Fenland Proteomics Fitness - Version 2

TG421

2024-01-02

## Load libraries

library(lattice)  
library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(haven)  
library(tidyr)  
library(tibble)  
library(caret)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(grid)  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

library(broom)  
library(purrr)

##   
## Attaching package: 'purrr'

## The following object is masked from 'package:caret':  
##   
## lift

library(readr)  
library(writexl)  
library(digest)  
  
  
baseDir <- "V:/Programme1\_DiabetesAetiology/Data/Fenland"

## Fetch Fenland Phase 1 data

# Import Fenland exposure data  
fenlandExposure <- read\_csv(file.path(baseDir,"Fenland\_R8a\_GEPHFENLANDR80002042018\_01082018/GEPHFENLANDR80002042018B\_02082018.csv"), show\_col\_types = FALSE)  
fenlandExposure <- fenlandExposure %>%  
 rowwise() %>%  
 mutate(SampleId\_nophase = paste0(na.omit(c(SerNo, OMICS\_ID, CoreExomeID, magicid)), collapse = "")) %>%  
 ungroup()  
  
  
# Join with proteomics data  
fenlandProteomics <- read\_dta(file.path(baseDir,"Somalogic/Fenland\_main\_03Jul2018/Cleaned\_ANML\_normalised\_Dec2019/V4-18-039.ANML.FINAL\_1128\_cleaned\_R211.dta"))  
fenlandPhase1 <- fenlandExposure %>%  
 left\_join(fenlandProteomics, by = "SampleId\_nophase") %>%  
 rename(id\_Fenland = SampleId\_nophase)  
  
rm(fenlandProteomics, fenlandExposure)  
  
  
# Exclude samples and retain only phase 1 data  
fenlandPhase1 <- fenlandPhase1 %>%  
 mutate(exclude = ifelse(is.na(QC\_flag\_reason\_ANML) & is.na(QC\_fail\_reason\_ANML), 0, 1)) %>%  
 filter(exclude == 0, phase == 1)  
  
  
# Prepare covariates  
fenlandPhase1 <- fenlandPhase1 %>%  
 mutate(sex2c = as.numeric(Sex == "M"),  
 testSite = as.factor(TestSite),  
 ethnicity2c = as.numeric(gq\_eth\_5c\_DER != 1),  
 smoker4c = case\_when(  
 G\_SMOKE == 0 ~ 0,  
 G\_SMOKE == 2 ~ 1,  
 G\_SMOKE == 4 ~ 2,  
 G\_SMOKE %in% c(1, 3, 9) ~ 3  
 ),  
 fitnessTBM = P\_TR\_FITNESS\_trunc15\_est,  
 ffm = 0.001 \* AD06i\_iDEXA\_Total\_lean\_mass,  
 fitnessFFM = ifelse(P\_TR\_FITNESS\_trunc15\_est \* E\_Weight / ffm < 0, NA, P\_TR\_FITNESS\_trunc15\_est \* E\_Weight / ffm),  
 date = ymd(gsub(" 00:00:00.000", "", AppDate\_Attended))  
 ) %>%  
 rename(shr = P\_SHR, bmi = E\_BMI, age = AgeAtTest\_DM\_Attended, height = E\_Height, weight = E\_Weight)  
  
  
# Drop data points with missing values on main variables  
fenlandPhase1 <- fenlandPhase1 %>%  
 drop\_na(fitnessTBM, fitnessFFM, age, sex2c, ethnicity2c, bmi, shr, testSite, smoker4c)  
  
  
for (var in c("age", "height", "weight", "fitnessTBM", "fitnessFFM", "shr")){  
 fenlandPhase1[[paste0(var, "\_std")]] <- scale(fenlandPhase1[[var]])  
}  
  
  
for (var in c("bmi", "ffm", grep("SeqId\_", names(fenlandPhase1), value = TRUE))) {  
 fenlandPhase1[[paste0(var, "\_lnstd")]] <- scale(log(fenlandPhase1[[var]]))  
}  
  
  
  
fenlandCovariates <- fenlandPhase1[,c("id\_Fenland",   
 "sex2c",   
 "testSite",  
 "ethnicity2c",  
 "smoker4c")]  
  
  
fenlandPhase1 <- fenlandPhase1[,c("id\_Fenland",   
 "fitnessTBM",  
 "fitnessFFM",  
 "shr",  
 "age",  
 "height",  
 "weight",  
 "bmi",  
 "ffm",  
 "date",  
 grep("\_std$", names(fenlandPhase1), value = TRUE),  
 grep("\_lnstd$", names(fenlandPhase1), value = TRUE)  
 )  
 ]  
  
rm(var)

## Fetch Fenland Phase 2 data

# Fitness from P2  
  
# Import fitness and data data  
fenlandP2\_1 <- read\_csv(file.path(baseDir,"Fenland\_R8a\_GEPHFENLANDR80002042018E\_20Jul2023/GEPHFENLANDR80002042018E\_P2\_Batch\_1.csv"), show\_col\_types = FALSE)  
fenlandP2\_1 <- fenlandP2\_1 %>%  
 rowwise() %>%  
 mutate(id\_Fenland = paste0(na.omit(c(Serno, OMICS\_ID, CoreExomeID, magicid)), collapse = "")) %>%  
 ungroup()  
  
fenlandP2\_1 <- fenlandP2\_1 %>% mutate(fitnessTBM = P\_TR\_FITNESS\_est\_P2,  
 date = ymd(gsub(" 00:00:00.000", "", AppDate\_Attended\_p2)),  
 weight = P\_weight\_P2  
 )  
  
  
  
fenlandP2\_1 <- fenlandPhase1 %>%  
 select(id\_Fenland, age, date) %>%  
 inner\_join(fenlandP2\_1, by = "id\_Fenland") %>%  
 mutate(age2 = round(age + (1/365.25)\*as.numeric(difftime(date.y, date.x, units = "days")), 1))  
   
  
fenlandP2\_1 <- fenlandP2\_1[,c("id\_Fenland", "fitnessTBM","weight","date.y","age2")] %>% rename(age =age2, date = date.y)  
  
  
# Import FFM  
  
fenlandP2\_2 <- read\_csv(file.path(baseDir,"Fenland\_R9\_ GEPHFENLANDR90002112019D\_28Nov2019/GEPHFENLANDR90002112019D\_28Nov2019.csv"), show\_col\_types = FALSE)  
fenlandP2\_2 <- fenlandP2\_2 %>%  
 rowwise() %>%  
 mutate(id\_Fenland = paste0(na.omit(c(SerNo, OMICS\_ID, CoreExomeID, magicid)), collapse = "")) %>%  
 ungroup() %>%  
 rename(ffm = AD06i\_iDEXA\_Total\_lean\_mass\_P2)   
   
fenlandP2\_2 <- fenlandP2\_2[,c("id\_Fenland", "ffm")]  
  
  
  
# Import Height and SHR  
  
fenlandP2\_3 <- read\_csv(file.path(baseDir,"Fenland\_R8a\_GEPHFENLANDR80002042018E\_20Jul2023/GEPHFENLANDR80002042018E\_P2\_Batch\_2.csv"), show\_col\_types = FALSE)

## Warning: One or more parsing issues, call `problems()` on your data frame for details,  
## e.g.:  
## dat <- vroom(...)  
## problems(dat)

fenlandP2\_3 <- fenlandP2\_3 %>%  
 rowwise() %>%  
 mutate(id\_Fenland = paste0(na.omit(c(Serno, OMICS\_ID, CoreExomeID, magicid)), collapse = "")) %>%  
 ungroup() %>%  
 rename(height = P\_height\_P2, shr = P\_SHR\_P2)  
  
fenlandP2\_3 <- fenlandP2\_3[,c("id\_Fenland", "height","shr")]  
  
  
  
# Import Proteomics  
  
fenlandP2\_4 <- read\_dta(file.path(baseDir,"Somalogic/Fenland\_main\_03Jul2018/Cleaned\_ANML\_normalised\_Dec2019/V4-18-039.ANML.FINAL\_1128\_cleaned\_R211.dta"))  
  
# Exclude samples and retain only phase 2 data  
fenlandP2\_4 <- fenlandP2\_4 %>%  
 mutate(exclude = ifelse(is.na(QC\_flag\_reason\_ANML) & is.na(QC\_fail\_reason\_ANML), 0, 1)) %>%  
 filter(exclude == 0, phase == 2) %>%  
 rename(id\_Fenland = SampleId\_nophase)  
   
  
fenlandP2\_4 <- fenlandP2\_4[,c("id\_Fenland",grep("^SeqId\_", names(fenlandP2\_4), value = TRUE))]  
  
  
  
fenlandPhase2 <- fenlandP2\_1 %>%  
 left\_join(fenlandP2\_2, by = "id\_Fenland") %>%  
 left\_join(fenlandP2\_3, by = "id\_Fenland") %>%  
 left\_join(fenlandP2\_4, by = "id\_Fenland") %>%  
 mutate(bmi = round(weight / height^2, 1),  
 fitnessFFM = fitnessTBM \* weight / (0.001 \* ffm)  
 ) %>%  
 mutate(  
 fitnessFFM = ifelse(fitnessFFM < 0, NA, fitnessFFM)  
 )  
  
  
  
# Drop data points with missing values on main variables and cutoff date for sample handling  
  
fenlandPhase2 <- fenlandPhase2 %>%  
 drop\_na(fitnessTBM, fitnessFFM, age, bmi, shr) %>%  
 filter(date >= ymd("2017-04-01"))  
  
  
  
for (var in c("age", "height", "weight", "fitnessTBM", "fitnessFFM", "shr")){  
 fenlandPhase2[[paste0(var, "\_std")]] <- scale(fenlandPhase2[[var]])  
}  
  
  
for (var in c("bmi", "ffm", grep("SeqId\_", names(fenlandPhase2), value = TRUE))) {  
 fenlandPhase2[[paste0(var, "\_lnstd")]] <- scale(log(fenlandPhase2[[var]]))  
}  
  
  
fenlandPhase2 <- fenlandPhase2[,c("id\_Fenland",   
 "fitnessTBM",  
 "fitnessFFM",  
 "shr",  
 "age",  
 "height",  
 "weight",  
 "bmi",  
 "ffm",  
 "date",  
 grep("\_std$", names(fenlandPhase2), value = TRUE),  
 grep("\_lnstd$", names(fenlandPhase2), value = TRUE)  
 )  
 ]  
  
  
rm(var,fenlandP2\_1,fenlandP2\_2,fenlandP2\_3,fenlandP2\_4)

## Create Fenland Table 1

This code creates a “Table 1” descriptive table from Fenland Phase 1 data.

# Function to generate stats table  
  
generateTable <- function(data, contVars, contSigs, catVars) {  
 # Get cont stats  
 contResults <- contVars %>%  
 map2\_df(contSigs, ~{  
 curVar <- .x  
 curSig <- .y  
   
 medianVal <- median(data[[curVar]], na.rm = TRUE)  
 q25 <- quantile(data[[curVar]], 0.25, na.rm = TRUE)  
 q75 <- quantile(data[[curVar]], 0.75, na.rm = TRUE)   
   
   
 tibble(  
 Variable = curVar,  
 Value = sprintf("Median: %.\*f (IQR: %.\*f - %.\*f)", curSig, medianVal, curSig, q25, curSig, q75)  
 )  
 })  
  
 # Get cat stats  
 catResults <- catVars %>%  
 map\_df(~{  
 curVar <- .x  
 curLevels <- sort(unique(data[[curVar]]))  
   
 stats <- map\_df(curLevels, ~{  
 curLevel <- .x  
 curCount <- sum(data[[curVar]] == curLevel, na.rm = TRUE)  
   
 tibble(  
 Variable = sprintf("%s: %s", curVar, as.character(curLevel)),  
 Value = sprintf("%.1f%% (%d)", (100 \* curCount / nrow(data)), curCount)  
 )  
 })  
   
 missingCount <- sum(is.na(data[[curVar]]))  
 if (missingCount > 0) {  
 stats <- bind\_rows(stats, tibble(  
 Variable = sprintf("%s: Missing", curVar),  
 Value = sprintf("%.1f%% (%d)", (100 \* missingCount / nrow(data)), missingCount)  
 ))  
 }  
 stats  
 })  
  
 outTable <- bind\_rows(  
 tibble(Variable = "N", Value = as.character(nrow(data))),  
 contResults,  
 catResults  
 )  
   
 return(outTable)  
}  
  
  
  
# Generate tables and write to Excel on separate sheets  
write\_xlsx(list(Pooled = generateTable(fenlandPhase1 %>% left\_join(fenlandCovariates, by="id\_Fenland") ,   
 c("age", "height", "weight", "bmi", "ffm", "shr", "fitnessTBM"),   
 c(0, 0, 1, 1, 1, 0, 1),   
 c("ethnicity2c", "sex2c")  
 ),   
 Women = generateTable(fenlandPhase1 %>% left\_join(fenlandCovariates, by="id\_Fenland") %>% filter(sex2c == 0),   
 c("age", "height", "weight", "bmi", "ffm", "shr", "fitnessTBM"),   
 c(0, 0, 1, 1, 1, 0, 1),   
 c("ethnicity2c")  
 ),   
 Men = generateTable(fenlandPhase1 %>% left\_join(fenlandCovariates, by="id\_Fenland") %>% filter(sex2c == 1),   
 c("age", "height", "weight", "bmi", "ffm", "shr", "fitnessTBM"),   
 c(0, 0, 1, 1, 1, 0, 1),   
 c("ethnicity2c")  
 )  
 ),  
 file.path(baseDir, "Analysis/Proteomic\_fitness/TG421\_Analysis/Fenland\_Table\_1\_Final\_v2.xlsx")  
 )

## Perform single protein regressions

This code perfoms regression analysis of standardised fitness against standardised protein values, with progressive levels of adjustment.

Level 1: none Level 2: age, sex ethnicity, test site Level 3: additionally adjusted for bmi Level 4: additionally adjusted for protein:sex interaction Level 5: additionally adjusted for protein:age interaction

doRegressions <- function(data, outcome, predictor, adj, i) {  
   
 # Replace 'protein' with actual protein name in interaction terms  
 adjustedAdj <- lapply(adj, function(x) gsub("protein", predictor, x))  
  
 # Construct formula string with adjustment terms  
 formulaString <- paste0(outcome, "~", predictor,   
 if (length(adjustedAdj) > 0) {  
 paste(" + ", paste(adjustedAdj, collapse = " + "))  
 } else {  
 ""  
 })  
   
 model <- lm(as.formula(formulaString), data = data)  
 tidied <- broom::tidy(model, conf.int = TRUE)   
   
 result <- tidied %>%  
 #filter(term == predictor) %>%  
 mutate(protein = sub("\_lnstd", "", predictor),  
 adjustment.level = paste0("Adj\_", i),   
 adjustment.vars = ifelse(is.null(adj), "None", paste(adj, collapse = ", ")),   
 term = gsub(predictor, "protein", term)) %>%  
 select(protein,   
 adjustment.level,   
 adjustment.vars,  
 term,  
 estimate,   
 std.error,   
 conf.low,   
 conf.high,   
 p.value  
 )  
   
 return(result)  
}  
  
  
# Check if regression results exist  
  
if (file.exists(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/Protein\_regressions\_final\_v2.csv"))){  
  
 regResults <- read\_csv(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/Protein\_regressions\_final\_v2.csv"), show\_col\_types = FALSE)  
  
} else {  
   
 #Define objects to run regressions on proteins  
 regData <- fenlandPhase1 %>% left\_join(fenlandCovariates, by="id\_Fenland")   
 proteinList <- names(regData)[grepl("SeqId\_", names(regData))]  
   
 adjList <- list(NULL,   
 c("age\_std", "sex2c", "ethnicity2c", "testSite"),  
 c("age\_std", "sex2c", "ethnicity2c", "testSite", "bmi\_lnstd"),  
 c("age\_std", "sex2c", "ethnicity2c", "testSite", "bmi\_lnstd", "protein:sex2c"),  
 c("age\_std", "sex2c", "ethnicity2c", "testSite", "bmi\_lnstd", "protein:age\_std")  
 )  
   
 #Do regressions for each protein and adjustment combination  
 regResults <- map\_dfr(seq\_along(adjList), function(i){  
 curAdj <- adjList[[i]]  
 map\_dfr(proteinList, ~doRegressions(data = regData ,   
 outcome = "fitnessTBM\_std",   
 predictor = .x,   
 adj = curAdj,   
 i=i  
 ))  
 })  
   
 write.csv(regResults,file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/Protein\_regressions\_final\_v2.csv"))  
   
 rm(regData,proteinList,adjList)  
}

## New names:  
## • `` -> `...1`

## Data split and scaling

This code splits the phase 1 dataset into derivation and validation sets using a 70/30 split. We then scale the derivation and validation sets separately.

#Function to create data partition on two variables, and then scale each partition separately  
definePartition <- function(curData,   
 idVar,  
 varName,  
 curThresh,   
 curSeed,  
 excludedVars = c()  
 )  
 {  
 set.seed(curSeed)  
   
 trainIndex <- caret::createDataPartition(curData[[varName]],  
 p=curThresh,  
 list = FALSE,  
 times = 1)  
   
   
 excludedVars <- c(idVar, excludedVars) # Combine idVar and additional variables to exclude (that cant be scaled)  
  
 scaleColumns <- !(names(curData) %in% excludedVars) # Define which columns to scale  
   
 # Scale the derive and validate samples separately  
 derive <- curData[trainIndex,]  
 derive[, scaleColumns] <- scale(derive[, scaleColumns])  
   
 validate <- curData[-trainIndex,]  
 validate[, scaleColumns] <- scale(validate[, scaleColumns])  
  
 return(list(derive = derive, validate = validate))   
   
 }  
  
# Create derivation and validation sets  
  
partitionResult <- definePartition(fenlandPhase1,  
 "id\_Fenland",   
 "fitnessTBM",  
 0.70,  
 1234,  
 excludedVars = "date"   
 )  
  
dataDerive <- partitionResult$derive %>% left\_join(fenlandCovariates, by = "id\_Fenland")   
dataValidate <- partitionResult$validate %>% left\_join(fenlandCovariates, by = "id\_Fenland")   
  
rm(partitionResult)

## LASSO model derivation

This code fits a LASSO model to the Fenland proteins against fitness, using either the full protein list or the filtered protein list that intersects with CARDIA.

# Define function for unadjusted or adjusted LASSO model  
  
modelLASSO <- function(curData,   
 depVar,   
 indVars,   
 covVars=NULL,   
 curSeed,  
 curControl  
 )  
 {  
 set.seed(curSeed)   
   
 allVars <- c(depVar, covVars, indVars)  
 curData <- curData[, allVars]  
   
   
 # Create a penalty factor vector with 1s for indVars and 0s for controlVars  
 penaltyFactor <- c(rep(0, length(covVars)), rep(1, length(indVars)))  
   
  
 # Create a formula string from the dependent variable and all predictors  
 curFormula <- paste(depVar, "~", paste(allVars[-1], collapse = " + "))  
  
 curModel <- caret::train(as.formula(curFormula),  
 data = curData,  
 method = "glmnet",  
 family = "gaussian",  
 penalty.factor = penaltyFactor,   
 tuneGrid=expand.grid(alpha=1, lambda= 10\*\*(seq(-5,5,by=0.2))),  
 trControl=curControl  
 )  
 return(curModel)  
 }  
  
  
  
# Check if models exits  
  
if (file.exists(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/FenlandModels.Rdata"))){  
   
 # If does, bring into memory  
   
 load(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/FenlandModels.Rdata"))  
  
} else {  
   
 # If does not, derive models (this takes a long time)  
 # Define the proteinList  
  
 proteinList <- list(  
 Full = names(fenlandPhase1)[grepl("SeqId\_", names(fenlandPhase1))],  
 Filt = read\_csv(file.path(baseDir,"Analysis/Proteomic\_fitness/Coefs\_CARDIA\_Fen/exercise\_recalFenland\_lasso\_coefs.csv"), show\_col\_types = FALSE) %>%  
 filter(filtered\_protein\_set\_included == "Included") %>%  
 pull(SeqId\_SQL) %>%  
 paste0("\_lnstd")  
 )  
   
   
 # Define control parameters  
 lassoControl <- trainControl(method="repeatedcv",  
 number=10,  
 repeats=5,  
 returnData=FALSE,  
 savePredictions="all",  
 verboseIter=FALSE)  
   
   
 # Define dependent variable and control variables  
 depVar <- "fitnessTBM\_std"  
 covVars <- c("age\_std", "bmi\_lnstd", "sex2c", "ethnicity2c")  
 seedVal <- 1234  
   
   
 fenlandModels <- list(  
 fullLASSO = modelLASSO(dataDerive, depVar, proteinList$Full, covVars, seedVal, lassoControl),  
 filtLASSO = modelLASSO(dataDerive, depVar, proteinList$Filt, covVars, seedVal, lassoControl)  
 )  
   
 rm(lassoControl,depVar,covVars,seedVal)  
   
 save(fenlandModels, file = file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/FenlandModels.Rdata"))   
   
}

## Get coefficient lists from Fenland models and CARDIA models

This code computes the coefficients for models in Fenland and models derived in CARDIA.

# Functions to extract coefficients from models  
getFenlandCoefs <- function(model, lambda) {  
 coef(model$finalModel, lambda) %>%  
 as.matrix() %>%  
 as.data.frame() %>%  
 rownames\_to\_column(var="protein") %>%  
 filter(!protein %in% c("(Intercept)", "age\_std","bmi\_lnstd","sex2c","ethnicity2c")) %>%  
 rename(coef = s1)   
}  
  
getCardiaCoefs <- function(file\_path, protein\_col, coef\_col) {  
 read.csv(file\_path) %>%  
 select({{protein\_col}}, {{coef\_col}}) %>%  
 rename(protein = {{protein\_col}}, coef = {{coef\_col}})  
}  
  
  
  
  
# Coefficients from Fenland models  
fenlandCoefs <- list(  
 fullLASSO = getFenlandCoefs(fenlandModels$fullLASSO, fenlandModels$fullLASSO$bestTune$lambda),  
 filtLASSO = getFenlandCoefs(fenlandModels$filtLASSO, fenlandModels$filtLASSO$bestTune$lambda)  
)  
  
  
  
# Coefficients from CARDIA models  
cardiaCoefs <- list(  
 fullLASSO = getCardiaCoefs(file.path(baseDir,"Analysis/Proteomic\_fitness/Coefs\_CARDIA\_Fen/exercise\_recalFenland\_lasso\_coefs.csv"), SeqId\_SQL, G22DURTN\_score) %>%   
 mutate(protein = paste0(protein, "\_lnstd")) ,  
 filtLASSO = getCardiaCoefs(file.path(baseDir,"Analysis/Proteomic\_fitness/Coefs\_CARDIA\_Fen/exercise\_recalFenland\_lasso\_coefs.csv"), SeqId\_SQL, G22DURTN\_filtered\_score) %>%   
 mutate(protein = paste0(protein, "\_lnstd"))  
)  
  
write.csv(fenlandCoefs$fullLASSO,file.path(baseDir,"Analysis/Proteomic\_fitness/Coefs\_CARDIA\_Fen/fenland\_LASSO\_full.csv"))  
write.csv(fenlandCoefs$filtLASSO,file.path(baseDir,"Analysis/Proteomic\_fitness/Coefs\_CARDIA\_Fen/fenland\_LASSO\_filtered.csv"))

## Calculate model protein scores

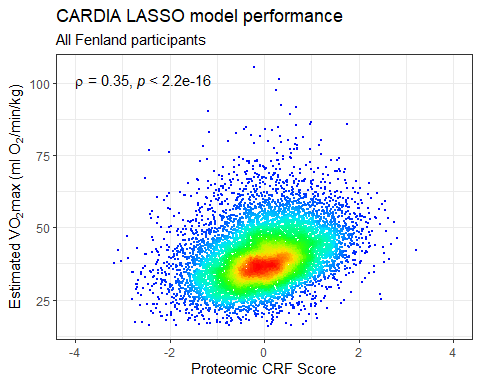
This code computes protein scores for the CARDIA models.

calcScores <- function(curCoefs, curData, curModel, idVar){  
  
 # Get the proteins that have coefficients  
 proteinList <- curCoefs[,1]  
   
 # Subset the data to only the proteins in proteinList  
 curData <- curData[, c(idVar, proteinList)]   
  
 scores <- data.frame(id\_Fenland = curData[[idVar]],   
 curScores = as.matrix(curData %>% select(all\_of(proteinList))) %\*% as.vector(curCoefs[,2])  
 )  
   
 scores <- setNames(scores, c("id\_Fenland", curModel))  
   
 return(scores)  
   
 }  
  
  
# Calc protein scores for CARDIA models in full Fenland P1 and P2 samples  
  
for (model in c("fullLASSO","filtLASSO")){  
 for (phase in c("1","2")){  
 curScore <- calcScores(cardiaCoefs[[model]], get(paste0("fenlandPhase", phase)), paste0(model, ".P", phase), "id\_Fenland")  
 if (!exists("cardiaScores")) {  
 cardiaScores <- curScore  
 } else {  
 cardiaScores <- left\_join(cardiaScores, curScore, by = "id\_Fenland")  
 }  
 }  
 rm(curScore, model, phase)  
}

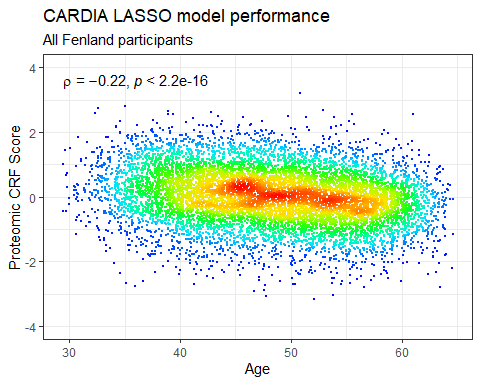
## Generate figures

This code generates summary figures that examine the association between computed scores with fitness, BMI age, sex, and race.

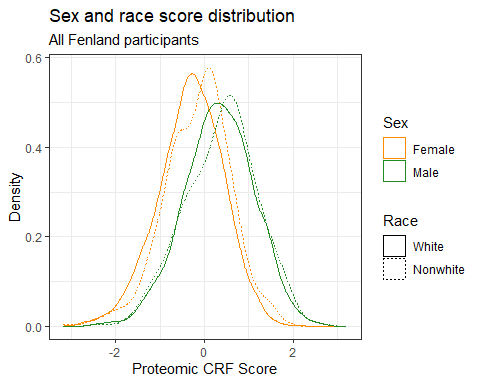
dataForPlots <- cardiaScores %>%  
 left\_join(fenlandCovariates, by="id\_Fenland") %>%  
 left\_join(fenlandPhase1 %>% select(id\_Fenland, fitnessTBM, age, bmi), by="id\_Fenland")   
  
  
# Correlation plot  
ggplot(data=dataForPlots, aes(x=fullLASSO.P1, y=fitnessTBM)) +  
 geom\_point(size=0.5, aes(col = densCols(fullLASSO.P1, fitnessTBM, colramp = colorRampPalette(rev(rainbow(10, end = 4/6)))))) +  
 scale\_color\_identity() +  
 theme\_bw() +  
 ggpubr::stat\_cor(cor.coef.name = "rho", method="spearman") +  
 xlim(-4, 4) +  
 labs(title = "CARDIA LASSO model performance",  
 subtitle = "All Fenland participants",  
 x = "Proteomic CRF Score",  
 y = expression("Estimated VO"[2]\*"max"\*" (ml O"[2]\*"/min/kg)"))



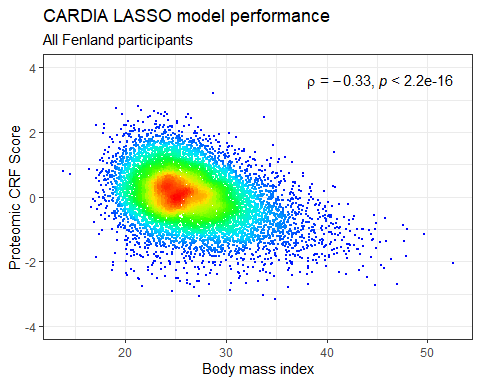
ggsave(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/score\_ett\_corr\_v2.pdf"), width = 4, height = 3)  
  
  
## Age  
  
ggplot(data=dataForPlots, aes(x=age, y=fullLASSO.P1)) +  
 geom\_point(size=0.5, aes(col = densCols(age, fullLASSO.P1, colramp = colorRampPalette(rev(rainbow(10, end = 4/6)))))) +  
 scale\_color\_identity() +  
 theme\_bw() +  
 ggpubr::stat\_cor(cor.coef.name = "rho", method="spearman", label.x.npc = "left", label.y.npc = "top") +  
 ylim(-4, 4) +  
 labs(title = "CARDIA LASSO model performance",  
 subtitle = "All Fenland participants",  
 y = "Proteomic CRF Score",  
 x = "Age")



ggsave(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/score\_age\_corr\_v2.pdf"), width = 4, height = 3)  
  
  
## Sex and race  
  
write.csv(tidy(lm(fullLASSO.P1 ~ sex2c\*ethnicity2c, data=dataForPlots)),file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/Table\_output.csv"))   
  
  
colorSet <- c("Female" = "darkorange", "Male" = "forestgreen")  
  
ggplot(dataForPlots, aes(x=fullLASSO.P1)) +  
 geom\_density(aes(colour = factor(sex2c, labels = c("Female", "Male")),   
 linetype = factor(ethnicity2c, labels = c("White", "Nonwhite")))) +  
 scale\_fill\_manual(values=colorSet) +  
 scale\_colour\_manual(values = colorSet) +  
 labs(x = "Proteomic CRF Score",  
 y = "Density",  
 title = "Sex and race score distribution",  
 subtitle = "All Fenland participants") +  
 theme\_bw() +  
 guides(color=guide\_legend(title = "Sex"),  
 linetype=guide\_legend(title = "Race"))



ggsave(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/score\_sexRace\_histos\_v2.pdf"), width = 4, height = 3)  
  
  
## BMI  
  
ggplot(data=dataForPlots, aes(x=bmi, y=fullLASSO.P1)) +  
 geom\_point(size=0.5, aes(col = densCols(bmi, fullLASSO.P1, colramp = colorRampPalette(rev(rainbow(10, end = 4/6)))))) +  
 scale\_color\_identity() +  
 theme\_bw() +  
 ggpubr::stat\_cor(cor.coef.name = "rho", method="spearman", label.x.npc = "right", hjust=1) +  
 ylim(-4, 4) +  
 labs(title = "CARDIA LASSO model performance",  
 subtitle = "All Fenland participants",  
 y = "Proteomic CRF Score",  
 x = "Body mass index")



ggsave(file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/score\_bmi\_corr\_v2.pdf"), width = 4, height = 3)  
  
  
rm(dataForPlots, colorSet)

## VO2max to protein score sensitivity analyses

This code performs regressions examining the relationship between fitenss and the derived CARDIA protein score, adjusting for various factors

doRegressions2 <- function(data, outcome, predictor, adj, i) {  
   
 # Replace 'protein' with actual protein name in interaction terms  
 adjustedAdj <- lapply(adj, function(x) gsub("score", predictor, x))  
  
 # Construct formula string with adjustment terms  
 formulaString <- paste0(outcome, "~", predictor,   
 if (length(adjustedAdj) > 0) {  
 paste(" + ", paste(adjustedAdj, collapse = " + "))  
 } else {  
 ""  
 })  
   
 model <- lm(as.formula(formulaString), data = data)  
 tidied <- broom::tidy(model, conf.int = TRUE)   
   
 result <- tidied %>%  
 mutate(score = sub(".P1", "", predictor),  
 adjustment.level = paste0("Adj\_", i),   
 adjustment.vars = ifelse(is.null(adj), "None", paste(adj, collapse = ", ")),   
 term = gsub(predictor, "score", term)) %>%  
 select(score,   
 adjustment.level,   
 adjustment.vars,  
 term,  
 estimate,   
 std.error,   
 conf.low,   
 conf.high,   
 p.value  
 )  
   
 return(result)  
}  
  
  
#Define objects to run regressions  
regData <- cardiaScores %>%  
 left\_join(fenlandCovariates, by="id\_Fenland") %>%  
 left\_join(fenlandPhase1 %>% select(id\_Fenland, fitnessTBM\_std, age\_std, bmi\_lnstd), by="id\_Fenland")  
  
  
adjList <- list(c("age\_std", "sex2c", "ethnicity2c", "score:age\_std"),  
 c("age\_std", "sex2c", "ethnicity2c", "score:age\_std", "bmi\_lnstd"),  
 c("age\_std", "sex2c", "ethnicity2c", "score:sex2c"),  
 c("age\_std", "sex2c", "ethnicity2c", "score:sex2c", "bmi\_lnstd"),  
 #c("age\_std", "sex2c", "ethnicity2c", "score:age\_std", "score:sex2c"),  
 #c("age\_std", "sex2c", "ethnicity2c", "score:age\_std", "score:sex2c", "bmi\_lnstd"),  
 c("age\_std", "sex2c", "ethnicity2c", "score:ethnicity2c"),  
 c("age\_std", "sex2c", "ethnicity2c", "score:ethnicity2c", "bmi\_lnstd")  
 )  
  
scoreList <- "fullLASSO.P1"  
  
#Do regressions for each protein and adjustment combination  
regResults <- map\_dfr(seq\_along(adjList), function(i){  
 curAdj <- adjList[[i]]  
 map\_dfr(scoreList, ~doRegressions2(data = regData ,   
 outcome = "fitnessTBM\_std",   
 predictor = .x,   
 adj = curAdj,   
 i=i  
 ))  
})  
  
write.csv(regResults,file.path(baseDir,"Analysis/Proteomic\_fitness/TG421\_Analysis/Score\_regressions\_final\_v2.csv"))  
  
rm(regData,scoreList,adjList)

## Get MD5 checksum values

# File list  
fileList <- c("Protein\_regressions\_final\_v2.csv",  
 "score\_ett\_corr\_v2.pdf",  
 "score\_age\_corr\_v2.pdf",  
 "score\_sexRace\_histos\_v2.pdf",  
 "score\_bmi\_corr\_v2.pdf",  
 "Score\_regressions\_final\_v2.csv",  
 "Fenland\_Table\_1\_Final\_v2.xlsx")  
  
  
# Compute MD5 checksums  
checksums <- lapply(fileList, function(file) {  
 file\_path <- file.path(baseDir, "Analysis/Proteomic\_fitness/TG421\_Analysis", file)  
 if (file.exists(file\_path)) {  
 return(data.frame(fileName = file, md5 = digest(file\_path, algo = "md5", file = TRUE)))  
 } else {  
 return(data.frame(fileName = file, md5 = NA))  
 }  
})  
  
checksums <- do.call(rbind, checksums)  
  
write\_xlsx(checksums, file.path(baseDir, "Analysis/Proteomic\_fitness/TG421\_Analysis/Checksums\_v2.xlsx"))