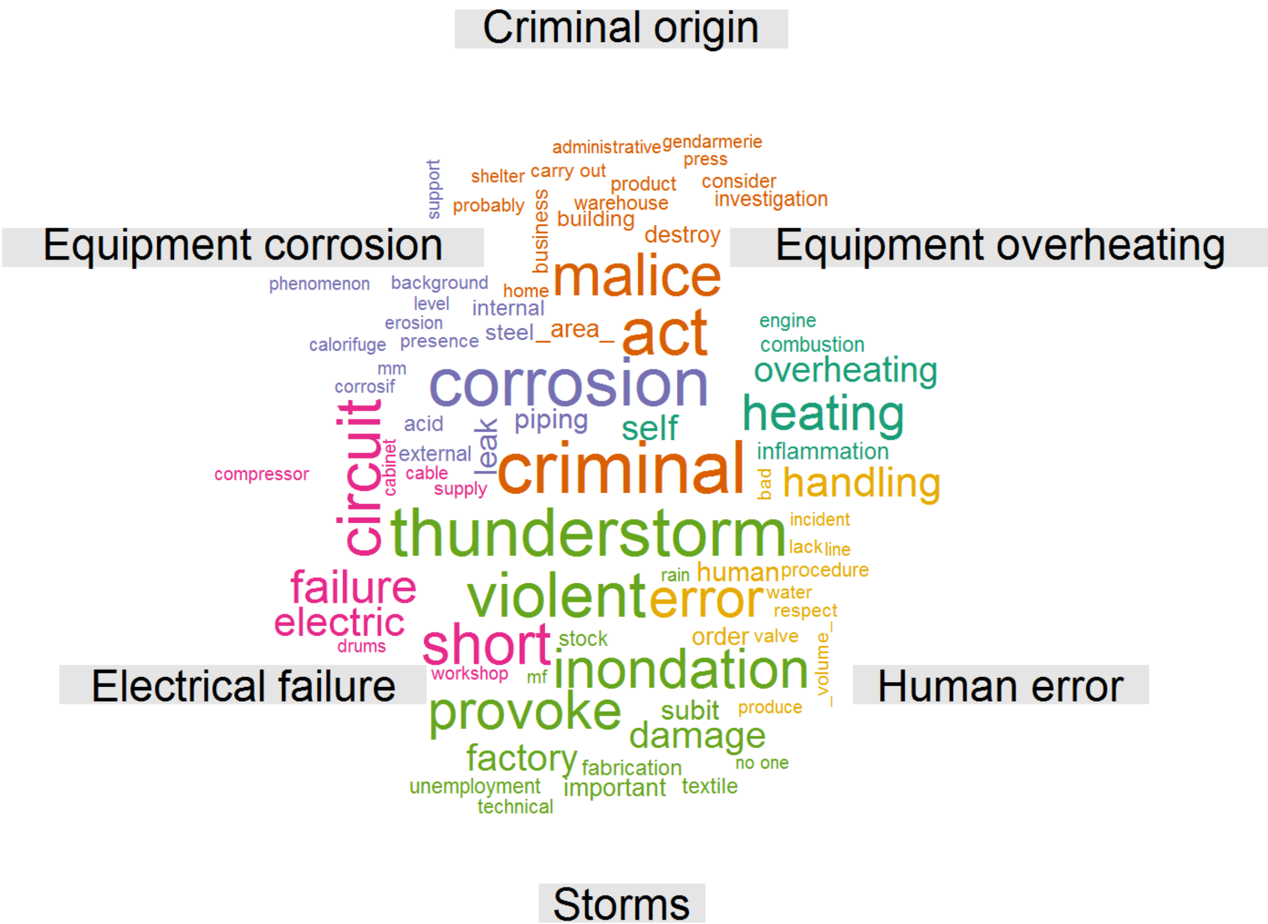



```
# Add clusters to initial data
causes.clus <- merge(df.causes,
                     data.frame(id_sent = names(SKMEANS$cluster), cluster = SKMEANS$cluster),
                     by = "id_sent", sort = F)
```

Causes analysis: presentation of few clusters

We present here some of the causes we identified thanks to the clustering:

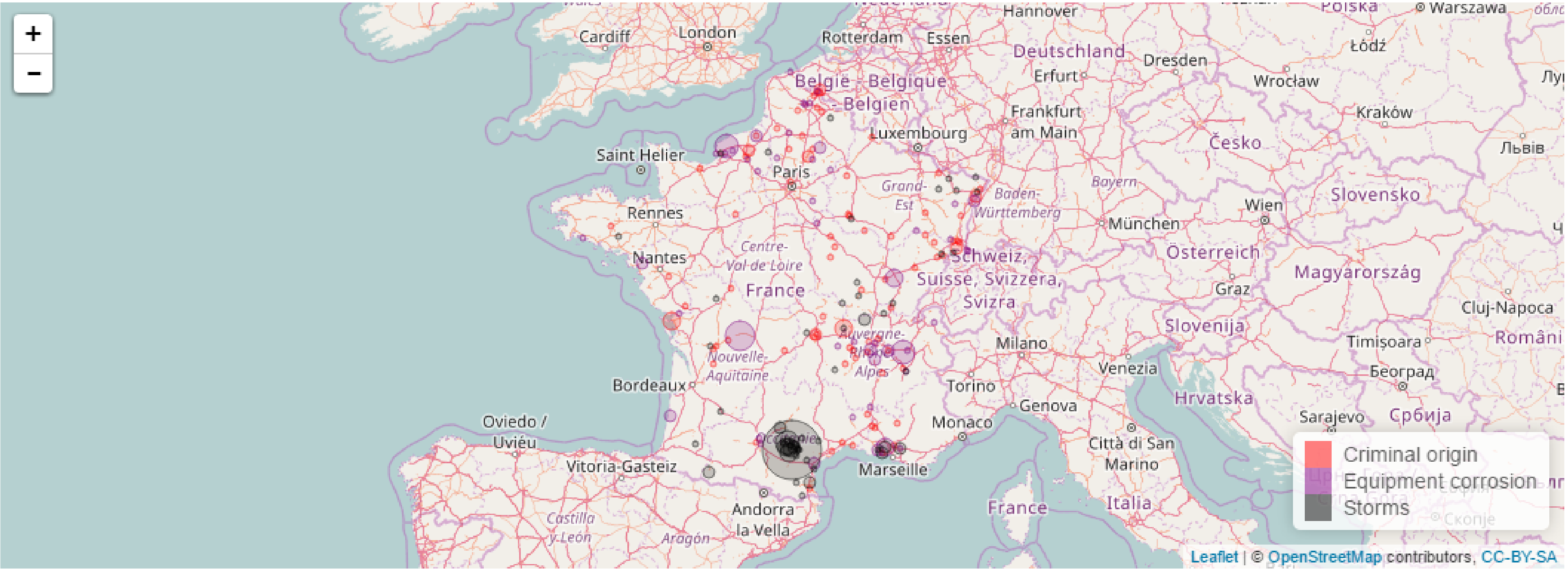
```
# Comparative cloud
causes.matrix <- slam::rollup(TDMclmt, 2, causes.clus$cluster.lab, na.rm = T)
causes.matrix <- as.matrix(causes.matrix)
comparison.cloud(causes.matrix[, clus], max.words=80, random.order=FALSE,
                 title.size = 1.5, colors=c(brewer.pal(8,"Dark2")), scale=c(2.5,0.5))
```



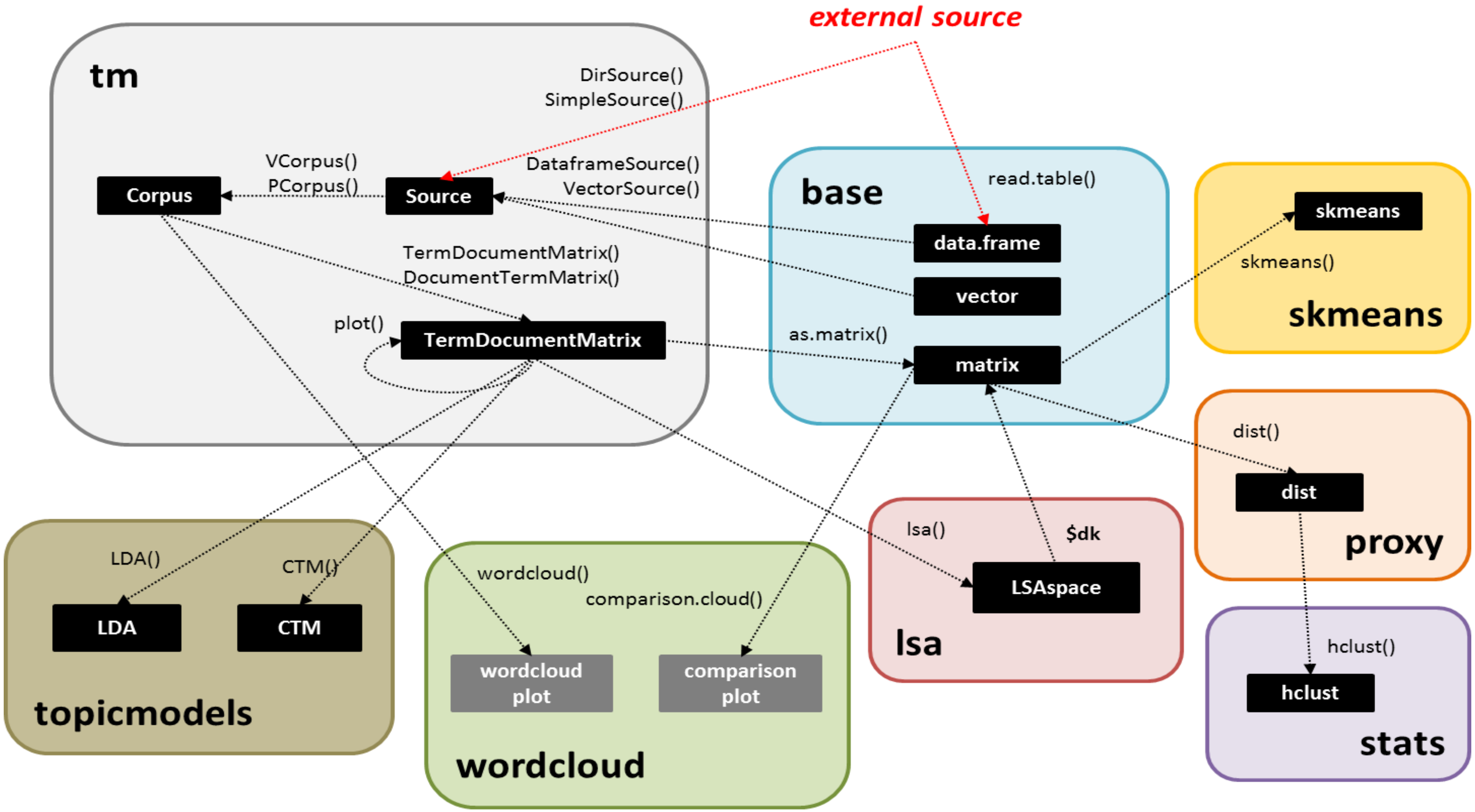
Criminal origin
Equipment corrosion
Equipment overheating
Electrical failure
Human error
Storms
administrative
gendarmerie
press
shelter
carry out
product
consider
investigation
support
probably
business
warehouse
building
destroy
malice
home
act
area
internal
steel
phenomenon
background
level
erosion
calorifuge
presence
mm
corrosif
corrosion
piping
acid
leak
self
engine
combustion
overheating
heating
inflammation
circuit
external
cabinet
cable
supply
compressor
criminal
bad
handling
thunderstorm
incident
lack
line
rain
human
procedure
failure
electric
drums
violent
error
water
respect
short
stock
order
valve
volume
workshop
mf
inondation
provoke
subit
produce
damage
factory
fabrication
no one
unemployment
important
textile
technical

Causes analysis: over French territory

```r
m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addCircleMarkers(lng=city.cause.freq$long, lat=city.cause.freq$lat,
                  popup = paste(city.cause.freq$CITY, ": ", city.cause.freq$freq),
                  radius = city.cause.freq$freq*2, weight = 1, color = city.cause.freq$color) %>%
  addLegend("bottomright", colors = c("red", "purple", "black"),
            labels = c('Criminal origin', 'Equipment corrosion', 'Storms'))
m  # Print the map
```



R packages and their interactions



Conclusion

- We have **described our corpus** of expert reports
- We have explored its content with **topic modeling** to obtain a **clustering of accidents**
- We have then been into further detail with a **clustering of causes**
- The results of this POC are convincing
- After these analyses, we could for example:
  - *create indicators to describe risks and their causes in each type of industry, over the territory*
  - *quantify each cause and cross it with known costs to imagine new prevention services that could interest companies*
- Now **it’s your turn** to try on your corpus!

References

1. Latent Semantic Analysis
   - *Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41(6)*
2. Latent Dirichlet Allocation
   - *Blei DM, Ng AY, Jordan MI (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3*
3. Correlated Topics Model
   - *Blei DM, Lafferty JD (2007). A Correlated Topic Model of Science. The Annals of Applied Statistics, 1(1)*
4. Perplexity
   - *Grun B, Hornik K (2011). Topic models: An R package for fitting topic models. Journal of Statistical Software, 40*
5. Spherical Kmeans
   - *Dhillon IS, Modha DS (2001). Concept Decompositions for Large Sparse Text Data Using Clustering. Machine Learning, 42(1)*

Thank you for your attention!