# Select, load, annotate, normalize, and process toxicogenomic raw data from GEO and ArrayExpress

Andreas Schüttler[1]

[1]Helmholtz Centre for Environmental Research - UFZ

dx.doi.org/10.17504/protocols.io.s24eggw

Andreas Schüttler

ABSTRACT

Gene expression databases like Gene Expression Omnibus or ArrayExpress by now contain a wealth of toxicogenomic datasets. This is a offers great possibilities for advanced data analyses, like meta-analyses, co-expression studies, etc.

However, automated retrieval of this data is still a challenge.

With this computational pipeline we retrieve toxicogenomic data of the *Danio rerio* (zebrafish) embryo from GEO and ArrayExpress.
To make the data as comparable as possible, we download the raw datasets, and re-map the probes or probesets to the most recent version.

In the end a matrix with logFC in response to different chemical treatments is compiled.

PROTOCOL STATUS

**In development**
We are still developing and optimizing this protocol

BEFORE STARTING

The pipeline makes use of the custom R-package "toxprofileR".
This package is accessible via https://git.ufz.de/schuettl/toxprofileR

## Select data from Gene expression databases

1 1. GEO: The first step is to retrieve metadata from Gene Expression Omnibus. This is achieved with the help of the R-package 'GEOmetadb'. From the metadata and from manually curated information, datasets are selected and list for downloading data are created.

COMMAND

```
rm(list = ls())

# load libraries ---------------------------------------------------------------------------
library("Biobase")
library("GEOquery")
library("GEOmetadb")
library("RSQLite")
library("DBI")


# 1. Select data from GEO database ---------------------------------------------------------------------

## load GEO metadata as SQLite database ------------------------------------------------

if (!file.exists('./data/GEOmetadb.sqlite')) {
   GEOmetadb::getSQLiteFile(destdir = "./data/")
}

## connect to database
con <-
   RSQLite::dbConnect(RSQLite::SQLite(), "./data/GEOmetadb.sqlite")

## query Danio rerio datasets
## query Danio rerio datasets
drerio_datasets <- DBI::dbGetQuery(
  con,
  "SELECT *
  FROM gse JOIN gsm JOIN gpl
  ON gse.gse=gsm.series_id
  AND gpl.gpl=gsm.gpl
  WHERE gsm.organism_ch1 like '%Danio%rerio%'
  AND gse.type='Expression profiling by array'"
)

names_gse <- dbListFields(con, 'gse')
names_gsm <- dbListFields(con, 'gsm')
names_gpl <- dbListFields(con, 'gpl')

colnames(drerio_datasets) <-
  c(paste0("gse.", names_gse),
    paste0("gsm.", names_gsm),
    paste0("gpl.", names_gpl))

# extract age information from title, description, source name or characteristics ----------------------------
drerio_datasets$agegroup <- NA

drerio_datasets$agegroup[grepl(pattern = "embryo|larva|hpf|egg",
                    x = drerio_datasets$gse.title,
                    ignore.case = T)] <- "embryo"

drerio_datasets$agegroup[grepl(pattern = "embryo|larva|hpf|egg",
                    x = drerio_datasets$gsm.title,
                    ignore.case = T)] <- "embryo"

drerio_datasets$agegroup[grepl(pattern = "embryo|larva|hpf|egg",
                    x = drerio_datasets$gsm.description,
                    ignore.case = T)] <- "embryo"

drerio_datasets$agegroup[grepl(pattern = "embryo|larva|hpf|egg",
                    x = drerio_datasets$gsm.source_name_ch1,
                    ignore.case = T)] <- "embryo"

drerio_datasets$agegroup[grepl(pattern = "embryo|larva|hpf|egg",
                    x = drerio_datasets$gsm.characteristics_ch1,
                    ignore.case = T)] <- "embryo"

drerio_datasets$agegroup[grepl(pattern = "adult|male",
                    x = drerio_datasets$gse.title,
                    ignore.case = T)] <- "adult"

drerio_datasets$agegroup[grepl(pattern = "adult|male",
                    x = drerio_datasets$gsm.title,
                    ignore.case = T)] <- "adult"

drerio_datasets$agegroup[grepl(pattern = "adult|male",
                    x = drerio_datasets$gsm.description,
                    ignore.case = T)] <- "adult"

drerio_datasets$agegroup[grepl(pattern = "adult|male",
                    x = drerio_datasets$gsm.source_name_ch1,
                    ignore.case = T)] <- "adult"

drerio_datasets$agegroup[grepl(pattern = "adult|male",
                    x = drerio_datasets$gsm.characteristics_ch1,
                    ignore.case = T)] <- "adult"

# manually annotate some missing studies ---------------------------------------------
drerio_datasets$agegroup[drerio_datasets$gse.gse %in% c(
  "GSE11893",
  "GSE22634",
  "GSE27067",
  "GSE42084",
  "GSE61155",
  "GSE50718",
  "GSE69444",
  "GSE75245",
  "GSE84906",
  "GSE77148"
)] <- "embryo"

drerio_datasets$agegroup[drerio_datasets$gse.gse %in% c(
  "GSE3048",
  "GSE12140",
  "GSE18861",
  "GSE19908",
  "GSE27707",
  "GSE34716",
  "GSE43675",
  "GSE47039",
  "GSE48427",
  "GSE49915",
  "GSE67600",
  "GSE66362",
  "GSE71270",
  "GSE51434",
  "GSE58205",
  "GSE62541",
  "GSE93367",
  "GSE110340"
)] <- "adult"


drerio_datasets$agegroup[drerio_datasets$gse.gse == "GSE53086"] <-
  "cell_line"

# retreive manual table listing chemical treatments ------------------------------------
chem_data <-
  read.table(
    "./data/MetaData_curated/gse_chem.csv",
    header = T,
    sep = "\t",
    quote = ""
  )

drerio_datasets$chemical_treatment <-
  chem_data$chemical_treatment[match(drerio_datasets$gse.gse, chem_data$gse)]

# select embryo datasets with chemical treatment ----------------------------------------
zfe_tox_geo <-
  unique(drerio_datasets[(drerio_datasets$agegroup == "embryo") &
                (drerio_datasets$chemical_treatment == TRUE), ], MARGIN = 1)

# remove amputation experiments -----------------------------------------------
zfe_tox_geo <-
  zfe_tox_geo[!grepl("amputation", zfe_tox_geo$gse.title), ]

# remove arrays from custom array manufacturer --------------------------------
zfe_tox_geo <-
  zfe_tox_geo[zfe_tox_geo$gpl.distribution != "non-commercial" &
          zfe_tox_geo$gpl.manufacturer != "MWG" &
          zfe_tox_geo$gpl.manufacturer != "NimbleGen", ]

# save dataset ----------------------------------------------------------------
save(zfe_tox_geo, file = "./data/zfe_tox_geo.Rd")

# write ftp_download_list for array data --------------------------------------------
ftp_download_list_geo <-
  paste0(
    "ftp://ftp.ncbi.nlm.nih.gov/geo/series/",
    gsub('.{3}$', 'nnn', unique(zfe_tox_geo$gse.gse)),
      "/",
      unique(zfe_tox_geo$gse.gse),
      "/suppl/",
      unique(zfe_tox_geo$gse.gse),
      "_RAW.tar"
   )

# write table
write.table(
   x = ftp_download_list_geo,
   file = "./data/download_lists/ftp_download_list_geo.txt",
   quote = F,
   row.names = F,
   col.names = F
)


# save list of Platform IDs (add GPL7244 which is alternative to GPL2878) ------

platformfiles_geo <-
   data.frame(gpl_id = as.character(unique(c(
      zfe_tox_geo$gsm.gpl, "GPL7244"
   ))))

platformfiles_geo$annotation_data <-
   paste0("./data/PlatformData/soft/",
          platformfiles_geo$gpl_id,
          "_family.soft")

save(platformfiles_geo, file = "./data/platformfiles_geo.Rd")

ftp_download_list_platforms_geo <-
   paste0(
      "ftp://ftp.ncbi.nlm.nih.gov/geo/platforms/",
      gsub('.{3}$', 'nnn', platformfiles_geo$gpl_id),
      "/",
      platformfiles_geo$gpl_id,
      "/soft/*"
   )

write.table(
   x = ftp_download_list_platforms_geo,
   file = "./data/download_lists/ftp_download_list_platforms_geo.txt",
   quote = F,
   row.names = F,
   col.names = F
)
```

R

2 2. ArrayExpress: The same as for GEO is done for ArrayExpress.

COMMAND

```
rm(list = ls())

# load libraries ------------------------------------------------------------------------
library("ArrayExpress")
library("plyr")

# Select data from ArrayExpress database -------------------------------------------------------
sets <- ArrayExpress::queryAE(species = "danio+rerio")
sets <-
   sets[!grepl("GEOD", sets$ID), ] # remove all sets also present on GEO
chemIDs <- as.character(sets$ID[grepl("COMPOUND",
                             as.character(sets$ExperimentFactors),
                  ignore.case = T) &
                  !grepl(
                    "morpholino|morphilino|male|month|cell-line",
                    as.character(sets$ExperimentFactors),
                    ignore.case = T
                  )])

# manually remove studies from set ----------------------------------------------------
deselect <- c(
  "E-TABM-547",
  # adult
  "E-MEXP-2948",
  # microinjection
  "E-MEXP-1301",
  # adult
  "E-MEXP-818",
  # microinjection/animal caps
  "E-TABM-105",
  # adult
  "E-MEXP-736",
  # cell line
  "E-MTAB-43" # no raw-data available, no danio rerio
)

chemIDs <- chemIDs[!chemIDs %in% deselect]

sets <- sets[sets$ID %in% chemIDs, ]

# retrieve metadata-frames ----------------------------------------------------

meta_data_all <- list()
for (id in chemIDs) {
  meta_data_all[[id]] <-
    read.table(
      file = paste0(
        "https://www.ebi.ac.uk/arrayexpress/files/",
        id,
        "/",
        id,
        ".sdrf.txt"
      ),
      header = T,
      sep = "\t",
      quote = ""
    )
}

zfe_tox_ae <- do.call("rbind.fill", meta_data_all)

# remove studies from metadataset ---------------------------------------------
zfe_tox_ae <-
  zfe_tox_ae[!grepl("male", zfe_tox_ae$Characteristics.sex., ignore.case = T) &
          !grepl("adult",
             zfe_tox_ae$Characteristics.developmental.stage.,
             ignore.case = T) &
          !grepl(
            "heart|liver|extracardiac body tissue",
            zfe_tox_ae$Characteristics..OrganismPart.,
            ignore.case = T
          ) &
          !grepl("sequencing", zfe_tox_ae$Technology.Type, ignore.case = T)
        ,]

# save metadataset -------------------------------------------------------------
save(zfe_tox_ae, file = "./data/zfe_tox_ae.Rd")

# write ftp download list -----------------------------------------------------
ftp_download_list_ae <-
  as.character(unique(zfe_tox_ae$Comment..ArrayExpress.FTP.file.))
```

```
write.table(
   x = ftp_download_list_ae,
   file = "./data/download_lists/ftp_download_list_ae.txt",
   quote = F,
   row.names = F,
   col.names = F
)

# save list of Platform IDs
platformfiles_ae <-
   data.frame(gpl_id =  as.character(unique(zfe_tox_ae$Array.Design.REF)))

platformfiles_ae$annotation_data <-
   paste0("./data/PlatformData/soft/",
        platformfiles_ae$gpl_id,
        ".adf.txt")

save(platformfiles_ae, file = "./data/platformfiles_ae.Rd")

ftp_download_list_platforms_ae <-
   platformfiles_ae$gpl_id[!grepl("AFFY", platformfiles_ae$gpl_id)]

ftp_download_list_platforms_ae <-
   ftp_download_list_platforms_ae[!ftp_download_list_platforms_ae %in% c(#Affymetrix
      "A-GEOD-16933",
      # = GPL18967
      "A-GEOD-18967")]

ftp_download_list_platforms_ae <-
   paste0(
      "https://www.ebi.ac.uk/arrayexpress/files/",
      as.character(ftp_download_list_platforms_ae),
      "/",
      as.character(ftp_download_list_platforms_ae),
      ".adf.txt"
   )

write.table(
   x = ftp_download_list_platforms_ae,
   file = "./data/download_lists/ftp_download_list_platforms_ae.txt",
   quote = F,
   row.names = F,
   col.names = F
)
```

R

## Download Array and Platform data

3 Next step is to download the array and platform data selected in step 1.

We create a "data" directory where all data is downloaded to.

COMMAND

```
#!/bin/bash


# download GEO data
cat ./data/download_lists/ftp_download_list_geo.txt | parallel --gnu "wget {} -P data/ArrayData/"

cd ./data/ArrayData/

for f in *.tar; do
 d=`basename $f .tar`
 mkdir $d
 (cd $d && tar xf ../$f)
done
# (snippet from https://lonelycoding.com/how-can-you-untar-more-than-one-file-at-a-time/)

find . -name "*.gz" | while read filename; do gzip -d "`dirname "$filename"`" "$filename"; done;

cd ../../

rm ./data/ArrayData/*.tar

# download ArrayExpress Datasets

cat ./data/download_lists/ftp_download_list_ae.txt | parallel --gnu "wget {} -P data/ArrayData/"

cd ./data/ArrayData/

for f in *.zip; do
unzip "$f" -d "${f%*[[:punct:]]*[[:punct:]]*}";
done

cd ../../

rm ./data/ArrayData/*.zip


# download GEO Platform-Files

cat ./data/download_lists/ftp_download_list_platforms_geo.txt | parallel --gnu "wget {} -P data/PlatformData/soft/"

cd ./data/PlatformData/soft/

gunzip *.gz


cd ../../../

# download ArrayExpress Platform-Files
cat ./data/download_lists/ftp_download_list_platforms_ae.txt | parallel --gnu "wget {} -P data/PlatformData/soft/"


# download sequence information for Affy Arrays (need to be logged in)
wget http://www.affymetrix.com/Auth/analysis/downloads/lf/wt/ZebGene-1_1-st-v1/ZebGene-1_1-st-v1.zv9.probe.fa.zip -P data/PlatformData/fasta/
wget http://www.affymetrix.com/Auth/analysis/downloads/data/Zebrafish.probe_fasta.zip -P data/PlatformData/fasta/



# get 2bit file of latest Danio rerio genome assembly
wget "http://hgdownload-test.cse.ucsc.edu/goldenPath/currentGenomes/Danio_rerio/bigZips/danRer11.2bit" -P data/genomedata/

# get fasta file of cDNA from ensemble
wget "ftp://ftp.ensembl.org/pub/release-93/fasta/danio_rerio/cdna/Danio_rerio.GRCz11.cdna.all.fa.gz" -P data/genomedata/

# get fasta file of ncRNA from ensemble
wget "ftp://ftp.ensembl.org/pub/release-93/fasta/danio_rerio/ncrna/Danio_rerio.GRCz11.ncrna.fa.gz" -P data/genomedata/
```

Bash

## Probe mapping

4 Since microarrays are designed for different genome version, it is necessary to re-map the probes to the recent genome version (here dRer11).

For probe mapping, as a first step, fasta-files have to be created from the downloaded platform-files from GEO and ArrayExpress.

COMMAND

```
rm(list = ls())

# load platform information geo---------------------------------------------------------------
load("./data/platformfiles_geo.Rd")
platform_info_geo <-
  read.csv(
    file = "./data/MetaData_curated/platform_info_geo.csv",
    header = T,
    fill = T,
    sep = "\t",
    as.is = T
  )

lapply(platformfiles_geo$gpl_id, function(gpl_id) {
  if (platform_info_geo$platform_type[platform_info_geo$gpl_id == gpl_id] == "yes") {


    # determine start and end of platform table in .soft file
    start_end_read <-
      grep(
        "!platform_table_begin|!platform_table_end",
        readLines(platformfiles_geo$annotation_data[platformfiles_geo$gpl_id ==
                                                      gpl_id])
      )

    # read platform annotation (soft) file downloaded from GEO
    platform_annotation <-
      read.table(
        platformfiles_geo$annotation_data[platformfiles_geo$gpl_id == gpl_id],
        skip = start_end_read[1],
        nrow = start_end_read[2] - start_end_read[1],
        header = T,
        sep = "\t",
        fill = T,
        quote = "",
        comment.char = ""
      )

    # extract ProbeIDs and Sequences
    ProbeIDs <-
      as.character(platform_annotation[, platform_info_geo$ID_Column_Nr[platform_info_geo$gpl_id ==
                                                                          gpl_id]])
    Sequence <-
      as.character(platform_annotation[, platform_info_geo$Sequence_Column_Number[platform_info_geo$gpl_id ==
                                                                                    gpl_id]])

    # if ProbeType given, delete Control Probes for mapping
    if (!is.na(platform_info_geo$Type_Column_Name[platform_info_geo$gpl_id ==
                                                  gpl_id])) {
      ProbeType <-
        as.character(platform_annotation[, platform_info_geo$Type_Column_Number[platform_info_geo$gpl_id ==
                                                                                  gpl_id]])
      ProbeIDs <- ProbeIDs[ProbeType == platform_info_geo$Type_Entry_ExperimentalProbes[platform_info_geo$gpl_id ==
                                                                                        gpl_id]]
      Sequence <- Sequence[ProbeType == platform_info_geo$Type_Entry_ExperimentalProbes[platform_info_geo$gpl_id ==
                                                                                        gpl_id]]
    }

    platform_sequence_data <-
      data.frame(ProbeIDs = ProbeIDs, Sequence = Sequence)

    # write table with only ProbeID and Sequence
    write.table(
      platform_sequence_data,
      file = paste0("./data/PlatformData/soft/", gpl_id, "_columns.txt"),
      quote = F,
      sep = "\t",
        row.names = F,
        col.names = F
      )

      # use gawk to write fasta file
      system(paste(
        "gawk '{print \">\"$1\"\\n\"$2}'",
        paste0("./data/PlatformData/soft/", gpl_id, "_columns.txt"),
        ">",
        paste0("./data/PlatformData/fasta/", gpl_id, ".fa")
      ))

    }
})


# Platform information Array Express
load("./data/platformfiles_ae.Rd")
platform_info_ae <-
  read.csv(
    file = "./data/MetaData_curated/platform_info_ae.csv",
    header = T,
    fill = T,
    sep = "\t"
  )

lapply(platformfiles_ae$gpl_id, function(gpl_id) {
  if (platform_info_ae$platform_type[platform_info_ae$gpl_id == gpl_id] == "yes") {


  platform_annotation <-
    read.table(
      platformfiles_ae$annotation_data[platformfiles_ae$gpl_id == gpl_id],
      skip =  grep(
        "[main]",
        readLines(platformfiles_ae$annotation_data[platformfiles_ae$gpl_id == gpl_id]), fixed=T),
      header = T,
      sep = "\t",
      fill = T,
      quote = "",
      comment.char = ""
    )

# extract ProbeIDs and Sequences
ProbeIDs <-
  as.character(platform_annotation[, platform_info_ae$ID_Column_Nr[platform_info_ae$gpl_id ==
                                                  gpl_id]])
Sequence <-
  as.character(platform_annotation[, platform_info_ae$Sequence_Column_Number[platform_info_ae$gpl_id ==
                                                       gpl_id]])

# if ProbeType given, delete Control Probes for mapping
if (!is.na(platform_info_ae$Type_Column_Name[platform_info_ae$gpl_id ==
                                  gpl_id])) {
  ProbeType <-
    as.character(platform_annotation[, platform_info_ae$Type_Column_Number[platform_info_ae$gpl_id ==
                                                       gpl_id]])
  ProbeIDs <- ProbeIDs[ProbeType == ""]
  Sequence <- Sequence[ProbeType == ""]
}

platform_sequence_data <-
  data.frame(ProbeIDs = ProbeIDs, Sequence = Sequence)

# write table with only ProbeID and Sequence
write.table(
  platform_sequence_data,
  file = paste0("./data/PlatformData/soft/", gpl_id, "_columns.txt"),
  quote = F,
  sep = "\t",
  row.names = F,
  col.names = F
)

# use gawk to write fasta file
system(paste(
  "gawk '{print \">\"$1\"\\n\"$2}'",
  paste0("./data/PlatformData/soft/", gpl_id, "_columns.txt"),
  ">",
  paste0("./data/PlatformData/fasta/", gpl_id, ".fa")
))
  }
}
)
```

R

5 Perform Blat

COMMAND

```
#!/bin/bash

skriptdir=$(pwd)

cd ./data/PlatformData/fasta/

find $directory -type f -name "*.fa"|while read file
do
echo $file

# Perform blat on genome
twoBitFile="danRer11.2bit"
inFASTA="$file"
echo "perform Blat on genome"
blat ${twoBitFile} ${inFASTA} -maxIntron=380000 -minIdentity=95 -tileSize=9 -stepSize=5 -minScore=19 "${file}_danRer11_blatOut.psl"
echo "ok"
echo "writin to bed-file.."
cat "${file}_danRer11_blatOut.psl"  | perl ${skriptdir}/psl2fullBed.pl -fracIdentCO 0.95 > "../annotation/${file%%.fa}_hits_danRer11.bed"
echo "ok"

# Perform blat on cDNA
twoBitFile="Danio_rerio.GRCz11.cdna.all.fa"
inFASTA="$file"
echo "perform Blat on cDNA"
blat ${twoBitFile} ${inFASTA} -maxIntron=0  -minIdentity=95 -tileSize=9 -stepSize=5 -minScore=19 "${file}_danRer11_blatOut_cDNA.psl"
echo "ok"
echo "writin to bed-file.."
cat "${file}_danRer11_blatOut_cDNA.psl"  | perl ${skriptdir}/psl2fullBed.pl -fracIdentCO 0.95 > "../annotation/${file%%.fa}_hits_danRer11_cDNA.bed"
echo "ok"

# Perform blat on ncRNA
twoBitFile="Danio_rerio.GRCz11.ncrna.fa"
inFASTA="$file"
echo "perform Blat on ncRNA"
blat ${twoBitFile} ${inFASTA} -maxIntron=0  -minIdentity=95 -tileSize=9 -stepSize=5 -minScore=19 "${file}_danRer11_blatOut_ncrna.psl"
echo "ok"
echo "writin to bed-file.."
cat "${file}_danRer11_blatOut_ncrna.psl"  | perl ${skriptdir}/psl2fullBed.pl -fracIdentCO 0.95 > "../annotation/${file%%.fa}_hits_danRer11_ncrna.bed"
echo "ok"

done
```

6 map with gene annotation

COMMAND

```
rm(list = ls())

# load libraries -------------------------------------------------------------
library("seqinr")
library("biomaRt")
library("GenomicRanges")
library("AnnotationDbi")
library("pbapply")
library("data.table")
library("toxprofileR")

# global parameters ---------------------------------------------
maxmismatch <- 1
martversion <- 93

# load platform information
```

```
## Array Express
platform_info_ae <-
  data.table::fread(
    file = "./data/MetaData_curated/platform_info_ae.csv",
    header = T,
    fill = T,
    sep = "\t"
  )
load("./data/platformfiles_ae.Rd")
platform_info_ae <- merge(platform_info_ae, platformfiles_ae, all = T)
platform_info_geo <-
  data.table::fread(
    file = "./data/MetaData_curated/platform_info_geo.csv",
    header = T,
    fill = T,
    sep = "\t"
  )

## GEO
load("./data/platformfiles_geo.Rd")
platform_info_geo <-
  merge(platform_info_geo, platformfiles_geo, all = T)

## combine AE and GEO
platform_info <- rbind(platform_info_geo, platform_info_ae)
platform_info$gpl_id <- as.character(platform_info$gpl_id)
rm(
  list = c(
    "platform_info_ae",
    "platform_info_geo",
    "platformfiles_ae",
    "platformfiles_geo"
  )
)

# load annotation databases ---------------------------------------------
mart <- biomaRt::useEnsembl(biomart = "ensembl",
                       dataset = "drerio_gene_ensembl",
                       version = martversion)

if (!file.exists(file = paste0("./data/exbygene_drerio_ensembl_v", martversion, ".Rd"))) {
  library("GenomicFeatures")
  if (!file.exists(paste0(
    "./data/drerio_annotationdb_ensembl_",
    martversion,
    ".db"
  ))) {
    annotationdb <-
      GenomicFeatures::makeTxDbFromBiomart(
        biomart = mart@biomart,
        dataset = mart@dataset,
        host = unlist(strsplit(mart@host, ":80", fixed = T))[1]
      )
    AnnotationDbi::saveDb(
      annotationdb,
      file = paste0(
        "./data/drerio_annotationdb_ensembl_",
        martversion,
        ".db"
      )
    )
  } else{
    annotationdb <-
      AnnotationDbi::loadDb(file = paste0(
        "./data/drerio_annotationdb_ensembl_",
        martversion,
        ".db"
      ))
  }
  ,

  # extract exons by genes -------------------------------------------
  exbygene <- GenomicFeatures::exonsBy(annotationdb, by = "gene")
  save(exbygene,
     file = paste0("./data/exbygene_drerio_ensembl_v", martversion, ".Rd"))
} else {
  load(file = paste0("./data/exbygene_drerio_ensembl_v", martversion, ".Rd"))
}

# apply annotation to all platform files --------------------------------------------

lapply(platform_info$gpl_id, function(GPL) {

  message(paste("process platform", GPL))
  plat_info <- platform_info[gpl_id == GPL]

  # if annotation is identical to another platform
  if (grepl("GPL", plat_info$platform_type)) {

    # check if other file already exists, otherwise process this one first
    if (file.exists(paste0(
      "./data/PlatformData/final_annotation/",
      GPL,
      "annotation.Rds"
    ))) {
      message(paste(
        "take annotation data from",
        plat_info$platform_type
      ))
      table_annot <-
        readRDS(
          paste0(
            "./data/PlatformData/final_annotation/",
            plat_info$platform_type,
            "annotation.Rds"
          )
        )
      saveRDS(
        table_annot,
        file = paste0(
          "./data/PlatformData/final_annotation/",
          GPL,
          "annotation.Rds"
        )
      )
      return(NULL)
    } else {
      GPL_old <- GPL
      GPL <- plat_info$platform_type
      message(paste("process platform", GPL))
      plat_info <- platform_info[gpl_id == GPL]
    }
  }

  # Oaklabs array (with confidential fasta file)
  if (plat_info$platform_type == "BIOTOX") {
    fasta_file <-
      "../../ArrayAnnotation/data/069507_D_Fasta_20140902.fa"

    cDNA_bed <-
      "./data/PlatformData/annotation/Oaklabs_ArrayXS_Danio_rerio_V1_hits_danRer11_cDNA.bed"

    DNA_bed <-
      "./data/PlatformData/annotation/Oaklabs_ArrayXS_Danio_rerio_V1_hits_danRer11.bed"

    ncrna_bed <-
      "./data/PlatformData/annotation/Oaklabs_ArrayXS_Danio_rerio_V1_hits_danRer11_ncrna.bed"
    arraytype <- "regular"
  }

  # take old annotation if there was no sequence data available --------------------
  if (plat_info$platform_type == "no") {
    message("Sequence Data not available for ", GPL)
    message("loading old Annotation")

    start_end_read <-
      grep(
        "!platform_table_begin|!platform_table_end",
        readLines(plat_info$annotation_data)
      )

    # read platform annotation (soft) file downloaded from GEO
    platform_annotation <-
      read.table(
        plat_info$annotation_data,
        skip = start_end_read[1],
        nrow = start_end_read[2] - start_end_read[1],
        header = T,
        sep = "\t",
        fill = T,
        quote = "",
        comment.char = ""
      )

    ProbeIDs <-
      as.character(platform_annotation[, plat_info$ID_Column_Nr])
    GeneIdentifier <-
      as.character(platform_annotation[, plat_info$GeneIdentifier_ColumnName])

    table_annot <-
      data.frame(ProbeIDs = ProbeIDs, GeneIDs = GeneIdentifier)

    message("Biomart query")

    BM <-
      toxprofileR::getBM_annotation(
        values = as.character(table_annot$GeneIDs),
        filter = as.character(plat_info$GeneIdentifier_Type),
        mart = mart
      )

    table_annot <-
      merge.data.frame(
        table_annot,
        BM,
        by.x = "GeneIDs",
        by.y = as.character(plat_info$GeneIdentifier_Type),
        all.x = T,
        sort = F
      )

    message("saving")
    saveRDS(
      table_annot,
      file = paste0(
        "./data/PlatformData/final_annotation/",
        GPL,
        "annotation.Rds"
      )
    )
    return(NULL)
  }

  # Affy Arrays
  if (plat_info$platform_type == "Affy") {
    # check if Affy annotation is already present and copy annotation file from there
    affy_ids <-
      as.character(platform_info$gpl_id[platform_info$platform_type == "Affy"])
    if (sum(file.exists(
      paste0(
        "./data/PlatformData/final_annotation/",
        affy_ids,
        "annotation.Rds"
      )
    )) > 0) {
      root_file <-
        paste0("./data/PlatformData/final_annotation/",
               affy_ids,
               "annotation.Rds")[file.exists(
                 paste0(
                   "./data/PlatformData/final_annotation/",
                   affy_ids,
                   "annotation.Rds"
                 )
               )][1]
      table_annot <- readRDS(root_file)
      saveRDS(
        table_annot,
        file = paste0(
          "./data/PlatformData/final_annotation/",
          GPL,
          "annotation.Rds"
        )
      )
      return(NULL)
    }
    fasta_file <-
      "./data/PlatformData/fasta/Zebrafish.probe_fasta.fa"
    cDNA_bed <-
      "./data/PlatformData/annotation/Zebrafish.probe_fasta_hits_danRer11_cDNA.bed"

    DNA_bed <-
      "./data/PlatformData/annotation/Zebrafish.probe_fasta_hits_danRer11.bed"

    ncrna_bed <-
      "./data/PlatformData/annotation/Zebrafish.probe_fasta_hits_danRer11_ncrna.bed"

    arraytype <- "regular"
  }

  if (plat_info$platform_type == "Affy_ST") {
    # check if Affy annotation is already present and copy annotation file from there
    affy_ids <-
      as.character(platform_info$gpl_id[platform_info$platform_type == "Affy_ST"])
    if (sum(file.exists(
      paste0(
        "./data/PlatformData/final_annotation/",
        affy_ids,
        "annotation.Rds"
      )
    )) > 0) {
      root_file <-
        paste0("./data/PlatformData/final_annotation/",
               affy_ids,
               "annotation.Rds")[file.exists(
                 paste0(
                   "./data/PlatformData/final_annotation/",
                   affy_ids,
                   "annotation.Rds"
                 )
               )][1]
      table_annot <- readRDS(root_file)
      saveRDS(
        table_annot,
        file = paste0(
          "./data/PlatformData/final_annotation/",
          GPL,
          "annotation.Rds"
        )
      )
      return(NULL)
    }
    fasta_file <-
      "./data/PlatformData/fasta/ZebGene-1_1-st-v1.zv9.probe.fa"
    cDNA_bed <-
      "./data/PlatformData/annotation/ZebGene-1_1-st-v1.zv9.probe_hits_danRer11_cDNA.bed"
    DNA_bed <-
      "./data/PlatformData/annotation/ZebGene-1_1-st-v1.zv9.probe_hits_danRer11.bed"
    ncrna_bed <-
      "./data/PlatformData/annotation/ZebGene-1_1-st-v1.zv9.probe_hits_danRer11_ncrna.bed"
    arraytype <- "Affy_ST"
  }


  if (plat_info$platform_type == "yes") {
    fasta_file <- paste0("./data/PlatformData/fasta/", GPL, ".fa")

    cDNA_bed <-
      paste0("./data/PlatformData/annotation/",
           GPL,
           "_hits_danRer11_cDNA.bed")

    DNA_bed <-
      paste0("./data/PlatformData/annotation/",
           GPL,
           "_hits_danRer11.bed")

    ncrna_bed <-
      paste0("./data/PlatformData/annotation/",
           GPL,
           "_hits_danRer11_ncrna.bed")

    arraytype <- "regular"
  }



  # load FASTA ----------------------------------------------------------------
  platformfasta <- seqinr::read.fasta(file = fasta_file)
  if (sum(duplicated(names(platformfasta))) > 0) {
    message(paste(sum(duplicated(
      names(platformfasta)
    )), "duplicates in fasta file"))
  }

  probelengths <- unlist(lapply(platformfasta, length))
  #annotInfo$NrProbes <- length(platformfasta)
  rm(platformfasta)

  # cDNA
  aggr_table_cDNA <-
    toxprofileR::get_hits_exons(
      exon_bed = cDNA_bed,
      mart = mart,
      maxmismatch = maxmismatch,
      probelengths = probelengths,
      name = "cDNA",
      arraytype = arraytype
    )
  # ncrna
  aggr_table_ncrna <-
    toxprofileR::get_hits_exons(
      exon_bed = ncrna_bed,
      mart = mart,
      maxmismatch = maxmismatch,
      probelengths = probelengths,
      name = "ncrna",
      arraytype = arraytype
    )
  # genome
  aggr_table_genome <-
    toxprofileR::get_hits_genome(
      genome_bed = DNA_bed,
      mart = mart,
      maxmismatch = maxmismatch,
      probelengths = probelengths,
      arraytype = arraytype
    )

  # merge all three alignments -------------------------------------------------
  mapFinal.all <-
    as.data.table(merge(
      merge(
        aggr_table_genome,
        aggr_table_cDNA,
        by = "ProbeID",
        all = T
      ),
      aggr_table_ncrna,
      by = "ProbeID",
      all = T
    ))

  mapFinal.all <-
    mapFinal.all[, ensembl_gene_id_all := mapply(function(genome, cDNA, ncrna) {
      list(unique(c(
        unlist(genome), unlist(cDNA), unlist(ncrna)
      )[!is.na(c(unlist(genome), unlist(cDNA), unlist(ncrna)))]))
    },
    genome = ensembl_gene_id_genome_all,
    cDNA = ensembl_gene_id_cDNA_all,
    ncrna = ensembl_gene_id_ncrna_all)]

  mapFinal.all <-
    mapFinal.all[, n_all := mapply(function(ids_all) {
      length(unlist(ids_all))
    }, ids_all = ensembl_gene_id_all)]



  mapFinal.all <-
    mapFinal.all[, ensembl_gene_id := mapply(
      function(genome,
               cDNA,
               ncrna,
               score_genome,
               score_cDNA,
               score_ncrna) {
        c(unlist(genome), unlist(cDNA), unlist(ncrna))[which.max(c(
          max(unlist(score_genome), na.rm = T),
          max(unlist(score_cDNA), na.rm = T),
          max(unlist(score_ncrna), na.rm = T)
        ))]
      },
      genome = ensembl_gene_id_genome,
      cDNA = ensembl_gene_id_cDNA_all,
      ncrna = ensembl_gene_id_ncrna_all,
      score_genome = overlap_length,
      score_cDNA = score.x,
      score_ncrna = score.y
    )]

  sum(mapFinal.all$n_all == 1)

  ## flag genes with nonunique-hits
  mapFinal.all$unique <- FALSE
  mapFinal.all$unique[mapFinal.all$n_all == 1] <- TRUE

  # remove Probes with n_hits >= 100
  mapFinal.all <- mapFinal.all[n_all < 100]

  # make probecluster name to probe ID name for affy arrays
  if (plat_info$platform_type == "Affy") {
    mapFinal.all <- mapFinal.all[order(ProbeID)]
    mapFinal.all <-
      mapFinal.all[, ProbeID := unlist(lapply(
        X = strsplit(ProbeID, split = ":"),
        FUN = function(ProbeName) {
          ProbeName[3]
        }
      ))]
  }

  if (plat_info$platform_type == "Affy_ST") {
    mapFinal.all <- mapFinal.all[order(ProbeID)]
    mapFinal.all <-
      mapFinal.all[, ProbeID := unlist(lapply(
        X = strsplit(ProbeID, split = ":"),
        FUN = function(ProbeName) {
          ProbeName[3]
        }
      ))][,
        ProbeID := unlist(lapply(
          X = strsplit(ProbeID, split = "-"),
          FUN = function(ProbeName) {
            ProbeName[2]
          }
        ))][,
          ProbeID := unlist(lapply(
            X = strsplit(ProbeID, split = ";"),
            FUN = function(ProbeName) {
              ProbeName[1]
            }
          ))]
  }

  # determine ProbeSet-Annotation for Affy-Arrays ----------------------------------------
  if (plat_info$platform_type == "Affy_ST" |
    plat_info$platform_type == "Affy") {
    mapFinal.Probeset <-
      mapFinal.all[, .(
        ensembl_gene_id_all = list(ensembl_gene_id_all),
        ensembl_gene_id = list(ensembl_gene_id),
        unique = list(unique),
        n_all = list(n_all)
      ), by = ProbeID]

    mapFinal.Probeset <-
      mapFinal.Probeset[, ensembl_gene_ids_probeset := mapply(function(all_ids) {
        names(sort(table(unlist(
          all_ids
        )), decreasing = T))[1]
      }, all_ids = ensembl_gene_id_all)]

    mapFinal.Probeset <-
      mapFinal.Probeset[, ensembl_gene_ids_probeset_max_count := mapply(function(all_ids) {
        as.numeric(sort(table(unlist(
          all_ids
      )), decreasing = T))[1]
    }, all_ids = ensembl_gene_id_all)]

  mapFinal.Probeset <-
    mapFinal.Probeset[, ensembl_gene_ids_probeset_count_probes := mapply(function(all_ids) {
      length(unlist(all_ids))
    }, all_ids = ensembl_gene_id)]

  mapFinal.Probeset <-
    mapFinal.Probeset[, ensembl_gene_ids_probeset_count_unique := mapply(function(uniques) {
      sum(unlist(uniques))
    }, uniques = unique)]

  mapFinal.Probeset <-
    mapFinal.Probeset[, ensembl_gene_ids_probeset_percent_id := ensembl_gene_ids_probeset_max_count /
                     ensembl_gene_ids_probeset_count_probes]

  mapFinal.Probeset <-
    mapFinal.Probeset[, ensembl_gene_ids_probeset_percent_unique := ensembl_gene_ids_probeset_count_unique /
                     ensembl_gene_ids_probeset_count_probes]

  mapFinal.Probeset <-
    mapFinal.Probeset[, unique := mapply(function(percent_unique) {
      percent_unique > 0.5
    }, percent_unique = ensembl_gene_ids_probeset_percent_unique)]

  mapFinal.Probeset <-
    mapFinal.Probeset[, ensembl_gene_id := mapply(function(percent_id,
                                     ensembl_gene_ids_probeset) {
      if (percent_id > 0.5 &
        !is.na(percent_id)) {
        ensembl_gene_ids_probeset
      } else{
        NA
      }
    },
    percent_id = ensembl_gene_ids_probeset_percent_id,
    ensembl_gene_ids_probeset = ensembl_gene_ids_probeset)]

  mapFinal.all <- mapFinal.Probeset

  if (plat_info$platform_type == "Affy_ST") {
    transcript.probeset <-
      read.csv(
        file = "./data/PlatformData/fasta/ZebGene-1_1-st-v1.na33.zv9.probeset.csv",
        sep = ",",
        fill = T,
        header = T,
        as.is = T
      )
    transcript.probeset <-
      transcript.probeset[transcript.probeset$transcript_cluster_id != 0, c("probeset_id", "transcript_cluster_id")]
    transcript.probeset$transcript_cluster_id <-
      as.character(transcript.probeset$transcript_cluster_id)
    transcript.probeset$probeset_id <-
      as.character(transcript.probeset$probeset_id)

    mapFinal.all <-
      merge(
        mapFinal.all,
        transcript.probeset,
        by.x = "ProbeID",
        by.y = "transcript_cluster_id",
        all = T
      )
  } else{
    mapFinal.all$probeset_id <- NA
  }
```

```
  } else {
    mapFinal.all$probeset_id <- NA
  }

  mapFinal_reduced <-
    mapFinal.all[, c(
      "ProbeID",
      "ensembl_gene_id_all",
      "ensembl_gene_id",
      "unique",
      "n_all",
      "probeset_id"
    )]

  # get functional annotation ------------------------------------------------
  BM_all <-
    toxprofileR::getBM_annotation(values = mapFinal_reduced$ensembl_gene_id,
                           filter = "ensembl_gene_id",
                           mart)

  annotation_drer11 <-
    merge(
      mapFinal_reduced,
      BM_all,
      by = "ensembl_gene_id",
      all.x = T,
      sort = F
    )

  annotation_drer11 <-
    annotation_drer11[order(annotation_drer11$ProbeID), ]

  # save ---------------------------------------------------------------------
  saveRDS(
    annotation_drer11,
    file = paste0(
      "./data/PlatformData/final_annotation/",
      plat_info$gpl_id,
      "annotation.Rds"
    )
  )

  if (exists("GPL_old")) {
    saveRDS(
      annotation_drer11,
      file = paste0(
        "./data/PlatformData/final_annotation/",
        GPL_old,
        "annotation.Rds"
      )
    )
    rm(GPL_old)
  }

})
```

R

## Create target file and table of comparisons

7 Before loading the data, we compile a targets data frame and a table with all comparisons for logFC calculation

COMMAND

```
rm(list = ls())
```

```
# combine sample metadata from databases with manual annotation -------------------------
## Array Express -----------------------------------------------------------------------

# manual annotation
zfe_tox_ae_cure <-
   read.csv(
      "./data/MetaData_curated/zfe_tox_ae_cure.csv",
      sep = "\t",
      header = T,
      fill = T,
      as.is = T
   )

# database metadata
load(file = "./data/zfe_tox_ae.Rd")

zfe_tox_ae <- zfe_tox_ae[!duplicated(zfe_tox_ae$Array.Data.File), ]

zfe_tox_ae$study_id <-
   unlist(lapply(strsplit(
      x = as.character(zfe_tox_ae$Comment..ArrayExpress.FTP.file.),
      fixed = F,
      split = "[.]|[/]"
   ), function(x) {
      x[[(grep(pattern = "raw", x) - 1)]]
   }))
zfe_tox_ae$gpl_id <- as.character(zfe_tox_ae$Array.Design.REF)

# combine
zfe_tox_ae_complete <-
   merge.data.frame(zfe_tox_ae[, c("Array.Data.File", "study_id", "gpl_id")], zfe_tox_ae_cure, by =
                    "Array.Data.File")

zfe_tox_ae_complete$gsm.gsm <- zfe_tox_ae_complete$Array.Data.File

## GEO ---------------------------------------------------------------------------------

# manual annotation
zfe_tox_geo_cure <-
   read.csv(
      "./data/MetaData_curated/zfe_tox_geo_cure.csv",
      sep = "\t",
      header = T,
      fill = T,
      as.is = T
   )

# database metadata
load(file = "./data/zfe_tox_geo.Rd")

zfe_tox_geo$study_id <- as.character(zfe_tox_geo$gse.gse)
zfe_tox_geo$gpl_id <- as.character(zfe_tox_geo$gpl.gpl)
zfe_tox_geo$Array.Data.File <- unlist(lapply(strsplit(unlist(
   lapply(
      X = strsplit(
         zfe_tox_geo$gsm.supplementary_file,
         split = ";",
         fixed = T
      ),
      FUN = function(x) {
         x[[1]]
      }
   )
), split = "[/]|.gz"), function(x) {
   x[[length(x)]]
}))

# combine
```

```
zfe_tox_geo_complete <-
  merge.data.frame(zfe_tox_geo[, c("Array.Data.File", "study_id", "gpl_id", "gsm.gsm")], zfe_tox_geo_cure, by =
                    "gsm.gsm")

# merge AE and GEO -------------------------------------------------------------
zfe_tox_targets <- rbind(zfe_tox_ae_complete, zfe_tox_geo_complete)

# remove lines which are marked to disregard
zfe_tox_targets <-
  zfe_tox_targets[zfe_tox_targets$disregard == "" |
                    is.na(zfe_tox_targets$disregard), ]

# make sample names
zfe_tox_targets$SampleName <-
  make.names(
    names = paste(
      zfe_tox_targets$Comp_SubstanceName_Trivial,
      zfe_tox_targets$Exp_Messzeit_hpf,
      zfe_tox_targets$Exp_Conc,
      zfe_tox_targets$Exp_Conc_Unit,
      sep = "_"
    ),
    unique = T
  )

# only take one color experiments
zfe_tox_targets <-
  zfe_tox_targets[
    zfe_tox_targets$Exp_Design == "OneColor", ]


# correct units
zfe_tox_targets$Exp_Conc_Unit <-
  gsub(
    pattern = "µm_L",
    replacement = "µM_L",
    x = zfe_tox_targets$Exp_Conc_Unit,
    fixed = T
  )
zfe_tox_targets$Exp_Conc_Unit <-
  gsub(
    pattern = "mg/l",
    replacement = "mg_L",
    x = zfe_tox_targets$Exp_Conc_Unit,
    fixed = T
  )

# remove flawed datasets
message("remove dataset GSM668015")
zfe_tox_targets <-
  zfe_tox_targets[zfe_tox_targets$gsm.gsm != "GSM668015", ] ##flawed Dataset

message("remove dataset GSM957452")
zfe_tox_targets <-
  zfe_tox_targets[zfe_tox_targets$gsm.gsm != "GSM957452", ] ##flawed Dataset


# create comparisons -----------------------------------------------------------
zfe_tox_targets$treat_id <-
  as.numeric(factor(apply(zfe_tox_targets[, c(
    "study_id",
    "Exp_Expositionsstart_hpf",
    "Exp_Expositionsstop_hpf",
    "Exp_Messzeit_hpf",
    "Comp_SubstanceName_Trivial",
    "Comp_Tr_Ctrl",
    "Exp_Conc",
    "Exp_tissue"
  )], 1, function(x) {
```

```
  )], 1, function(x) {
    paste(x, collapse = "_")
  })))

comparisons <-
  zfe_tox_targets[!duplicated(zfe_tox_targets[, "treat_id"]), ]

comparisons_control <-
  comparisons[comparisons$Comp_Tr_Ctrl == "Control", !(colnames(comparisons) %in%
                                             c("Array.Data.File", "gsm.gsm"))]

comparisons_treatment <-
  comparisons[comparisons$Comp_Tr_Ctrl == "Treatment", !(colnames(comparisons) %in% c("Array.Data.File", "gsm.gsm"))]

comparisons_merge <-
  merge.data.frame(
    comparisons_treatment,
    comparisons_control,
    by = c(
      "study_id",
      "gpl_id",
      "Exp_Expositionsstart_hpf",
      "Exp_Expositionsstop_hpf",
      "Exp_Messzeit_hpf",
      "Exp_Expositionsdauer_h",
      "Exp_tissue",
      "Data_RawFormat"
    ),
    all = T,
    suffixes = c(".treatment", ".control")
  )

comparisons_merge$compareID <-
  c(1:length(rownames(comparisons_merge)))

save(list=c("comparisons_merge", "zfe_tox_targets"), file = "./data/targets_comparisons.Rd")
```

R

## Read raw data

8 Based on the R-packages "limma" and "oligo", data is loaded into R.

COMMAND

```
rm(list = ls())

library("toxprofileR")

# load targets file
load(file = "./data/targets_comparisons.Rd")

# create ProcessedData Folder
if (!dir.exists("./data/ProcessedData/")) {
  dir.create("./data/ProcessedData/")
}

lapply(unique(zfe_tox_targets$study_id), function(study) {

  # retrieve study metadata
  metadata <- zfe_tox_targets[zfe_tox_targets$study_id == study, ]
  datadir <- paste0("./data/ArrayData/", study, "_RAW/")
  rawformat <- as.character(metadata$Data_RawFormat[1])
  betweenArrayNorm <- "cyclicloess"

  # read data
  assign(
    x = paste0("data.norm.", metadata$study_id[1]),
    value = toxprofileR::read_raw_public(
      datadir = datadir,
      rawformat = rawformat,
      betweenArrayNorm = betweenArrayNorm,
      metadata = metadata
    )
  )

  # save data
  save(
    list = paste0("data.norm.", metadata$study_id[1]),
    file = paste0(
      "./data/ProcessedData/",
      metadata$study_id[1],
      "_norm.Rdata"
    )
  )


})
```

R

## Create logFC matrix

9 Last but not least a logFC matrix is created from all normalized data.

COMMAND

```
rm(list = ls())

library("toxprofileR")
library("tidyverse")

message("catching target file")
load(file = "./data/targets_comparisons.Rd")

logFC_list <-
  lapply(unique(comparisons_merge$study_id), function(study_id) {

    message("processing dataset ", study_id)
    comparisons_study <-
      comparisons_merge[comparisons_merge$study_id == study_id, ]

    # load data
    load(file = paste0(
      "./data/ProcessedData/",
      comparisons_study$study_id[1],
      "_norm.Rdata"
    ))
    data <- get(paste0("data.norm.", comparisons_study$study_id[1]))

    #just for safety...
    data$targets$Exp_Conc <- as.numeric(data$targets$Exp_Conc)
    data$targets$Exp_Messzeit_hpf <-
      as.numeric(data$targets$Exp_Messzeit_hpf)


    fc_list_study <-
      lapply(comparisons_study$compareID, function(compareID) {
        message("Comparison # ", compareID)
        comparison <-
          comparisons_study[comparisons_study$compareID == compareID, ]

        # logFC
        logFCframe <-
          toxprofileR::calc_logfc_public(data = data, comparison = comparison)

        if (is.data.frame(logFCframe)) {
          return(logFCframe)
        }
      })

    fc_frame_study <- do.call("cbind", fc_list_study)
    fc_frame_study$ensembl_gene_id <- row.names(fc_frame_study)

    return(fc_frame_study)
  })

# remove studies with no output
logFC_list <- logFC_list[unlist(lapply(logFC_list,class))!="NULL"]

# merge all study frames together
logFC_frame <-  logFC_list %>% reduce(full_join, by = "ensembl_gene_id")
row.names(logFC_frame) <- logFC_frame$ensembl_gene_id
logFC_frame <- logFC_frame[,!colnames(logFC_frame)=="ensembl_gene_id"]

# save
save(logFC_frame, file = "./data/ProcessedData/logFC_frame.Rd")
```

R