Pipeline_GTExV7

Jurado Ruiz, Federico

16/5/2019

Functions for the GTEx in his seventh version.

coe.getCovariatesGTex.

coe.getCovariatesGTex is a function developed to extract from a global dataframe with all the covariates coming from the GTEx and separate them in 53 differente files (to be able to handle them more easily) separated by their tissue. It recieve as parameters a dataframe with samples as rows and covariates as columns, and as second parameter a directory to write the files.

```
coe.getCovariatesGTex = function(samples, output_dir){

  attach(samples)
  GENDER = factor(SEX, levels=c("Male","Female"),labels=c(0,1))
  SAMPID = gsub("-",".",SAMPID)
  covariates = data.frame(SAMPID,SMTSD,SMCENTER,SMRIN,SMGEBTCH,
                          SMNABTCH,SMATSSCR,GENDER,AGE)
  ordered_covariates = covariates[order(SMTSD),]
  sp = split(ordered_covariates, ordered_covariates$SMTSD)
  for (data in sp){
    write.csv(data,file = paste0(output_dir,data$SMTSD[1],".csv"))
  }
}
```

coe.findGenesWithCovsAll.

The next two functions match the files generated by getCovariatesGTex with their genes at the global matrix. With the match it groups them by tissue and generates a new file with the gene expression results for the samples in that tissue. We call to the main function coe.findGenesWithCovsAll and give it as parameters the covariates directory, and output directory and the global matrix.

```
coe.findGenesWithCovs = function(tissue_covs,genesmatrix){

  names <- intersect(colnames(genesmatrix),tissue_covs$SAMPID)
  genes.withcovs =data.frame(genesmatrix[,names],row.names = genesmatrix[,1])
  return (genes.withcovs)
}

coe.findGenesWithCovsAll=function(tissue_covs_dir,output_dir,genematrix){

  files = list.files(tissue_covs_dir)
  for (file in files){
    tissue_covs = read.csv(paste0(tissue_covs_dir,"/",file))
    genes.withcovs= coe.findGenesWithCovs(tissue_covs,genematrix)
    write.csv(genes.withcovs,file = paste0(output_dir,"/",gsub(".csv","",file),"_GE.csv"))
  }
}
```

coe.plotMDS_batch.

With coe.plotMDS_batch we call limma library to visualice the possible batch effect by plotting it. As required parameters we need the a matrix with the gene expression levels, the covariates for that tissue and from those, the covariate we think is responsible of the batch effect. As optional parameters we can plot it without the legend, and save it at an specific output directory.

```
coe.plotMDS_Batch=function(tissue_GE,covs_tissue,cov_as_batch,
                           legend=T,save=F,output_dir=getwd(),
                           tissue_name=deparse(substitute(tissue_GE))){

  mask = sample(1:nrow(tissue_GE),40)
  colors = rainbow(length(unique(as.numeric(covs_tissue[,cov_as_batch]))))
  finalcolors = colors[as.numeric(covs_tissue[,cov_as_batch])]
  if (save == TRUE){
    pdf(file = paste0(output_dir,"/",gsub(" ","_",tissue_name), "_MDS_using_",cov_as_batch,".pdf"))
  }
  limma::plotMDS(tissue_GE[,mask],col=finalcolors,
          main=paste0(tissue_name, "_MDS_using_",cov_as_batch))
  if (legend == TRUE){
    legend("topright",fill=colors,
           legend=levels(covs_tissue[,cov_as_batch]))
  }
  if (save == TRUE){
    dev.off()
  }

}
```

coe.prepareForCombat and coe.combatCorrect.

The next function, coe.prepareForCombat prepare the gene expression file by filtering genes which are not expressed by following the rule specified in 3.1 (Materials and methodology) of this work. This function also normalize the data and return the correspoding normalized expression matrix and covariates in a list of two elements.

```
coe.prepareForCombat = function(tissue_GE,covs_tissue,
                                lower_limit = 0.1,upper_limit = 0.8){

  covs_tissue = covs_tissue[complete.cases(covs_tissue[,"SMCENTER"]),]
  covs_tissue = covs_tissue[complete.cases(covs_tissue[,"SMGEBTCH"]),]
  names = intersect(colnames(tissue_GE),rownames(covs_tissue))
  covs = covs_tissue[names,]
  tissue_GE = tissue_GE[,names]

  rpkms_qn = normalize.quantiles(as.matrix(tissue_GE))
  stopifnot(nrow(covs) == ncol(rpkms_qn))
  expressed.genes = rowSums(tissue_GE > lower_limit)  > (upper_limit * ncol(tissue_GE))
  tissue_GE = tissue_GE[expressed.genes,]
  covs$SMCENTER = as.factor(covs$SMCENTER)
  return(list(normalized_expression=as.matrix(tissue_GE),covariates=covs))
}
```

From the list generated by the previous function, we can retrieve our dataframe and our covariates and begin with the batch correction. In the following function we specify as a requisite the dataframe and the covariates, and optionally we can retrieve the plots for the correlation (of the specified covariates) before and after the correction on the specified directory.

```
coe.combatCorrect = function (
  rpkms_qn,covs,
  covsCols=c("AGE","GENDER","SMCENTER","SMRIN","SMGEBTCH"),
  PCA_plots = F,
  tissue_name=deparse(substitute(rpkms_qn)),
  output_dir = paste0(getwd(),"/pca_plots_",tissue_name))
  {


  if (length(unique(levels(covs$SMCENTER))) > 1){
    cat("Correcting batch for SMCENTER\n")
    rpkms_combat = ComBat(dat=rpkms_qn,batch=gsub(",","",covs$SMCENTER),
                          mod=model.matrix(~1,data=covs[,c("GENDER","AGE","SMRIN")]))
    rpkms_combat = rpkms_combat - min(rpkms_combat)
  }else{rpkms_combat = rpkms_qn}
  cat("Done\n")
  cat("Correcting batch for SMGEBTCH\n")
  rpkms_combat = ComBat(dat=rpkms_combat,batch=as.factor(gsub("-","",covs$SMGEBTCH)),
                        mod=model.matrix(~1,data=covs[,c("GENDER","AGE","SMRIN")]))
  rpkms_combat = rpkms_combat - min(rpkms_combat)
  cat("Done\n")
  if (length(unique(levels(covs$SMCENTER))) < 2){
      covsCols=covsCols[-3]
  }
  if (PCA_plots ==T){
    if(!(dir.exists(output_dir))){
      cat("Directory for the plots not found, creating one.\n")
```

```
      dir.create(output_dir)
    }
    cat("PCA uncorrected and plotting in process\n")
    pdf(file = paste0(output_dir,"/",deparse(substitute(rpkms_qn)), "_PCA_uncorrected.pdf"))
    pcres = prince(rpkms_qn,covs[,covsCols],top=20)
    coexp.princePlot(prince=pcres,main=paste0("PCA_for_" ,deparse(substitute(rpkms_qn)),"_uncorrected"))
    dev.off()
    cat("PCA uncorrected done.\nPCA corrected and plotting in process\n")
    pdf(file = paste0(output_dir,"/",deparse(substitute(rpkms_qn)), "_PCA_corrected.pdf"))
    pcres = prince(rpkms_combat,covs[,covsCols],top=20)
    coexp.princePlot(prince=pcres,main=paste0("PCA_for_" ,deparse(substitute(rpkms_qn)),"_corrected"))
    dev.off()
    cat("PCA corrected done\n")
  }

  return (rpkms_combat)
}
```

coe.svaCorrection.

The following function, coe.svaCorrection, performs a correction over a dataseta. This correction is done in two steps, first, it locates de surrogated variables (SVA) and then with a regression model correct the effect of the covariates along with these SVA. By default it generates plots before and after the correction, but these can be turned off by setting plots = F. This function requires off a dataframe with the gen expression, and a data frame with the covariates.

```
coe.svaCorrection = function(rpkms_combat,covs,
                             covsCols=c("AGE","GENDER","SMCENTER","SMRIN","SMGEBTCH"),
                             model_matrix = model.matrix(~ GENDER + AGE + SMRIN,data=covs),
                             plots = T,
                             tissue_name=deparse(substitute(rpkms_combat)),
                             output_dir = paste0(getwd(),"/svas_plots_",tissue_name),
                             save = T,
                             resids_output=paste0(getwd(),"/Tissues_residuals/",tissue_name,".rds")){


  if (length(unique(levels(covs$SMCENTER))) < 2){
    covsCols=covsCols[-3]
  }
  mm = model_matrix
  nullmm = model.matrix(~ 1,data=covs)
  cat("Launching svaseq, this will take some time\n")
  svas = tryCatch({
    return(svaseq(dat=as.matrix(tissue_combat),mod=mm,mod0=nullmm))
    cat("\nSVAs obteined\n")
    },error = function(e){
    svas = list(0)
    svas$n.sv = 0
    return (svas)
    cat ("\nsvaseq failed, assuming 0 SV\n")
  })

  if (svas$n.sv != 0){
  ## LM correction.
    cat("Starting linear model correction\n")
    numeric.covs = covs[,covsCols]
    linp = matrix(ncol=svas$n.sv,nrow=ncol(numeric.covs))
    rownames(linp) = colnames(numeric.covs)
    colnames(linp) = paste0("SV",1:svas$n.sv)
    linp[] = 0
    for(cov in 1:ncol(numeric.covs)){
      for(sva in 1:svas$n.sv){ if(svas$n.sv == 1)
        axis = svas$sv else
          axis = svas$sv[,sva]
        linp[cov,sva] = cor.test(as.numeric(numeric.covs[,cov]),axis)$p.value
      }
    }
    smallest = -10
    linp10 <- log10(linp)
    linp10 <- replace(linp10, linp10 <= smallest, smallest)
    tonote <- signif(linp, 1)
  }
  #Resid generation.
```

```
  if(svas$n.sv != 0){
    covs.rs = covs[,match(covsCols,colnames(covs))]
    cat("Creating residuals\n")
    resids <- apply(rpkms_combat, 1, function(y){
      lm( y ~ . , data=cbind(covs.rs,svas$sv))$residuals
      })
  }else{
    covs.rs = covs[,match(covsCols,colnames(covs))]
    cat("Creating residuals\n")
    resids <- apply(rpkms_combat, 1, function(y){
      lm( y ~ . , data=cbind(covs.rs))$residuals
    })
  }
  cat("Done\n")
  if ((plots == T) & (svas$n.sv != 0)){
    cat("Creating plots\n")
    if(!(dir.exists(output_dir))){
      cat("Directory for the plots not found, creating one.")
      dir.create(output_dir)
    }
    pdf(file = paste0(output_dir,"/",tissue_name, "_Corr_of_SVs_and_covs.pdf"))
    heatmap.2(linp10, Colv = F, Rowv = F, dendrogram = "none",
              trace = "none", symbreaks = F, symkey = F,
              breaks = seq(-20, 0, length.out = 100),
              key = T, colsep = NULL, rowsep = NULL, sepcolor = "black",
              sepwidth = c(0.05, 0.05), main = "Corr. of SVs and covs., Combat-FPKM QN",
              labCol = colnames(linp10),
              labRow = covsCols,
              xlab = "Surrogate variables")
    dev.off()
    pdf(file = paste0(output_dir,"/",tissue_name, "_residuals_corr.pdf"))
    pcres = prince(as.matrix(coexp.trasposeDataFrame(resids,F)),covs.rs,top=20)
    coexp.princePlot(prince=pcres, main=paste0(tissue_name,": residual cor. with GENDER, AGE, SMRIN"))
    dev.off()
    cat("Plots done\n")
  }else if ((plots == T) & (svas$n.sv == 0)){
    pdf(file = paste0(output_dir,"/",tissue_name, "_residuals_corr.pdf"))
    pcres = prince(as.matrix(coexp.trasposeDataFrame(resids,F)),covs.rs,top=20)
    coexp.princePlot(prince=pcres, main=paste0(tissue_name,": residual cor. with GENDER, AGE, SMRIN"))
    dev.off()
  }

  if(save==T){
    cat("Saving residuals\n")
    if(!(dir.exists(resids_output))){
      cat("Directory for the residuals not found, creating one.")
      dir.create(resids_output)
    }
    saveRDS(resids,paste0(resids_output,"/",tissue_name,".rds"))
    cat(paste0("Residuales saved at: \n",resids_output))
  }
  return(resids)
}
```

coexp.getDownStreamNetwork.

This function, is the main call for all the package. With it all the previous function will be called along with coexp.getDownStreamNetwork (for net generation), coexp.getGProfilerOnNet (for gprofiler) and genAnnotationCellType (for the celltypes). As a result, this function requires a gene expression dataframe, and the covariates also as a dataframe, and if no output is specified, it will generate in the working directory multiple folders with the plos, the residuals, the network and the cluster profiling.

```
coe.getNetOneTissue = function(tissue_GE,covs_tissue,output=getwd(),name){
  name=gsub(" ","_",name)
  if(!(dir.exists(paste0(output,"/",name)))){
    cat("Directory for the job not found, creating one.\n")
    dir.create(paste0(output,"/",name))
    dir.create(paste0(output,"/",name,"/Plots"))
    dir.create(paste0(output,"/",name,"/Plots/Plots_MDS_Batch"))
    dir.create(paste0(output,"/",name,"/Plots/Plots_PCA_Batch"))
    dir.create(paste0(output,"/",name,"/Plots/Plots_SVas"))
    dir.create(paste0(output,"/",name,"/Plots/Plots_Net"))
    dir.create(paste0(output,"/",name,"/RDS"))
    dir.create(paste0(output,"/",name,"/RDS/Net"))
    dir.create(paste0(output,"/",name,"/RDS/Gprofiler"))
    dir.create(paste0(output,"/",name,"/RDS/Residuals"))
    dir.create(paste0(output,"/",name,"/temp"))
  }
  cat("Starting MDS plots\n")
  coe.plotMDS_Batch(tissue_GE,covs_tissue,"SMCENTER",
                    save = T,
                    output_dir = paste0(output,"/",name,"/Plots/Plots_MDS_Batch"),
                    tissue_name=name)
  coe.plotMDS_Batch(tissue_GE,covs_tissue,"SMGEBTCH",
                    save = T,
                    output_dir = paste0(output,"/",name,"/Plots/Plots_MDS_Batch"),
                    tissue_name=name)
  cat("Done.\nPreparing data for combat correction\n")
  Quantiles_and_covs = coe.prepareForCombat(tissue_GE,covs_tissue)
  tissue_qn = Quantiles_and_covs$normalized_expression
  covs = Quantiles_and_covs$covariates
  cat("Done.\nCorrecting with combat.\n")
  tissue_combat=coe.combatCorrect(rpkms_qn=tissue_qn,covs = covs,
                                  PCA_plots = T,
                                  tissue_name = name,
                                  output_dir = paste0(output,"/",name,"/Plots/Plots_PCA_Batch"))
  cat("Done.\nCorrecting SVas.\n")
  resids = coe.svaCorrection(tissue_combat,covs,
                             plots = T,
                             tissue_name = name,
                             save=T,
                             output_dir = paste0(output,"/",name,"/Plots/Plots_SVas"),
                             resids_output = paste0(output,"/",name,"/RDS/Residuals"))
  cat("Done.\nStarting to create the net\n")
  net = coexp.getDownstreamNetwork(tissue=name,
                                   n.iterations=50,
                                   net.type = "signed",
                                   debug=F,
                                   expr.data=resids,
```

```
                                    job.path=paste0(output,"/",name,"/temp"))
  cat("Done.\nStarting gprofiler\n")
  tissue_net = readRDS(net$net)
  names(tissue_net$moduleColors) = unlist(lapply(str_split(names(tissue_net$moduleColors),"\\."),functi
  gprof = coexp.getGProfilerOnNet(net.file = tissue_net,
                                  ensembl = F,
                                  out.file = paste0(output,"/",name,"/RDS/Gprofiler/",name,"_gprof.tsv"))
  ct = genAnnotationCellType(net.in = tissue_net,return.processed = F, which.one = "new")

  write.table(ct,file=paste0(output,"/",name,"/RDS/Gprofiler/",name,"_celltype.tsv"),sep = "\t")
  path=paste0(output,"/",name)
  for (files in list.files(path=path,pattern = ".pdf")){
    file.move(paste0(path,"/",files),paste0(path,"/Plots/Plots_Net"))
  }
  for (files in list.files(path=paste0(path,"/temp"),pattern = ".pdf")){
    file.move(paste0(path,"/temp/",files),paste0(path,"/Plots/Plots_Net"))
  }
  for (files in list.files(path=paste0(path,"/temp"),pattern = ".rds")){
    file.move(paste0(path,"/temp/",files),paste0(path,"/RDS/Net"))
  }
  unlink(paste0(path,"/temp"),recursive = T)
  cat("All done.")
}
```

coe.getBioTypeFromBioMart.

This function serves from biomart to retrieve the gene biotypes. It recieves as a parameter one of the residuals files generated by this pipeline and return a dataframe with the ensemble ID, the external name, and the biotype.

```
coe.getBioTypeFromBioMart <- function(genes.file){
  cat("\nChoosing Mart\n")
  ensembl <- useMart(biomart="ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl")
  attributes <- c("ensembl_gene_id","external_gene_name","gene_biotype")
  cat("Reading Files\n")
  ensembl.genes <- readRDS(genes.file)
  names = str_remove(colnames(ensembl.genes),"\\..?.?")
  cat("Starting query\n")
  return(getBM(attributes=attributes, filters='ensembl_gene_id', values=names,mart=ensembl))
}
```