Supplementary source code

This section provides the source code for the identification of circulating lncRNAs associated with gallbladder cancer risk by tissue-based preselection, cis-eQTL validation, and analysis of association with genotype-based expression

Note: The working directories have to be set accordingly in each R script.

Structure:

1. Preselection of differentially expressed lncRNAs along the sequence GS -> Dys -> GBC through two-sided Jonckheere-Terpstra test
2. Selection of the best model for prediction using robust AIC
3. Genotype-based lncRNA expression prediction and association analysis
4. Preselection of differentially expressed lncRNAs along the sequence GS -> Dys -> GBC through the XGBoost algorithm

Source code in R

```
##############################################################################
#
# program name:         01_LINC00662_preselection.R
# program title:        Preselection of differentially expressed lncRNAs
#                       along the sequence GS -> Dys -> GBC
# author:               Alice Blandino
# version:              1.0
# date:                 15.11.2021
#
# description:          Calculation of two-sided Jonckheere-Terpstra test
#
# input files:          01_data_LINC00662_preselection.txt
#
# Available at          www.biometrie.uni-heidelberg.de/
#                       StatisticalGenetics/Software_and_Data
#
##############################################################################

# "01_data_LINC00662_preselection.txt"
#
# A text file with a header line, and then one line per study participant
# with the following two fields:
#
# LINC00662             expression of LINC00662 in FFPE tissue
# group                 patients' status (gallstones,dysplasia,GBC)

# install and activate package to run two-sided J-T test
install.packages("DescTools", dependencies = TRUE)
library(DescTools)
# load data of study participants
setwd("*Path:\*")
data_preselection <- read.table("01_data_LINC00662_preselection.txt", header=T)
# order the group variable
data_preselection$group <- factor(data_preselection$group,
                        levels=c("GBC", "dysplasia", "gallstones"),
                        ordered=TRUE)
# perform J-T test
jt.test<-JonckheereTerpstraTest(data_preselection$LINC00662,
```

```
                data_preselection$group,
                alternative = "two.sided",nperm = 5000)


##############################################################################
#
# program name:         02_LINC00662_validation.R
# program title:        Selection of best model for prediction
# author:               Alice Blandino
# version:              1.0
# date:                 15.11.2021
#
# description:          Model selection based on robust AIC from robust
#                       linear regression models
# input files:          02_data_LINC00662_validation.txt
# Available at          www.biometrie.uni-heidelberg.de/
#                       StatisticalGenetics/Software_and_Data
#
##############################################################################

# "02_data_LINC00662_validation.txt"
#
# A text file with a header line, and then one line per study participant
# with the following two fields:
#
# LINC00662             LINC00662 expression in serum
# rs11083486            genotype for rs11083486 (0=G/G ;1=G/T ;2=T/T)
# rs142521755           genotype for rs142521755 (0=A/A ;1=A/T ;2=T/T)
# age                   study participants' age
# gender                study participants' gender
# PC1-PC10              first 10 PCs

# install and activate package to add variables to dataframe
install.packages("dplyr", dependencies = TRUE)
library(dplyr)

setwd("*Path:\*")
data_validation <- read.table("02_data_LINC00662_validation.txt", header=T)

# add new variables where:
# rs11083486 is once encoded dominantly (0+1 vs. 2), once encoded recessively (0 vs. 1+2)
# rs142521755 is encoded dominantly (0+1 vs. 2)
data_validation_new<-data_validation%>%
            mutate(rs11083486.dominant=ifelse(rs11083486=="0",1,rs11083486),
                rs11083486.recessive=ifelse(rs11083486=="2",1,rs11083486),
                rs142521755.dominant=ifelse(rs142521755=="0",1,rs142521755))

# model selection
# install and activate package to run robust linear regression models
install.packages(c("MASS","clickR",”AICcmodavg”), dependencies = TRUE)
library(MASS)
library(clickR)
library(AICcmodavg)
# 1.
# MODELS WITH rs11083486 ONLY
# additive
model.rs11083486.additive<-
rlm(LINC00662~rs11083486+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10,data=data_validati
on_new)
# three-genotypes
```

```
model.rs11083486.three<-
rlm(LINC00662~as.factor(rs11083486)+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10,data=da
ta_validation_new)
# dominant
model.rs11083486.dom<-
rlm(LINC00662~rs11083486.dominant+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10,data=dat
a_validation_new)
# recessive
model.rs11083486.rec<-
rlm(LINC00662~rs11083486.recessive+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10,data=dat
a_validation_new)

# 2.
# MODEL WITH rs142521755 ONLY
# rs142521755 dominant
model.rs142521755.dom<-
rlm(LINC00662~rs142521755.dominant+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10,data=d
ata_validation_new)

# 3.
# MODELS WITH BOTH rs11083486 AND rs142521755
# rs11083486 additive & rs142521755 dominant
model.add.dom<-
rlm(LINC00662~rs11083486+rs142521755.dominant+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+
PC10,data=data_validation_new)
# rs11083486 three-genotypes & rs142521755 dominant
model.three.dom<-
rlm(LINC00662~as.factor(rs11083486)+rs142521755.dominant+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+P
C8+PC9+PC10,data=data_validation_new)
# rs11083486 dominant & rs142521755 dominant
model.dom.dom<-
rlm(LINC00662~rs11083486.dominant+rs142521755.dominant+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+P
C8+PC9+PC10,data=data_validation_new)
# rs11083486 recessive & rs142521755 dominant
model.rec.dom<-
rlm(LINC00662~rs11083486.recessive+rs142521755.dominant+age+gender+PC1+PC2+PC3+PC4+PC5+PC6+PC7+P
C8+PC9+PC10,data=data_validation_new)

# create a dataframe with each model's name and its AIC
# vector with AICs:
AICs<-
c(AIC(model.rs11083486.additive),AIC(model.rs11083486.three),AIC(model.rs11083486.dom),AIC(model.rs1108348
6.rec),

AIC(model.rs142521755.dom),AIC(model.add.dom),AIC(model.three.dom),AIC(model.dom.dom),AIC(model.rec.dom
))
# vector with models' characteristics
models<-c("rs11083486.additive","rs11083486.three","rs11083486.dominant","rs11083486.recessive",
          "rs142521755.dominant","additive+dominant","three+dominant","dominant+dominant","recessive+dominant")
# dataframe with both AIC and models' characteristics
summary.AIC<-data.frame(AICs,models)
# find which model has the lowest RAIC
summary.AIC[order(summary.AIC$AICs),,drop=FALSE] [1,]
```

```
##############################################################################
#
# program name:          03_LINC00662_prediction.R
# program title:         Genotype-based lncRNA expression prediction  and association
#                        analysis
# author:                Alice Blandino
# version:               1.0
# date:                  15.11.2021
#
# description:           prediction of lncRNA based on individual genotype data
#                        and quantification of GBC risk associated to it
# input files:           03_data_LINC00662_prediction.txt
# Available at           www.biometrie.uni-heidelberg.de/
#                        StatisticalGenetics/Software_and_Data
#
##############################################################################

# "03_data_LINC00662_prediction.txt"
#
# A text file with a header line, and then one line per study participant
# with the following two fields:
#
# rs11083486             genotype for rs11083486 (0=T/T ;1=G/T ;2=G/G)
# rs142521755            genotype for rs142521755 (0=A/A ;1=A/T ;2=T/T)
# pheno                  patients' status (Control, Case)
# age                    study participants' age
# gender                 study participants' gender
# PC1-PC10               first 10 PCs

# install and activate package to add variables to dataframe
install.packages(c("robustbase",”dplyr”), dependencies = TRUE)
library(robustbase)
library(dplyr)

setwd("*Path:\*")
data_prediction <- read.table("03_data_LINC00662_prediction.txt", header=T)

# calculate the SNP-based expression
data_prediction_calculation<-data_prediction%>%
  mutate(rs11083486.coeff=ifelse(rs11083486=="0",-0.7352*0,ifelse(rs11083486=="1",-0.7352*1,-0.7352*2)),
         rs142521755.coeff=ifelse(rs142521755=="0",1.0797*0,ifelse(rs142521755=="1",1.0797*0,1.0797)),
         predicted.LINC00662=0.9267+rs11083486.coeff+rs142521755.coeff)

# association analysis fitting robust logistic regression model

# set controls as baseline category
data_prediction_calculation$pheno<-ordered(data_prediction_calculation$pheno, levels = c("Control", "Case"))

# model fitting
mod<-
glmrob(as.factor(data_prediction_calculation$pheno)~predicted.LINC00662+age+gender+PC1+PC2+PC3+PC4+PC5+
PC6+PC7+PC8+PC9+PC10,family = binomial, method= "Mqle",control=
glmrobMqle.control(tcc=1.2),data=data_prediction_calculation)
summary(mod)

# extract Oddsratio for Cases
exp(summary(mod)$coefficients[2])

# extract lower and upper limits for confidence intervals
exp(summary(mod)$coefficients[2] + qnorm(c(0.5,0.025,0.975)) * summary(mod)$coefficients[2,2])[2]
exp(summary(mod)$coefficients[2] + qnorm(c(0.5,0.025,0.975)) * summary(mod)$coefficients[2,2])[3]
```

```
##############################################################################
#
# program name:          04_ML_models.R
# program title:         XGBoost algorithm
# author:                Sinan U. Umu
# version:               1.0
# date:                  15.11.2021
#
# description:           Extreme gradient boosting (XGBoost) algorithm
#                        to train three-class classification ML models
#
##############################################################################

require(xgboost)
require(parallel)
require(doParallel)
require(tidyverse)
require(h2o)

setwd("*Path:\*")
lcrna_gbc=openxlsx::read.xlsx("lncRNAs_normalized_filtered.xlsx") %>% filter(!ID %in% excluded_samples)

h2o.init(nthreads = 15,max_mem_size = "200G")

function_h2o_data_split=function(lcrna_gbc,ratio=0.70,seed=1) { #seed=1

  dysplasia=h2o.splitFrame(as.h2o(lcrna_gbc   %>% mutate(group=factor(group)) %>% filter(group == "dysplasia"))
,ratios = ratio,seed = seed)
  gallstones=h2o.splitFrame(as.h2o(lcrna_gbc   %>% mutate(group=factor(group)) %>% filter(group == "gallstones"))
,ratios = ratio,seed = seed)

  GBC=h2o.splitFrame(as.h2o(lcrna_gbc   %>% mutate(group=factor(group))%>% filter(group == "GBC")) ,ratios =
ratio,seed = seed)

  train_x=c(dysplasia[[1]],gallstones[[1]],GBC[[1]]) %>% purrr::map(~as_tibble(.)) %>% bind_rows()
  test_x=c(dysplasia[[2]],gallstones[[2]],GBC[[2]]) %>% purrr::map(~as_tibble(.)) %>% bind_rows()




  return(list(as.h2o(train_x),as.h2o(test_x)))
}


lncrna_split=function_h2o_data_split(lcrna_gbc,seed=1) #create the datasets training, validation

train_x=as.h2o(upSample(as.data.frame(lncrna_split[[1]]),as.data.frame(lncrna_split[[1]]) %>% pull(group))) #training
dataset
test_x=lncrna_split[[2]] #validation dataset

y=c("group")
x =setdiff(names(train_x), c(y,"Class","age","sex","ID")) #predictors, only lncRNAs left



hyper_params <- list(ntrees = seq(10, 300, 1),
                     learn_rate = seq(0.005, 0.3, 0.01),
                     max_depth = seq(1, 7, 1),
                     sample_rate = seq(0.1, 1.0, 0.01),
                     col_sample_rate = seq(0.2, 1.0, 0.01),
                     reg_alpha=seq(0,0.3,0.025),
```

```
                gamma=seq(3,7,1),
                reg_lambda=seq(0,0.3,0.025)

)



search_criteria <- list(strategy = "RandomDiscrete",
                max_models = 50
)

#grid search for hyperparamater tuning via crossvalidation
xgb_grid <- h2o.grid(algorithm = "xgboost",
                grid_id = "xgboostgbc_grid_random_seed8",
                x = x, y = y,
                training_frame = train_x,
                #validation_frame=test_x,
                #reg_lambda=0,

                auc_type = "WEIGHTED_OVR",

                #validation_frame=test_x,
                nfolds = 5,
                seed = 8,
                #stopping_metric="mean_per_class_error",
                #stopping_tolerance=0.01,
                #stopping_rounds=10,

                hyper_params = hyper_params_hard_code,
                search_criteria = search_criteria)

best_model <- h2o.getModel(xgb_grid@model_ids[[1]])



## extract best variables
as.tibble(h2o.varimp(best_model))        %>%        dplyr::mutate(median=median(relative_importance))        %>%
dplyr::mutate(f=ifelse(relative_importance >= median,"PASS","FAIL"))

##train the best using the hyperparamaters on the training dataset

function_get_model_params=function(best_model,param_name) {

  best_model@allparameters[[param_name]]
}


#train the best using the hyperparamaters
test_model <- h2o.xgboost(x = x,
                y = y,
                training_frame = train_x,


                booster= function_get_model_params(best_model,"booster"),

                normalize_type=function_get_model_params(best_model,"normalize_type"),
                seed = 8,

                ntrees = function_get_model_params(best_model,"ntrees"),
                #nround=function_get_model_params(best_model,"nround"),
                learn_rate = function_get_model_params(best_model,"learn_rate"),
```

```
                eta=function_get_model_params(best_model,"eta"),
                max_depth = function_get_model_params(best_model,"max_depth"),
                sample_rate = function_get_model_params(best_model,"sample_rate"),
                subsample = function_get_model_params(best_model,"subsample"),
                col_sample_rate = function_get_model_params(best_model,"col_sample_rate"),
                colsample_bylevel=function_get_model_params(best_model,"colsample_bylevel"),
                reg_alpha=function_get_model_params(best_model,"reg_alpha"),
                reg_lambda=function_get_model_params(best_model,"reg_lambda"),
                colsample_bytree=function_get_model_params(best_model,"colsample_bytree"),
                col_sample_rate_per_tree=function_get_model_params(best_model,"col_sample_rate_per_tree"),
                colsample_bynode=function_get_model_params(best_model,"colsample_bynode"),
                gamma=function_get_model_params(best_model,"gamma"),
                min_split_improvement=function_get_model_params(best_model,"min_split_improvement"),
                gainslift_bins=-1,
                grow_policy="depthwise",
                fold_assignment="Random",
                tweedie_power=1.5,
                max_leaves=0,
                max_bins=function_get_model_params(best_model,"max_bins"),
                min_rows=function_get_model_params(best_model,"min_rows"),
                min_child_weight=function_get_model_params(best_model,"min_child_weight"),
                #stopping_tolerance=0.01,
                auc_type = "WEIGHTED_OVR",
                nfolds=5
)


#get model performance on the test dataset (this is reported in the manuscript)
h2o.performance(test_model,test_x)


#these are the most important hyperparamaters, found via crossvalidation
hyper_params_found_by_the_best_model <- list(ntrees = 245,
                                             learn_rate = 0.115,
                                             max_depth = 6,
                                             sample_rate = 0.66,
                                             col_sample_rate = 0.47,
                                             reg_alpha=0,
                                             gamma=3,
                                             reg_lambda=0.3

)
```