library(NLP)  
library(xml2)  
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)  
library(stringr)  
library(bitops)  
library(RCurl)  
library(rjson)  
library(sp)  
library(postGIStools)  
library(rstudioapi)

path1<-paste0(getwd(),"/Documents")  
path2<-paste0(getwd(),"/Documents\_backup")  
path3<-paste0(getwd(),"/HTML\_LINKS")  
path4<-paste0(getwd(),"/Phantom/phantomjs-2.1.1-windows/bin")

##CLEANING AND CONSTRUCTION OF TEXT MATRIX  
  
## Step 1: To read the files - Folder:  
setwd(path1)  
files\_names<-list.files(getwd())   
doc.list<-lapply(files\_names,read\_file)  
N.docs<-length(doc.list)  
## Step 2: To construct Corpus of documents  
my.docs<-VectorSource(c(doc.list))  
my.docs$names<-sub(".txt","",files\_names)  
mydocs<-Corpus(my.docs)  
## Step 3: To clean special characters  
removeSpecialChars<-function(x) gsub("[^-a-zA-z0-9. ]","",x)  
mydocs<-tm\_map(mydocs,removeSpecialChars)  
## Step 4: To clean stopwords()  
mydocs<-tm\_map(mydocs,removeWords,stopwords('en'))  
## Step 5: To clean doble whitespaces between words  
mydocs<-tm\_map(mydocs,stripWhitespace)  
## Step 6: To clean information before the word "Abstract"  
removeAbstract<-function(x) gsub("(.\*?)A(?i)bstract(?-i)","",x)  
mydocs<-tm\_map(mydocs,removeAbstract)  
## Step 7: To clean information after the word "Conclusions"  
removeConclusions<-function(x) sub("C(?i)onclusion(?-i)(.\*)","",x)  
mydocs<-tm\_map(mydocs,removeConclusions)  
## Step 8: To divide the text in four main parts: Abstract, Introduction, Materials and Methods and Results  
## Step 8.1: Abstract  
removeIntroduction\_down<-function(x) sub("I(?i)ntroduction(?-i)(.\*)","",x)  
str1<-tm\_map(mydocs,removeIntroduction\_down)  
## Step 8.2: Introduction  
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)  
## Step 8.3: Materials and Methods  
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)  
## Step 8.4: Results  
removeResults\_up<-function(x) sub("(.\*?)R(?i)esults(?-i)","",x)  
str4<-tm\_map(mydocs,removeResults\_up)  
## Step 9: To save information in data.frames for posterior construction of VCorpus  
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)  
## Step 10: To create 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$text,stringsAsFactors = F)##Abstract  
VCorpus\_docs1<-VCorpus(DataframeSource(df\_str1))  
df\_str2<-data.frame(doc\_id=seq(1:N.docs),text=extract\_str2$text,stringsAsFactors = F)##Introduction  
VCorpus\_docs2<-VCorpus(DataframeSource(df\_str2))  
df\_str3<-data.frame(doc\_id=seq(1:N.docs),text=extract\_str3$text,stringsAsFactors = F)##Materials and methods  
VCorpus\_docs3<-VCorpus(DataframeSource(df\_str3))  
df\_str4<-data.frame(doc\_id=seq(1:N.docs),text=extract\_str4$text,stringsAsFactors = F)##Results  
VCorpus\_docs4<-VCorpus(DataframeSource(df\_str4))

##PARAMETER\_1 AND PARAMETER\_2: COUNTRY AND COUNTRY\_CODE  
## Step 1: To read the dataset of countries in the world - Folder:  
setwd(path2)  
countries<-read.delim("GEODATASOURCE-COUNTRY.txt", header = TRUE, sep = "\t")  
countries\_names<-as.vector(countries[,4])  
countries\_codes<-as.vector(countries[,1])  
## Step 2: To extract 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))  
## Step 3: To extract the country name  
## Rules  
## 1- SP's: Abstract, Materials and Methods  
## 2- Tokanizer: min=1, max=2  
 for (m in 1:N.docs){  
  
 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]  
 }  
 }  
 }  
 }

##PARAMETER\_3: MUNICIPALITY  
## Step 1: To read the dataset of cities in the world - Folder:  
setwd(path2)  
cities<-fread("GEODATASOURCE-CITIES-FREE.txt", header = TRUE, sep = "\t")  
cities\_data<-as.data.frame(cities)  
## Step 2: To extract the parameter City\_name  
## Rules  
## 1- SP's: Materials and Methods  
## 2- Tokanizer: min=1, max=4  
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]  
 }  
 }  
}

##PARAMETER\_4 AND PARAMETER\_5: COORDINATES (LATITUDE,LONGITUDE)  
##Step 1.1: To extract coordinates using API from Gogle Maps based on the name of the city (MUNICIPALITY)  
LONGITUDE<-c(rep(NA,N.docs))  
LATITUDE<-c(rep(NA,N.docs))  
getGeoData <- function(location){  
 location <- gsub(' ','+',location)  
 geo\_data <- getURL(paste("https://maps.googleapis.com/maps/api/geocode/json?address=",location,"&key=AIzaSyD3r4Ckm47KQ0luZEPy7TuaH8TxMyRC0GQ", sep=""))  
 raw\_data\_2 <- fromJSON(geo\_data)  
 return(raw\_data\_2)  
}  
  
for (m in 1:N.docs){  
   
 if (is.na(MUNICIPALITY[m])=="FALSE"){  
   
 coordinates<-unlist(getGeoData(MUNICIPALITY[m]))  
   
 for (a in 1:length(coordinates)){  
 if (names(coordinates)[a]=="results.geometry.location.lat"){  
 LATITUDE[m]<-unname(coordinates[a])  
 }  
 if (names(coordinates)[a]=="results.geometry.location.lng"){  
 LONGITUDE[m]<-unname(coordinates[a])  
 }  
 }  
 }  
##Step 1.2: To extract coordinates directly from the text  
## Rules  
## 1- SP's: Materials and Methods  
## 2- Tokanizer: min=1, max=1  
   
 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\_6: TYPE OF WETLAND  
##Step 1: To define matrix of keywords  
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")  
  
##Step 2: To define cluster of expressions in matrix of keywords  
cluster<-c("VF",  
 "HSSF",  
 "VF",  
 "HF",  
 "VF",  
 "FWS",  
 "HSSF")  
##Step 3: To extract parameter  
## Rules  
## 1- SP's: Abstract, Materials and Methods, Results  
## 2- Tokanizer: min=1, max=2  
  
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\_6: AREA  
##Step 1: To extract parameter  
## Rules  
## 1- SP's: Abstract, Materials and Methods  
## 2- Tokanizer: min=2, max=2  
  
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)  
  
 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)  
   
 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]  
 }  
 }   
 }  
 }  
}

##PARAMETER\_7: TYPE OF WASTEWATER  
##Step 1: To define matrix of keywords  
matrix\_keyword<-c("^(?i)domestic(?-i)$",  
 "^(?i)industrial(?-i)$",  
 "^(?i)urban runoff(?-i)$",  
 "^(?i)food processing industry(?-i)$",  
 "^(?i)agriculture(?-i)$",  
 "^(?i)agricultural(?-i)$",  
 "^(?i)eutrophic lake water(?-i)$",  
 "^(?i)Dariy milking parlor(?-i)$",  
 "^(?i)Potato starch processing(?-i)$",  
 "^(?i)swine urine(?-i)$")  
  
matrix\_keyword\_nregex<-c("domestic",  
 "industrial",  
 "urban runoff",  
 "food processing industry",  
 "agriculture",  
 "agricultural",  
 "eutrophic lake water",  
 "Dariy milking parlor",  
 "Potato starch processing",  
 "swine urine")  
  
##Step 2: To define cluster of expressions in matrix of keywords  
cluster<-c("MUNICIPAL",  
 "INDUSTRIAL",  
 "STORMWATER",  
 "INDUSTRIAL",  
 "AGRICULTURAL",  
 "AGRICULTURAL",  
 "INDUSTRIAL",  
 "INDUSTRIAL",  
 "INDUSTRIAL")  
##Step 3: To extract parameter  
## Rules  
## 1- SP's: Materials and Methods  
## 2- Tokanizer: min=1, max=3  
  
variable\_cluster<-vector(mode="character", length=10)  
n.words<-length(matrix\_keyword)  
TYPE\_WASTEWATER<-c(rep(NA,N.docs))  
  
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=paste0("(?i)",matrix\_keyword[i],"(?-i)"),kw\_typewastewater$keywords\_typewastewater)))  
 if((length\_filtered\_terms\_type>=1)&(is.na(TYPE\_WASTEWATER[m])=="FALSE"))  
 {  
 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)&(is.na(TYPE\_WASTEWATER[m])=="TRUE"))  
 {  
 c<-c+1  
 variable\_cluster[c]<-cluster[i]  
 TYPE\_WASTEWATER[m]<-cluster[i]  
 }  
 }  
}

##PARAMETER\_8: TYPE OF PLANTS  
##Step 1: To set the folder:  
setwd(path2)  
##Step 2: To read matrix of keywords  
matrix<-as.data.frame(read.xlsx("Type\_plants.xlsx"))  
matrix\_keywords<-as.vector(matrix[,1])  
##Step 3: To read cluster of expressions in matrix of keywords  
variable\_cluster<-vector(mode="character", length=10)  
cluster<-as.vector(matrix[,2])  
##Step 4: To extract parameter  
## Rules  
## 1- SP's: Materials and Methods  
## 2- Tokanizer: min=1, max=4  
  
n.words<-length(matrix\_keywords)  
TYPE\_PLANT<-vector(mode="character", length=N.docs)  
  
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=paste0("(?i)",matrix\_keywords[i],"(?-i)"),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\_9: BOD5 INFLUENT, BOD5 EFLUENT, JOURNAL NAME, PUBLISHER, YEAR OF PUBLICATION, TITLE  
##The following approach works for pre-reviwed articles, which information is available in HTML version e.g. ELSEVIER  
## Step 1: To read the dataset of links - Folder:  
setwd(path3)  
HTML\_links<-read.xlsx("HTML\_links.xlsx")  
HTML\_names<-as.vector(HTML\_links[,1])  
HTML\_url<-as.vector(HTML\_links[,2])  
## Step 2: To set PHANTOM folder:  
setwd(path4)  
## Step 3: To extract the parameters  
  
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<-vector(mode="character", length=N.docs)  
  
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\_names[j]){  
 url<-HTML\_url[j]  
 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: ELSEVIERR  
 journal\_IWA<-grepl("iwaponline",url,ignore.case=FALSE)##Journal:IWA  
 journal\_researchgate<-grepl("researchgate",url,ignore.case=FALSE)##Journal:Researchgate  
 journal\_ncbi<-grepl("ncbi",url,ignore.case=FALSE)##Journal:NCBI  
   
 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()  
 TITLE[m]<-title\_article  
 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],".")  
   
 }  
 }  
 PUBLISHER[m]<-authors  
 publication\_year[1]<-sub("â\u0080\u0093","-",publication\_year[1])  
 part1<-sub("(.\*?)[,]","",publication\_year[1])  
 part2<-sub("(.\*?)[,] ","",part1)  
 part3<-sub("[,](.\*)","",part2)  
 YEAR[m]<-str\_sub(part3, start= -4)  
 p<-as.numeric(YEAR[m])  
   
 if ((is.na(p)=="TRUE")|(p<0)){  
 part1<-sub("(.\*?)[,]","",publication\_year[1])  
 part2<-sub("[,](.\*)","",part1)  
 YEAR[m]<-str\_sub(part2, start= -4)  
 }  
   
 JOURN\_NAME[m]<-journal\_name  
 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()  
 PUBLISHER[m]<-name[1]  
 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()  
 TITLE[m]<-journal\_name[2]  
 name<-page\_html %>% html\_nodes("div.auths")%>%html\_text()  
 PUBLISHER[m]<-name  
 publication\_year<-page\_html %>% html\_nodes("div.cit")%>%html\_text()  
 JOURN\_NAME[m]<-sub("[.](.\*)","",publication\_year[1])  
 part1<-sub("(.\*?)[.] ","",publication\_year[1])  
 part2<-sub("[;](.\*)","",part1)  
 YEAR[m]<-gsub("[a-zA-z ]","",part2)  
 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  
   
 if(length(table)>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]])  
}

##CREATION MATRIX OF PARAMETER: The parameters extracted from the text are integrated in a data.frame  
##Step 1: To open the database in PostgreSQL  
  
pw <- {  
 "mauricio"  
}##pw=password for accesing to the database  
  
drv <- dbDriver("PostgreSQL")  
con <- dbConnect(drv, dbname = "postgres",  
 host = "localhost", port = 5432,  
 user = "openpg", password = pw)  
  
##Step 2: To read the tables in the PostgreSQL database: SITES, SYSTEMS, C\_PLANTS, LITERATURE, C\_ORG, CELLS, COUNTRY,LITERATURE  
con2=dbConnect(PostgreSQL(),user = "postgres",password="wetlands",dbname = "postgres")  
SITES <- as.data.frame(dbGetQuery(con2, 'SELECT \* from "SITES"'))  
SITES$CTRY\_CODE[is.na(SITES$CTRY\_CODE)=="TRUE"]<-"NULL"  
SITES$MUNI[is.na(SITES$MUNI)=="TRUE"]<-"NULL"  
SITES$LONG[is.na(SITES$LONG)=="TRUE"]<-0  
SYSTEMS<-as.data.frame(dbGetQuery(con2, 'SELECT \* from "SYSTEMS"'))  
C\_PLANTS<-as.data.frame(dbGetQuery(con2, 'SELECT \* from "C\_PLANTS"'))  
C\_PLANTS$PLANT\_SPEC[is.na(C\_PLANTS$PLANT\_SPEC)=="TRUE"]<-"NULL"  
LITERATURE<-as.data.frame(dbGetQuery(con2, 'SELECT \* from "LITERATURE"'))  
LITERATURE$TITLE[is.na(LITERATURE$TITLE)=="TRUE"]<-"NULL"  
C\_ORG<-as.data.frame(dbGetQuery(con2, 'SELECT \* from "C\_ORG"'))  
CELLS<-as.data.frame(dbGetQuery(con2, 'SELECT \* from "CELLS"'))  
COUNTRY<-as.data.frame(dbGetQuery(con2, 'SELECT \* from "COUNTRY"'))  
LITERATURE\_comparison<-data.frame(LIT\_ID=LITERATURE$LIT\_ID,TITLE=LITERATURE$TITLE)  
C\_PLANTS\_comparison<-data.frame(PLANT\_ID=C\_PLANTS$PLANT\_ID,PLANT\_SPEC=C\_PLANTS$PLANT\_SPEC)  
SITES\_comparison<-data.frame(SITE\_ID=SITES$SITE\_ID,CTRY\_CODE=SITES$CTRY\_CODE,MUNI=SITES$MUNI,LONG=SITES$LONG)  
  
##Step 3: To read the length of the tables in the PostgreSQL database  
row\_C\_ORG<-nrow(C\_ORG)  
row\_LITERATURE<-nrow(LITERATURE)  
row\_PLANT<-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)  
b<-row\_SYSTEM+1  
  
##Step 4: To change the NA's value by "NULL"or 0 (Depending the type of character: numeric or string)  
MUNICIPALITY[is.na(MUNICIPALITY)=="TRUE"]<-"NULL"  
country\_code[is.na(country\_code)=="TRUE"]<-"NULL"  
country[is.na(country)=="TRUE"]<-"NULL"  
LONGITUDE[is.na(LONGITUDE)=="TRUE"]<-0  
LATITUDE[is.na(LATITUDE)=="TRUE"]<-0  
CITATION[is.na(CITATION)=="TRUE"]<-"NULL"  
URL[is.na(URL)=="TRUE"]<-"NULL"  
JOURN\_NAME[is.na(JOURN\_NAME)=="TRUE"]<-"NULL"  
PUBLISHER[is.na(PUBLISHER)=="TRUE"]<-"NULL"  
LIT\_TYPE[is.na(LIT\_TYPE)=="TRUE"]<-"NULL"  
YEAR<-as.numeric(YEAR)  
YEAR[is.na(YEAR)=="TRUE"]<-0  
TITLE[is.na(TITLE)=="TRUE"]<-"NULL"  
TYPE\_PLANT[TYPE\_PLANT==""]<-"NULL"  
TYPE\_WASTEWATER[is.na(TYPE\_WASTEWATER)=="TRUE"]<-"NULL"  
TYPE\_WETLANDS[TYPE\_WETLANDS==""]<-"NULL"  
AREA[AREA==""]<-"NULL"  
  
##Step 5: To declare a data.table with column names equal to the column names of the tables in the PostgreSQL database  
  
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=numeric(),  
 LAT=numeric(),  
 LIT\_TYPE=character(),  
 TITLE=character(),  
 YEAR=numeric(),  
 JOURN\_NAME=character(),  
 PUBLISHER=character(),  
 URL=character(),  
 CELL\_AREA=character(),  
 PLANT\_SPEC=character(),  
 WW\_TYPE=character(),  
 C\_BOD\_IN=numeric(),  
 C\_BOD\_OUT=numeric(),  
 CW\_TYPE=character(),  
 stringsAsFactors=FALSE)  
  
for (v in 1:N.docs){  
   
 l1<-length(BOD\_inlist[[v]])  
 l2<-length(BOD\_outlist[[v]])  
 location<-c(l1,l2)  
 max\_location<-which(location == max(location), arr.ind = TRUE)  
 max<-max(location)  
 control<-0  
   
 if(length(max\_location)==2){  
   
 LIT\_ID<-vector(mode="character",length=max)  
 lit\_type<-rep(LIT\_TYPE[v],max)  
 url<-rep(URL[v],max)  
 year<-rep(YEAR[v],max)  
 journ\_name<-rep(JOURN\_NAME[v],max)  
 publisher<-rep(PUBLISHER[v],max)  
 Country<-rep(country[v],max)  
 ctr\_code<-rep(country\_code[v],max)  
 longitude<-rep(LONGITUDE[v],max)  
 latitude<-rep(LATITUDE[v],max)  
 typewetlands<-rep(TYPE\_WETLANDS[v],max)  
 typewaster<-rep(TYPE\_WASTEWATER[v],max)  
 typeplant<-rep(TYPE\_PLANT[v],max)  
 municipality<-rep(MUNICIPALITY[v],max)  
 area<-rep(AREA[v],max)  
 bod\_outlist<-rep(BOD\_outlist[[v]],max)  
 bod\_inlist<-rep(BOD\_inlist[[v]],max)  
 document<-rep(my.docs$names[v],max)  
 SITE\_ID<-vector(mode="character",length=max)  
 PLANT\_ID<-vector(mode="character",length=max)  
 SYSTEM\_ID<-vector(mode="character",length=max)  
 C\_ORG\_ID<-vector(mode="character",length=max)  
 CELL\_ID<-vector(mode="character",length=max)  
 row\_SYSTEM\_end<-row\_SYSTEM+max  
 cont<-0  
 for (r in row\_SYSTEM\_seq:row\_SYSTEM\_end){  
 cont<-cont+1  
 SYSTEM\_ID[cont]<-paste0("SYSTEM\_",b)  
 C\_ORG\_ID[cont]<-paste0("C\_ORG\_ID\_",r)  
 CELL\_ID[cont]<-paste0("CELL\_",r)  
 }  
 b<-b+1  
 row\_SYSTEM<-row\_SYSTEM\_end  
 row\_SYSTEM\_seq<-row\_SYSTEM+1  
 title<-rep(TITLE[v],max)  
   
 for (i in 1:max){  
   
 row\_SITES\_comparison<-nrow(SITES\_comparison)  
   
 for (m in 1:row\_SITES\_comparison){  
   
 if ((SITES\_comparison$CTRY\_CODE[m]==ctr\_code[i])&(SITES\_comparison$MUNI[m]==municipality[i])&(SITES\_comparison$LONG[m]==longitude[i])){  
 temporal\_sites<-as.vector(SITES\_comparison$SITE\_ID)  
 SITE\_ID[i]<-temporal\_sites[m]  
 }  
 }  
   
 if (SITE\_ID[i]==""){  
 row\_SITES\_seq<-row\_SITES\_comparison+1  
 SITE\_ID[i]<-paste0("SITE\_",row\_SITES\_seq)  
 add\_SITES<-data.frame(SITE\_ID=SITE\_ID[i],CTRY\_CODE=ctr\_code[i],MUNI=municipality[i],LONG=longitude[i])  
 SITES\_comparison<-rbind(SITES\_comparison,add\_SITES)  
 }  
 }  
   
 for (i in 1:max){  
   
 row\_C\_PLANTS\_comparison<-nrow(C\_PLANTS\_comparison)  
   
 for (m in 1:row\_C\_PLANTS\_comparison){  
   
 if (C\_PLANTS\_comparison$PLANT\_SPEC[m]==typeplant[i]){  
 temporal\_plant<-as.vector(C\_PLANTS\_comparison$PLANT\_ID)  
 PLANT\_ID[i]<-temporal\_plant[m]  
 }  
 }  
   
 if (PLANT\_ID[i]==""){  
 row\_PLANT\_seq<-row\_C\_PLANTS\_comparison+1  
 PLANT\_ID[i]<-paste0("PLANT\_",row\_PLANT\_seq)  
 add\_PLANT<-data.frame(PLANT\_ID=PLANT\_ID[i],PLANT\_SPEC=typeplant[i])  
 C\_PLANTS\_comparison<-rbind(C\_PLANTS\_comparison,add\_PLANT)  
 }  
 }  
   
   
 for (i in 1:max){  
   
 row\_LITERATURE\_comparison<-nrow(LITERATURE\_comparison)  
   
 for (m in 1:row\_LITERATURE\_comparison){  
   
 if (LITERATURE\_comparison$TITLE[m]==title[i]){  
 temporal\_lit<-as.vector(LITERATURE\_comparison$LIT\_ID)  
 LIT\_ID[i]<-temporal\_lit[m]  
 }  
 }  
   
 if (LIT\_ID[i]==""){  
 row\_LITERATURE\_seq<-row\_LITERATURE\_comparison+1  
 LIT\_ID[i]<-paste0("LIT\_",row\_LITERATURE\_seq)  
 add\_LITERATURE<-data.frame(LIT\_ID=LIT\_ID[i],TITLE=title[i])  
 LITERATURE\_comparison<-rbind(LITERATURE\_comparison,add\_LITERATURE)  
 }  
 }  
   
 database\_add<-data.table(DOCUMENT=document,  
 CTRY\_NAME=Country,  
 MUNI=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,  
 CW\_TYPE=typewetlands,  
 C\_ORG\_ID=C\_ORG\_ID,  
 C\_BOD\_IN=BOD\_inlist[[v]],  
 C\_BOD\_OUT=BOD\_outlist[[v]])  
   
 database<-rbind(database,database\_add)  
 control<-1  
 }  
   
 if ((max\_location==1)&(control!=1)){  
 year<-rep(YEAR[v],max)  
 lit\_type<-rep(LIT\_TYPE[v],max)  
 url<-rep(URL[v],max)  
 journ\_name<-rep(JOURN\_NAME[v],max)  
 publisher<-rep(PUBLISHER[v],max)  
 Country<-rep(country[v],max)  
 ctr\_code<-rep(country\_code[v],max)  
 title<-rep(TITLE[v],max)  
 longitude<-rep(LONGITUDE[v],max)  
 latitude<-rep(LATITUDE[v],max)  
 typewetlands<-rep(TYPE\_WETLANDS[v],max)  
 typewaster<-rep(TYPE\_WASTEWATER[v],max)  
 typeplant<-rep(TYPE\_PLANT[v],max)  
 municipality<-rep(MUNICIPALITY[v],max)  
 area<-rep(AREA[v],max)  
 document<-rep(my.docs$names[v],max)  
 SITE\_ID<-vector(mode="character",length=max)  
 PLANT\_ID<-vector(mode="character",length=max)  
 LIT\_ID<-vector(mode="character",length=max)  
 bod\_outlist<-rep(BOD\_outlist[[v]],length.out=max)  
 SYSTEM\_ID<-vector(mode="character",length=max)  
 C\_ORG\_ID<-vector(mode="character",length=max)  
 CELL\_ID<-vector(mode="character",length=max)  
 row\_SYSTEM\_end<-row\_SYSTEM+max  
 cont<-0  
   
 for (r in row\_SYSTEM\_seq:row\_SYSTEM\_end){  
 cont<-cont+1  
 SYSTEM\_ID[cont]<-paste0("SYSTEM\_",b)  
 C\_ORG\_ID[cont]<-paste0("C\_ORG\_ID\_",r)  
 CELL\_ID[cont]<-paste0("CELL\_",r)  
 }  
 b<-b+1  
 row\_SYSTEM<-row\_SYSTEM\_end  
 row\_SYSTEM\_seq<-row\_SYSTEM+1  
  
   
 for (i in 1:max){  
   
 row\_SITES\_comparison<-nrow(SITES\_comparison)  
   
 for (m in 1:row\_SITES\_comparison){  
   
 if ((SITES\_comparison$CTRY\_CODE[m]==ctr\_code[i])&(SITES\_comparison$MUNI[m]==municipality[i])&(SITES\_comparison$LONG[m]==longitude[i])){  
 temporal\_sites<-as.vector(SITES\_comparison$SITE\_ID)  
 SITE\_ID[i]<-temporal\_sites[m]  
 }  
 }  
   
 if (SITE\_ID[i]==""){  
 row\_SITES\_seq<-row\_SITES\_comparison+1  
 SITE\_ID[i]<-paste0("SITE\_",row\_SITES\_seq)  
 add\_SITES<-data.frame(SITE\_ID=SITE\_ID[i],CTRY\_CODE=ctr\_code[i],MUNI=municipality[i],LONG=longitude[i])  
 SITES\_comparison<-rbind(SITES\_comparison,add\_SITES)  
 }  
 }  
   
   
 for (i in 1:max){  
   
 row\_C\_PLANTS\_comparison<-nrow(C\_PLANTS\_comparison)  
   
 for (m in 1:row\_C\_PLANTS\_comparison){  
   
 if (C\_PLANTS\_comparison$PLANT\_SPEC[m]==typeplant[i]){  
 temporal\_plant<-as.vector(C\_PLANTS\_comparison$PLANT\_ID)  
 PLANT\_ID[i]<-temporal\_plant[m]  
 }  
 }  
   
 if (PLANT\_ID[i]==""){  
 row\_PLANT\_seq<-row\_C\_PLANTS\_comparison+1  
 PLANT\_ID[i]<-paste0("PLANT\_",row\_PLANT\_seq)  
 add\_PLANT<-data.frame(PLANT\_ID=PLANT\_ID[i],PLANT\_SPEC=typeplant[i])  
 C\_PLANTS\_comparison<-rbind(C\_PLANTS\_comparison,add\_PLANT)  
 }  
 }  
   
   
 for (i in 1:max){  
   
 row\_LITERATURE\_comparison<-nrow(LITERATURE\_comparison)  
   
 for (m in 1:row\_LITERATURE\_comparison){  
   
 if (LITERATURE\_comparison$TITLE[m]==title[i]){  
 temporal\_lit<-as.vector(LITERATURE\_comparison$LIT\_ID)  
 LIT\_ID[i]<-temporal\_lit[m]  
 }  
 }  
   
 if (LIT\_ID[i]==""){  
 row\_LITERATURE\_seq<-row\_LITERATURE\_comparison+1  
 LIT\_ID[i]<-paste0("LIT\_",row\_LITERATURE\_seq)  
 add\_LITERATURE<-data.frame(LIT\_ID=LIT\_ID[i],TITLE=title[i])  
 LITERATURE\_comparison<-rbind(LITERATURE\_comparison,add\_LITERATURE)  
 }  
 }  
   
 database\_add<-data.table(DOCUMENT=document,  
 CTRY\_NAME=Country,  
 MUNI=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,  
 CW\_TYPE=typewetlands,  
 C\_ORG\_ID=C\_ORG\_ID,  
 C\_BOD\_IN=BOD\_inlist[[v]],  
 C\_BOD\_OUT=bod\_outlist)  
   
 database<-rbind(database,database\_add)  
 }  
   
 if ((max\_location==2)&(control!=1)){  
 year<-rep(YEAR[v],max)  
 lit\_type<-rep(LIT\_TYPE[v],max)  
 url<-rep(URL[v],max)  
 journ\_name<-rep(JOURN\_NAME[v],max)  
 publisher<-rep(PUBLISHER[v],max)  
 Country<-rep(country[v],max)  
 title<-rep(TITLE[v],max)  
 ctr\_code<-rep(country\_code[v],max)  
 longitude<-rep(LONGITUDE[v],max)  
 latitude<-rep(LATITUDE[v],max)  
 typewetlands<-rep(TYPE\_WETLANDS[v],max)  
 typewaster<-rep(TYPE\_WASTEWATER[v],max)  
 typeplant<-rep(TYPE\_PLANT[v],max)  
 municipality<-rep(MUNICIPALITY[v],max)  
 area<-rep(AREA[v],max)  
 document<-rep(my.docs$names[v],max)  
 SITE\_ID<-vector(mode="character",length=max)  
 PLANT\_ID<-vector(mode="character",length=max)  
 LIT\_ID<-vector(mode="character",length=max)  
 bod\_inlist<-rep(BOD\_inlist[[v]],length.out=max)  
 SYSTEM\_ID<-vector(mode="character",length=max)  
 C\_ORG\_ID<-vector(mode="character",length=max)  
 CELL\_ID<-vector(mode="character",length=max)  
 row\_SYSTEM\_end<-row\_SYSTEM+max  
 cont<-0  
 for (r in row\_SYSTEM\_seq:row\_SYSTEM\_end){  
 cont<-cont+1  
 SYSTEM\_ID[cont]<-paste0("SYSTEM\_",b)  
 C\_ORG\_ID[cont]<-paste0("C\_ORG\_ID\_",r)  
 CELL\_ID[cont]<-paste0("CELL\_",r)  
 }  
 b+1  
 row\_SYSTEM<-row\_SYSTEM\_end  
 row\_SYSTEM\_seq<-row\_SYSTEM+1  
   
 for (i in 1:max){  
   
 row\_SITES\_comparison<-nrow(SITES\_comparison)  
   
 for (m in 1:row\_SITES\_comparison){  
   
 if ((SITES\_comparison$CTRY\_CODE[m]==ctr\_code[i])&(SITES\_comparison$MUNI[m]==municipality[i])&(SITES\_comparison$LONG[m]==longitude[i])){  
 temporal\_sites<-as.vector(SITES\_comparison$SITE\_ID)  
 SITE\_ID[i]<-temporal\_sites[m]  
 }  
 }  
   
 if (SITE\_ID[i]==""){  
 row\_SITES\_seq<-row\_SITES\_comparison+1  
 SITE\_ID[i]<-paste0("SITE\_",row\_SITES\_seq)  
 add\_SITES<-data.frame(SITE\_ID=SITE\_ID[i],CTRY\_CODE=ctr\_code[i],MUNI=municipality[i],LONG=longitude[i])  
 SITES\_comparison<-rbind(SITES\_comparison,add\_SITES)  
 }  
 }  
   
   
 for (i in 1:max){  
   
 row\_C\_PLANTS\_comparison<-nrow(C\_PLANTS\_comparison)  
   
 for (m in 1:row\_C\_PLANTS\_comparison){  
   
 if (C\_PLANTS\_comparison$PLANT\_SPEC[m]==typeplant[i]){  
 temporal\_plant<-as.vector(C\_PLANTS\_comparison$PLANT\_ID)  
 PLANT\_ID[i]<-temporal\_plant[m]  
 }  
 }  
   
 if (PLANT\_ID[i]==""){  
 row\_PLANT\_seq<-row\_C\_PLANTS\_comparison+1  
 PLANT\_ID[i]<-paste0("PLANT\_",row\_PLANT\_seq)  
 add\_PLANT<-data.frame(PLANT\_ID=PLANT\_ID[i],PLANT\_SPEC=typeplant[i])  
 C\_PLANTS\_comparison<-rbind(C\_PLANTS\_comparison,add\_PLANT)  
 }  
 }  
   
   
 for (i in 1:max){  
   
 row\_LITERATURE\_comparison<-nrow(LITERATURE\_comparison)  
   
 for (m in 1:row\_LITERATURE\_comparison){  
   
 if (LITERATURE\_comparison$TITLE[m]==title[i]){  
 temporal\_lit<-as.vector(LITERATURE\_comparison$LIT\_ID)  
 LIT\_ID[i]<-temporal\_lit[m]  
 }  
 }  
   
 if (LIT\_ID[i]==""){  
 row\_LITERATURE\_seq<-row\_LITERATURE\_comparison+1  
 LIT\_ID[i]<-paste0("LIT\_",row\_LITERATURE\_seq)  
 add\_LITERATURE<-data.frame(LIT\_ID=LIT\_ID[i],TITLE=title[i])  
 LITERATURE\_comparison<-rbind(LITERATURE\_comparison,add\_LITERATURE)  
 }  
 }  
 database\_add<-data.table(DOCUMENT=document,  
 CTRY\_NAME=Country,  
 MUNI=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,  
 CW\_TYPE=typewetlands,  
 C\_ORG\_ID=C\_ORG\_ID,  
 C\_BOD\_IN=bod\_inlist,  
 C\_BOD\_OUT=BOD\_outlist[[v]])  
   
 database<-rbind(database,database\_add)  
 }   
}

database[database=="NULL"]<-NA  
database[database==0]<-NA

head(database)

## DOCUMENT CTRY\_CODE SYSTEM\_ID SITE\_ID LIT\_ID PLANT\_ID CELL\_ID CTRY\_NAME  
## 1: Doc1 TS SYSTEM\_15 SITE\_2 LIT\_2 PLANT\_2 CELL\_15 Tunisia  
## 2: Doc10 IT SYSTEM\_16 SITE\_3 LIT\_3 PLANT\_2 CELL\_16 Italy  
## 3: Doc11 TU SYSTEM\_17 SITE\_4 LIT\_4 <NA> CELL\_17 Turkey  
## 4: Doc12 FR SYSTEM\_18 SITE\_5 LIT\_5 <NA> CELL\_18 France  
## 5: Doc13 CH SYSTEM\_19 SITE\_6 LIT\_6 PLANT\_4 CELL\_19 China  
## 6: Doc2 IT SYSTEM\_20 SITE\_15 LIT\_7 PLANT\_5 CELL\_20 Italy  
## C\_ORG\_ID MUNI LONG LAT LIT\_TYPE  
## 1: C\_ORG\_ID\_15 Tunis NA NA PRE-REVIEW ARTICLE  
## 2: C\_ORG\_ID\_16 <NA> NA NA PRE-REVIEW ARTICLE  
## 3: C\_ORG\_ID\_17 Marmara NA NA PRE-REVIEW ARTICLE  
## 4: C\_ORG\_ID\_18 Savoie NA NA PRE-REVIEW ARTICLE  
## 5: C\_ORG\_ID\_19 Guangzhou NA NA PRE-REVIEW ARTICLE  
## 6: C\_ORG\_ID\_20 <NA> 14.41667 37.26667 PRE-REVIEW ARTICLE  
## TITLE  
## 1: Constructed wetland as a low cost and sustainable solution for wastewater treatment adapted to rural settlements: the Chorfech wastewater treatment pilot plant.  
## 2: Hydraulic behaviour and removal efficiencies of two H-SSF constructed wetlands for wastewater reuse with different operational life.  
## 3: Effect of recirculation on organic matter removal in a hybrid constructed wetland system.  
## 4: Recirculation on a single stage of vertical flow constructed wetland: Treatment limits and operation modes  
## 5: Performance study of vegetated sequencing batch coal slag bed treating domestic wastewater in suburban area.  
## 6: Growth and biomass production of different plant species in two different constructed wetland systems in Sicily  
## YEAR JOURN\_NAME  
## 1: 2011 Water Sci Technol  
## 2: 2011 Water Sci Technol  
## 3: 2011 Water Sci Technol  
## 4: 2012 Ecological Engineering  
## 5: 2008 Bioresour Technol  
## 6: 2009 Desalination  
## PUBLISHER  
## 1: Ghrabi A1, Bousselmi L, Masi F, Regelsberger M.  
## 2: Barbagallo S1, Cirelli GL, Marzo A, Milani M, Toscano A.  
## 3: Ayaz SC1, Findik N, Akça L, Erdogan N, Kinaci C.  
## 4: Prost-Boucle S.,and P. Molle.  
## 5: Chan SY1, Tsang YF, Chua H, Sin SN, Cui LH.  
## 6: Barbera Antonio C.,Giuseppe L. Cirelli,Valeria Cavallaro,Isabella Di Silvestro,Paola Pacifici,Vincenzo Castiglione,Attilio Toscano,and Mirco Milani.  
## URL  
## 1: https://www.ncbi.nlm.nih.gov/pubmed/22049731  
## 2: https://www.ncbi.nlm.nih.gov/pubmed/22214048  
## 3: https://www.ncbi.nlm.nih.gov/pubmed/21977661  
## 4: https://www.sciencedirect.com/science/article/pii/S0925857412000729  
## 5: https://www.ncbi.nlm.nih.gov/pubmed/17709244  
## 6: https://www.sciencedirect.com/science/article/pii/S0011916409004275  
## CELL\_AREA PLANT\_SPEC WW\_TYPE  
## 1: <NA> Phragmites Australis AGRICULTURAL  
## 2: 2000 Phragmites Australis <NA>  
## 3: 13.7-18 <NA> MUNICIPAL  
## 4: 1.11.6 <NA> <NA>  
## 5: <NA> Cyperus alternifolius MUNICIPAL-AGRICULTURAL  
## 6: 4.5 Phragmites spp;Phragmites Australis <NA>  
## C\_BOD\_IN C\_BOD\_OUT CW\_TYPE  
## 1: NA NA HF-VF-HSSF  
## 2: NA NA HSSF  
## 3: NA NA VF-HF-HSSF  
## 4: NA NA VF-HSSF  
## 5: NA NA VF  
## 6: NA NA VF-HSSF