Large-scale genomic investigation of the gut microbiome on Parkinson's disease etiology - Code

Rodrigo Sandon

11/27/2021

Preprocessing GWAS Summary Datasets

The first step of this pipeline was to identify datasets we'd be performing our association studies on. Go to https://mibiogen.gcc.rug.nl/ to find the repository of datasets provided by Kurilshikov et. al, 2021 (https://pubmed.ncbi.nlm.nih.gov/33462485/). On this website, you'll find 6 links:

1. MBG.allHits.p1e4.txt -> top hit variants ($p < 1e-4$) for any level of the bacterial hierarchy
2. MiBioGen_QmbQTL_summary_phylum.zip (2.4 GB) -> summary statistics of bacterial phyla
3. MiBioGen_QmbQTL_summary_class.zip (4.4 GB) -> summary statistics of bacterial classes
4. MiBioGen_QmbQTL_summary_order.zip (5.4 GB) ->summary statistics of bacterial orders
5. MiBioGen_QmbQTL_summary_family.zip (9.5 GB) -> summary statistics of bacterial families
6. MiBioGen_QmbQTL_summary_genus.zip (35.0 GB) -> summary statistics of bacterial genera

In our case, we wanted explore the bacterial genera taxanomy. In latter analyses, we hope to run family, order, class, and phylum taxanomies of bacteria. When you unzip the MiBioGen_QmbQTL_summary_genus.zip, it will look something like this:

- MiBioGen_QmbQTL_summary_genus:
    - genus.Clostridiuminnocuumgroup.id.14397.summary.txt.gz
    - genus.Eubacteriumbrachygroup.id.11296.summary.txt.gz

Unzip the subfolders and you get a table like this:

bac	chr	bp	rsID	ref.allele	eff.allele	beta	SE	Z weighted	P weighted	N	Cohorts
name	5	71186626	rs6890185	C	T	0.113	0.023	4.868	1.122e-06	4166	20

Now perform these data pre-processing steps (on sample data):

Import "reticulate" to incorporate python functionalities.

```r
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=80), tidy=TRUE)

library(reticulate)
use_virtualenv("base")
use_python("/Volumes/T7Touch/Applications/anaconda3/bin/python")
```

Each .gz file of the sum stats will give a txt file, convert these into csv files.

```
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=80), tidy=TRUE)

sumstats_root <- "/Users/rodrigosandon/Documents/GitHub/ParkisonsMicrobiomeStudy/MB-PD_Association_Stud

source_python("Utilities.py")
out_paths <- txt_to_csv_files_in_root(sumstats_root)
# If this doesn't work, run the py file itself with the process execution
```

Add a "p" column to each sum stat csv file.

```
source_python("Utilities.py")

knitr::opts_chunk$set(tidy.opts=list(width.cutoff=80), tidy=TRUE)

add_p_col_to_df <- function(bac_sumstat_path, new_col_name, out_path) {

  df <- read.csv(bac_sumstat_path, header = TRUE, sep = ",")
  df[new_col_name] <- 2*pnorm(-abs(df$beta/df$SE))
  eles_of_bac_path <- strsplit(bac_sumstat_path, "/")
  name_of_bac_sumstat <- eles_of_bac_path[[1]][length(eles_of_bac_path[[1]])]
  # ^finding the name original name of the bac sumstat file

  new_name <- paste("addedP_", name_of_bac_sumstat, sep = "")
  new_path <- paste(out_path, new_name, sep = "")
  #print(new_path)
  # Export
  write.csv(df, new_path, row.names = FALSE, quote = FALSE)

}

create_lst_of_file_paths <- function(root_path, files_endswith_str) {
  files <- list.files(path = root_path, pattern = files_endswith_str, full.names = TRUE,
                      recursive = FALSE)
  return (files)
}

add_p_col_to_csvs_in_root <- function(root_path, out_path, files_endswith_str,
                                      new_col_name, omit_csvs_that_contain) {
  dir.create(out_path, showWarnings = FALSE) #uncomment if dir hasn't been created yet, else, keep comm
  files <- create_lst_of_file_paths(root_path, files_endswith_str)
  # Reminder: current files in the list should be in csv format

  for (bac_sumstat_path in files) {
    print(paste("Working on ...", bac_sumstat_path))
    start.time <- Sys.time()
    if (grepl(omit_csvs_that_contain, bac_sumstat_path, fixed = TRUE) == FALSE) {
      add_p_col_to_df(bac_sumstat_path, new_col_name, out_path)
    }
    end.time <- Sys.time()
    print(paste("Time to process: ", end.time - start.time))
  }
}
```

```
ROOT_PATH <- "/Users/rodrigosandon/Documents/GitHub/ParkisonsMicrobiomeStudy/MB-PD_Association_Study_Pip
OUT_PATH <- "/Users/rodrigosandon/Documents/GitHub/ParkisonsMicrobiomeStudy/MB-PD_Association_Study_Pipe

#perform this function only for the bacterial genera in which the names are known
add_p_col_to_csvs_in_root(ROOT_PATH, OUT_PATH, files_endswith_str = "*.csv",
                          new_col_name = "pDerived", omit_csvs_that_contain = "unknown")
```

Polygenic Risk Score Analysis

Make a txt file of all of the genera you will be performing PRS on

```
#Identify the path where you've modified the sumstats
sumstats_path <- "/Users/rodrigosandon/Documents/GitHub/ParkisonsMicrobiomeStudy/MB-PD_Association_Stud

#Define the group of files to identify within ^ this root path
look_for <- "addedP_"

#Identify path where you'll locate the txt file
txt_out <- "/Users/rodrigosandon/Documents/GitHub/ParkisonsMicrobiomeStudy/MB-PD_Association_Study_Pipel

source_python("Utilities.py")

files_lst <- find_paths_startswith(sumstats_path, look_for)
listdir_to_txt_file(files_lst, txt_out)
```

Using PRSice.R, perform PRS and acquire the output file

```
### Polygenic risk score analyses of 119 bacteria genuses versus PD risk

## Make a list of summary stats file
ls *csv > /Users/rodrigosandon/Documents/GitHub/ParkisonsMicrobiomeStudy/MB-PD_Association_Study_Pipeli

## Format these files
cat genuses.txt | while read LINE
do
  echo $LINE
  sed 's/\"//g' $LINE | sed 's/,/ /g' > temp.txt
  awk '{print $0"\t"$2":"$3}' temp.txt | sed 's/chr\:bp/ID/' > $LINE.temp_formatted.txt
  rm temp.txt
done

## Identify independent risk SNPs using our in-house LD reference data for European populations (/data/

Rscript /data/LNG/pdMeta5v2/leaveOneOutPrsice/PRSice_linux/PRSice.R --cov-file /data/LNG/saraB/WGS/noag
Rscript /data/LNG/pdMeta5v2/leaveOneOutPrsice/PRSice_linux/PRSice.R --cov-file /data/LNG/saraB/WGS/noag

## Remove NeuroX individuals & extract nominated variants
cat genuses_formatted_list.txt | while read LINE
do
```

```
plink --bfile /data/LNG/saraB/concat_HARDCALLS_PD_september_2018_no_cousins --remove-fam NeuroX.txt --e
done

## Make score files
cat genuses_formatted_list.txt | while read LINE
do
awk '{print $14, $6, $7}' addedPgenus.$LINE.summary.txt.csv.temp_formatted.txt | sed '1d' > $LINE.tosco
done

## Make sure score files have 3 expected fields rather than 2
cat genuses_formatted_list.txt | while read LINE
do
grep ":" $LINE.toscore.txt > true_$LINE.toscore.txt
done

## Calculate scores
cat genuses_formatted_list.txt | while read LINE
do
plink --bfile pruned_$LINE --score $LINE.toscore.txt --make-bed --out pruned_$LINE
done
```

Run PRS (logistic regression) on R

```
#install.packages("data.table")
library("data.table")
listOfProfiles <- read.table("genera_formatted_list.txt", header = T)
names(listOfProfiles) <- c("id")
covs1 <- fread("/data/LNG/saraB/WGS/noage_toPRSice_phenosAndCovs_renamed.tab", header = T)
covs2 <- fread("/data/LNG/saraB/concat_HARDCALLS_PD_september_2018_no_cousins.fam", header = F)
colnames(covs2) <- c("FID", "IID", "MAT", "PAT", "SEX", "PHENO")
covsfinal <- merge (covs1, covs2, by ="FID")
covsfinal$CASE <- covsfinal$PHENO.x - 1
outPut <- matrix(ncol = 4, nrow = length(listOfProfiles$id), NA)
colnames(outPut) <- c("genus","b","se","p")
for(i in 1:length(listOfProfiles$id))
{
    profileName <-  as.character(listOfProfiles$id[i])
    profile <- fread(file = paste(profileName, ".profile", sep = ""), header = T)
    profile$index <- paste(profile$FID, profile$IID, sep = "")
    data <- merge(covsfinal, profile, by = "index")
    meanControls <- mean(data$SCORE[data$CASE == 0])
    sdControls <- sd(data$SCORE[data$CASE == 0])
    data$zSCORE <- (data$SCORE - meanControls)/sdControls
    grsTest <- glm(CASE ~ zSCORE + SEX + PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC8 + PC9 + PC10 + D
    beta <- summary(grsTest)$coefficients["zSCORE","Estimate"]
    se <- summary(grsTest)$coefficients["zSCORE","Std. Error"]
    p <- summary(grsTest)$coefficients["zSCORE","Pr(>|z|)"]
    outPut[i,1] <- profileName
    outPut[i,2] <- beta
    outPut[i,3] <- se
    outPut[i,4] <- p
}
write.table(outPut, "Genus_PRS.tab", quote = F, sep = "\t", row.names = F)
```

```
## SAMPLE RESULT:

# Call:
# glm(formula = CASE ~ zSCORE + SEX + PC1 + PC2 + PC3 + PC4 + PC5 +
#     PC6 + PC7 + PC8 + PC9 + PC10, family = "binomial", data = data)
#
# Deviance Residuals:
#    Min       1Q  Median       3Q      Max
# -1.919  -1.003  -0.810   1.278   1.796
#
# Coefficients:
#              Estimate Std. Error z value Pr(>|z|)
# (Intercept)   0.43040    0.03974  10.831  < 2e-16 ***
# zSCORE        0.03021    0.01424   2.121   0.0339 *
# SEX          -0.58575    0.02603 -22.505  < 2e-16 ***
# PC1         -35.38600    2.73911 -12.919  < 2e-16 ***
# PC2          50.17593    2.86877  17.490  < 2e-16 ***
# PC3          10.63757    2.68575   3.961 7.47e-05 ***
# PC4           0.63048    2.65991   0.237   0.8126
# PC5          16.19265    2.70187   5.993 2.06e-09 ***
# PC6         -24.20283    2.74525  -8.816  < 2e-16 ***
# PC7           1.61607    2.62627   0.615   0.5383
# PC8          12.66905    2.70987   4.675 2.94e-06 ***
# PC9          -5.96044    2.64009  -2.258   0.0240 *
# PC10        -16.18338    2.65955  -6.085 1.16e-09 ***
# ---
# Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
#     Null deviance: 35275  on 26385  degrees of freedom
# Residual deviance: 34124  on 26373  degrees of freedom
# AIC: 34150
#
# Number of Fisher Scoring iterations: 4
```

Linkage Disequilibirum Score Regression (LDSC) Analysis

Data Preprocessing

```
pre_ldsc_chrPosRs <- source_python("/Users/rodrigosandon/Documents/GitHub/ParkisonsMicrobiomeStudy/MB-P

# chrPosRs.tab available upon request
# reformatted2_META5_all_with_rsid.txt available upon request
#
```

LDSC

```
import pandas as pd
import csv
import os
import time
import os.path
from os import path

# the PD file for LD SR run is : /Volumes/T7Touch/NIHSummer2021/Data/LDSR_analysis/nalls_onlyRsIDs.sums

# Ex: /Volumes/Passport/119Bacs_addedP/addedPgenus..Clostridiuminnocuumgroup.id.14397.summary.txt.csv


def formatSumStatsForBac(csvPath):
    newFileName = "formatted." + csvPath.split("/")[4].replace(
        ".txt", "").replace("..", ".").replace("addedPgenus", "addedP.genus")
    newFilePath = "/Volumes/Passport/formatted119Bacs_addedP/%s" % (
        newFileName)

    if path.exists(newFilePath) == False:
        df = pd.read_csv(csvPath)
        newDf = df[['rsID', 'eff.allele', 'ref.allele',
                    'beta', 'SE', 'N', 'pDerived']].copy()
        newDf.insert(3, 'Zscore', newDf['beta'] / newDf['SE'], True)
        newDf = newDf.drop(['beta', 'SE'], axis=1)
        newDf = newDf.rename(columns={
                             'rsID': 'snpid', 'pDerived': 'P-value', 'eff.allele': 'A1', 'ref.allele':

        # Now formatted.addedP.genus.Clostridiuminnocuumgroup.id.14397.summary.csv

        newDf.to_csv(newFilePath, index=False)
    else:
        print("File %s already exists" % (newFilePath))
    return newFilePath


def CSVtoTXT(csv_file, txtFile):
    with open(txtFile, "w") as my_output_file:
        with open(csv_file, "r") as my_input_file:
            [my_output_file.write(" ".join(row)+'\n')
             for row in csv.reader(my_input_file)]
        my_output_file.close()

# ex csv file now: /Volumes/Passport/formatted119Bacs_addedP/formatted.addedP.genus.Clostridiuminnocuum


def mungeDataCall(csv_file):

    txtFilePath = csv_file.replace(".csv", ".txt")
    # /Volumes/Passport/formatted119Bacs_addedP/formatted.addedP.genus.Clostridiuminnocuumgroup.id.1439

    newFilePath = txtFilePath.replace(
        "formatted119Bacs_addedP", "munge119Bacs_output").replace(".txt", "")

    if path.exists(txtFilePath) == False:
        CSVtoTXT(csv_file, txtFilePath)
    else:
        print("File %s already munged" % (newFilePath))
    # os.chdir("/Users/rodrigosandon/ldsc")

    cmd = "./munge_sumstats.py \
        --sumstats %s \
        --out %s \
        --merge-alleles /Volumes/T7Touch/NIHSummer2021/Data/LDSR_analysis/ldsc/w_hm3.snplist" % (txtFil

    if path.exists(newFilePath + ".sumstats.gz") == False:  # only if .gz file don't exist
        os.system(cmd)

    return newFilePath

# ex gx bac name munged: /Volumes/Passport/munge119Bacs_output/formatted.addedP.genus.Clostridiuminnocu


def LDscore_regression(munged_bac_output):
    print("munged_bac_output: ", munged_bac_output)
    LDSR_outName = "/Volumes/T7Touch/NIHSummer2021/Code/LDSC_analysis2/results/%s_ldscResults" % (
        munged_bac_output.split("/")[4].split(".")[3])  # <--only bac name

    # os.chdir("/Users/rodrigosandon/ldsc")

    cmd = "./ldsc.py \
        --rg %s,/Volumes/T7Touch/NIHSummer2021/Code/LDSC_analysis2/munged_META5_all_with_rsid.sumstats.
        --ref-ld-chr /Volumes/T7Touch/NIHSummer2021/Data/LDSR_analysis/ldsc/eur_w_ld_chr/ \
        --w-ld-chr /Volumes/T7Touch/NIHSummer2021/Data/LDSR_analysis/ldsc/eur_w_ld_chr/ \
        --out %s " % (munged_bac_output + ".sumstats.gz", LDSR_outName)

    os.system(cmd)

    return LDSR_outName

###MAIN###


masterDir = "/Volumes/Passport/119Bacs_addedP/"


for root, dirs, files in os.walk(masterDir):
    for name in files:
        start = time.time()
        bacPathToProcess = os.path.join(root, name)
        print("Processing", bacPathToProcess)

        newFilePath1 = formatSumStatsForBac(bacPathToProcess)

        newFilePath2 = mungeDataCall(newFilePath1)

        LDSR_outName = LDscore_regression(newFilePath2)

        print("Results can be found in", LDSR_outName)
        end = time.time()
        print("Time to perform LDSR:", (end-start)/60, "mins")
```

```
# BEFORE RUNNING FILE, MUNGE NEW PD GWAS w/ w_hm3 snplist
cmd = "./munge_sumstats.py \
        --sumstats /Volumes/T7Touch/NIHSummer2021/Code/LDSC_analysis2/reformatted2_META5_all_with_rsid.
        --out /Volumes/T7Touch/NIHSummer2021/Code/LDSC_analysis2/munged_META5_all_with_rsid \
        --merge-alleles /Volumes/T7Touch/NIHSummer2021/Data/LDSR_analysis/ldsc/w_hm3.snplist"
```

Including Plots

You can also embed plots, for example:

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.