```
################################################################################
##### RNA-SEQ DATA ANALYSIS - PSEUDOMONAS S5 - R FUNCTIONS
################################################################################
##### Spring 2016 - MLS - UNIL - Marie Zufferey
##### !!! some hard-coded parameters, file shape and formats not checked
library(edgeR)
library(readr)
library(ggplot2)
library(pheatmap)
library(reshape2)
library(rtracklayer)
library(magrittr)
library(dplyr)
library(grid)
library(ggthemes)
library(genoPlotR)
library(gtable)
library(tm)           # wc
library(SnowballC)    # wc
library(wordcloud)    # wc
library(RColorBrewer) # wc
setwd("PATH_TO_FOLDER")

#########################################################################
###### Read abundance files
#########################################################################

getRawCounts <- function(abd_fld){
  # abd_fld <- path_to_abundances <- "../data/abundances/"
  files <- dir(abd_fld, pattern=".tsv$")
  # paste the path to your files - make sure that you have the path in front the files
  files <- paste0(abd_fld, files)
  samples <- paste0(rep(c('LM','SA','WL','WR'), each = 4), rep(1:4,4))

  # read kallisto files again to get counts of transcripts
  transcripts <- readDGE(files,
                         columns = c(1,4),
                         group = rep(c('LM','SA','WL','WR'), each = 4),
                         labels = samples)

  # get counts of transcripts
  tr_counts <- transcripts$counts
  return(tr_counts)
}

#########################################################################
###### Create DGEList object
#########################################################################

getRawData <- function(abd_fld, threshold=T){

    # get counts of transcripts
    tr_counts <- getRawCounts(abd_fld)
    tr_cpms <- cpm(tr_counts)
    keep <- rowSums(tr_cpms > 1) >= 4
    tr_counts_clean <- tr_counts[keep,]
#    sum(!keep)
    dt <- DGEList(counts = tr_counts_clean,
                  group = rep(c('LM','SA','WL','WR'), each = 4))

    return(dt)
}


#########################################################################
###### "Normalization" (library size and dispersion)
#########################################################################

getDGE <- function(abd_fld, threshold=T){

  dt <- getRawData(abd_fld, threshold=T)

  dt <- calcNormFactors(dt)
  dt <- estimateCommonDisp(dt)
  dt <- estimateTagwiseDisp(dt)

  return(dt)
}

#########################################################################
###### Get mean for each conditions (mean of replicates)
#########################################################################

getMeanData <- function(data){
  data %<>% as.data.frame
  LMcol <- colnames(data)[regexpr("LM", colnames(data))>0]
  data$LM <- rowMeans(subset(data, select = LMcol), na.rm = TRUE)
  SAcol <- colnames(data)[regexpr("SA", colnames(data))>0]
  data$SA <- rowMeans(subset(data, select = SAcol), na.rm = TRUE)
  WRcol <- colnames(data)[regexpr("WR", colnames(data))>0]
  data$WR <- rowMeans(subset(data, select = WRcol), na.rm = TRUE)
  WLcol <- colnames(data)[regexpr("WL", colnames(data))>0]
  data$WL <- rowMeans(subset(data, select = WLcol), na.rm = TRUE)
  dataMean <- data[,c("LM", "SA", "WR", "WL")]
  return(dataMean)
}

#########################################################################
###### RPKM normalization (retrieve from Kamil and Andrea's tutorial)
#########################################################################

## rpkm calculations
myrpkm <- function(counts, lengths) {
  rate <- counts / lengths
  return(rate / sum(counts) * 1e9)
}

#########################################################################
###### Pairwise exact test of differential expression
#########################################################################

# returns a dataframe with i.a. logFC, logCPM, pval, FDR
# for a pairwise exact test for a given pair of conditions
# genes are identified with "Transcript" column (e.g. "S5_genome_4011")
# a column "Pair" is added with information about conditions tested (e.g. "LM/SA")
pairTestGenes <- function(dt, sel_genes, cond1, cond2){
  # dt is the DGEList object
  # data_annot <- annot <- read.csv("../data/annot_mot.csv", sep=",")
  # cond1 <- "LM";cond2 <- "SA"
  # cond1 and cond2 the conditions to test
  # sel_genes <- data_annot$Gene_position
  # the genes we want to select (position = transcript name)
  # this imports the .tsv files in your R environment
  # select only the motility
  de.tr <- exactTest(dt, pair = c(cond1, cond2))
  tT.transcripts <- topTags(de.tr, n = nrow(dt), p.value = 0.05)
  det.list <- tT.transcripts$table
  det.list %<>% as.data.frame
  det.list$Transcript <- row.names(det.list)
  det.list$Pair <- rep(paste0(cond1,"/", cond2), nrow(det.list))
  det.list <- det.list[which(rownames(det.list) %in% sel_genes),]
  rownames(det.list) <- NULL
  return(det.list)
}


#########################################################################
###### DIFFERENT FUNCTIONS FOR COLOR SETTING
#########################################################################

# Function to set colors according to motility type
# (use for ggplot axis labels)
setMyCol_mot <- function(mot){
  if(is.na(mot)){          # must be the first condition tested !
    return("black")
  }
  else if(mot == "pili"){
    return("darkblue")
```

```r
  }else if(mot =="flagella"){
    return("red3")
  }else if(mot =="swarming"){
    return("forestgreen")
  }else{
    return("black")
  }
}

setMyCol_smear <- function(mot){
  if(is.na(mot)){           # must be the first condition tested !
    return("black")
  }
  else if(mot == "pili"){
    return("darkblue")
  }else if(mot =="flagella"){
    return("gold")
  }else if(mot =="swarming"){
    return("forestgreen")
  }else{
    return("black")
  }
}

setMyFill <- function(mot){
  if(is.na(mot)){           # must be the first condition tested !
    return(NA)
  }
  else if(mot == "pili"){
    return("darkblue")
  }else if(mot =="flagella"){
    return("red3")
  }else if(mot =="swarming"){
    return("forestgreen")
  }else{
    return(NA)
  }
}


# Function to set colors according to significance
# (use for ggplot dots)
setMyCol_fdr <- function(fdr){
  if(fdr < 0.05){
    return("green")
  }else{
    return("gray48")
  }
}

# Function to set colors according to sign of FC
# (used for barplot)
setMyCol_fc <- function(x){
  if(is.na(x)){
    return("black")
  }else if(x<0){
    return("red")
  }else if(x>0){
    return("green")
  }else{
    return("black")
  }
}

# Function to set colors according to sign of the strand
# (used for line of logCPM, plot with 2 y axis)
setMyCol_strand <- function(x){
  if(is.na(x)){
    return("black")
  }else if(x=="+"){
    return("maroon3")
  }else if(x=="-"){
    return("mediumaquamarine")
  }else{
    return("black")
  }
}

setMyCol_PCA <- function(x){
  if(is.na(x)){
    return("black")
  }else if(x=="flagella"){
    return("tomato")
  }else if(x=="pili"){
    return("lightblue")
  }else if(x=="swarming"){
    return("yellowgreen")
  }else{
    return("black")
  }
}
setMyPch_PCA <- function(x){
  if(is.na(x)){
    return(4)
  }else if(x=="flagella"){
    return(1)
  }else if(x=="pili"){
    return(2)
  }else if(x=="swarming"){
    return(3)
  }else{
    return(4)
  }
}


#########################################################################
###### HEATMAPS FOR DIFFERENTIAL EXPRESSION
#########################################################################

heatmapPairs <- function(data, gbkData, annotData, tit){
  # retrieve the chromosomal position
  # data <- allPairs
  # annotData <- annot
  # gbkData <- gbkData
  # tit <- "logFC motility associated genes all pairs of conditions"
  #nrow(data)  #298
  data %<>% left_join(., gbkData, by =c("Transcript"="Locus_tag"))
  #nrow(data)  #298
  data %<>% left_join(., annotData, by =c("Transcript"="Gene_position"))

  data$Start %<>% as.character %>% as.numeric  # because stored as factor
  data <- data[order(data$Start),]  # order by starting position

  # need "unique" for the labels (and their colors) of the x axis
  # because x-axis in ggplot is the start position
  # duplicated because of different conditions
  uniqData <- unique(data[,c("Start", "Motility_type", "Gene_name")])
  # nrow(uniqData) # 82
  # -> ok, tested with labels=1:82 in ggplot axis.text.x

  # set colors of x-axis labels according to motility type
  labCol <- sapply(uniqData$Motility_type, function(x){setMyCol_mot(x)})

  # replace LM/SA -> SA vs. LM
  data$Pair <- gsub('(^.{2})/(.{2}$)', '\\2 vs. \\1', data$Pair)

  p <- ggplot(data, aes(x=Pair,y=as.factor(Start)))+
    geom_tile(aes(fill = logFC))+
    scale_fill_gradient2(low="red", high="green", mid="black", name="logFC")+
    scale_y_discrete("Genes",labels=uniqData$Gene_name)+
    scale_x_discrete("Conditions", expand=c(0,0))+
    ggtitle(tit)+
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.background = element_rect(fill="gray"),
          legend.title = element_text(face="bold"),
          axis.text.x = element_text(angle=45, hjust=1, size=10, colour="black"),
          axis.text.y = element_text(size=10,colour=labCol),
          axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          plot.title =  element_text(colour="darkslateblue", size=20))
  return(p)
}


##########################################################################
###### SCATTERPLOT MATRIX OF DIFFERENTIAL EXPRESSION COMPARISONS
##########################################################################

# written by François Gillet ("Numerical ecology with R") !!!
# -> MZ: color changed
panel.hist <- function(x, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5) )
  h <- hist(x, plot = FALSE)
  breaks <- h$breaks; nB <- length(breaks)
  y <- h$counts; y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y, col="steelblue", ...)
}

# https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/pairs.html
# -> MZ: changed behaviour with NA and spearman as method,
# -> the sign and the size
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- (cor(x, y,use= "pairwise.complete.obs", method="spearman"))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * (r+0.1))
}


##########################################################################
###### VOLCANO PLOT FOR ALL GENES (motility genes coloured; if wished, top X genes labelled)
##########################################################################
# with label for the top X genes if plotAnnot=T, label the myT genes most differentially expressed
volcanoAllPoints<- function(dt, annot, cond1, cond2, plotAnnot=F, myT=5){
  #abd_fld <- "../data/abundances/"
  #dt <- getDGE(abd_fld)
  #cond1 = "LM"
  #cond2 = "SA"

  de.tr <- exactTest(dt, pair = c(cond1, cond2))

  tT.transcripts <- topTags(de.tr, n = nrow(dt), p.value = 0.05)
  det.list <- tT.transcripts$table

  allDF <- det.list
  allDF$Locus <- rownames(allDF)
  rownames(allDF) <- NULL
  allDF$log10p <- -log10(allDF$FDR)

  allExp <- full_join(allDF, annot, by=c("Locus"="Gene_position"))
  colPoints <- sapply(allExp$Motility_type, function(x){setMyCol_mot(x)})

  colFill <- sapply(allExp$Motility_type, function(x){setMyFill(x)})

  ytit <-expression(bold(paste("-",log[10]~" p-value (FDR)")))
  co <- c("darkblue",  "red3", "forestgreen")
  limx <- max(abs(allExp$logFC[which(!is.na(allExp$logFC))]))
  p <-  ggplot(allExp, aes(x=logFC, y=log10p,group=1))+
    geom_point(size=2, colour=colPoints, fill=colFill, shape=21)+
    #scale_y_continuous(bquote("-log"~ [10]~ "(Pval)"))+
    scale_y_continuous(ytit)+
    scale_x_continuous("log 2 fold change", limits=c(-limx,limx))+
    scale_fill_manual(name="ello", values=co)+
    ggtitle(paste0("Volcano plot ", cond2, " vs. ", cond1))+
    theme(axis.title.y = element_text( colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5,
size=12,lineheight=5,hjust=1),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank(),
          panel.grid.minor.x=element_blank(),
          panel.grid.major.x=element_blank())
  ### Add locus label for the 5 gene with most changing expression
  # have tried first with the annotation
  # but the most changing expression was not annotated (so after that see blast perl script !!!)
  #fction <- gbkData[,c("Product", "Gene_id", "Locus_tag")]
  #fctionData <- full_join(fction, allExp, by=c("Locus_tag"="Locus"))

  if(plotAnnot){

    topX <- allExp[order(abs(allExp$logFC), decreasing=T),][1:myT,]

    for(i in 1:myT){
      p <- p + ggplot2::annotate("text", x = topX$logFC[i], y = (topX$log10p[i]+3), size=4,
                        label = gsub("S5_genome_", "",topX$Locus[i]), colour="darkorange4")
    }
  }
  return(p)
}



#########################################################################
###### VOLCANO PLOTS OF MOTILITY-ASSOCIATED GENES WITH LABELS
#########################################################################

volcanoMotilityPointsAnnot <- function(dt, annot, cond1, cond2){
  #abd_fld <- "../data/abundances/"
  #dt <- getDGE(abd_fld)
  #cond1 = "LM"
  #cond2 = "SA"
  # Do the normalization according to the dispersions of the genes.

  de.tr <- exactTest(dt, pair = c(cond1, cond2))

  tT.transcripts <- topTags(de.tr, n = nrow(dt), p.value = 0.05)
  det.list <- tT.transcripts$table

  allDF <- det.list
  allDF$Locus <- rownames(allDF)
  rownames(allDF) <- NULL
  allDF$log10p <- -log10(allDF$FDR)
  allExp <- full_join(allDF, annot, by=c("Locus"="Gene_position"))
  allExp <- allExp[!is.na(allExp$Motility_type),]
  ytit <-expression(bold(paste("-",log[10]~" p-value (FDR)")))
  p <-  ggplot(allExp, aes(x=logFC, y=log10p,group=1,label=Gene_name))+
    geom_label(aes(fill = factor(Motility_type)), show.legend=T, size=3, color="black")+
    guides(fill=guide_legend(title=NULL))+
    scale_y_continuous(ytit)+
    scale_x_continuous("Log 2 fold change",
    #                   expand=c(0,0))+
    limits=c(-2.75,2.75))+
# limits = c(-max(abs(allExp$logFC)),max(abs(allExp$logFC))))+
#     limits=c(-max(abs(allExp$logFC))-1,-max(abs(allExp$logFC))+1))+
  scale_fill_manual(name = "Motility type", values=c("tomato", "lightblue","yellowgreen"))+
  #scale_color_manual(name = "Motility type", values=c("red3", "darkblue","forestgreen"))+
    ggtitle(paste0("Volcano plot ", cond2, " vs. ", cond1))+
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5,
size=12,lineheight=5,hjust=1),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank(),
          panel.grid.minor.x=element_blank(),
           panel.grid.major.x=element_blank())
  return(p)
}

########################################################################
###### BARPLOT FOR KEGG PATHWAYS
########################################################################
# print only the pathways for which at least threshold occurences

keggHisto <- function(dt, annot, path_S5, cond1, cond2, threshold){

    de.tr <- exactTest(dt, pair = c(cond1, cond2))
    tT.transcripts <- topTags(de.tr, n = nrow(dt), p.value = 0.05)
    det.list <- tT.transcripts$table
    det.list$S5_genome_id <- rownames(det.list)
    # suspense
    dataKegg <- inner_join(det.list, path_S5, by="S5_genome_id")

    countPos <- plyr::count(dataKegg[which(dataKegg$logFC>0),], "path_name")
    colnames(countPos) <- c("path_name", "freq.pos")
    countNeg <- plyr::count(dataKegg[which(dataKegg$logFC<0),], "path_name")
    colnames(countNeg) <- c("path_name", "freq.neg")

    allCounts <- full_join(countPos, countNeg, "path_name")

    plotCounts <- melt(allCounts,
                       id.vars = "path_name" )
    colnames(plotCounts) <- c("path_name", "variable", "value")

    plotCounts <- plotCounts[order(plotCounts$value, decreasing=T),]

    if(nrow(plotCounts)>20){
      plotCounts <- plotCounts[c(1:20),]
    }
#     plotCounts <- plotCounts[which(plotCounts$value>threshold),]
    plotCounts <- plotCounts[which(plotCounts$path_name!="<NA>"),]
    tit = paste0("Up- and downregulation by pathway (", cond1, "/", cond2, ")")
    p <- ggplot(plotCounts,
                aes(x=path_name, y=value, fill=factor(variable)))+
      geom_bar(stat="identity", position="dodge")+
      coord_flip()+
      ggtitle(tit)+
      ylab("Number of occurences")+
      #scale_y_continuous(limits = c(0, max(plotCounts$value)))+
      xlab("Pathways")+
      scale_fill_manual(name="",values=c("green", "red"),
                        label=c("Upregulation",
                                "Downregulation")) +
      theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
            axis.title.x = element_text(face="bold", colour="#990000", size=15),
            axis.text.y = element_text(colour="black", size=12),
            axis.text.x  = element_text(angle=90, vjust=0.5,
size=16,lineheight=5,hjust=1),
            plot.title =  element_text(colour="darkslateblue", size=15),
            panel.grid.minor.y=element_blank(),
            #          panel.grid.major.y=element_blank(),
            panel.grid.minor.x=element_blank(),
            panel.grid.major.x=element_blank())

    return(p)
}

########################################################################
###### BARPLOT GO CATEGORIES
########################################################################
# print only the categories for which at least threshold occurences
# if withMot = T => print motility category bar, regardless of its number of occurence

goHisto <- function(dt, annot, go_S5, cond1, cond2, threshold, withMot=F){

  de.tr <- exactTest(dt, pair = c(cond1, cond2))
  tT.transcripts <- topTags(de.tr, n = nrow(dt), p.value = 0.05)
  det.list <- tT.transcripts$table
  det.list$S5_genome_id <- rownames(det.list)
```

```r
  # suspense
  dataGO <- inner_join(det.list, go_S5, by="S5_genome_id")

  countPos <- plyr::count(dataGO[which(dataGO$logFC>0),], "GO.Term")
  colnames(countPos) <- c("GO.Term", "freq.pos")
  countNeg <- plyr::count(dataGO[which(dataGO$logFC<0),], "GO.Term")
  colnames(countNeg) <- c("GO.Term", "freq.neg")

  allCounts <- full_join(countPos, countNeg, "GO.Term")

  plotCounts <- melt(allCounts,
                     id.vars = "GO.Term" )
  plotCounts <- na.omit(plotCounts)
  colnames(plotCounts) <- c("GO.Term", "variable", "value")
  if(withMot){
    plotCounts <- plotCounts[which(plotCounts$value>threshold | plotCounts$GO.Term=="motility"),]
  }else{
    plotCounts <- plotCounts[which(plotCounts$value>threshold),]
  }
  plotCounts <- plotCounts[which(plotCounts$GO.Term!="<NA>"),]
  tit = paste0("Up- and downregulation by GO (", cond2, " vs. ", cond1, ")")
  p <- ggplot(plotCounts,
              aes(x=GO.Term, y=value, fill=factor(variable)))+
    geom_bar(stat="identity", position="dodge")+
    coord_flip()+
    ggtitle(tit)+
    ylab("Number of occurences")+
    #scale_y_continuous(limits = c(0, max(plotCounts$value)))+
    xlab("GO category")+
    scale_fill_manual(name="",values=c("green", "red"),
                      label=c("Upregulation",
                              "Downregulation")) +
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5,
size=16,lineheight=5,hjust=1),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          #          panel.grid.major.y=element_blank(),
          panel.grid.minor.x=element_blank(),
          panel.grid.major.x=element_blank())
  return(p)
}

##########################################################################
###### UP- AND DOWNREGULATION OF GENES
##########################################################################
# firstly used 2 y-axis with the right y-axis giving average log CPM (one line on the plot)
# but later, this was removed
# adapted from http://heareresearch.blogspot.ch
# annot <- read.csv("../data/annot_mot.csv", sep=",")
# gbkData <- read.csv("../data/S5_gbk_short.csv", sep=",")
# abd_fld <- "../data/abundances/"
# dt <- getDGE(abd_fld)  # compute it once here
# dataLMSA <- pairTestGenes(dt, annot$Gene_position , "LM", "SA")  #1
# tit="logFC and logCPM (SA vs. LM)"
# annotData <- annot
# dataPair = table for pairwise test
# dataPair <- dataLMSA

fc_barAndCpm_line <- function(dataPair, annotData, gbkData, tit, pt=T){
  #### DATA PREPARATION
  data <- left_join(dataPair, gbkData, by =c("Transcript"="Locus_tag"))
  #nrow(data)  #49
  data %<>% left_join(., annotData, by =c("Transcript"="Gene_position"))

  data$Start %<>% as.character %>% as.numeric  # because stored as factor
  data <- data[order(data$Start),]  # order by starting position

  # set colors of x-axis labels according to motility type
  labCol <- sapply(data$Motility_type, function(x){setMyCol_mot(x)})
  # set colors of line point according to strand
  pointCol  <- sapply(data$Strand, function(x){setMyCol_strand(x)})
  # set colors of bars according to sign of logFC
  barCol <- sapply(data$logFC, function(x){setMyCol_fc(x)})

  #### PLOT
  grid.newpage()
  # the two plots
  pFC <- ggplot(data, aes(x=as.factor(Start), y=logFC)) +
    scale_y_continuous("log 2 fold change", limits = c(-max(abs(data$logFC)),max(abs(data$logFC))))+
    scale_x_discrete("Genes", labels=data$Gene_name)+
    ggtitle(tit)+
    geom_bar(stat="identity",position = "identity", colour=barCol, fill=barCol) +
    geom_hline(yintercept=1,linetype="dashed", color="gray48")+
    geom_hline(yintercept=-1,linetype="dashed", color="gray48")+
#     theme_few()+
  # theme(axis.text.x = element_text(angle = 90, hjust = 1, colour=labCol,vjust=0.5),
  theme(axis.text.x = element_text(angle = 90, hjust = 1, colour=labCol,vjust=0.5, size=5),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank(),
          panel.grid.minor.x= element_line(color = "red"))
#           panel.grid.minor.x=element_line(colour="black", size=1))


  pCPM <- ggplot(data, aes(x=as.factor(Start),y= logCPM, group=1)) +
    geom_line(colour = "darkorange4") +
#     geom_point(size=2, colour=pointCol)+
    scale_x_discrete("Genes", labels=data$Gene_name)+
    scale_y_continuous("mean log CPM")+
    theme_few()+
    theme(panel.background = element_rect(fill = NA),
          axis.title.y = element_text(colour="darkorange4", angle=90))

  if(pt){
    pCPM <- pCPM + geom_point(size=2, colour=pointCol)
  }

  # extract gtable
  g1 <- ggplot_gtable(ggplot_build(pFC))
  g2 <- ggplot_gtable(ggplot_build(pCPM))

  # overlap the panel of 2nd plot on that of 1st plot
  pp <- c(subset(g1$layout, name == "panel", se = t:r))
  g <- gtable_add_grob(g1, g2$grobs[[which(g2$layout$name == "panel")]], pp$t,
                       pp$l, pp$b, pp$l)

  # axis tweaks
  ia <- which(g2$layout$name == "axis-l")
  ga <- g2$grobs[[ia]]
  ax <- ga$children[[2]]
  ax$widths <- rev(ax$widths)
  ax$grobs <- rev(ax$grobs)
  ax$grobs[[1]]$x <- ax$grobs[[1]]$x - unit(1, "npc") + unit(0.15, "cm")
  g <- gtable_add_cols(g, g2$widths[g2$layout[ia, ]$l], length(g$widths) - 1)
  g <- gtable_add_grob(g, ax, pp$t, length(g$widths) - 1, pp$b)

  ia2 <- which(g2$layout$name == "ylab")
  ga2 <- g2$grobs[[ia2]]
  ga2$rot <- 90
  g <- gtable_add_cols(g, g2$widths[g2$layout[ia2, ]$l], length(g$widths) - 1)
  g <- gtable_add_grob(g, ga2, pp$t, length(g$widths) - 1, pp$b)

 return(pFC)
#return(g)
}



########################################################################
###### BLOXPLOT FOR ALL CONDITIONS, BY MOTILITY TYPE
########################################################################

#data = getMeanData(dt$counts)
#data$Transcript <- rownames(data)
#data = left_join(annot, data, by=c("Gene_position" = "Transcript"))
```

```
boxplotMotGenes <- function(data, annot, maintit, chemo=F, allRep = F){
  meltData <- melt(data, id=c("Gene_position", "Gene_name", "Motility_type"))
  meltData$value <- log(meltData$value)
  meltData$Var <- substr(meltData$variable, 1, 2)
  if(chemo){
  fillCol <-c( "orangered2", "dodgerblue3", "forestgreen","thistle")
  } else{
    fillCol <- c( "orangered2", "dodgerblue3", "forestgreen")
  }
  if(allRep){
    my_x <- "variable"
  }else{
    my_x <- "Var"
  }
  p <-ggplot(meltData, aes_string(x=my_x, y="value", fill="Motility_type")) +
    geom_boxplot()+
    ggtitle(maintit)+
    scale_y_continuous("log(RPKM)")+
    #scale_colour_discrete(name  ="Experimental conditions")+
    scale_x_discrete("Experimental conditions")+
  scale_fill_manual(name="Gene associated with", values=fillCol)+
#    stat_summary(fun.y=mean, geom="line", aes(group=Motility_type, colour=fillCol))  +
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black"),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.x  = element_text(angle=90, vjust=0.5, size=10),
          plot.title =  element_text(colour="darkslateblue", size=20),
          legend.text=element_text(size=15),
          legend.title=element_text(size=15, face="bold"),
          panel.grid.minor.y=element_blank(),panel.grid.major.y=element_blank())+
    guides(colour=FALSE)
  return(p)
}

#########################################################################
###### MULTIPLOT GGPLOT
#########################################################################

# used for plotting multiple ggplots on the same window
# Source : http://www.cookbook-r.com/
# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}


#########################################################################
###### RDA PLOT
#########################################################################
# used to plot RDA result

rdaFct <- function(mes, env){
  # where mes is the response matrix and env the explanatory variables
  mes<-as.data.frame(scale(mes))
  env<-as.data.frame(scale(env))
  # rda(Y,X,W) where Y is the response matrix,X is the matrix of explanatory variables
  # and W is an optional matrix of covariables
  rda.site<-rda(mes~.,env)   #same as : rda(mes,env), but need formula for anova
  #summary(rda.site)
  coef(rda.site)
  # percentage of variance explained by axis 1
  a1 <- rda.site$CCA$eig[1]/rda.site$tot.chi
  xl =paste0("RDA1 - ", round(a1*100,2), "%")
  # percentage of variance explained by axis 2
  a2 <- rda.site$CCA$eig[2]/rda.site$tot.chi
  yl =paste0("RDA2 - ", round(a2*100,2), "%")
  # R-squared and adjusted-R2
  (R2 <- RsquareAdj(rda.site)$r.squared)
  (R2_adj <- RsquareAdj(rda.site)$adj.r.squared)
  plot(rda.site, xlab=xl, ylab=yl)
}


#########################################################################
###### GENO PLOT R
#########################################################################
# used to plot RDA result
# used to compare genome position of motility-associated genes Pseud. S5 and Pseud. f. Pf5

geneMapMot <- function(P_mot,agl=45, mt="") {
  names1 <- P_mot$fonc_short_pf
  names2 <- P_mot$fonc_short_pf
  starts1 <- as.numeric(as.character(P_mot$Start_ps))
  starts2 <- as.numeric(as.character(P_mot$start_pf))
  ends1 <- as.numeric(as.character(P_mot$End_ps))
  ends2 <- as.numeric(as.character(P_mot$end_pf))
  strands1 <- as.character(P_mot$Strand_ps)
  strands2 <- as.character(P_mot$strand_pf)
  strands1 <- sapply(strands1, numStrand)
  strands2 <- sapply(strands2, numStrand)

  #cols1 <- palette(rainbow(87))
  cols1 <- colorRampPalette(c("red", "blue"))(length(starts1))
  cols2 <- colorRampPalette(c("red", "blue"))(length(starts2))
  df1 <- data.frame(name=names1, start=starts1, end=ends1, strand=strands1, col=cols1)
  dna_seg1 <- dna_seg(df1)

  df2 <- data.frame(name=names2, start=starts2, end=ends2, strand=strands2, col=cols2)
  dna_seg2 <- dna_seg(df2)
  #plot_gene_map(list(dna_seg1, dna_seg2))
  df3 <- df1
  colnames(df3) %<>% paste0(., "1")
  df4 <- df2
  colnames(df4) %<>% paste0(., "2")
  df5 <- cbind(df3,df4)
  df5 <- df5[,c(2,3,7,8)]
  df5$col <- colorRampPalette(c("red", "blue"))(nrow(df5))
  comparison1 <- as.comparison(df5)
  dna_segs = list(dna_seg1, dna_seg2)
  mid_pos <- middle(dna_segs[[1]])

  annot <-  annotation(x1=mid_pos, x2=rep(NA, length(mid_pos)), rot=rep(agl, length(mid_pos)),
                      text=dna_segs[[1]]$name)

  plot_gene_map(dna_segs=dna_segs, comparisons=list(comparison1), annotations=annot, scale=T,
                annotation_cex=0.9, main = mt,
                dna_seg_labels=c("Pseudomonas S5","Pseudomonas fluorescens Pf5"))
}


##########################################################################
###### PCA plots - Gillet & Borcard 2012
##########################################################################

"cleanplot.pca" <- function(res.pca, mycol="black",ax1=1, ax2=2, point=FALSE,
                            ahead=0.07, cex=0.7)
{
  # A function to draw two biplots (scaling 1 and scaling 2) from an object
  # of class "rda" (PCA or RDA result from vegan's rda() function)
  #
  # License: GPL-2
  # Authors: Francois Gillet & Daniel Borcard, 24 August 2012
  # MODIFICATION MZUFFEREY: add % for each axis
  require("vegan")

  e1 <- round((res.pca$CA$eig[ax1]/sum(res.pca$CA$eig)*100),2)
  e2 <- round((res.pca$CA$eig[ax2]/sum(res.pca$CA$eig)*100),2)

  xl <- paste0("PC",ax1, " - ", e1, "%")
  yl <- paste0("PC",ax2, " - ", e2, "%")

  par(mfrow=c(1,2))
  p <- length(res.pca$CA$eig)

  # Scaling 1: "species" scores scaled to relative eigenvalues
  sit.sc1 <- scores(res.pca, display="wa", scaling=1, choices=c(1:p))
  spe.sc1 <- scores(res.pca, display="sp", scaling=1, choices=c(1:p))
  plot(res.pca, choices=c(ax1, ax2), display=c("wa", "sp"), type="n",
       main="PCA - scaling 1", scaling=1, xlab=xl, ylab=yl)
  if (point)
  {
    points(sit.sc1[,ax1], sit.sc1[,ax2], pch=20, col=mycol)
    text(res.pca, display="wa", choices=c(ax1, ax2), cex=cex, pos=3, scaling=1, col=mycol)
  }
  else
  {
    text(res.pca, display="wa", choices=c(ax1, ax2), cex=cex, scaling=1, col=mycol)
  }
  text(res.pca, display="sp", choices=c(ax1, ax2), cex=cex, pos=4,
       col="red", scaling=1)
  arrows(0, 0, spe.sc1[,ax1], spe.sc1[,ax2], length=ahead, angle=20, col="red")
  pcacircle(res.pca)

  # Scaling 2: site scores scaled to relative eigenvalues
  sit.sc2 <- scores(res.pca, display="wa", choices=c(1:p))
  spe.sc2 <- scores(res.pca, display="sp", choices=c(1:p))
  plot(res.pca, choices=c(ax1,ax2), display=c("wa","sp"), type="n",
       main="PCA - scaling 2",xlab=xl, ylab=yl, col=mycol)
  if (point) {
    points(sit.sc2[,ax1], sit.sc2[,ax2], pch=20, col=mycol)
    text(res.pca, display="wa", choices=c(ax1 ,ax2), cex=cex, pos=3, col=mycol)
  }
  else
  {
    text(res.pca, display="wa", choices=c(ax1, ax2), cex=cex, col=mycol)
  }
  text(res.pca, display="sp", choices=c(ax1, ax2), cex=cex, pos=4, col="red")
  arrows(0, 0, spe.sc2[,ax1], spe.sc2[,ax2], length=ahead, angle=20, col="red")
}
```

```
"pcacircle" <- function (pca)
{
  # Draws a circle of equilibrium contribution on a PCA plot
  # generated from a vegan analysis.
  # vegan uses special constants for its outputs, hence
  # the 'const' value below.

  eigenv <- pca$CA$eig
  p <- length(eigenv)
  n <- nrow(pca$CA$u)
  tot <- sum(eigenv)
  const <- ((n - 1) * tot)^0.25
  radius <- (2/p)^0.5
  radius <- radius * const
  symbols(0, 0, circles=radius, inches=FALSE, add=TRUE, fg=2)
}



#*************************************************************************************************
#*************************************************************************************************
# FURTHER FUNCTIONS FINALLY NOT USED NEITHER FOR THE REPORT NOR FOR THE SUPP. DATA
#*************************************************************************************************
#*************************************************************************************************

#**************************************************
#################### PLOT 1: line for 1 condition
#**************************************************
# plot line of the logFC value for comparison between given 2 conditions, for motility-associated genes
# (in fact not useful because can be done with next function ...)
## plot for logfc data for a given pair of conditions
# separately for each test
# x-axis: genes ordered by their position along chromosome
# y-axis: logFC
# dots colored by significance, x-axis label by motility type
# with line connecting the dots
# annotData -> contains match between transcript ID and gene ID
# gbkData -> contains at least the 2 columns "Start" "Locus_tag" (=transcript ID)
# tit -> string with title for the plot
# returns the created plot

oneTestLinePlot <- function(data, gbkData, annotData, tit){

  data %<>% left_join(., gbkData, by =c("Transcript"="Locus_tag"))

  data %<>% left_join(., annotData, by =c("Transcript"="Gene_position"))

  data$Start %<>% as.character %>% as.numeric  # because stored as factor
  data <- data[order(data$Start),]  # order by starting position

  # set colors of x-axis labels according to motility type
  labCol <- sapply(data$Motility_type, function(x){setMyCol_mot(x)})

  # set colors of the points according to significance (FDR)
  pointCol  <- sapply(data$FDR, function(x){setMyCol_fdr(x)})

  p <- ggplot(data, aes(x=as.factor(Start), y=logFC, group=1))+
    geom_point(size=3, colour=pointCol)+
    geom_line(colour="hotpink")+
    scale_y_continuous("Log2 fold change")+
    scale_x_discrete("Motility genes", labels=data$Gene_name)+
    ggtitle(tit)+
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5, size=12,lineheight=5,hjust=1,
colour=labCol),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank())
  return(p)
}
```

```r
#**************************************************
#################### PLOT 2: many lines in one plot (log2FC for pairs of conditions)
#**************************************************
# plot the lines of logFC values for all comparisons for motility-associated genes
# same parameters as the previous function
# used to plot in the same figure more than one pair of conditions tested
# x-axis: genes ordred by chromosomal position
# y-axis log2fc
# colour of the dots: green if significant (FDR), else gray

manyLinePlot <- function(data, gbkData, annotData, tit){
  data %<>% left_join(., gbkData, by =c("Transcript"="Locus_tag"))
  #nrow(data)  #298
  data %<>% left_join(., annotData, by =c("Transcript"="Gene_position"))

  data$Start %<>% as.character %>% as.numeric  # because stored as factor
  data <- data[order(data$Start),]  # order by starting position

  # need "unique" for the labels (and their colors) of the x axis
  # because x-axis in ggplot is the start position
  # duplicated because of different conditions
  uniqData <- unique(data[,c("Start", "Motility_type", "Gene_name")])
  # nrow(uniqData) # 82
  # -> ok, tested with labels=1:82 in ggplot axis.text.x

  # set colors of x-axis labels according to motility type
  labCol <- sapply(uniqData$Motility_type, function(x){setMyCol_mot(x)})

  # set colors of the points according to significance (FDR)
  # is.numeric(data$FDR) # TRUE -> ok
  pointCol  <- sapply(data$FDR, function(x){setMyCol_fdr(x)})

  p <- ggplot(data, aes(x=as.factor(Start), y=logFC, group=Pair))+
    geom_point(size=3, colour=pointCol, aes(group=Pair))+
    geom_line(aes(colour=Pair,group=Pair))+
    scale_color_discrete(name="Tested pairs")+
    scale_y_continuous("Log2 fold change")+
    # scale_x_discrete("Motility genes", labels=1:82)+
    scale_x_discrete("Motility genes", labels=uniqData$Gene_name)+
    ggtitle(tit)+
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5, size=12,lineheight=5,hjust=1,
colour=labCol),
          plot.title =  element_text(colour="darkslateblue", size=15),
          legend.title = element_text(face="bold"),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank())

  return(p)
}

#**************************************************
#################### PLOT 3: smear plot for 2 given conditions, colour by motility
#**************************************************
# custom smearplot function (at the end used built-in edgeR function in the supp. data)

smearPlotsAllPoints <- function(dt, annot, cond1, cond2){

  de.tr <- exactTest(dt, pair = c(cond1, cond2))

  # allDF <- tT.transcripts$table   # => no one near 0 log2fc
  allDF <- de.tr$table
  allDF$Locus <- rownames(allDF)
  rownames(allDF) <- NULL
  allExp <- full_join(allDF, annot, by=c("Locus"="Gene_position"))
  colPoints <- sapply(allExp$Motility_type, function(x){setMyCol_mot(x)})
  lm <- max(abs(allExp$logFC))
  p <-  ggplot(allExp, aes(x=logCPM, y=logFC,group=1))+
    geom_point(size=2, colour=colPoints)+
    scale_y_continuous("Log 2 fold change",limits = c(-lm,lm))+
    scale_x_continuous("Average logCPM")+
    ggtitle(paste0("Smear plot ", cond1, "/", cond2))+
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5,
size=12,lineheight=5,hjust=1),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank(),
          panel.grid.minor.x=element_blank(),
          panel.grid.major.x=element_blank())
  return(p)
}

#**************************************************
##### PLOT 4 : smear plot with colour for given 2 conditions to test - motility-associated genes only
#**************************************************
# custom smearplot function (at the end used built-in edgeR function in the supp. data)
# plot only motility-associated genes

smearPlotsMotility <- function(dt, annot, cond1, cond2){
  de.tr <- exactTest(dt, pair = c(cond1, cond2))
  allDF <- de.tr$table
  allDF$Locus <- rownames(allDF)
  rownames(allDF) <- NULL
  allExp <- full_join(allDF, annot, by=c("Locus"="Gene_position"))
  allExp <- allExp[!is.na(allExp$Motility_type),]
  lm <- max(abs(allExp$logFC))
  p <-  ggplot(allExp, aes(x=logCPM, y=logFC,group=1))+
    geom_point(aes(color=factor(Motility_type)),size=2)+
    scale_y_continuous("Log 2 fold change",limits = c(-lm,lm))+
    scale_x_continuous("Average logCPM")+
    scale_color_manual(name = "Motility type", values=c("red3", "forestgreen", "darkblue"))+
    ggtitle(paste0("Smear plot ", cond1, "/", cond2))+
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5,
size=12,lineheight=5,hjust=1),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank(),
          panel.grid.minor.x=element_blank(),
          panel.grid.major.x=element_blank())
  return(p)
}

#**************************************************
##### PLOT 5 : idem plot 4 but with labels
#**************************************************
# custom smearplot function (with labels instead of points)

smearPlotsMotility_label <- function(dt, annot, cond1, cond2){
  de.tr <- exactTest(dt, pair = c(cond1, cond2))
  allDF <- de.tr$table
  allDF$Locus <- rownames(allDF)
  rownames(allDF) <- NULL
  allExp <- full_join(allDF, annot, by=c("Locus"="Gene_position"))
  allExp <- allExp[!is.na(allExp$Motility_type),]
  lm <- max(abs(allExp$logFC))
  p <-  ggplot(allExp, aes(x=logCPM, y=logFC,group=1, label=Gene_name))+
    geom_label(aes(fill = factor(Motility_type)), show.legend=T)+
    guides(fill=guide_legend(title=NULL))+
    scale_y_continuous("Log 2 fold change",limits = c(-lm,lm))+
    scale_x_continuous("Average logCPM")+
    ggtitle(paste0("Smear plot ", cond1, "/", cond2))+
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5,
size=12,lineheight=5,hjust=1),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank(),
```

```r
          panel.grid.minor.x=element_blank(),
          panel.grid.major.x=element_blank())
  return(p)
}

#**************************************************
##### PLOT 6 : volcano plot for motility associated genes
#**************************************************
# volcano plot for motility-associated genes (points not labels)

volcanoMotilityPoints<- function(dt, annot, cond1, cond2){

  de.tr <- exactTest(dt, pair = c(cond1, cond2))

  tT.transcripts <- topTags(de.tr, n = nrow(dt), p.value = 0.05)
  det.list <- tT.transcripts$table

  allDF <- det.list
  allDF$Locus <- rownames(allDF)
  rownames(allDF) <- NULL
  allDF$log10p <- -log10(allDF$FDR)
  allExp <- full_join(allDF, annot, by=c("Locus"="Gene_position"))
  allExp <- allExp[!is.na(allExp$Motility_type),]
  p <-  ggplot(allExp, aes(x=logFC, y=log10p,group=1))+
    geom_point(aes(color=factor(Motility_type), name="Motility type"),size=2)+
    scale_y_continuous("Log 2 fold change")+
    scale_x_continuous("Average logCPM")+
    scale_color_manual(name = "Motility type", values=c("red3", "forestgreen", "darkblue"))+
    ggtitle(paste0("Smear plot ", cond1, "/", cond2))+
    theme(axis.title.y = element_text(face="bold", colour="#990000", size=15),
          axis.title.x = element_text(face="bold", colour="#990000", size=15),
          axis.text.y = element_text(colour="black", size=12),
          axis.text.x  = element_text(angle=90, vjust=0.5,
size=12,lineheight=5,hjust=1),
          plot.title =  element_text(colour="darkslateblue", size=15),
          panel.grid.minor.y=element_blank(),
          panel.grid.major.y=element_blank(),
          panel.grid.minor.x=element_blank(),
          panel.grid.major.x=element_blank())
  return(p)
}

#**************************************************
##### PLOT 7 : BARPLOT FC
#**************************************************
# barplot similar to the one used for the report

fc_bar <- function(dataPair, annotData, gbkData, tit){
  #### DATA PREPARATION
  data <- left_join(dataPair, gbkData, by =c("Transcript"="Locus_tag"))
  #nrow(data)  #49
  data %<>% left_join(., annotData, by =c("Transcript"="Gene_position"))

  data$Start %<>% as.character %>% as.numeric  # because stored as factor
  data <- data[order(data$Start),]  # order by starting position

  # set colors of x-axis labels according to motility type
  labCol <- sapply(data$Motility_type, function(x){setMyCol_mot(x)})

  # set colors of bars according to sign of logFC
  barCol <- sapply(data$logFC, function(x){setMyCol_fc(x)})

  #### PLOT
  pFC <- ggplot(data, aes(x=as.factor(Start), y=logFC)) +
    scale_y_continuous("log 2 fold change", limits = c(-max(abs(data$logFC)),max(abs(data$logFC))))+
    scale_x_discrete("Genes", labels=data$Gene_name)+
    ggtitle(tit)+
    geom_bar(stat="identity",position = "identity", colour=barCol, fill=barCol) +
    theme_few()+
    theme(axis.text.x = element_text(angle = 90, hjust = 1, colour=labCol),
          plot.title =  element_text(colour="darkslateblue", size=15))

  return(pFC)
}
```

```
#*************************************************
##### Plot 8: word clouds
#*************************************************
# used for the presentation 1st slide
# adapted from https://georeferenced.wordpress.com

wordCld <- function(x, myStopW="",minF=1, maxW=200){
  text <- as.character(x)
  # corpus = liste de documents
  # Charger les données comme un corpus
  docs <- Corpus(VectorSource(text))
  toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
  docs <- tm_map(docs, toSpace, "/")
  docs <- tm_map(docs, toSpace, "@")
  docs <- tm_map(docs, toSpace, "\\|")
  # convert lower case
  docs <- tm_map(docs, content_transformer(tolower))
  # remove numbers
  docs <- tm_map(docs, removeNumbers)
  # remove stop words english
  docs <- tm_map(docs, removeWords, stopwords("english"))
  # remove stop words passed in parameters
  docs <- tm_map(docs, removeWords, myStopW)
  # remove punctuation signs
  docs <- tm_map(docs, removePunctuation)
  # remove white spaces
  docs <- tm_map(docs, stripWhitespace)

  dtm <- TermDocumentMatrix(docs)
  m <- as.matrix(dtm)
  v <- sort(rowSums(m),decreasing=TRUE)
  d <- data.frame(word = names(v),freq=v)
  wordcloud(words = d$word, freq = d$freq, min.freq = minF,
            max.words=maxW, random.order=FALSE, rot.per=0.35, random.color=F,
            colors=c
('#9e0142','#d53e4f','#f46d43','#fdae61','#fee08b','#e6f598','#abdda4','#66c2a5','#3288bd','#5e4fa2'))
  # author colors:
  # palette(rainbow(8)))
  # colors=brewer.pal(12, "Spectral"))
  #colorRampPalette(c("red", "blue"))(20))#brewer.pal(8, "Dark2"))
  # palette(rainbow(6))     # six color rainbow
  # (palette(gray(seq(0,.9,len = 25)))) #grey scale
}
```