library(gsubf)

library(proto)

library(RSQLite)

library(readr)

library(tm)

library(SnowballC)

library(RWeka)

library(sqldf)

library(dplyr)

library(data.table)

library(ggmap)

library(RODBC)

library(XML)

library(rvest)

library(xml2)

library(openxlsx)

library(RPostgreSQL)

library(stringr)

library(data.table)

rm(list=ls())

##Parameter\_1 : Location - Country

##Set the working directory with the ubication of the files

setwd("C:/Users/Mauricio/Desktop/WETLANDS\_EXTRATION/Documents")

## The documents should be prepared before previous analysis:

## Step 1: Convertion of the document from PDF to .txt by opening the PDF file in Microsoft WORD an then save it as .txt

## Step 2: Elimination of all whitespaces and tap-spaces by using the program Note++ and the following regular expression [\t\r\n\s]+

##Read the documents you want for analysis (Documents located in a specific folder in the computer)

files\_names<-list.files(getwd()) ##The name of the documents should be in the form doc[#].txt e.g. doc1.txt

doc.list<-lapply(files\_names,read\_file)

N.docs<-length(doc.list)

##Construction of Corpus of documents

my.docs<-VectorSource(c(doc.list))

my.docs$names<-sub(".txt","",files\_names)

mydocs<-Corpus(my.docs)

removeSpecialChars<-function(x) gsub("[^a-zA-z0-9. ]","",x)

##Cleaning 1= Removing special characters

mydocs<-tm\_map(mydocs,removeSpecialChars)

##Cleaning 2= Removing stopwords()

mydocs<-tm\_map(mydocs,removeWords,stopwords('en'))

##Cleaning 3= Removing doble whitespaces between words

mydocs<-tm\_map(mydocs,stripWhitespace)

##Remove information before the word "Abstract"

removeAbstract<-function(x) gsub("(.\*?)A(?i)bstract(?-i)","",x)

mydocs<-tm\_map(mydocs,removeAbstract)

##Remove information after the word "Conclusions"

removeConclusions<-function(x) sub("C(?i)onclusion(?-i)(.\*)","",x)

mydocs<-tm\_map(mydocs,removeConclusions)

##Structure number 1: Abstract - Introduction

removeIntroduction\_down<-function(x) sub("I(?i)ntroduction(?-i)(.\*)","",x)

str1<-tm\_map(mydocs,removeIntroduction\_down)

##Structure number 2: Introduction - Methods

removeIntroduction\_up<-function(x) sub("(.\*?)I(?i)ntroduction(?-i)","",x)

str2\_fs1.1<-tm\_map(mydocs,removeIntroduction\_up)

removeMethods\_down<-function(x) sub("M(?i)ethods(?-i)(.\*)","",x)

str2\_fs1.2<-tm\_map(str2\_fs1.1,removeMethods\_down)

removeMethods\_down<-function(x) sub("M(?i)aterial(?-i)(.\*)","",x)

str2<-tm\_map(str2\_fs1.2,removeMethods\_down)

##Structure number 3: Method - Results

removeMethods\_up<-function(x) sub("(.\*?)M(?i)aterial(?-i)","",x)

str3\_fs1.1<-tm\_map(mydocs,removeMethods\_up)

removeMethods\_up<-function(x) sub("(.\*?)M(?i)ethods(?-i)","",x)

str3\_fs1.2<-tm\_map(str3\_fs1.1,removeMethods\_up)

removeResults\_down<-function(x) sub("R(?i)esults(?-i)(.\*)","",x)

str3<-tm\_map(str3\_fs1.2,removeResults\_down)

##Structure number 4: Results-Conclusions

removeResults\_up<-function(x) sub("(.\*?)R(?i)esults(?-i)","",x)

str4<-tm\_map(mydocs,removeResults\_up)

dataframe\_docs<-data.frame(text=sapply(mydocs, as.character),stringsAsFactors = FALSE)

extract\_str1<-data.frame(text=sapply(str1, as.character),stringsAsFactors = FALSE)

extract\_str2<-data.frame(text=sapply(str2, as.character),stringsAsFactors = FALSE)

extract\_str3<-data.frame(text=sapply(str3, as.character),stringsAsFactors = FALSE)

extract\_str4<-data.frame(text=sapply(str4, as.character),stringsAsFactors = FALSE)

#Creation of VCorpus for N-gram analysis

df<-data.frame(doc\_id=seq(1:N.docs),text=dataframe\_docs$text,stringsAsFactors = F)

VCorpus\_docs<-VCorpus(DataframeSource(df))

df\_str1<-data.frame(doc\_id=seq(1:N.docs),text=extract\_str1$text,stringsAsFactors = F)

VCorpus\_docs1<-VCorpus(DataframeSource(df\_str1))

df\_str2<-data.frame(doc\_id=seq(1:N.docs),text=extract\_str2$text,stringsAsFactors = F)

VCorpus\_docs2<-VCorpus(DataframeSource(df\_str2))

df\_str3<-data.frame(doc\_id=seq(1:N.docs),text=extract\_str3$text,stringsAsFactors = F)

VCorpus\_docs3<-VCorpus(DataframeSource(df\_str3))

df\_str4<-data.frame(doc\_id=seq(1:N.docs),text=extract\_str4$text,stringsAsFactors = F)

VCorpus\_docs4<-VCorpus(DataframeSource(df\_str4))

##Reading the dataset of countries in the world

setwd("C:/Users/Mauricio/Desktop//WETLANDS\_EXTRATION/Documents\_backup")

countries<-read.delim("GEODATASOURCE-COUNTRY.txt", header = TRUE, sep = "\t")

countries\_names<-as.vector(countries[,4])

countries\_codes<-as.vector(countries[,1])

##Extracting the number of countries in the dataset

l\_countries<-length(countries\_names)

country\_code<- c(rep(NA,N.docs))

COUNTRY<-c(rep(NA,N.docs))

##Coding for comparing match in the dataframe of the Term-Document-Matrix and the dataframe of countries names

for (m in 1:N.docs){

#Defining Tokanizer for Term-Document-Matrix = # of words in each row of the matrix

Tokenizer<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=2))

head(NGramTokenizer(VCorpus\_docs3[m],Weka\_control(min=1,max=2)))

dtm3<-TermDocumentMatrix(VCorpus\_docs3[m],control=list(tokenize=Tokenizer,tolower=FALSE))

bio3<-as.data.frame(t(as.matrix(dtm3)))

keywords3<-colnames(bio3)

kw\_country3<-as.data.frame(keywords3)

colnames(kw\_country3)[1]<-"keywords\_country"

Tokenizer<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=2))

head(NGramTokenizer(VCorpus\_docs1[m],Weka\_control(min=1,max=2)))

dtm1<-TermDocumentMatrix(VCorpus\_docs1[m],control=list(tokenize=Tokenizer,tolower=FALSE))

bio1<-as.data.frame(t(as.matrix(dtm1)))

keywords1<-colnames(bio1)

kw\_country1<-as.data.frame(keywords1)

colnames(kw\_country1)[1]<-"keywords\_country"

kw\_country<-rbind(kw\_country3,kw\_country1)

l\_keywords<-length(keywords1)+length(keywords3)

extract\_country<-as.data.frame(lapply(kw\_country, function(x) lapply(countries\_names,function(y) grepl(y,x))))

for (i in 1:l\_keywords){

for (j in 1:l\_countries){

if (extract\_country[i,j]=="TRUE"){

COUNTRY[m]<-countries\_names[j]

country\_code[m]<-countries\_codes[j]

}

}

}

}

rm(dtm,bio,keywords,kw\_country,l\_keywords,extract\_country)

##Parameter\_2 : Location - City

cities<-fread("GEODATASOURCE-CITIES-FREE.txt", header = TRUE, sep = "\t")

cities\_data<-as.data.frame(cities)

MUNICIPALITY<-c(rep(NA,N.docs))

for (m in 1:N.docs){

if (is.na(country\_code[m])==FALSE){

Tokenizer\_city<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=4))

head(NGramTokenizer(VCorpus\_docs3[m],Weka\_control(min=1,max=4)))

dtm\_city<-TermDocumentMatrix(VCorpus\_docs3[m],control=list(tokenize=Tokenizer\_city,tolower=FALSE))

bio\_city<-as.data.frame(t(as.matrix(dtm\_city)))

keywords\_city<-colnames(bio\_city)

kw\_city<-as.vector(keywords\_city)

l\_keywords\_city<-length(keywords\_city)

filtration\_cities<-subset(cities\_data,CC\_FIPS==country\_code[m])

filtration\_cities<-as.vector(filtration\_cities[,2])

l\_cities<-length(filtration\_cities)

filtration\_cities<-paste("^",filtration\_cities,"$",sep="")

extract\_cities<-lapply(filtration\_cities,function(x)grep(x,kw\_city,value=TRUE))

extract\_cities<-extract\_cities[lapply(extract\_cities,length)>0]

if ((NROW(extract\_cities)==1)&(length(extract\_cities[[1]])==1)){

MUNICIPALITY[m]<-extract\_cities[[1]][1]

}

}

rm(dtm\_city,bio\_city,keywords\_city,kw\_city,l\_keywords\_city,extract\_cities,filtration\_cities)

}

##Parameter\_3: Location - Coordinates

## API from Gogle maps for extracting coordinates based on the name of the city (MUNICIPALITY)

LONGITUDE<-c(rep(NA,N.docs))

LATITUDE<-c(rep(NA,N.docs))

for (m in 1:N.docs){

if (is.na(MUNICIPALITY[m])==FALSE){

coordinates<-geocode(MUNICIPALITY[m])

LONGITUDE[m]<-coordinates[1,1]

LATITUDE[m]<-coordinates[1,2]

}

if (is.na(MUNICIPALITY[m])==TRUE){

Tokenizer<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=1))

head(NGramTokenizer(VCorpus\_docs3[m],Weka\_control(min=1,max=1)))

dtm\_coordinates<-TermDocumentMatrix(VCorpus\_docs3[m],control=list(tokenize=Tokenizer,tolower=FALSE))

bio\_coordinates<-as.data.frame(t(as.matrix(dtm\_coordinates)))

keywords\_coordinates<-colnames(bio\_coordinates)

kw\_coordinates<-as.data.frame(keywords\_coordinates)

longitud\_filter\_E<-as.vector(filter(kw\_coordinates, grepl(pattern="^[0-9]{5}E",kw\_coordinates$keywords\_coordinates))[1,])

longitud\_filter\_W<-as.vector(filter(kw\_coordinates, grepl(pattern="^[0-9]{5}W",kw\_coordinates$keywords\_coordinates))[1,])

latitud\_filter\_N<-as.vector(filter(kw\_coordinates, grepl(pattern="^[0-9]{5}N",kw\_coordinates$keywords\_coordinates))[1,])

latitud\_filter\_S<-as.vector(filter(kw\_coordinates, grepl(pattern="^[0-9]{5}S",kw\_coordinates$keywords\_coordinates))[1,])

if (is.na(longitud\_filter\_E[1])=="FALSE"){

lon\_number<-sub("(^[^0-9]\*)(\\d+)([^0-9].\*)", "\\2", longitud\_filter\_E[1])

grade\_longitud<-as.numeric(sub('([0-9]{2}).\*', '\\1', lon\_number))

minutes\_longitud<-as.numeric(sub(".\*(\\d+{2}).\*$", "\\1", lon\_number))

LONGITUDE[m]<-grade\_longitud+minutes\_longitud/60

}

if (is.na(latitud\_filter\_N[1])=="FALSE"){

lat\_number<-sub("(^[^0-9]\*)(\\d+)([^0-9].\*)", "\\2", latitud\_filter\_N[1])

grade\_latitud<-as.numeric(sub('([0-9]{2}).\*', '\\1', lat\_number))

minutes\_latitud<-as.numeric(sub(".\*(\\d+{2}).\*$", "\\1", lat\_number))

LATITUDE[m]<-grade\_latitud+minutes\_latitud/60

}

if (is.na(longitud\_filter\_W[1])=="FALSE"){

lon\_number<-sub("(^[^0-9]\*)(\\d+)([^0-9].\*)", "\\2", longitud\_filter\_W[1])

grade\_longitud<-as.numeric(sub('([0-9]{2}).\*', '\\1', lon\_number))

minutes\_longitud<-as.numeric(sub(".\*(\\d+{2}).\*$", "\\1", lon\_number))

LONGITUDE[m]<-(grade\_longitud+minutes\_longitud/60)\*(-1)

}

if (is.na(latitud\_filter\_S[1])=="FALSE"){

lat\_number<-sub("(^[^0-9]\*)(\\d+)([^0-9].\*)", "\\2", latitud\_filter\_S[1])

grade\_latitud<-as.numeric(sub('([0-9]{2}).\*', '\\1', lat\_number))

minutes\_latitud<-as.numeric(sub(".\*(\\d+{2}).\*$", "\\1", lat\_number))

LATITUDE[m]<-(grade\_latitud+minutes\_latitud/60)\*-1

}

}

}

##Parameter\_4: Type of wetland

matrix\_keyword<-c("^VSSF$","^HSSF$","^VFCW$","^HFCW$","^(?i)vertical(?-i)$","^(?i)surface flow(?-i)$","^(?i)subsurface flow(?-i)$")

matrix\_keyword\_nregex<-c("VSSF","HSSF","VFCW","HFCW","vertical","surface flow","subsurface flow")

cluster<-c("VERTICAL","HORIZONTAL","VERTICAL","HORIZONTAL","VERTICAL","HORIZONTAL","HORIZONTAL")

n.words<-length(matrix\_keyword)

TYPE\_WETLANDS<-vector(mode="character", length=N.docs)

variable\_cluster<-vector(mode="character", length=10)

for (m in 1:N.docs){

Tokanizer\_typewetlands3<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=2))

head(NGramTokenizer(VCorpus\_docs3[m],Weka\_control(min=1,max=2)))

dtm\_typewetlands3<-TermDocumentMatrix(VCorpus\_docs3[m],control = list(tokenize=Tokanizer\_typewetlands3,tolower=FALSE))

bio\_typewetlands3<-as.data.frame(t(as.matrix(dtm\_typewetlands3)))

keywords\_typewetlands3<-colnames(bio\_typewetlands3)

kw\_typewetlands3<-as.data.frame(keywords\_typewetlands3)

colnames(kw\_typewetlands3)[1]<-"keywords\_typewetlands"

Tokanizer\_typewetlands4<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=2))

head(NGramTokenizer(VCorpus\_docs4[m],Weka\_control(min=1,max=2)))

dtm\_typewetlands4<-TermDocumentMatrix(VCorpus\_docs4[m],control = list(tokenize=Tokanizer\_typewetlands4,tolower=FALSE))

bio\_typewetlands4<-as.data.frame(t(as.matrix(dtm\_typewetlands4)))

keywords\_typewetlands4<-colnames(bio\_typewetlands4)

kw\_typewetlands4<-as.data.frame(keywords\_typewetlands4)

colnames(kw\_typewetlands4)[1]<-"keywords\_typewetlands"

Tokanizer\_typewetlands1<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=2))

head(NGramTokenizer(VCorpus\_docs1[m],Weka\_control(min=1,max=2)))

dtm\_typewetlands1<-TermDocumentMatrix(VCorpus\_docs1[m],control = list(tokenize=Tokanizer\_typewetlands4,tolower=FALSE))

bio\_typewetlands1<-as.data.frame(t(as.matrix(dtm\_typewetlands1)))

keywords\_typewetlands1<-colnames(bio\_typewetlands1)

kw\_typewetlands1<-as.data.frame(keywords\_typewetlands1)

colnames(kw\_typewetlands1)[1]<-"keywords\_typewetlands"

kw\_typewetlands<-rbind(kw\_typewetlands3,kw\_typewetlands4,kw\_typewetlands1)

c<-0

for(i in 1:n.words){

length\_filtered\_terms\_type<-nrow(filter(kw\_typewetlands, grepl(pattern=matrix\_keyword[i],kw\_typewetlands$keywords\_typewetlands)))

if((length\_filtered\_terms\_type>=1)&(TYPE\_WETLANDS[m]!=""))

{

con<-0

for (l in 1:c){

if (variable\_cluster[l]==cluster[i]){

con<-1

}

}

if (con==0){

c<-c+1

variable\_cluster[c]<-cluster[i]

TYPE\_WETLANDS[m]<-paste0(TYPE\_WETLANDS[m],"-",cluster[i])

}

}

if((length\_filtered\_terms\_type>=1)&(TYPE\_WETLANDS[m]==""))

{

c<-c+1

variable\_cluster[c]<-cluster[i]

TYPE\_WETLANDS[m]<-cluster[i]

}

}

}

##Parameter\_5: Area

##Matrix of words from the text based in the Tokanizer (minimum and maximum number of words)

AREA<-vector(mode="character", length=N.docs)

NLPBigramTokenizer <- function(x) {

unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)

}

for (m in 1:N.docs){

dtm\_area<-TermDocumentMatrix(VCorpus\_docs1[m],control = list(tokenize=NLPBigramTokenizer))

bio\_area<-as.data.frame(t(as.matrix(dtm\_area)))

keywords\_area<-colnames(bio\_area)

kw\_area<-as.data.frame(keywords\_area)

filtered\_terms\_area1 <- filter(kw\_area, grepl(pattern="[0-9]m2.$",kw\_area$keywords\_area))

filtered\_terms\_area2 <- filter(kw\_area, grepl(pattern="[0-9] m2.$",kw\_area$keywords\_area))

filtered\_terms\_area3 <- filter(kw\_area, grepl(pattern="[[0-9]m2$",kw\_area$keywords\_area))

filtered\_terms\_area4 <- filter(kw\_area, grepl(pattern="[0-9] m2$",kw\_area$keywords\_area))

filtered\_terms\_area<-rbind(filtered\_terms\_area1,filtered\_terms\_area2,filtered\_terms\_area3,filtered\_terms\_area4)

filtered\_terms\_area<-as.vector(filtered\_terms\_area$keywords\_area)

#Another filtering process for just extracting the numbers

if (length(filtered\_terms\_area)>0){

for(i in 1:length(filtered\_terms\_area)){

if(AREA[m]!="")

{

filtered\_terms\_area[i]<-sub("m(.\*)","",gsub(gsub("[0-9].\*","",filtered\_terms\_area[i]),"",filtered\_terms\_area[i]))

filtered\_terms\_area[i]<-gsub("[[:space:]]+","",filtered\_terms\_area[i])

AREA[m]<-paste0(AREA[m],"-",filtered\_terms\_area[i])

}

if(AREA[m]=="")

{

filtered\_terms\_area[i]<-sub("m(.\*)","",gsub(gsub("[0-9].\*","",filtered\_terms\_area[i]),"",filtered\_terms\_area[i]))

filtered\_terms\_area[i]<-gsub("[[:space:]]+","",filtered\_terms\_area[i])

AREA[m]<-filtered\_terms\_area[i]

}

}

}

if (length(filtered\_terms\_area)==0){

dtm\_area<-TermDocumentMatrix(VCorpus\_docs3[m],control = list(tokenize=NLPBigramTokenizer))

bio\_area<-as.data.frame(t(as.matrix(dtm\_area)))

keywords\_area<-colnames(bio\_area)

kw\_area<-as.data.frame(keywords\_area)

filtered\_terms\_area1 <- filter(kw\_area, grepl(pattern="[0-9]m2.$",kw\_area$keywords\_area))

filtered\_terms\_area2 <- filter(kw\_area, grepl(pattern="[0-9] m2.$",kw\_area$keywords\_area))

filtered\_terms\_area3 <- filter(kw\_area, grepl(pattern="[[0-9]m2$",kw\_area$keywords\_area))

filtered\_terms\_area4 <- filter(kw\_area, grepl(pattern="[0-9] m2$",kw\_area$keywords\_area))

filtered\_terms\_area<-rbind(filtered\_terms\_area1,filtered\_terms\_area2,filtered\_terms\_area3,filtered\_terms\_area4)

filtered\_terms\_area<-as.vector(filtered\_terms\_area$keywords\_area)

for(i in 1:length(filtered\_terms\_area)){

if(AREA[m]!="")

{

filtered\_terms\_area[i]<-sub("m(.\*)","",gsub(gsub("[0-9].\*","",filtered\_terms\_area[i]),"",filtered\_terms\_area[i]))

filtered\_terms\_area[i]<-gsub("[[:space:]]+","",filtered\_terms\_area[i])

AREA[m]<-paste0(AREA[m],"-",filtered\_terms\_area[i])

}

if(AREA[m]=="")

{

filtered\_terms\_area[i]<-sub("m(.\*)","",gsub(gsub("[0-9].\*","",filtered\_terms\_area[i]),"",filtered\_terms\_area[i]))

filtered\_terms\_area[i]<-gsub("[[:space:]]+","",filtered\_terms\_area[i])

AREA[m]<-filtered\_terms\_area[i]

}

}

}

}

##Parameter\_6: Type of wastewaster

TYPE\_WASTEWATER<-vector(mode="character", length=N.docs)

matrix\_keyword<-c("^(?i)domestic(?-i)$","^(?i)industrial(?-i)$","^(?i)mixed(?-i)$","^(?i)urban runoff(?-i)$","^(?i)food processing industry(?-i)$")

matrix\_keyword\_nregex<-c("domestic","industrial","mixed","urban runoff","food processing industry")

cluster<-c("DOMESTIC","INDUSTRIAL","DOMESTIC","RUNOFF","INDUSTRIAL")

variable\_cluster<-vector(mode="character", length=10)

n.words<-length(matrix\_keyword)

for (m in 1:N.docs){

Tokanizer\_typewastewater<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=3))

head(NGramTokenizer(VCorpus\_docs3[m],Weka\_control(min=1,max=3)))

dtm\_typewastewater<-TermDocumentMatrix(VCorpus\_docs3[m],control = list(tokenize=Tokanizer\_typewastewater,tolower=FALSE))

bio\_typewastewater<-as.data.frame(t(as.matrix(dtm\_typewastewater)))

keywords\_typewastewater<-colnames(bio\_typewastewater)

kw\_typewastewater<-as.data.frame(keywords\_typewastewater)

c<-0

for(i in 1:n.words){

length\_filtered\_terms\_type<-nrow(filter(kw\_typewastewater, grepl(pattern=matrix\_keyword[i],kw\_typewastewater$keywords\_typewastewater)))

if((length\_filtered\_terms\_type>=1)&(TYPE\_WASTEWATER[m]!=""))

{

con<-0

for (l in 1:c){

if (variable\_cluster[l]==cluster[i]){

con<-1

}

}

if (con==0){

c<-c+1

variable\_cluster[c]<-cluster[i]

TYPE\_WASTEWATER[m]<-paste0(TYPE\_WASTEWATER[m],"-",cluster[i])

}

}

if((length\_filtered\_terms\_type>=1)&(TYPE\_WASTEWATER[m]==""))

{

c<-c+1

variable\_cluster[c]<-cluster[i]

TYPE\_WASTEWATER[m]<-cluster[i]

}

}

}

##Parameter\_7: Type of plants

setwd("C:/Users/Mauricio/Desktop/WETLANDS\_EXTRATION/Documents\_backup")

TYPE\_PLANT<-vector(mode="character", length=N.docs)

matrix<-as.data.frame(read.xlsx("Type\_plants.xlsx"))

matrix\_keywords<-as.vector(matrix[,1])

variable\_cluster<-vector(mode="character", length=10)

cluster<-as.vector(matrix[,2])

n.words<-length(matrix\_keywords)

for (m in 1:N.docs){

Tokanizer\_typeplant<-function(x) NGramTokenizer(x,Weka\_control(min=1,max=4))

head(NGramTokenizer(VCorpus\_docs3[m],Weka\_control(min=1,max=4)))

dtm\_typeplant<-TermDocumentMatrix(VCorpus\_docs3[m],control = list(tokenize=Tokanizer\_typeplant))

bio\_typeplant<-as.data.frame(t(as.matrix(dtm\_typeplant)))

keywords\_typeplant<-colnames(bio\_typeplant)

kw\_typeplant<-as.data.frame(keywords\_typeplant)

c<-0

for(i in 1:n.words){

length\_filtered\_terms\_type<-nrow(filter(kw\_typeplant, grepl(pattern="Volume",kw\_typeplant$keywords\_typeplant)))

if((length\_filtered\_terms\_type>=1)&(TYPE\_PLANT[m]!=""))

{

con<-0

for (l in 1:c){

if (variable\_cluster[l]==cluster[i]){

con<-1

}

}

if (con==0){

c<-c+1

variable\_cluster[c]<-cluster[i]

TYPE\_PLANT[m]<-paste0(TYPE\_PLANT[m],";",cluster[i])

}

}

if((length\_filtered\_terms\_type>=1)&(TYPE\_PLANT[m]==""))

{

c<-c+1

variable\_cluster[c]<-cluster[i]

TYPE\_PLANT[m]<-cluster[i]

}

}

}

##Parameter\_8: BOD5 influent and BOD5 efluent

##The following approach works for pre-reviwed articles, which information

##is available in HTML version e.g. ELSEVIER

BOD\_inlist<-vector(mode="list",length=N.docs)

BOD\_outlist<-vector(mode="list",length=N.docs)

BOD\_removallist<-vector(mode="list",length=N.docs)

CITATION<-c(rep(NA,N.docs))

URL<-c(rep(NA,N.docs))

JOURN\_NAME<-c(rep(NA,N.docs))

PUBLISHER<-c(rep(NA,N.docs))

LIT\_TYPE<-c(rep(NA,N.docs))

YEAR<-c(rep(NA,N.docs))

TITLE<-c(rep(NA,N.docs))

setwd("C:/Users/Mauricio/Desktop/WETLANDS\_EXTRATION/HTML\_LINKS")

HTML\_links<-read.xlsx("HTML\_links.xlsx")

setwd("C:/Users/Mauricio/Desktop/WETLANDS\_EXTRATION/Phantom/phantomjs-2.1.1-windows/bin")

for (m in 1:N.docs){

LIT\_TYPE[m]<-"PRE-REVIEW ARTICLE"

for (j in 1:nrow(HTML\_links)){

if (my.docs$names[m]==HTML\_links[j,1]){

url<-HTML\_links[j,2]

URL[m]<-url

name<-paste0("doc",m,".js")

writeLines(sprintf("var page = require('webpage').create();

page.open('%s', function () {

console.log(page.content); //page source

phantom.exit();

});", url), con=name)

name\_phantom<-paste0("phantomjs doc",m,".js")

name\_html<-paste0("doc",m,".html")

write(readLines(pipe(name\_phantom, "r")), name\_html)

journal\_elsevier<-grepl("sciencedirect",url,ignore.case=FALSE)

journal\_IWA<-grepl("iwaponline",url,ignore.case=FALSE)

journal\_researchgate<-grepl("researchgate",url,ignore.case=FALSE)

journal\_ncbi<-grepl("ncbi",url,ignore.case=FALSE)

page\_html<- read\_html(name\_html)

table<-page\_html %>% html\_nodes("table") %>% html\_table(fill=TRUE)

head\_lines<-page\_html %>% html\_nodes("span")%>%html\_text()

if (journal\_elsevier=="TRUE"){

title\_article<-page\_html %>% html\_nodes("span.title-text")%>%html\_text()

surname<-page\_html %>% html\_nodes("span.text.surname")%>%html\_text()

name<-page\_html %>% html\_nodes("span.text.given-name")%>%html\_text()

doi<-page\_html %>% html\_nodes("a.doi")%>%html\_text()

doi<-sub("(.\*?)doi.org","doi:",doi)

journal\_name<-page\_html %>% html\_nodes("h2#publication-title.publication-title")%>%html\_text()

publication\_year<-as.data.frame(page\_html %>% html\_nodes("div.text-xs")%>%html\_text())

colnames(publication\_year)[1]<-"YEAR"

publication\_year<-as.vector(filter(publication\_year,grepl(pattern="(?i)volume(?-i)",publication\_year$YEAR))[,1])

name<-sub("Ã¤","ä",name)

name<-sub("Ãµ","õ",name)

name<-sub("Ã\u009c","Ü",name)

surname<-sub("Ã\u0096Ã¶","Öö",surname)

for (z in 1:length(surname)){

if (z==1){

authors<-paste0(surname[1]," ",name[1],",")

}

if ((z!=1)&(z!=length(surname))){

authors<-paste0(authors,name[z]," ",surname[z],",")

}

if(z==length(surname)){

authors<-paste0(authors,"and ",name[z]," ",surname[z],".")

}

}

CITATION[m]<-paste0(authors,'"',title\_article,'"',".",journal\_name," ",publication\_year[1],".",doi)

}

if (journal\_IWA=="TRUE"){

title\_article<-page\_html %>% html\_nodes("h1#page-title.highwire-cite-title")%>%html\_text()

TITLE[m]<-title\_article

journal\_name<-page\_html %>% html\_nodes("div.region-inner.region-branding-inner")%>%html\_text()

JOURN\_NAME[m]<-journal\_name

name<-page\_html %>% html\_nodes("span.highwire-citation-authors")%>%html\_text()

doi\_publicationyear<-page\_html %>% html\_nodes("div.highwire-cite-metadata")%>%html\_text()

CITATION[m]<-paste0(name[1],".",'"',title\_article,'"',".Water Sci Technol.",doi\_publicationyear[1])

}

if (journal\_ncbi=="TRUE"){

journal\_name<-page\_html %>% html\_nodes("h1")%>%html\_text()

name<-page\_html %>% html\_nodes("div.auths")%>%html\_text()

publication\_year<-page\_html %>% html\_nodes("div.cit")%>%html\_text()

PMID<-page\_html %>% html\_nodes("dd")%>%html\_text()

CITATION[m]<-paste0(name,'"',journal\_name[2],'"',publication\_year,"PMID: ",PMID[2])

}

head\_tables<-vector(mode="character", length=20)

a<-0

for (i in 1:length(head\_lines)){

regex\_head<-grepl("Table",head\_lines[i],ignore.case=FALSE)

if (regex\_head=="TRUE"){

a<-a+1

head\_tables[a]<-head\_lines[i]

}

}

head\_tables<-head\_tables[head\_tables!=""]

head\_tables\_clean<-vector(mode="character", length=20)

c<-0

for (i in 1:length(head\_tables)){

number<-length(unlist(str\_extract\_all(head\_tables[i], "\\w+")))

if (number>2){

c<-c+1

head\_tables\_clean[c]<-head\_tables[i]

}

}

head\_tables\_clean<-head\_tables\_clean[head\_tables\_clean!=""]

BOD\_IN<-vector(mode="integer", length=20)

BOD\_OUT<-vector(mode="integer", length=20)

BOD\_REMOVAL<-vector(mode="integer", length=20)

sec\_in<-0

sec\_out<-0

sec\_removal<-0

for(b in 1:length(table)){

con5<-0

con6<-0

con2<-0

con3<-0

con4<-0

regex\_influent<-grepl("(?i)influent(?-i)",head\_tables\_clean[b],ignore.case=FALSE)|grepl("(?i)inflow(?-i)",head\_tables\_clean[b],ignore.case=FALSE)

if (regex\_influent==TRUE){

con2<-1

}

regex\_efluent<-grepl("(?i)efluent(?-i)",head\_tables\_clean[b],ignore.case=FALSE)|grepl("(?i)outflow(?-i)",head\_tables\_clean[b],ignore.case=FALSE)

if (regex\_efluent==TRUE){

con3<-1

}

regex\_removal<-grepl("(?i)removal(?i)",head\_tables\_clean[b],ignore.case=FALSE)

if (regex\_removal==TRUE){

con4<-1

}

con<-0

row<-nrow(table[[b]])

col<-ncol(table[[b]])

for(i in 1:row){

for (j in 1:col){

table[[b]][i,j]<-sub("(?i)Â(.\*)(?-i)","",table[[b]][i,j])

regex<-grepl("BOD",table[[b]][i,j],ignore.case=FALSE)|grepl("BOD",colnames(table[[b]])[j],ignore.case=FALSE)

if (regex=="TRUE"){

con<-con+1

}

}

}

table\_numeric<-data.frame(table[[b]],stringsAsFactors=FALSE)

table\_numeric<- as.data.frame(sapply(table\_numeric, as.numeric))

elimination\_colum<-vector(mode="integer", length=10)

elimination\_row<-vector(mode="integer", length=10)

if (con>=1){

for (j in 1:col){

regex\_deviation<-grepl("(?i)deviation(?-i)",table[[b]][1,j],ignore.case=FALSE)

regex\_deviation\_names<-grepl("(?i)deviation(?-i)",colnames(table[[b]][j]),ignore.case=FALSE)

if((regex\_deviation==TRUE)|(regex\_deviation\_names==TRUE)){

con5<-con5+1

elimination\_colum[con5]<-j

}

}

for (i in 1:row){

regex\_deviation<-grepl("(?i)deviation(?-i)",table[[b]][i,1],ignore.case=FALSE)

if(regex\_deviation==TRUE){

con6<-con6+1

elimination\_row[con6]<-i

}

}

elimination\_colum<-elimination\_colum[elimination\_colum!=0]

elimination\_row<-elimination\_row[elimination\_row!=0]

if ((length(elimination\_colum)!=0)&(length(elimination\_row)!=0)){

table[[b]]<-table[[b]][-elimination\_row,-elimination\_colum]

}

if ((length(elimination\_colum)==0)&(length(elimination\_row)!=0)){

table[[b]]<-table[[b]][-elimination\_row,]

}

if ((length(elimination\_colum)!=0)&(length(elimination\_row)==0)){

table[[b]]<-table[[b]][,-elimination\_colum]

}

}

row<-nrow(table[[b]])

col<-ncol(table[[b]])

if ((con2>=1)&(con>=1)){

for (i in 1:2){

for (j in 1:col){

regex\_in<-grepl("BOD",table[[b]][i,j],ignore.case=FALSE)

k<-i+1

if(k<=row){

if ((regex\_in=="TRUE")&(is.numeric(table\_numeric[k,j])==TRUE)&(is.na(table\_numeric[k,j])==FALSE)){

p<-i+1

for (u in p:row){

sec\_in<-sec\_in+1

BOD\_IN[sec\_in]<-table[[b]][u,j]

}

}

}

}

}

if (BOD\_IN[1]==0){

for (j in 1:col){

regex\_in<-grepl("BOD",colnames(table[[b]])[j],ignore.case=FALSE)&grepl("(?i)in(?-i)",table[[b]][1,j],ignore.case=FALSE)

if (regex\_in=="TRUE"){

for (i in 2:row){

sec\_in<-sec\_in+1

BOD\_IN[sec\_in]<-table[[b]][i,j]

}

}

}

}

for (i in 1:row) {

regex\_in<-grepl("BOD",table[[b]][i,1],ignore.case=FALSE)

if ((regex\_in=="TRUE")&(is.numeric(table\_numeric[i,2])==TRUE)&(is.na(table\_numeric[k,j])==FALSE)){

for (u in 2:col){

sec\_in<-sec\_in+1

BOD\_IN[sec\_in]<-table[[b]][i,u]

}

}

}

BOD\_IN<-BOD\_IN[BOD\_IN!=0]

BOD\_inlist[[m]]<-BOD\_IN

}

if ((con3>=1)&(con>=1)){

for (i in 1:2){

for (j in 1:col){

regex\_out<-grepl("BOD",table[[b]][i,j],ignore.case=FALSE)

k<-i+1

if(k<=row){

if ((regex\_out=="TRUE")&(is.numeric(table\_numeric[k,j])==TRUE)&(is.na(table\_numeric[k,j])==FALSE)){

p<-i+1

for (u in p:row){

sec\_out<-sec\_out+1

BOD\_OUT[sec\_out]<-table[[b]][u,j]

}

}

}

}

}

if (BOD\_OUT[1]==0){

for (j in 1:col){

regex\_out<-grepl("BOD",colnames(table[[b]])[j],ignore.case=FALSE)&grepl("(?i)out(?-i)",table[[b]][1,j],ignore.case=FALSE)

if (regex\_out=="TRUE"){

for (i in 2:row){

sec\_out<-sec\_out+1

BOD\_OUT[sec\_out]<-table[[b]][i,j]

}

}

}

}

for (i in 1:row) {

regex\_out<-grepl("BOD",table[[b]][i,1],ignore.case=FALSE)

if ((regex\_out=="TRUE")&(is.numeric(table\_numeric[i,2])==TRUE)&(is.na(table\_numeric[k,j])==FALSE)){

for (u in 2:col){

sec\_out<-sec\_out+1

BOD\_OUT[sec\_out]<-table[[b]][i,u]

}

}

}

BOD\_OUT<-BOD\_OUT[BOD\_OUT!=0]

BOD\_outlist[[m]]<-BOD\_OUT

}

if ((con4>=1)&(con>=1)){

sec<-0

for (i in 1:2){

for (j in 1:col){

regex\_removal<-grepl("BOD",table[[b]][i,j],ignore.case=FALSE)

k<-i+1

if(k<=row){

if ((regex\_removal=="TRUE")&(is.numeric(table\_numeric[k,j])==TRUE)&(is.na(table\_numeric[k,j])==FALSE)){

p<-i+1

for (u in p:row){

sec\_removal<-sec\_removal+1

BOD\_REMOVAL[sec\_removal]<-table[[b]][u,j]

}

}

}

}

}

for (i in 1:row) {

regex\_removal<-grepl("BOD",table[[b]][i,1],ignore.case=FALSE)

if ((regex\_removal=="TRUE")&(is.numeric(table\_numeric[i,2])==TRUE)&(is.na(table\_numeric[k,j])==FALSE)){

for (u in 2:col){

sec\_removal<-sec\_removal+1

BOD\_REMOVAL[sec\_removal]<-table[[b]][i,u]

}

}

}

BOD\_REMOVAL<-BOD\_REMOVAL[BOD\_REMOVAL!=0]

BOD\_removallist[[m]]<-BOD\_REMOVAL

}

}

}

}

}

for (i in 1:N.docs){

if (length(BOD\_inlist[[i]])==0){

BOD\_inlist[[i]]<-c(NA)

}

if (length(BOD\_outlist[[i]])==0){

BOD\_outlist[[i]]<-c(NA)

}

if (length(BOD\_removallist[[i]])==0){

BOD\_removallist[[i]]<-c(NA)

}

BOD\_removallist[[i]]<-as.numeric(BOD\_removallist[[i]])

BOD\_outlist[[i]]<-as.numeric(BOD\_outlist[[i]])

BOD\_inlist[[i]]<-as.numeric(BOD\_inlist[[i]])

}

##Extracting BOD values from other sources than tables

for (m in 1:N.docs){

x<-c(NA)

if (is.na(x)=="TRUE"){

Tokanizer\_BOD<-function(x) NGramTokenizer(x,Weka\_control(min=3,max=3))

head(NGramTokenizer(VCorpus\_docs1[m],Weka\_control(min=3,max=3)))

dtm\_BOD<-TermDocumentMatrix(VCorpus\_docs1[m],control = list(tokenize=Tokanizer\_BOD,tolower=FALSE))

bio\_BOD<-as.data.frame(t(as.matrix(dtm\_BOD)))

keywords\_BOD<-colnames(bio\_BOD)

kw\_BOD<-as.data.frame(keywords\_BOD)

filter\_BOD<-filter(kw\_BOD, grepl(pattern="BOD$",kw\_BOD$keywords\_BOD))

}

}

pw <- {

"mauricio"

}

drv <- dbDriver("PostgreSQL")

con <- dbConnect(drv, dbname = "postgres",

host = "localhost", port = 5432,

user = "openpg", password = pw)

con2=dbConnect(PostgreSQL(),user = "postgres",password="wetlands",dbname = "postgres")

SITES <- as.data.frame(dbGetQuery(con2, "SELECT \* from SITES"))

SYSTEMS<-as.data.frame(dbGetQuery(con2, "SELECT \* from SYSTEMS"))

C\_PLANTS<-as.data.frame(dbGetQuery(con2, "SELECT \* from C\_PLANTS"))

LITERATURE<-as.data.frame(dbGetQuery(con2, "SELECT \* from LITERATURE"))

C\_ORG<-as.data.frame(dbGetQuery(con2, "SELECT \* from C\_ORG"))

CELLS<-as.data.frame(dbGetQuery(con2, "SELECT \* from CELLS"))

row\_C\_ORG<-nrow(C\_ORG)

row\_LITERATURE<-nrow(C\_LITERATURE)

row\_PLANTS<-nrow(C\_PLANTS)

row\_SYSTEM<-nrow(SYSTEMS)

row\_SYSTEM\_seq<-row\_SYSTEM+1

row\_C\_ORG\_seq<-row\_C\_ORG+1

row\_SITES<-nrow(SITES)

##Combine all lists in one dataframe for exporting to PostGreSQL

database <- data.table( DOCUMENT=character(),

CTRY\_CODE=character(),

SYSTEM\_ID=character(),

SITE\_ID=character(),

LIT\_ID=character(),

PLANT\_ID=character(),

CELL\_ID=character(),

CTRY\_NAME=character(),

C\_ORG\_ID=character(),

MUNI=character(),

LONG=character(),

LAT=character(),

LIT\_TYPE=character(),

TITLE=character(),

YEAR=character(),

JOURN\_NAME=character(),

PUBLISHER=character(),

URL=character(),

CELL\_AREA=character(),

PLANT\_SPEC=character(),

WW\_TYPE=character(),

C\_BOD\_IN=integer(),

BOD\_outlist=integer(),

BOD\_removallist=integer(),

CW\_TYPE=character(),

stringsAsFactors=FALSE)

for (i in 1:N.docs){

l1<-length(BOD\_inlist[[i]])

l2<-length(BOD\_outlist[[i]])

l3<-length(BOD\_removallist[[i]])

location<-c(l1,l2,l3)

max\_location<-which(location == max(location), arr.ind = TRUE)

max<-max(location)

if (max\_location==1){

country<-rep(COUNTRY[i],max)

ctr\_code<-rep(country\_code[i],max)

longitude<-rep(LONGITUDE[i],max)

latitude<-rep(LATITUDE[i],max)

typewetlands<-rep(TYPE\_WETLANDS[i],max)

typewaster<-rep(TYPE\_WASTEWATER[i],max)

typeplant<-rep(TYPE\_PLANT[i],max)

municipality<-rep(MUNICIPALITY[i],max)

area<-rep(AREA[i],max)

bod\_outlist<-rep(BOD\_outlist[[i]],length.out=max)

bod\_removallist<-rep(BOD\_removallist[[i]],length.out=max)

document<-rep(my.docs$names[i],max)

SITE\_ID<-vector(mode="character",length=max)

PLANT\_ID<-vector(mode="character",length=max)

row\_SYSTEM\_end<-row\_SYSTEM+max

row\_C\_ORG\_end<-row\_C\_ORG\_end+max

n\_SYSTEM\_ID<-seq(row\_SYSTEM\_seq:row\_SYSTEM\_end)

n\_C\_ORG\_ID<-seq(row\_C\_ORG\_seq:row\_C\_ORG\_end)

C\_ORG\_ID<-unlist(lapply(n\_C\_ORG\_ID,function(x)paste0("C\_ORG\_ID\_",x)))

SYSTEM\_ID<-unlist(lapply(n\_SYSTEM\_ID,function(x)paste0("SYSTEM\_",x)))

CELL\_ID<-unlist(lapply(n\_SYSTEM\_ID,function(x)paste0("CELL\_",x)))

row\_SITES\_seq<-row\_SITES

row\_PLANT\_seq<-row\_PLANT

row\_LITERATURE\_seq<-row\_LITERATURE

for (i in 1:max){

for (m in 1:row\_SITES){

if(i==1){

if ((SITES$CTRY\_NAME[m]==COUNTRY[i])&(SITES$MUNI[m]==MUNIPALITY[i])&(SITES$LONG[m]==LONGITUDE[m])){

SITE\_ID[i]<-SITES$SITE\_ID[m]

}

}

if(i!=1){

if ((SITES$CTRY\_NAME[m]==COUNTRY[i])&(SITES$MUNI[m]==MUNIPALITY[i])&(SITES$LONG[m]==LONGITUDE[i])&(COUNTRY[i-1]==COUNTRY[i])&(LONGITUDE[i-1]==LONGITUDE[i])&(MUNIPALITY[i-1]==MUNIPALITY[i])){

SITE\_ID[i]<-SITES$SITE\_ID[m]

}

}

}

if (SITE\_ID[i]==""){

row\_SITES\_seq<-row\_SITES\_seq+1

SITE\_ID[i]<-paste0("SITE\_",row\_SITES)

}

}

for (i in 1:max){

for (m in 1:row\_PLANTS){

if(i==1){

if (C\_PLANTS$PLANT\_SPEC[m]==typeplant[i]){

PLANT\_ID[i]<-C\_PLANTS$PLANT\_ID[m]

}

}

if(i!=1){

if ((C\_PLANTS$PLANT\_SPEC[m]==typeplant[i])&(typeplant[i-1]==typeplant[i])){

PLANT\_ID[i]<-C\_PLANTS$PLANT\_ID[m]

}

}

}

if (PLANT\_ID[i]==""){

row\_PLANT\_seq<-row\_PLANT\_seq+1

PLANT\_ID[i]<-paste0("PLANT\_",row\_PLANT\_seq)

}

}

for (i in 1:max){

for (m in 1:row\_LITERATURE){

if(i==1){

if (LITERATURE$TITLE[m]==TITLE[i]){

LIT\_ID[i]<-LITERATURE$LIT\_ID[m]

}

}

if(i!=1){

if ((LITERATURE$TITLE[m]==TITLE[i])&(TITLE[i-1]==TITLE[i])){

LIT\_ID[i]<-LITERATURE$TITLE[m]

}

}

}

if (LIT\_ID[i]==""){

row\_LITERATURE\_seq<-row\_LITERATURE\_seq+1

LIT\_ID[i]<-paste0("LIT\_",row\_LITERATURE\_seq)

}

}

database\_add<-data.frame(DOCUMENT=document,

CTRY\_NAME=country,

MUN=municipality,

LONG=longitude,

LAT=latitude,

CELL\_AREA=area,

PLANT\_SPEC=typeplant,

WW\_TYPE=typewaster,

CTRY\_CODE=ctr\_code,

SYSTEM\_ID=SYSTEM\_ID,

SITE\_ID=SITE\_ID,

LIT\_ID=LIT\_ID,

PLANT\_ID=PLANT\_ID,

CELL\_ID=CELL\_ID,

LIT\_TYPE=LIT\_TYPE,

TITLE=TITLE,

YEAR=YEAR,

JOURN\_NAME=JOURN\_NAME,

PUBLISHER=PUBLISHER,

URL=URL,

C\_ORG\_ID=C\_ORG\_ID,

C\_BOD\_IN=BOD\_inlist[[i]],

BOD\_outlist=bod\_outlist,

BOD\_removallist=bod\_removallist,

CW\_TYPE=typewetlands)

database<-rbind(database,database\_add)

}

if (max\_location==2){

country<-rep(COUNTRY[i],max)

longitude<-rep(LONGITUDE[i],max)

latitude<-rep(LATITUDE[i],max)

typewaster<-rep(TYPE\_WASTEWATER[i],max)

typeplant<-rep(TYPE\_PLANT[i],max)

municipality<-rep(MUNICIPALITY[i],max)

area<-rep(AREA[i],max)

bod\_inlist<-rep(BOD\_inlist[[i]],length.out=max)

bod\_removallist<-rep(BOD\_removallist[[i]],length.out=max)

typewetlands<-rep(TYPE\_WETLANDS[i],max)

document<-rep(my.docs$names[i],max)

database\_add<-data.frame(DOCUMENT=document,

COUNTRY=country,

MUNICIPALITY=municipality,

LONGITUDE=longitude,

LATITUDE=latitude,

AREA=area,

TYPE\_PLANT=typeplant,

TYPE\_WASTEWATER=typewaster,

BOD\_inlist=bod\_inlist,

BOD\_outlist=BOD\_outlist[[i]],

BOD\_removallist=bod\_removallist,

TYPE\_WETLANDS=typewetlands)

database<-rbind(database,database\_add)

}

if (max\_location==3){

country<-rep(COUNTRY[i],max)

longitude<-rep(LONGITUDE[i],max)

latitude<-rep(LATITUDE[i],max)

typewaster<-rep(TYPE\_WASTEWATER[i],max)

typeplant<-rep(TYPE\_PLANT[i],max)

municipality<-rep(MUNICIPALITY[i],max)

area<-rep(AREA[i],max)

bod\_inlist<-rep(BOD\_inlist[[i]],length.out=max)

bod\_outlist<-rep(BOD\_outlist[[i]],length.out=max)

typewetlands<-rep(TYPE\_WETLANDS[i],max)

document<-rep(my.docs$names[i],max)

database\_add<-data.frame(DOCUMENT=document,

COUNTRY=country,

MUNICIPALITY=municipality,

LONGITUDE=longitude,

LATITUDE=latitude,

AREA=area,

TYPE\_PLANT=typeplant,

TYPE\_WASTEWATER=typewaster,

BOD\_inlist=bod\_inlist,

BOD\_outlist=bod\_outlist,

BOD\_removallist=BOD\_removallist[[i]],

TYPE\_WETLANDS=typewetlands)

database<-rbind(database,database\_add)

}

}

##Creating tables for exporting information to PostGRE i.e. based in UNU-Flores relational tables framework

##Coding for the ID parameters

##SITE\_ID: SITE\_01,SITE\_02,...

##SYSTEM\_ID: SYSTEM\_01,SYSTEM\_02,...

##LIT\_ID: LIT\_01,LIT\_02,...

##PLANT\_ID: PLANT\_01,PLANT\_02,...

##CELL\_ID: CELL\_01,CELL\_02,...

##C\_ORG\_ID: C\_ORG\_01,C\_ORG\_02,...

##1-SITES

SITES\_R <- data.table(SITE\_ID=character(),

SYSTEM\_ID=character(),

CTRY\_CODE=character(),

SUBN\_ADM\_L=character(),

MUNI=character(),

LAT=numeric(),

LONG=numeric(),

ELEVATION=numeric(),

WW\_TYPE=character(),

HYDR\_CATCH=character(),

TOTAL\_PE=numeric(),

ULT\_DISP\_M=character(),

SW\_CATCH\_A=character(),

stringsAsFactors=FALSE)

SITES\_R$SITE\_ID<-database$SITE\_ID

SITES\_R$SYSTEM\_ID<-database$SYSTEM\_ID

SITES\_R$CTRY\_CODE<-database$CTRY\_CODE

SITES\_R$MUNI<-database$MUNI

SITES\_R$LAT<-database$LAT

SITES\_R$LONG<-database$LONG

SITES\_R$WW\_TYPE<-database$WW\_TYPE

SITES<-rbind(SITES,SITES\_R)

SITES<-setkey(SITES, NULL)

SITES<-unique(SITES[,list(SITE\_ID)])

dbWriteTable(con2, "SITES",

value = SITES, append = TRUE, row.names = FALSE)

##2-COUNTRY

COUNTRY\_R<-data.table(CTRY\_CODE=character(),

CONTINENT=character(),

CTRY\_NAME=character(),

HDI=character(),

HD\_GROUP=character(),

LDC=character(),

LLDC=character(),

SIDS=character(),

stringsAsFactors=FALSE)

COUNTRY\_R$CTRY\_CODE<-database$CTRY\_CODE

COUNTRY\_R$CTRY\_NAME<-database$CTRY\_NAME

COUNTRY<-rbind(COUNTRY,COUNTRY\_R)

COUNTRY<-setkey(COUNTRY, NULL)

COUNTRY<-unique(COUNTRY[,list(CTRY\_CODE)])

dbWriteTable(con2, "COUNTRY",

value = COUNTRY, append = TRUE, row.names = FALSE)

##3-LITERATURE

LITERATURE\_R<-data.table(LIT\_ID=character(),

SITE\_ID=character(),

SYSTEM\_ID=character(),

LIT\_TYPE=character(),

TITLE=character(),

YEAR=character(),

JOURN\_NAME=character(),

PUBLISHER=character(),

URL=character(),

stringsAsFactors=FALSE)

LITERATURE\_R$LIT\_ID<-database$LIT\_ID

LITERATURE\_R$SITE\_ID<-database$SITE\_ID

LITERATURE\_R$LIT\_TYPE<-database$LIT\_TYPE

LITERATURE\_R$TITLE<-database$TITLE

LITERATURE\_R$YEAR<-database$YEAR

LITERATURE\_R$JOURN\_NAME<-database$JOURN\_NAME

LITERATURE\_R$PUBLISHER<-database$PUBLISHER

LITERATURE\_R$URL<-database$URL

LITERATURE<-rbind(LITERATURE,LITERATURE\_R)

LITERATURE<-setkey(LITERATURE, NULL)

LITERATURE<-unique(LITERATURE[,list(LIT\_ID)])

dbWriteTable(con2, "LITERATURE",

value = LITERATURE, append = TRUE, row.names = FALSE)

##4-C\_PLANTS

C\_PLANTS\_R<-data.table(PLANT\_ID=character(),

CELL\_ID=character(),

PLANT\_GEN=character(),

PLANT\_SPEC=character(),

stringsAsFactors=FALSE)

C\_PLANTS\_R$PLANT\_ID<-database$PLANT\_ID

C\_PLANTS\_R$CELL\_ID<-database$CELL\_ID

C\_PLANTS\_R$PLANT\_SPEC<-database$PLANT\_SPEC

C\_PLANTS<-rbind(C\_PLANTS,C\_PLANTS\_R)

C\_PLANTS<-setkey(C\_PLANTS, NULL)

C\_PLANTS<-unique(C\_PLANTS[,list(PLANT\_ID)])

dbWriteTable(con2, "C\_PLANTS",

value = C\_PLANTS, append = TRUE, row.names = FALSE)

##5-CELLS

CELLS\_R<- data.table(CELL\_ID=character(),

SYSTEM\_ID=character(),

C\_PRE\_TRT=character(),

C\_POST\_TRT=character(),

CW\_TYPE=character(),

CELL\_SHAPE=character(),

LENGTH=numeric(),

WIDTH=numeric(),

DIAMETER=numeric(),

DEPTH=numeric(),

CELL\_AREA=numeric(),

CELL\_VOL=numeric(),

INTENS\_MTH=character(),

SUBST\_TYP=character(),

SUBST\_PORO=numeric(),

SSFGSZMIN=numeric(),

SSFGSZMAX=numeric(),

C\_START\_DT=character(),

C\_END\_DATE=character(),

FLO\_REGIME=character(),

LD\_PERIOD=character(),

DRY\_PERIOD=character(),

stringsAsFactors=FALSE)

CELLS\_R$CELL\_ID<-database$CELL\_ID

CELLS\_R$SYSTEM\_ID<-database$SYSTEM\_ID

CELLS\_R$CW\_TYPE<-database$CW\_TYPE

CELLS\_R$CELL\_AREA<-database$CELL\_AREA

CELLS<-rbind(CELLS,CELLS\_R)

CELLS<-setkey(CELLS, NULL)

CELLS<-unique(CELLS[,list(CELL\_ID)])

dbWriteTable(con2, "CELLS",

value = CELLS, append = TRUE, row.names = FALSE)

##6-C\_ORG

C\_ORG <- data.frame(C\_ORG\_ID=character(),

SITE\_ID=character(),

SYSTEM\_ID=character(),

CELL\_ID=character(),

C\_BOD\_IN=numeric(),

C\_BOD\_OUT=numeric(),

C\_BOD\_MT=numeric(),

C\_BOD\_ST=numeric(),

C\_COD\_IN=numeric(),

C\_COD\_OUT=numeric(),

C\_COD\_MT=numeric(),

C\_COD\_ST=numeric(),

C\_TOC\_IN=numeric(),

C\_TOC\_OUT=numeric(),

C\_TOC\_MT=numeric(),

C\_TOC\_ST=numeric(),

C\_ORG\_RD=numeric(),

stringsAsFactors=FALSE)

##7-SYSTEMS

SYSTEMS <- data.frame(SYSTEM\_ID=character(),

SITE\_ID=character(),

SYS\_NAME=character(),

SYS\_AREA=numeric(),

SYS\_SCALE=numeric(),

S\_PRE\_TRT=character(),

S\_POST\_TRT=character(),

DES\_FLOW=numeric(),

DES\_LIFE=character(),

START\_DATE=character(),

END\_DATE=character(),

INV\_COST=numeric(),

NV\_COST\_C=numeric(),

AN\_COST=numeric(),

AN\_COST\_C=numeric(),

stringsAsFactors=FALSE)

pw <- {

"mauricio"

}

drv <- dbDriver("PostgreSQL")

con <- dbConnect(drv, dbname = "postgres",

host = "localhost", port = 5432,

user = "openpg", password = pw)

con2=dbConnect(PostgreSQL(),user = "postgres",password="wetlands",dbname = "postgres")

df\_postgres <- dbGetQuery(con2, "SELECT \* from database")

row\_postgres<-nrow(df\_postgres)+1

lenght<-nrow(database)

row\_postgres\_end<-nrow(df\_postgres)+lenght

ID<-seq(row\_postgres:row\_postgres\_end)

database$ID<-ID

dbWriteTable(con2, "database",

value = database, append = TRUE, row.names = FALSE)

##Copy the files in the folder Documents\_backup and eliminate from the folder Documents

current.folder <- "C:/Users/Mauricio/Desktop/WETLANDS\_EXTRATION/Documents"

new.folder <- "C:/Users/Mauricio/Desktop/WETLANDS\_EXTRATION/Documents\_backup"

list.of.files <- list.files(current.folder,full.names = TRUE)

file.copy(list.of.files, new.folder)

file.remove(list.of.files)