# Additional file 1

This additional file to ENEV-D-19-00031 includes all code that is referred to in the manuscript. The code is organized into subsections in order to clarify. When using the code, each part highlighted in yellow needs to be filled in by the user (e.g. working directory, API).

## Code to add abstracts to list of articles downloaded from Scopus

This script can be used to retrieve abstract information from Scopus using R. The csv file exported from Scopus needs to be saved as txt file.

*# Packages*

install.packages("rscopus")

install.packages("readxl")

install.packages("foreach")

install.packages("xlsx")

*# Library*

library("rscopus")

library("foreach")

library("readxl")

library("readr")

*# Adding Abstracts to cleaned excel file from scopus (including title, eid etc.) and save the new file as csv in working directory*

*# read txt file from Scopus and combine*

Search\_1a <- read.delim("output file Scopus ", sep = "\t")

Search\_1b <- read.delim("output file Scopus ", sep = "\t")

Search\_1 <- rbind(Search\_1a, Search\_1b)

Search\_1 <- subset(Search\_1, !duplicated(Search\_1$EID)) #27,517 unique EIDs

*# Converting Year to a factor and EID, authors, title, doi, link to character*

Search\_1 <- within(Search\_1, {

 Year <- factor(Year)

 EID <- as.character(EID)

 Authors <- as.character(Authors)

 Title <- as.character(Title)

 DOI <- as.character(DOI)

 Link <- as.character(Link)

})

*# Adding column named Abstract to the dataframe Search\_1\_no\_author*

Search\_1["Abstract"] <- ""

*# Add abstract to article using the Rscopus package*

*# You need to change api\_key after 10,000 records*

set\_api\_key("YOUR API KEY") *# Set API key*

x <- abstract\_retrieval("2-s2.0-85038924323", identifier = "eid", verbose = TRUE) *# TEST!*

x <- x[["content"]][["abstracts-retrieval-response"]][["coredata"]][["dc:description"]] *# Now, x contains the abstract for EID =* 2-s2.0-85038924323

*# Make a list of the EIDs*

Search\_1$EID <- as.character(Search\_1$EID)

list.eid <- with(Search\_1, c(EID[1:length(EID)]))

*# Find abstract for each EID*

abstract<- function(x) {if (!is.null(x)) paste(x[["content"]][["abstracts-retrieval-response"]][["coredata"]][["dc:description"]])}

*# abstract\_retrieval returns list of elements, function abstract returns only abstract information*

Search\_1$Abstract[1:9000] <- lapply((lapply(list.eid[1:9000], abstract\_retrieval, identifier = "eid")), abstract)

*# after 9000 records, we change the API*

*# Save dataframe in between adding abstracts*

Search\_1\_incl\_Abstracts <- as.data.frame(Search\_1)

Search\_1\_incl\_Abstracts <- apply(Search\_1\_incl\_Abstracts ,2,as.character)

write.table(Search\_1\_incl\_Abstracts, file = " ")

set\_api\_key("YOUR API KEY") # Set API key

Search\_1$Abstract[9001:18000] <- lapply((lapply(list.eid[9001:18000], abstract\_retrieval, identifier = "eid")), abstract) *# 18000records, change API*

*# Save dataframe in between adding abstracts*

Search\_1\_incl\_Abstracts <- as.data.frame(Search\_1)

Search\_1\_incl\_Abstracts <- apply(Search\_1\_incl\_Abstracts ,2,as.character)

write.table(Search\_1\_incl\_Abstracts, file = “How and where to save the file?” sep = "\t")

set\_api\_key("YOUR API KEY ") *# Set API key*

Search\_1$Abstract[18001:22570] <- lapply((lapply(list.eid[18001:22570], abstract\_retrieval, identifier = "eid")), abstract)

*# Save*

Search\_1\_incl\_Abstracts <- as.data.frame(Search\_1)

Search\_1\_incl\_Abstracts <- apply(Search\_1\_incl\_Abstracts ,2,as.character)

write.table(Search\_1\_incl\_Abstracts, = “How and where to save the file?”, sep = "\t")

## Code for primary study selection using data driven clustering with Adjutant

Import file can be any other exported txt file from Scopus. Each line starting with an # is an explanatory line to explain the code that comes after the line. The # is necessary, as when the script is put into the R console, it will be recognized as not part of the code.

*# Install packages*

options(repos= "http://cran.rstudio.com/") *# to avoid error see https://stackoverflow.com/questions/25599943/unable-to-install-packages-in-latest-version-of-rstudio-and-r-version-3-1-1*

install.packages("devtools")

install.packages("rscopus")

install.packages("readxl")

install.packages("foreach")

install.packages("xlsx")

install.packages("tm", dependencies=TRUE, repos="http://cran.rstudio.com/")

install.packages("tidytext", dependencies=TRUE, repos='http://cran.rstudio.com/')

install.packages("stringi")

install.packages('ggplot2', dep=TRUE, lib=NULL)

install.packages("doBy") # keeping only unique EIDs when combining search 1,2,3

install.packages("SnowballC", dependencies=TRUE, repos='http://cran.rstudio.com/') # wordStem

install.packages("dbscan", dependencies=TRUE, repos='http://cran.rstudio.com/')

*# Library*

library("foreach")

library("readxl")

library("devtools")

library("tm")

library("tidytext")

library("stringi")

library("dplyr") # need this package for inner\_join function

library("ggplot2")

library("doBy")

library("stringr", lib.loc="~/R/R\_library") # function str\_length needed for tidycorpus

library("SnowballC")

library("dbscan")

*# Install Adjutant*

devtools::install\_github("amcrisan/Adjutant")

*# Get R ready to run Adjutant*

library(adjutant)

*# Set a seed - there is some randomness in the analysis*

set.seed(416)

*# Because Adjutant is developed for Pubmed search results and not for Scopus, we will not use the interface but work with a corpus*

*# More information on Adjutant: https://github.com/amcrisan/Adjutant#download*

*# Import dataset into R*

calibration\_1 <- read.delim(“Open text file from Scopus”, sep = "\t")

*# Add extra column PMID, but fill it with EID. Also, make title and abstract a character variable in order to match the output style of pubmed*

calibration\_1 <- within(calibration\_1,{

 PMID <- as.factor(EID)

 Title <- as.character(Title)

 Abstract <- as.character(Abstract)

})

*# Clean abstract & remove words*

# *1. Some abstracts contain Â© 2018, 2017 etc. --> remove*

Abstract <- as.list(calibration\_1$Abstract)

*# TidyCopus will remove any number, but does apparently not recognize these patterns*

Abstract\_cleaned <- sapply(Abstract, gsub, pattern= "Â© 2018|Â© 2017|Â© 2016|Â© 2015|Â© 2014|Â© 2013|Â© 2012|Â© 2011|Â© 2010|Â© 2009|Â© 2008|Â© 2007|Â© 2006|Â© 2005|Â© 2004|Â© 2003", replacement = "")

*# 2. Some abstracts might be written in sections (lower or higher cases, without spaces in front or after the section title), remove these sections*

Abstract\_cleaned <- sapply(Abstract\_cleaned, gsub, pattern = "INTRODUCTION|BACKGROUND|BACKGROUND AND|PURPOSE|OBJECTIVES|OBJECTIVE|RESULTS|Material|Methods|BACKGROUND AIMS|CONCLUSION|Introduction|Background|Background and|Purpose|Objectives|Objective|Results|Material|Methods|Background aims|Conclusion", replacement = "")

*# 3. Remove most prominent licensing information*

Abstract\_cleaned <- sapply(Abstract\_cleaned, gsub, pattern = "This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.|licensee MDPI, Basel, Switzerland.|Licensee MDPI, Basel, Switzerland.|licensee BioMed Central.|Licensee BioMed Central.| Licensee: AOSIS OpenJournals.|licensee: AOSIS OpenJournals.|licensee Springer.|Licensee Springer.|licensee Chemistry Central Ltd.|Licensee Chemistry Central Ltd.|Elsevier", replacement = "")

*# 4. Remove words from abstract that are too common in this list of records: e.g. study, studies, studied, water etc.*

Abstract\_cleaned<- sapply(Abstract\_cleaned, gsub, pattern = "water|Water |study|studies|studied|species|region|stable", replacement = "")

*# Put the abstract\_cleaned list in Scopus\_output1\_cleaned*

calibration\_1\_cleaned <- calibration\_1

calibration\_1\_cleaned <- within(calibration\_1\_cleaned, {

 rm(Abstract)

 Abstract <- Abstract\_cleaned

})

*# Create tidycorpus and save*

tidy\_corpus1 <- tidyCorpus(corpus = calibration\_1\_cleaned)

*# Step 3: Performing a dimensionality reduction using t-SNE on tidy\_corpus1*

tsneObj <- runTSNE(tidy\_corpus1, check\_duplicates=FALSE)

*# Add t-SNE coordinates to tidy\_corpus1 object*

calibration\_1\_cleaned <- inner\_join(calibration\_1\_cleaned,tsneObj$Y,by="PMID")

*# Plot the t-SNE results*

ggplot(calibration\_1\_cleaned,aes(x=tsneComp1,y=tsneComp2))+

 geom\_point(alpha=0.2)+

 theme\_bw()

*# Step 4:Perform an unsupervised clustering using hdbscan*

*# Run HDBSCAN and select the optimal cluster parameters automatically*

optClusters <- optimalParam(calibration\_1\_cleaned) # explained @ https://rdrr.io/github/amcrisan/Adjutant/man/optimalParam.html

optClusters <- as.data.frame(optClusters["retItems"]) # Need to adjust label, for some reason

optClusters <- within(optClusters, {

 PMID <- retItems.PMID

})

*# Add the new cluster ID’s to the dataset*

calibration\_1\_cleaned <- inner\_join(calibration\_1\_cleaned, optClusters, by = "PMID") %>%

 mutate(tsneClusterStatus = ifelse(optClusters$retItems.tsneCluster == 0, "not-clustered","clustered"))

*# Plot the HDBSCAN clusters (no names yet)*

clusterNames <- calibration\_1\_cleaned %>%

 dplyr::group\_by(retItems.tsneCluster) %>%

 dplyr::summarise(medX = median(tsneComp1),

 medY = median(tsneComp2)) %>%

 dplyr::filter(retItems.tsneCluster != 0)

ggplot(calibration\_1\_cleaned,aes(x=tsneComp1,y=tsneComp2,group= retItems.tsneCluster))+

 geom\_point(aes(colour = tsneClusterStatus),alpha=0.2)+

 geom\_label(data=clusterNames,aes(x=medX,y=medY,label= retItems.tsneCluster),size=2,colour="red")+

 stat\_ellipse(aes(alpha=tsneClusterStatus))+

 scale\_colour\_manual(values=c("black","blue"),name="cluster status")+

 scale\_alpha\_manual(values=c(1,0),name="cluster status")+ #remove the cluster for noise

 theme\_bw()

*# Step 5 Naming the clusters*

clustNames <- calibration\_1\_cleaned %>%

 group\_by(retItems.tsneCluster)%>%

 mutate(tsneClusterNames = getTopTerms(clustPMID = PMID,clustValue= retItems.tsneCluster,topNVal = 2,tidyCorpus= tidy\_corpus1)) %>%

 select(PMID,tsneClusterNames) %>%

 ungroup()

*# Update document corpus with cluster names*

calibration\_1\_cleaned <- inner\_join(calibration\_1\_cleaned,clustNames,by=c("PMID","retItems.tsneCluster"))

*#Re-plot the clusters*

clusterNames <- calibration\_1\_cleaned %>%

 dplyr::group\_by(tsneClusterNames) %>%

 dplyr::summarise(medX = median(tsneComp1),

 medY = median(tsneComp2)) %>%

 dplyr::filter(tsneClusterNames != "Noise")

ggplot(calibration\_1\_cleaned,aes(x=tsneComp1,y=tsneComp2,group=tsneClusterNames))+

 geom\_point(aes(colour = tsneClusterStatus),alpha=0.2)+

 stat\_ellipse(aes(alpha=tsneClusterStatus))+

 geom\_label(data=clusterNames,aes(x=medX,y=medY,label=tsneClusterNames),size=3,colour="red")+

 scale\_colour\_manual(values=c("black","blue"),name="cluster status")+

 scale\_alpha\_manual(values=c(1,0),name="cluster status")+ #remove the cluster for noise

 theme\_bw()

# *Save document with clusterinformation*

write.table(calibration\_1\_cleaned, file = "/s-schijf/hartmanj/Artikel 2/R/Data/P18.txt", sep = "\t")

## Code for primary study selection using number of matching search terms

*# Analyzing relevant clusters using tm package*

*# July 2018, Julia Hartmann*

# Install packages

options(repos= "http://cran.rstudio.com/")

install.packages("tm")

install.packages("corpus")

install.packages("readxl", dependencies = TRUE)

install.packages("tokenizers")

# Attach packages

library("tidyr") # unnest text into sentences

library("stringi") #stri\_enc\_toutf8(str, is\_unknown\_8bit = FALSE, validate = FALSE)

library("tm")

library("readxl")

library("quanteda")

library("tokenizers") #tokenize sentences

library("corpus")

library("dplyr")

library("matrixStats", lib.loc="N:/data/R\_library")

library("data.table")

library("stringr") # for defining pattern to only keep words > 2 characters

*# Load dataset*

calibration\_1 <- read.delim("Scopus output file", sep = "\t")

*# Remove irrelevant columns and create corpus*

calibration\_1\_cleaned <- calibration\_1[ , c("Title","Abstract","EID")]

calibration\_1\_cleaned <- within(calibration\_1\_cleaned, {

 doc\_id <- as.character(EID)

 text <- paste(Title, Abstract, sep = " ")

 rm(Title)

 rm(Abstract)

 rm(EID)

})

calibration\_1\_cleaned <- calibration\_1\_cleaned[, c("doc\_id", "text")] # rearranging columns to be able to run DataframeSource and than Corpus function from tm package

### # 1. GENERAL SORTING BASED ON SEARCH TERMS 1 (quanteda)

*# Only keep the words defined in searchterms*

searchterms\_1 <- c("contaminant", "contaminants", "substance", "substances","compound", "compounds", "chemical", "chemicals", "pathogen", "pathogenic", "pathogens", "pathogenicity", "microorganism", "microorganisms", "infectious disease", "infectious diseases", "virus","viri","bacteria","protozoa","component", "components", "agent", "agents", "metabolite", "metabolites", "transformation product", "transformation products" , "pollutant", "pollutants", "nanoparticle", "nanoparticles", "analysis", "analyzed", "analysed", "analyses", "detect", "detected", "detection", "determinate", "determinated", "determination", "monitor","monitoring","monitored","occurrence","occur", "occurred", "screen", "screening", "screened", "surveilled", "surveillance", "test", "tested", "sensor", "sensoring", "report", "reported", "reporting", "prevalence",

 "identify", "identification", "identified", "recent","novel", "new","emerging", "first", "effluent", "effluents", "surface","surfacewater", "river", "rivers", "lake","lakes", "storm",

 "stormwater","stream","streams", "process water","processwater","treated water","treatedwater","tap","tapwater","drinking" ,"drinkingwater","potable",

 "potablewater","ground","groundwater","fresh","freshwater","river", "rivers", "aquatic", "aquifer","aquifers", "sewage","waste","wastewater","rain","waterborne","catchment", "waterrelated")

calibration\_1\_corpus <- corpus(calibration\_1\_cleaned) # using the quanteda package

*# Tokenize*

calibration\_1\_tokens <- tokens(calibration\_1\_corpus, remove\_punct = TRUE, remove\_separators = FALSE)

*# Analyzing data based on searchterms 1*

# select only relevant relevant searchterms

calibration\_1\_tokens\_S1 <- tokens\_select(calibration\_1\_tokens, searchterms\_1, padding = TRUE) # case\_insensitive = TRUE is default, therefore ignores cases when matching, if TRUE

*# create dfm*

S1\_calibration1\_dfm <- dfm(calibration\_1\_tokens\_S1, tolower = TRUE)

*# create dataframe*

S1\_calibration1\_dfm\_df <- convert(S1\_calibration1\_dfm, to = "data.frame")

S1\_calibration1\_dfm\_df <- within(S1\_calibration1\_dfm\_df, {

 rm(V1)

})

# Do not want a frequency dfm but merely a present/not present document feature matrix

presentnotpresent <- function(x) {ifelse(x >= 1, 1, 0)}

S1\_calibration1\_dfm\_df\_pnp <- as.data.frame(S1\_calibration1\_dfm\_df)

S1\_calibration1\_dfm\_df\_pnp[ , 2:(length( names(S1\_calibration1\_dfm\_df\_pnp)))] <- lapply(S1\_calibration1\_dfm\_df\_pnp[ , c(2:(length( names(S1\_calibration1\_dfm\_df\_pnp))))], presentnotpresent)

*# add column with total count*

S1\_calibration1\_dfm\_df\_pnp\_sum <-within(S1\_calibration1\_dfm\_df\_pnp, {

 rm(document)

}) # making new df without document information, just to fill total\_row

S1\_calibration1\_dfm\_df\_pnp$Total\_Row <- rowSums(S1\_calibration1\_dfm\_df\_pnp\_sum)

S1\_calibration1\_dfm\_df\_pnp <- S1\_calibration1\_dfm\_df\_pnp[order(S1\_calibration1\_dfm\_df\_pnp$Total\_Row),] # order df based on total\_row

summary(S1\_calibration1\_dfm\_df\_pnp$Total\_Row) # max 20

*# Make new df with information including abstract, title, year, author information (?), link, removing terms*

S1\_calibration1\_final <- S1\_calibration1\_dfm\_df\_pnp

S1\_calibration1\_final <- S1\_calibration1\_final[ , -c(2:(length( names(S1\_calibration1\_final))-1))]

S1\_calibration1\_final\_all <- S1\_calibration1\_dfm\_df\_pnp # show all columns, in order to check search terms : why also articles with total\_row =1 this can not be

S1\_calibration1\_final <- merge(x = S1\_calibration1\_final, y = calibration\_1[, c("EID", "Title", "Abstract", "Year", "Authors")], by.x = "document", by.y = "EID")

S1\_calibration1\_final\_all <- merge(x = S1\_calibration1\_final\_all, y = calibration\_1[, c("EID", "Title", "Abstract", "Year", "Authors")], by.x = "document", by.y = "EID")

*# Save dataframe*

write.table(“Where and how to save file”, sep = "\t")

*# What is the distribution of Total\_Row?*

S1\_calibration1\_dfm\_df\_pnp <- within(S1\_calibration1\_dfm\_df\_pnp, {

 Total\_Row <- as.factor(Total\_Row)

})

plot\_calibration1\_searchterms <- plot(S1\_calibration1\_dfm\_df\_pnp$Total\_Row)

### # 3aI. TM: WHICH FEATURES ARE MOST USED BY RELEVANT ARTICLES?

*(https://projets.pasteur.fr/projects/rap-r/wiki/Text\_Mining\_tm\_pipeline\_####*

library("tm")

*# # Load data*

calibration\_1 <- read.delim("scopus output file", sep = "\t")

#*Remove irrelevant columns and create corpus*

calibration\_1\_cleaned <- calibration\_1[ , c("Abstract","EID","Title")]

calibration\_1\_cleaned <- within(calibration\_1\_cleaned, {

 doc\_id <- as.character(EID)

 text <- paste(Title, Abstract, sep = " ")

 rm(Title)

 rm(Abstract)

 rm(EID)

})

calibration\_1\_cleaned <- calibration\_1\_cleaned[, c("doc\_id", "text")] # rearranging columns to be able to run DataframeSource and than Corpus function from tm package

*# Subset only relevant articles*

calibration\_1\_relevant <- calibration\_1\_cleaned [ which(calibration\_1\_cleaned$doc\_id == "2-s2.0-36549047400" | calibration\_1\_cleaned$doc\_id == "2-s2.0-79951811164"

 | calibration\_1\_cleaned$doc\_id == "2-s2.0-40049101875" | calibration\_1\_cleaned$doc\_id == "2-s2.0-54349095319"

 | calibration\_1\_cleaned$doc\_id == "2-s2.0-77957306688" | calibration\_1\_cleaned$doc\_id == "2-s2.0-33646571543"

 | calibration\_1\_cleaned$doc\_id == "2-s2.0-84861863606" | calibration\_1\_cleaned$doc\_id == "2-s2.0-67649506353"

 | calibration\_1\_cleaned$doc\_id == "2-s2.0-76749156103" | calibration\_1\_cleaned$doc\_id == "2-s2.0-54049133601"

 | calibration\_1\_cleaned$doc\_id == "2-s2.0-56549084144" | calibration\_1\_cleaned$doc\_id == "2-s2.0-70350434146"

 | calibration\_1\_cleaned$doc\_id == "2-s2.0-80051667278"| calibration\_1\_cleaned$doc\_id == "2-s2.0-77953018842"), ]

*# 2-s2.0-54049133601 is not part ofcalibration\_1, no mention of a first detection in abstract/title, only full text*

*# create corpus*

calibration\_1\_relevant\_corpus <- Corpus(DataframeSource(calibration\_1\_relevant)) # using the tm package

*# Inspect Corpus*

summary(calibration\_1\_relevant\_corpus)

inspect(calibration\_1\_relevant\_corpus[3])

*# Clean up corpus###*

## remove specific characters

*## to lower case*

calibration\_1\_relevant\_corpus <- tm\_map(calibration\_1\_relevant\_corpus, tolower)

*## remove punctuation*

calibration\_1\_relevant\_corpus <- tm\_map(calibration\_1\_relevant\_corpus, removePunctuation)

*## remove stop words*

calibration\_1\_relevant\_corpus <- tm\_map(calibration\_1\_relevant\_corpus, removeWords,stopwords("english"))

calibration\_1\_relevant\_corpus <- tm\_map(calibration\_1\_relevant\_corpus, removeWords,c("water","study","studies", "paper")) # remove own specific words

*## remove numbers*

calibration\_1\_relevant\_corpus <- tm\_map(calibration\_1\_relevant\_corpus, removeNumbers)

*## Strip white spaces*

calibration\_1\_relevant\_corpus <- tm\_map(calibration\_1\_relevant\_corpus, stripWhitespace)

*# Compute Doc Term Matrix (DTM)*

dtm <- DocumentTermMatrix(calibration\_1\_relevant\_corpus)

*# find frequent terms*

findMostFreqTerms(dtm) # niet handig, te uiteenlopend, zegt niks

table(findMostFreqTerms(dtm))

*# 3bI : FREQUENCY SEARCH TERMS IN RELEVANT DOCS (QUANTEDA PACKAGE)* (https://docs.quanteda.io/articles/pkgdown/examples/plotting.html) ####

*# searchterms*

*# Only keep the words defined in search terms*

searchterms\_1 <- c("contaminant", "contaminants", "substance", "substances","compound", "compounds", "chemical", "chemicals", "pathogen", "pathogenic", "pathogens", "pathogenicity", "microorganism", "microorganisms", "infectious disease", "infectious diseases", "virus","viri","bacteria","protozoa","component", "components", "agent", "agents", "metabolite", "metabolites", "transformation product", "transformation products" , "pollutant", "pollutants", "nanoparticle", "nanoparticles", "analysis", "analyzed", "analysed", "analyses", "detect", "detected", "detection", "determinate", "determinated", "determination", "monitor","monitoring","monitored","occurrence","occur", "occurred", "screen", "screening", "screened", "surveilled", "surveillance", "test", "tested", "sensor", "sensoring", "report", "reported", "reporting", "prevalence",

 "recent","novel","late","new","emerging", "first", "effluent", "effluents", "surface","surfacewater", "river", "rivers", "lake","lakes", "storm",

 "stormwater","stream","streams", "process water","processwater","treated water","treatedwater","tap","tapwater","drinking" ,"drinkingwater","potable",

 "potablewater","ground","groundwater","fresh","freshwater","river", "rivers", "aquatic", "aquifer","aquifers", "sewage","waste","wastewater","rain","waterborne","catchment", "waterrelated")

*# create corpus*

calibration\_1\_relevant\_corpus <- corpus(calibration\_1\_relevant) # using the quanteda package

# *Tokenize*

calibration\_1\_relevant\_tokens <- tokens(calibration\_1\_relevant\_corpus, remove\_punct = TRUE, remove\_separators = FALSE)

### # Analyzing data based on searchterms 1

# select only relevant relevant searchterms

calibration\_1\_relevant\_tokens\_S1 <- tokens\_select(calibration\_1\_relevant\_tokens, searchterms\_1, padding = FALSE) # case\_insensitive = TRUE is default, therefore ignores cases when matching, if TRUE

*# create dfm*

S1\_calibration1\_relevant\_dfm <- dfm(calibration\_1\_relevant\_tokens\_S1, tolower = TRUE)

*# Calculate frequency by relevant articles*

freq\_weight <- textstat\_frequency(S1\_calibration1\_relevant\_dfm, n = 10, groups = c( "2-s2.0-36549047400", "2-s2.0-79951811164"

 , "2-s2.0-40049101875" , "2-s2.0-54349095319"

 , "2-s2.0-77957306688" , "2-s2.0-33646571543"

 , "2-s2.0-84861863606" , "2-s2.0-67649506353"

 , "2-s2.0-76749156103" , "2-s2.0-56549084144", "2-s2.0-70350434146"

 , "2-s2.0-80051667278", "2-s2.0-77953018842"))

*# Plot frequency*

ggplot(data = freq\_weight, aes(x = nrow(freq\_weight):1, y = frequency)) +

 geom\_point() +

 facet\_wrap(~ group, scales = "free") +

 coord\_flip() +

 scale\_x\_continuous(breaks = nrow(freq\_weight):1,

 labels = freq\_weight$feature) +

 labs(x = NULL, y = "Frequency")

ggplot(data = freq\_weight\_all, aes(x = nrow(freq\_weight):1, y = frequency)) +

 geom\_point() +

 facet\_wrap(~ group, scales = "free") +

 coord\_flip() +

 scale\_x\_continuous(breaks = nrow(freq\_weight):1,

 labels = freq\_weight$feature) +

 labs(x = NULL, y = "Frequency")

## Code for primary study selection using pattern matching

*# These sentences are again based on the relevant articles selected for search 1*

pattern2f <- "((for the first time|previously unknown|new(.){0,70}identif|identif(.){0,70}new|new(.){0,70}detect|detect(.){0,70}new|new(.){0,70}determin|determin(.){0,70}new|first(.){0,70}detect|detect(.){0,70}first|first(.){0,70}ident|ident(.){0,70}first|first(.){0,70}determin|determin(.){0,70}first|first(.){0,70}occur|occur(.){0,70}first|new(.){0,70}occur|occur(.){0,70}new|first(.){0,70}report|report(.){0,70}first|new(.){0,70}report|report(.){0,70}new))"

*# 0-70 characters between word(.)and word*

*# Load data Calibration Search 1*

calibration\_1 <- read.delim("Scopus output file", sep = "\t")

*# Remove irrelevant columns and combine abstract and title to tekst*

calibration\_1\_cleaned <- with(calibration\_1, calibration\_1[, c("Title","EID", "Abstract")])

calibration\_1\_cleaned <- within(calibration\_1\_cleaned, {

 doc\_id <- as.character(EID)

 text <- paste(Title, Abstract, sep = " ")

 rm(Title)

 rm(Abstract)

 rm(EID)

})

*# rearranging columns to be able to run DataframeSource and than Corpus function from tm package*

calibration\_1\_cleaned <- calibration\_1\_cleaned[, c("doc\_id", "text")]

*# tokenize calibration\_1\_cleaned$text into sentences*

calibration\_1\_cleaned$text <- stri\_enc\_toutf8(calibration\_1\_cleaned$text, is\_unknown\_8bit = TRUE, validate = TRUE) # encode text to utf-8

calibration\_1\_cleaned$sentences <- tokenize\_sentences(calibration\_1\_cleaned$text)

calibration\_1\_cleaned <- unnest(calibration\_1\_cleaned, sentences) # unlist column sentences

*# find relevant sentences and add to calibration 1*

string\_sentences <- calibration\_1\_cleaned$sentences

calibration\_1\_cleaned$relevantsentences <- grepl(pattern2f, string\_sentences, ignore.case = TRUE)

calibration\_1\_pattern2f <- subset.data.frame(calibration\_1\_cleaned, calibration\_1\_cleaned$relevantsentences == "TRUE") # keep only records with relevant sentences

calibration\_1\_pattern2f <- within(calibration\_1\_pattern2f, {

 rm(relevantsentences)

 }) # remove relevantsentences column (is all TRUE now)

*# save calibration\_1\_pattern2f with all information*

calibration\_1\_pattern2f <- merge(x = calibration\_1, y = calibration\_1\_pattern2f, by.x = "EID", by.y = "doc\_id")

# save calibration\_1\_cleaned, in order to compare with the use of different pattern

write.table(calibration\_1\_pattern2f, file = "How and where do you want to save the results?", sep = "\t")

## Code to automatically eliminating some false positives

library("stringi") #stri\_enc\_toutf8(str, is\_unknown\_8bit = FALSE, validate = FALSE)

# import pattern matching result

Appendix\_B <- read.delim(“Pattern matching result”, sep = "\t")

Appendix\_B$sentences <- stri\_enc\_toutf8(Appendix\_B$sentences, is\_unknown\_8bit = TRUE, validate = TRUE) # encode text to utf-8

# Remove records with sentences where "New" refers to a county or city (e.g. New Zealand).

# i. "New" only appears in the title and abstract with an upper case "N" AND

# ii. when "New" appears it is never at the beginning of a sentence

Appendix\_B\_new1 <- subset(Appendix\_B, !grepl(" New", Appendix\_B$sentences, ignore.case = FALSE))

# It may be possible to eliminate false positives by automatically identifying articles where "First" is used as a numerical transition word.

Appendix\_B\_first1 <- subset(Appendix\_B\_new1, !grepl("^First,", Appendix\_B\_new1$sentences, ignore.case = FALSE))

Appendix\_B\_first2 <- subset(Appendix\_B\_first1, !grepl("first second|first minute|first hour|first day|first week|first month|first year", Appendix\_B\_first1$sentences, ignore.case = TRUE))

# save

write.table(Appendix\_B\_aquatic1, file = "Where and how do you want to save the result", sep = "\t")

# Suggestion for adjusted pattern

Pattern\_adjusted <- "((for the first time|previously unknown|new(.){0,70}identif|identif(.){0,70}new|new(.){0,70}detect|detect(.){0,70}new|new(.){0,70}determin|determin(.){0,70}new|first(.){0,70}detect|detect(.){0,70}first|first(.){0,70}ident|ident(.){0,70}first|first(.){0,70}determin|determin(.){0,70}first|first(.){0,70}occur|occur(.){0,70}first|new(.){0,70}occur|occur(.){0,70}new|first(.){0,70}report|report(.){0,70}first|new(.){0,70}report|report(.){0,70}new|novel(.){0,70}species|new(.){0,70}species|undescribed(.){0,70}species|first outbreak|first description))"