Supplementary Material - OCD GWAS

Methods: Analysis Code

1. Filtering and aligning of GWAS summary statistics

A bash/awk script that iterates over each summary statistic file, and filters and aligns them:

- The directory structure is created first if it doesn't already exist.
- Each summary statistic file is read in as a data.table, filtered (MAF > 1% and INFO > 0.8), and then processed
- Merging, strand-flipping, and alignment to the HRC reference are done using merge functions.

Required:

- A file (called "files" here) with the names of each summary statistic file (one file name per line). Sumstat files have to be in the same folder ("00_daner")and need to be in daner-format (https://docs.google.com/document/d/1TWIhr8-qpCXB13WCXcU1_HDio8IC_MeWoAg2jlggrtU/edit?tab=t.0#heading=h.4008addvumol)
- HRC-reference file which contains the following information:
  SNP(rs-number) A1 A2 CHR_BP_A1_A2 MAF
  (can be downloaded HERE)

This script is arguably a bit difficult to follow (and probably to manipulate according to your needs). The same script was also created as an R-script (but not extensively tested, so use with caution), it can be found after the bash script.

```
#!/bin/bash

# Create folder structure
mkdir 01_filtered 02_newSNPIDs 03_alignedtoHRC 04_finalfiles 04_finalfiles_gz

#read in files:
NAMES="$(< files)"

# Creates files containing the header of each sumstats.
for NAME in $NAMES; do
cat 00_daner/daner_${NAME} |head -n1 > header_${NAME}

# filter all files to MAF > 0.01, INFO > 0.8 & < 1.2, and create SNP IDs in the form CHR_BP_A1_A2
awk 'NR == 1; NR > 1{if ($8>0.8 && $8<1.2 && $6>0.01 && $6<0.99 && $7>0.01 && $7<0.99) print
$1,$1"_"$3"_"$4"_"$5,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12}' < daner_${NAME} >
01_filtered/daner_${NAME}_Info0.8_MAF0.01

# Create switched & strand-flipped SNP IDs in the form CHR_BP_A2_A1, CHR_BP_A1_A2flipped,
CHR_BP_A2_A2flipped
awk '{if ($1=="CHR") print $0,"CHR_BP_A2_A1","CHR_BP_A1_A2flipped","CHR_BP_A2_A2flipped";
else if ($4=="A" && $5=="G") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_T_C",$1"_"$3"_C_T"; else if
($4=="G" && $5=="A") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_C_T",$1"_"$3"_T_C";else if
($4=="T" && $5=="C") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_A_G",$1"_"$3"_G_A";else if
($4=="C" && $5=="T") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_G_A",$1"_"$3"_A_G";else if
($4=="G" && $5=="T") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_C_A",$1"_"$3"_A_C";else if
($4=="T" && $5=="G") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_A_C",$1"_"$3"_C_A";else if
($4=="C" && $5=="A") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_G_T",$1"_"$3"_T_G";else if
($4=="A" && $5=="C") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,$1"_"$3"_T_G",$1"_"$3"_G_T";else if
($4=="A" && $5=="T") print $1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,"NA","NA";else
if ($4=="T" && $5=="A") print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,"NA","NA";else if ($4=="C" && $5=="G")
print $1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,"NA","NA";else if ($4=="G" &&
$5=="C") print $1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,"NA","NA"; else if (($4=="I")
|| ($5=="I") || ($4=="D") || ($5=="D")) print
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$1"_"$3"_"$5"_"$4,"NA","NA"}' <
01_filtered/daner_${NAME}_Info0.8_MAF0.01 >
02_newSNPIDs/daner_${NAME}_Info0.8_MAF0.01_newSNPID

#Compare the 4 SNP-ID versions to the ID in the HRC reference, merge them
awk 'NR==FNR{a[$4]=$0;next} ($2 in a){print $0 FS a[$2]}'
HRCSNPlist_chr1-22_reference_frq_BP_19052020
```

```
02_newSNPIDs/daner_${NAME}_Info0.8_MAF0.01_newSNPID> 03_alignedtoHRC/${NAME}_1
awk 'NR==FNR{a[$4]=$0;next} ($13 in a){print $0 FS a[$13]}'
HRCSNPlist_chr1-22_reference_frq_BP_19052020
02_newSNPIDs/daner_${NAME}_Info0.8_MAF0.01_newSNPID> 03_alignedtoHRC/${NAME}_2
awk 'NR==FNR{a[$4]=$0;next} ($14 in a){print $0 FS a[$14]}'
HRCSNPlist_chr1-22_reference_frq_BP_19052020
02_newSNPIDs/daner_${NAME}_Info0.8_MAF0.01_newSNPID> 03_alignedtoHRC/${NAME}_3
awk 'NR==FNR{a[$4]=$0;next} ($15 in a){print $0 FS a[$15]}'
HRCSNPlist_chr1-22_reference_frq_BP_19052020
02_newSNPIDs/daner_${NAME}_Info0.8_MAF0.01_newSNPID> 03_alignedtoHRC/${NAME}_4
cat 03_alignedtoHRC/${NAME}_1 03_alignedtoHRC/${NAME}_2 03_alignedtoHRC/${NAME}_3
03_alignedtoHRC/${NAME}_4 >
03_alignedtoHRC/daner_${NAME}_Info0.8_MAF0.01_newSNPID_merged

#Align to the HRC reference. Take the SNP-rs number from the reference. If alleles are strand-flipped,
flip them back. For ambigous A/T, C/G SNPs, check if in both files their allele frequency is either below
0.4 or above 0.6, then take as is, if one is below <0.4 and one is above 0.6 we assume a strand flip, so
we flip back; if allele frequency is between 0.4 and 0.6, exclude SNP.
awk '{if (($19==$2)&& (((($4=="C")&&($5=="A"))||
((($4=="A")&&($5=="C"))||(($4=="G")&&($5=="A"))||(($4=="A")&&($5=="G"))||(($4=="T")&&($5=="G"))||((
$4=="G")&&($5=="T"))||(($4=="T")&&($5=="C"))||(($4=="C")&&($5=="T")))) print
$1"\t"$16"\t"$3"\t"$4"\t"$5"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12;else if (($19==$13)&&
(((($4=="C")&&($5=="A"))||
((($4=="A")&&($5=="C"))||(($4=="G")&&($5=="A"))||(($4=="A")&&($5=="G"))||(($4=="T")&&($5=="G"))||((
$4=="G")&&($5=="T"))||(($4=="T")&&($5=="C"))||(($4=="C")&&($5=="T")))) print
$1"\t"$16"\t"$3"\t"$4"\t"$5"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12; else if (($19==$14)&&
(((($4=="C")&&($5=="A"))||
((($4=="A")&&($5=="C"))||(($4=="G")&&($5=="A"))||(($4=="A")&&($5=="G"))||(($4=="T")&&($5=="G"))||((
$4=="G")&&($5=="T"))||(($4=="T")&&($5=="C"))||(($4=="C")&&($5=="T")))) print
$1"\t"$16"\t"$3"\t"$17"\t"$18"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12; else if (($19==$15)&&
(((($4=="C")&&($5=="A"))||
((($4=="A")&&($5=="C"))||(($4=="G")&&($5=="A"))||(($4=="A")&&($5=="G"))||(($4=="T")&&($5=="G"))||((
$4=="G")&&($5=="T"))||(($4=="T")&&($5=="C"))||(($4=="C")&&($5=="T")))) print
$1"\t"$16"\t"$3"\t"$18"\t"$17"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12; else if (($19==$2)&&
(((($4=="T")&&($5=="A")) ||(($4=="A")&&($5=="T"))
||(($4=="G")&&($5=="C"))||(($4=="C")&&($5=="G")))&&(($6<0.4)||($6>0.6))&&((($6<0.5) && ($20<0.5))
|| (($6>0.5) && ($20>0.5)))) print  $1"\t"$16"\t"$3"\t"$4"\t"$5"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12;
else if (($19==$13)&& (((($4=="T")&&($5=="A")) ||(($4=="A")&&($5=="T"))
||(($4=="G")&&($5=="C"))||(($4=="C")&&($5=="G")))&& (($6<0.4)||($6>0.6))&& ((($6<0.5) &&
($20>0.5)) || (($6>0.5) && ($20<0.5)))) print
$1"\t"$16"\t"$3"\t"$4"\t"$5"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12}' <
03_alignedtoHRC/daner_${NAME}_Info0.8_MAF0.01_newSNPID_merged >
04_finalfiles/daner_${NAME}_Info0.8_MAF0.01_newSNPID_aligned_noheader

# add header and gzip file.
cat 00_header/header_daner_${NAME}
04_finalfiles/daner_${NAME}_Info0.8_MAF0.01_newSNPID_aligned_noheader >
04_finalfiles/daner_${NAME}_Info0.8_MAF0.01_newSNPID_aligned
rm 04_finalfiles/daner_${NAME}_Info0.8_MAF0.01_newSNPID_aligned_noheader
mv 04_finalfiles/daner_${NAME}_Info0.8_MAF0.01_newSNPID_aligned
04_finalfiles/daner_final_aligned_${NAME}
cp 04_finalfiles/daner_final_aligned_${NAME}  04_finalfiles_gz
gzip 04_finalfiles_gz/daner_final_aligned_${NAME}

done
```

```
# Load necessary library
library(data.table)

# Set up directory structure
dir.create("01_filtered", showWarnings = FALSE)
dir.create("02_newSNPIDs", showWarnings = FALSE)
dir.create("03_alignedtoHRC", showWarnings = FALSE)
dir.create("04_finalfiles", showWarnings = FALSE)
dir.create("04_finalfiles_gz", showWarnings = FALSE)

# Read in list of files
files <- readLines("files")

# Load HRC SNP reference file
hrc_ref <- fread("HRCSNPlist_chr1-22_reference_frq_BP_19052020")

# function for alining, merging etc.:

for (file_name in files) {   # Loop through each file in `files`

  daner_data <- fread(paste0("00_daner/daner_", file_name)) # Load the daner-format summary file

  header <- colnames(daner_data) # Extract header

  write.table(header, paste0("header_", file_name), row.names = FALSE, col.names = FALSE, quote =
FALSE)

  filtered_data <- daner_data[INFO > 0.8 & INFO < 1.2 & MAF > 0.01 & MAF < 0.99] # Filter based on
MAF and INFO thresholds

  filtered_data[, SNP_ID := paste(CHR, BP, A1, A2, sep = "_")] # Create SNP IDs in the form
CHR_BP_A1_A2 and switched/strand-flipped versions

  filtered_data[, `:=`(

    SNP_ID_A2_A1 = ifelse(A1 == "A" & A2 == "G", paste(CHR, BP, A2, A1, sep = "_"),

                  ifelse(A1 == "G" & A2 == "A", paste(CHR, BP, A2, A1, sep = "_"),

                  ifelse(A1 == "C" & A2 == "T", paste(CHR, BP, A2, A1, sep = "_"), NA))),

    SNP_ID_flipped_A1_A2 = ifelse(A1 == "T" & A2 == "C", paste(CHR, BP, "A", "G", sep = "_"), NA),

    SNP_ID_flipped_A2_A1 = ifelse(A1 == "G" & A2 == "T", paste(CHR, BP, "C", "A", sep = "_"), NA)

  )]
```

```
  fwrite(filtered_data, paste0("01_filtered/daner_", file_name, "_Info0.8_MAF0.01"), sep = "\t",
row.names = FALSE)  # Save filtered file

  # Merge with HRC reference using different SNP IDs

  merged_1 <- merge(hrc_ref, filtered_data, by.x = "CHR_BP_A1_A2", by.y = "SNP_ID", all = FALSE)

  merged_2 <- merge(hrc_ref, filtered_data, by.x = "CHR_BP_A1_A2", by.y = "SNP_ID_A2_A1", all =
FALSE)

  merged_3 <- merge(hrc_ref, filtered_data, by.x = "CHR_BP_A1_A2", by.y =
"SNP_ID_flipped_A1_A2", all = FALSE)

  merged_4 <- merge(hrc_ref, filtered_data, by.x = "CHR_BP_A1_A2", by.y =
"SNP_ID_flipped_A2_A1", all = FALSE)

  # Combine all merged results

  combined_merged <- rbindlist(list(merged_1, merged_2, merged_3, merged_4), fill = TRUE)

  fwrite(combined_merged, paste0("03_alignedtoHRC/daner_", file_name,
"_Info0.8_MAF0.01_newSNPID_merged"), sep = "\t", row.names = FALSE)

  # Handle strand flipping and align to HRC reference

  aligned_data <- combined_merged[,

    .(CHR, BP, A1, A2, MAF, SNP_ID = CHR_BP_A1_A2,

      rs_number = i.SNP, info, other_columns)]

  # Final processing and write output

  aligned_data_with_header <- rbind(header, aligned_data)

  final_file <- paste0("04_finalfiles/daner_", file_name, "_Info0.8_MAF0.01_newSNPID_aligned")

  fwrite(aligned_data_with_header, final_file, sep = "\t", row.names = FALSE)

  # Compress the final file

  gzip_file <- paste0("04_finalfiles_gz/daner_final_aligned_", file_name, ".gz")

  write.table(aligned_data_with_header, file = final_file, sep = "\t", row.names = FALSE, col.names =
FALSE, quote = FALSE)

  system(paste("gzip", final_file))

}
```

2. SNP-based fine-mapping GCTA-COJO
3. Multi-trait analysis of ascertainment subgroups (MTAG)
4. GenomicSEM

4.1. Models without individual SNP effects

The analysis strictly follows the GenomicSEM tutorial:
https://github.com/GenomicSEM/GenomicSEM/wiki/3.-Genome%E2%80%90wide-Models
You will need to download the reference files “eur_w_ld_chr/" (can be found under the link above).
Change all filenames etc. according to your analysis.
Resulting Supplementary Figure 39 was created ‘by hand’ in LibreOffice Draw.

```
#!/usr/bin/env Rscript

require(GenomicSEM)

require(Matrix)

require(stats)

#################
## Step 1: Munge the summary statistics
#################

# change filenames, traitnames, and N accordingly.

# In our case, we have a column “N” in the summary statistics file that lists the effective N per SNP. If that is the case, N can be “NA” in the munge function.

munge(files=c("daner_OCD_fullv6_BB_120522_BETA_LDSCNeffdouble_noBeta0","daner_OCD_fullv6_clinical_120522_BETA_LDSCNeffdouble_noBeta0","daner_OCD_fullv6_otherascertained_120522_BETA_LDSCNeffdouble_noBeta0"),hm3="w_hm3.noMHC.snplist",trait.names=c("OCD_BB_170522_noBeta0","OCD_clinical_170522_noBeta0","OCD_otherasc_170522_noBeta0"),N=c(NA,NA,NA),info.filter=0.9,maf.filter=0.01)

#################
## Step 2: Run multivariable LDSC
#################

traits <- c("OCD_otherasc_170522_noBeta0.sumstats.gz","OCD_clinical_170522_noBeta0.sumstats.gz","OCD_23andMe_170422.sumstats.gz","OCD_BB_170522_noBeta0.sumstats.gz") # output of step 1

sample.prev <- c(0.5,0.5,0.5,0.5) #e.g. (.28,.36,.4) #if Neff was given in step 1, write 0.05 here.
```

```
population.prev <- c(0.02,0.02,0.02) # we assume the same population prevalence for each subgroup here.

ld <- "eur_w_ld_chr/"

wld <- "eur_w_ld_chr/"

trait.names= c("OCD_otherasc_170522_noBeta0","OCD_clinical_170522_noBeta0","OCD_23andMe","OCD_BB_170522_noBeta0")

#apply:

LDSCoutput <- ldsc(traits=traits,sample.prev=sample.prev,population.prev=population.prev,ld=ld,wld=wld,trait.names=trait.names)

####save as an R-object for later use:

save(LDSCoutput, file="LDSCoutput_OCDsubgroups_otherascertained_clinical_23andMe_BB_2_141122")

#################
## Step 3: Run exploratory factor analysis (EFA)
#################

covmat<-LDSCoutput$S
covmat <- round(covmat,2)
#smooth the S matrix for EFA using the nearPD function in the Matrix package.
Ssmooth<-as.matrix((nearPD(covmat, corr = FALSE))$mat)

# Run EFA:
EFA <- factanal (covmat=Ssmooth , factors=1, rotation="promax", control = list(trace =T))
EFA
EFA$loadings

#################
## Step 4: Run confirmatory factor analysis (CFA)
## using the pre-packaged CFA model
#################
CommonFactor_DWLS<- commonfactor(covstruc = LDSCoutput, estimation="DWLS")

CommonFactor_DWLS

CommonFactor_DWLS$loadings
```

4.2. Multivariate GWAS of the common factor

The analysis strictly follows the GenomicSEM tutorial:
https://github.com/GenomicSEM/GenomicSEM/wiki/4.-Common-Factor-GWAS

```
#!/usr/bin/env Rscript

require(GenomicSEM)

require(Matrix)

require(stats)

#################
# Step 1 and 2 are the same as above. You can either use the saved R-object from the GenomicSEM model without SNP effects or run step 1 and 2 now.
#################

load("LDSCoutput_OCDsubgroups_otherascertained_clinical_23andMe_BB_2_141122", envir = parent.frame())

#################
## Step 3: Prepare the summary statistics for GWAS
#################

# The sumstats function takes care of a few things prior to running multivariate GWAS. In order to run a multivariate GWAS, the SNPs, and corresponding SNP effects, need to be coded across phenotypes so that the same allele is the reference allele in all cases. In all cases, coefficients and their SEs are then further transformed such that they are scaled relative to unit-variance scaled phenotypes.

# Please also note that the sumstats function requires that the variables be listed in the same order as they are listed for the ldsc function above.

files<-c("daner_OCD_fullv6_otherascertained_120522_BETA_LDSCNeffdouble_noBeta0","daner_OCD_fullv6_clinical_120522_BETA_LDSCNeffdouble_noBeta0","daner_OCD_full_only23andMe_190522_Neffhalfdouble", "daner_OCD_fullv6_BB_120522_BETA_LDSCNeffdouble_noBeta0")

ref= "reference.1000G.maf.0.005.txt"

trait.names<-c("OCD_otherasc_170522_noBeta0","OCD_clinical_170522_noBeta0","OCD_23andMe","OCD_BB_170522_noBeta0")

se.logit=c(T,T,T,T)

info.filter=.6

maf.filter=0.01

OLS=NULL

N=NULL
```

```
PSYCH_sumstats
<-sumstats(files=files,ref=ref,trait.names=trait.names,se.logit=se.logit,OLS=NULL,N=NULL,info.filter=in
fo.filter,maf.filter=maf.filter,keep.indel=FALSE,parallel=FALSE,cores=NULL)

save(PSYCH_sumstats,
file="PSYCH_sumstats_output_OCDsubgroups_otherascertained_clinical_23andMe_BB_241122")

#################
## Step 4: Combine the summary statistics and LDSC output and run the common factor GWAS
#################

#specify the model

model<-"F1 =~ NA*OCD_otherasc_170522_noBeta0 + OCD_clinical_170522_noBeta0 +
OCD_23andMe + OCD_BB_170522_noBeta0

F1 ~ SNP

F1~~1*F1" # fix the variance of the factor to 1

onefactor <- userGWAS(covstruc = LDSCoutput, SNPs=PSYCH_sumstats, estimation="DWLS",
model=model, printwarn=TRUE, sub="F1~SNP")

write.table(onefactor,
file="CommonfactorOCDmodel_comorbid_clinical_23andMe_BB_2811022_noBeta0_unitvarianceidenti
fication.txt", row.names=F, sep=" ", quote=FALSE)
```

5. LDSC: SNP heritability & genetic correlation

Heritability and genetic correlation analyses follow the standard procedure as outlined here: https://github.com/bulik/ldsc/wiki/Heritability-and-Genetic-Correlation

R-script to create figure 3:
The input file should contain the following columns, with p1 being “OCD”, p2 all other phenotypes.

p1	p2	Category	rg	se	p

```
rm(list=ls())
library(ggplot2)
library(ggpubr)
library(RColorBrewer)

# read in data:
data <- read.table("OCD_full_correlations_260822.csv", header=TRUE, sep="\t") # change
```

```
file name accordingly

# Correct p-value according to FDR:
data$P_FDR <- p.adjust(c(data$p),method="fdr") # fdr correct dataset
data$FDR_sig = ifelse(data$P_FDR<0.05,"yes","no")  # write yes in new column if significant, no if not

# subset data: (data1 will be the left side of the plot, data2 the right side)
data1 <- subset(data, subset = Category %in%
c("Psychiatric","Substance","Cognition/SES","Personality"))
data2 <- subset(data, subset= Category %in%
c("Psychological","Neurological","Autoimmune","Cardiovasc.","Antropomorphic/Diet","Fertility"
,"Other"))


# order each trait according to what order you want.
data1$order <- factor(data1$p2, levels=c("Risktaking-Auto
speeding","Agreeableness","Extraversion", "Risktaking","Openness","Risktaking-Nr. Sexual
Partners","Conscientiousness","Neurot. Fed-up","Neurot. Miserable","Neurot.
Irritability","Neurot. Loneliness","Neurot. Worry a. Embarras.","Neurot. Depressive
Cluster","Neurot. Moodswings","Neurot. Guilt","Neurot. Hurt","Neuroticism","Neurot. Suffer fr.
Nerves","Neurot. Nervous","Neurot. Worry", "Neurot. Worry Cluster","Neurot. Tense",
                                "Life satisfaction (finance)","VN Reasoning","Intelligence", "Adult
IQ","Educational Attainment", "Job Satisfaction","Household Income","Memory","Childhood
IQ","Reaction Time","Townsend Deprivation",
                                "Alcohol Dependence","Alcohol Consumption","Smoking
Cessation","Age Smoking Initiation", "Caffeine","Drinks per Week","Cannabis ever vs
never","Smoking Initiation","Cigarettes per Day","Opioid use disorder","Cannabis Use
Disorder","Nicotine Dependence",
                                "Bipolar Disorder", "ADHD", "Autism Spectrum Disorder","Serious
mental illness","Schizophrenia","Mania Symptoms","Psychosis Symptoms","Depressive
Symptoms","PTSD","Tourette Syndrome","Mood disturbance", "Anorexia Nervosa","Major
Depressive Disorder","Depressive Disorder","Anxiety"))


data2$order <- factor(data2$p2, levels=c("Age at first Birth","Age at Menopause","Age at
Menarche","Number of Children",
                                "LDL Cholesterol","Type2 Diabetes", "Total Cholesterol","Heart Rate
Variability","HDL Cholesterol","Triglycerides","Coronary Artery Disease","Myocardial
Infarction",
                                "Inflammatory Bowel Disease","Ulcerative Colitis","Crohns
Disease","Lupus","Reumathoid Arthritis","Atopic Dermatitis","Asthma Childhood
onset","Multiple Sclerosis","Asthma Adult onset",
                                "Self-rated Health","Sleep Duration","Mothers age at
death","Parents age at death","Chronotype Morningness","Fathers age at
death","Tiredness","Childhood Maltreatment",
                                "Diet
(Fat)","BMI","Bodyfat","Hip-Circumference","Waist-Circumference","Height","Diet
(Protein)","Waist-Hip Ratio","Birth Weight","Diet (Sugar)","Diet (Carbohydrate)",
                                "Epilepsy Generalized", "Parkinsons Disease","Alzheimers
```

```
Disease","ALS","Migraine","Epilepsy Focal",
                                "Subjective Wellbeing","Family relationship sat.","Friend relationship
sat.", "Freq of friend/family visits","Loneliness","Suicide attempt"))

# order each category according to what order you want.
data1$Category_order <-factor(data1$Category,
levels=c("Psychiatric","Substance","Cognition/SES","Personality"))
data2$Category_order <-factor(data2$Category,
levels=c("Psychological","Neurological","Autoimmune","Cardiovasc.","Antropomorphic/Diet","F
ertility","Other"))
gap=0.04

# make confidence intervals out of the SEs
data1$CI <- data1$se * 1.96
data2$CI <- data2$se * 1.96
data$CI <- data$se*1.96

# Save the data with adjiusted P-values and CIs
#write.table(data, "OCD_full_LDSC_results_260822_CI_FDR.csv", quote = FALSE,
col.names = TRUE, row.names = FALSE, sep=",")


# Plot

p1 <- ggplot( data1, aes(y=rg,x=order)) + # add fill=FDR_sig in case I want to change the
colour of the dots according to significance
  geom_errorbar(aes(ymin = rg-CI, ymax = rg+CI),
                width =  0.6, size=1, color="darkgray") +   # width of the error bars whiskers + colour
  geom_point(aes(colour=FDR_sig), size=4) +
  scale_color_manual(values=c("no"="black","yes"="#BD4545"),
name="",breaks=c("yes"),labels=c("Significant after FDR correction"))+
  geom_hline(yintercept=0, color="darkgray", linetype="dashed") +
  ylim(-2,2) + # adapt here so that the error bars are within this range.
    facet_grid(Category_order ~ .,scales="free", space="free") + #indicates that Category is
used for "grouping" in "separate" plots
  theme(strip.text.x= order)    # this line and the above ensures that the y-axis ticks only get
shown once (no overlay of all ticks)


p1   + coord_flip() +  theme_light(base_size = 15) + # theme_light(base_size = 9) adds nice
colour scheme,
   theme(legend.position = "top", axis.title.y=element_blank())
#theme(axis.title.y=element_blank()) deletes y axis labels, scale_fill_manual changes the
legend.

p2 <- ggplot( data2, aes(y=rg,x=order)) +  # add fill=FDR_sig in case I want to change the
colour of the dots according to significance
  geom_errorbar(aes(ymin = rg-CI, ymax = rg+CI),
                width =  0.6, size=1, color="darkgray") +   # width of the error bars whiskers + colour
  geom_point(aes(colour=FDR_sig), size=4) +
```

```
  scale_color_manual(values=c("no"="black","yes"="#BD4545"),
name="",breaks=c("yes"),labels=c("Significant after FDR correction"))+
  geom_hline(yintercept=0, color="darkgray", linetype="dashed") +
  ylim(-1.1,1.1) +
  facet_grid(Category_order ~ .,scales="free", space="free") + #indicates that Category is
used for "grouping" in "separate" plots
  theme(strip.text.x=order)    # this line and the above ensures that the y-axis ticks only get
shown once (no overlay of all ticks)

p2  + coord_flip() +  theme_light(base_size = 15) +  theme(legend.position = "top",
axis.title.y=element_blank()) #theme(axis.title.y=element_blank()) deletes y axis labels,
scale_fill_manual changes the legend.

sign(1)

arrangedplot <- ggarrange(p1 + coord_flip(ylim=c(-1,1)) + theme_light(base_size = 15) +
theme(axis.title.y=element_blank(),legend.position = "none") ,
                      p2 + coord_flip(ylim=c(-1,1)) + theme_light(base_size = 15) +
theme(axis.title.y=element_blank(), legend.position = "top") ,
                      ncol = 2, nrow = 1)
arrangedplot

# save
jpeg("rg_OCD_full_131223.jpg", units="mm", res=1500, width=430.2, height=300)
arrangedplot
dev.off()
```

6. Gene-based analyses

6.1. MBAT-Combo

```
#!/bin/bash
#PBS -N mbat-combo
#PBS -l mem=30GB
#PBS -l walltime=48:00:00

./gcta-1.94.1-linux-kernel-3-x86_64/gcta-1.94.1 --bfile 1000G.EUR.QC.MERGED
--mBAT-combo ocd_full.ma --mBAT-gene-list glist_ensgid_hg19_v40.txt --mBAT-print-all-p --out
OCD_mbat_combo_all_p.txt --thread-num 10
```

6.2. Transcriptome-Wide Association Study (TWAS)

```
#!/bin/bash
#SBATCH --job-name=ocd_pec_twas
#SBATCH --nodes=1
#SBATCH --mem=5gb
#SBATCH --time=2:00:00
#SBATCH --array=1-22
#SBATCH --output=ocd_array-%A_%a.out

module load R/4.3.2

Rscript FUSION.assoc_test.R --sumstats ocd.sumstats.gz --weights
./weights/PEC.BRAIN.RNASEQ.SUBSET_N.pos --weights_dir ./weights/ --ref_ld_chr
./ld/1000G_EUR_Phase3_plink/1000G.EUR.QC. --chr $SLURM_ARRAY_TASK_ID --out
./results/twas/ocd_coloc_chr$SLURM_ARRAY_TASK_ID.dat  --coloc_P 0.999999 --GWASN
103135 --PANELN weight_n_panel.txt
```

6.3. Summary-Based Mendelian Randomisation (SMR)

```
#!/bin/bash
#SBATCH --job-name=smr_ocd_brainmeta
#SBATCH --nodes=1
#SBATCH --mem=10gb
#SBATCH --time=02:00:00
#SBATCH --array=1-22
#SBATCH --output=smr_ocd_brainmeta-%A_%a.out

./smr --bfile ./twas/ld/LDREF/1000G.EUR.$SLURM_ARRAY_TASK_ID --gwas-summary
./gwas/ocd/original_sumstats/ocd_PGC_2023.ma --beqtl-summary
./smr/BrainMeta_cis_eqtl_summary/BrainMeta_cis_eQTL_chr$SLURM_ARRAY_TASK_ID --out
./results/smr/ocd_brainmeta_chr$SLURM_ARRAY_TASK_ID --thread-num 10

#!/bin/bash
#SBATCH --job-name=smr_ocd_eqtlgen
#SBATCH --nodes=1
#SBATCH --mem=20gb
#SBATCH --time=05:00:00
#SBATCH --output=smr_ocd_eqtlgen.log

./smr --bfile ./smr/ld/1000G.EUR.merged --gwas-summary
./gwas/ocd/original_sumstats/ocd_PGC_2023.ma --beqtl-summary
./smr/cis-eQTLs-full_eQTLGen_AF_incl_nr_formatted_20191212.new.txt_besd-dense --out
~/projects/ocd/results/smr/ocd_original_eqtlgen --thread-num 10
```

6.4. Psychiatric Omnilocus Prioritization Score

```
python PsyOPS.py --GWAS-hit-file ./processed/psyops_ocd.txt --output-file
./results/psyops/psyops_ocd.tsv
```

6.5. Protein-wide association study (PWAS)

```
#PBS -N fusion_rosmap_pwas
#PBS -l mem=5GB
#PBS -l walltime=10:00:00

module load R/3.5.1

for i in {1..22}
do
  Rscript FUSION.assoc_test.R \
  --sumstats ocd.sumstats.gz \
  --weights ./PWAS/ROSMAP.n376.fusion.WEIGHTS/train_weights_n.pos \
  --weights_dir ./PWAS/ROSMAP.n376.fusion.WEIGHTS/ \
  --ref_ld_chr ./LDREF/1000G_EUR_Phase3_plink/1000G.EUR.QC. \
  --chr $i \
  --out ./pwas_ocd_chr$i.dat
done

#PBS -N fusion_banner_pwas
#PBS -l mem=5GB
#PBS -l walltime=10:00:00

module load R/3.5.1

for i in {1..22}
do
  Rscript FUSION.assoc_test.R \
  --sumstats ocd.sumstats.gz \
  --weights ./PWAS/Banner.n152.fusion.WEIGHTS/train_weights_n.pos \
  --weights_dir ./PWAS/Banner.n152.fusion.WEIGHTS/ \
  --ref_ld_chr ./LDREF/1000G_EUR_Phase3_plink/1000G.EUR.QC. \
  --chr $i \
  --out ./pwas_banner_ocd_chr$i.dat
done
```

7. Tissue and cell-type enrichment analysis

R script for tissue and broad celltype analysis using both LDSC and MAGMA, makes Figure 2B

```
## LIBRARY
library(ggplot2)

## PARAM
PALETTE=c("Both"="#E69F00","MAGMA"="#0072B2",
        "LDSC"="#56B4E9","Neither"="#CC79A7")
MAGMA_COLS_KEEP=c("CATEGORY","NGENES","BETA","BETA_STD","SE","P")
LDSC_COLS_KEEP=c("CATEGORY","PROP_SNPS","PROP_H2","PROP_H2_STD_ERROR",
               "ENRICHMENT","ENRICHMENT_STD_ERROR","ENRICHMENT_P",
               "COEFFICIENT","COEFFICIENT_STD_ERROR","COEFFICIENT_Z_SCORE",
               "P")

Main <- function() {
  ARGS <- commandArgs(trailingOnly=T)
  if (length(ARGS) != 6) {
    Usage()
  }

  # get ARGS
  dataset.analysis.MAGMA.csv<-ARGS[1]
  dataset.analysis.LDSC.csv<-ARGS[2]
  dataset_name<-ARGS[3]
  top_n_results<-as.numeric(ARGS[4])
  out.csv<-ARGS[5]
  out.pdf<-ARGS[6]

  # read input datasets
  magma <- read.csv(dataset.analysis.MAGMA.csv, stringsAsFactors=F)
  ldsc <- read.csv(dataset.analysis.LDSC.csv, stringsAsFactors=F)

  # subset on key columns, relabel col names to avoid redundancy in merging
  magma <- magma[,MAGMA_COLS_KEEP]
  for (i in 2:ncol(magma)) {
    col_i=colnames(magma)[i]
    colnames(magma)[i]=paste0("MAGMA_",col_i)
  }
  ldsc <- ldsc[,LDSC_COLS_KEEP]
  for (i in 2:ncol(ldsc)) {
    col_i=colnames(ldsc)[i]
    colnames(ldsc)[i]=paste0("LDSC_",col_i)
  }

  # merge magma and ldsc tables
  magmaldsc <- merge(magma, ldsc, by="CATEGORY")

  # some datasets (mainly GTEx) have wierd formatting issues in CATEGORY,
  # run some commands to fix it
  magmaldsc$CATEGORY <- gsub("\\.\\.\\.", " - ", magmaldsc$CATEGORY)
  magmaldsc$CATEGORY <- gsub("\\.\\.", " (", magmaldsc$CATEGORY)
```

```
magmaldsc$CATEGORY <- gsub("\\.$", ")", magmaldsc$CATEGORY)
magmaldsc$CATEGORY <- gsub("\\.", " ", magmaldsc$CATEGORY)

# add columns for FDR-adjusted p-values
magmaldsc$MAGMA_P_FDR <- p.adjust(magmaldsc$MAGMA_P, method='fdr')
magmaldsc$LDSC_P_FDR   <- p.adjust(magmaldsc$LDSC_P, method='fdr')

# get -log10 p-values
magmaldsc$MAGMA_LOG10P=-log10(magmaldsc$MAGMA_P)
magmaldsc$LDSC_LOG10P=-log10(magmaldsc$LDSC_P)

# get mean -log10 p-values btwn magma and ldsc, sort by this value
magmaldsc$MEAN_LOG10P=rowMeans(magmaldsc[,c("MAGMA_LOG10P","LDSC_LOG10P")])
magmaldsc<-magmaldsc[rev(order(magmaldsc$MEAN_LOG10P)),]

cat("Criteria for significance : FDR-corrected p < 0.05\n")

# how many results have p < thresh in both magma and ldsc?
magmaldsc.11 <- subset(magmaldsc,
                    (MAGMA_P_FDR < 0.05)
                    &
                    (LDSC_P_FDR < 0.05)
                   )
cat("N categories significant in both MAGMA and LDSC :",
   nrow(magmaldsc.11),"\n")
magmaldsc.11.categ <- magmaldsc.11$CATEGORY

# how many results have p < thresh in magma but not ldsc?
magmaldsc.10 <- subset(magmaldsc,
                    (MAGMA_P_FDR < 0.05)
                    &
                    (LDSC_P_FDR >= 0.05)
                    )
cat("N categories significant in MAGMA but not in LDSC :",
   nrow(magmaldsc.10),"\n")
magmaldsc.10.categ <- magmaldsc.10$CATEGORY

# how many results have p < thresh in ldsc but not magma?
magmaldsc.01 <- subset(magmaldsc,
                    (MAGMA_P_FDR >= 0.05)
                    &
                    (LDSC_P_FDR < 0.05)
                   )
cat("N categories significant in LDSC but not in MAGMA :",
   nrow(magmaldsc.01),"\n")
magmaldsc.01.categ <- magmaldsc.01$CATEGORY

# how many results have p < thresh in neither ldsc or magma?
magmaldsc.00 <- subset(magmaldsc,
                         (MAGMA_P_FDR >= 0.05)
                    &
                    (LDSC_P_FDR >= 0.05)
                    )
cat("N categories significant in neither MAGMA or LDSC :",
   nrow(magmaldsc.00),"\n")
```

```
  magmaldsc.00.categ <- magmaldsc.00$CATEGORY

  # store classif for significance for each category in full table
  magmaldsc$Significant=rep("Neither",nrow(magmaldsc))
  magmaldsc$Significant=ifelse(magmaldsc$CATEGORY %in% magmaldsc.11.categ,
                          "Both", magmaldsc$Significant)
  magmaldsc$Significant=ifelse(magmaldsc$CATEGORY %in% magmaldsc.10.categ,
                          "MAGMA", magmaldsc$Significant)
  magmaldsc$Significant=ifelse(magmaldsc$CATEGORY %in% magmaldsc.01.categ,
                          "LDSC", magmaldsc$Significant)

  # write full result table to output csv
  write.csv(magmaldsc,
            file=out.csv,
            row.names=F, quote=F)

  # factorize significant column
  magmaldsc$Significant <- factor(magmaldsc$Significant,
                             levels=c("Both","MAGMA","LDSC","Neither"))

  # take the top X rows before plotting
  rownames(magmaldsc) <- 1:nrow(magmaldsc)
  magmaldsc <- magmaldsc[1:top_n_results, , drop=F]

  # make sure order of categories stays constant
  magmaldsc$CATEGORY <- factor(magmaldsc$CATEGORY,
                          levels=rev(magmaldsc$CATEGORY))

  # make a ggplot with the top X results based on mean(log10p)
  p<-ggplot(data=magmaldsc,
            aes(x=CATEGORY, y=MEAN_LOG10P, fill=Significant)) +
     geom_bar(stat="identity") +
     xlab("Category") +
     ylab("Mean(-log10p)")

  # change to colorblind palette
  p<-p+scale_fill_manual(values=PALETTE)

  # add a line for log10 pvalue threshold
  # p<-p+geom_hline(yintercept=bonf_thresh_dataset_log10)

  # flip x/y coordinates
  p<-p+coord_flip()

  # save the object
  ggsave(out.pdf, plot=p)

}

Usage <- function(){
  cat("scRNA_MAGMA_LDSC_dataset_analysis.R",
      "<dataset.analysis.MAGMA.csv> <dataset.analysis.LDSC.csv>",
      "<dataset_name> <top_n_results>",
      "<out.csv> <out.pdf>\n")
  q()
```

```
}

if (interactive() == F) {
  Main()
}
```

R script for specific celltype and celltype group analysis (produces raw Figure 2C):

```
## LIBRARY
library(ggplot2)

## PARAM
PALETTE=c(
        "Yes"="#0072B2",
        "No"="#CC79A7"
          )
MAGMA_COLS_KEEP=c("CATEGORY","NGENES","BETA","BETA_STD","SE","P")
DATASET_NAME="Zeisel et al. 2018 (all 265 celltypes from mouse nervous system"

Main <- function() {
  ARGS <- commandArgs(trailingOnly=T)
  if (length(ARGS) != 6) {
    Usage()
  }

  # get ARGS
  dataset.analysis.MAGMA.csv<-ARGS[1]
  out.csv<-ARGS[2]
  out.pdf<-ARGS[3]
  zeisel.celltypes_info.tsv<-ARGS[4]
  out.clustered.csv<-ARGS[5]
  out.clustered.pdf<-ARGS[6]

  # read input datasets
  magma <- read.csv(dataset.analysis.MAGMA.csv,
                    stringsAsFactors=F)

  # subset on key columns, relabel col names to avoid redundancy in merging
  magma <- magma[,MAGMA_COLS_KEEP]

  # some datasets (mainly GTEx) have wierd formatting issues in CATEGORY,
  # run some commands to fix it
  magma$CATEGORY <- gsub("\\.\\.\\.", " - ", magma$CATEGORY)
  magma$CATEGORY <- gsub("\\.\\.", " (", magma$CATEGORY)
  magma$CATEGORY <- gsub("\\.$", ")", magma$CATEGORY)
  magma$CATEGORY <- gsub("\\.", " ", magma$CATEGORY)

  # get bonferroni threshold
  bonf_thresh = 0.05 / nrow(magma)
  cat("Bonferrroni threshold = 0.05 /",nrow(magma),"=",bonf_thresh,"\n")

  # get log10p
  magma$LOG10P <- -log10(magma$P)
```

```
# order by log10p
magma <- magma[rev(order(magma$LOG10P)), ]

# derive fdr_p
magma$FDR_P <- p.adjust(magma$P, method='BH')

# how many results have p < thresh?
magma.1 <- subset(magma, FDR_P < 0.05)
cat("N categories significant  (5% FDR) :",
   nrow(magma.1),"\n")

# how many results have p>= thresh?
magma.0 <- subset(magma, FDR_P >= 0.05)
cat("N categories not significant (5% FDR):",
   nrow(magma.0), "\n")

# how many results have p<thresh (bonferroni)?
magma.1 <- subset(magma, P < bonf_thresh)
cat("N categories significant  (bonferroni) :",
   nrow(magma.1),"\n")

# how many results have p>= thresh?
magma.0 <- subset(magma, P >= bonf_thresh)
cat("N categories not significant (bonferroni):",
   nrow(magma.0), "\n")

# store classif for significance for each category in full table
magma$Significant=ifelse(magma$FDR_P < 0.05, "Yes", "No")

# write full result table to output csv
write.csv(magma,
        file=out.csv,
        row.names=F, quote=F)

# factorize significant column
magma$Significant <- factor(magma$Significant,
                        levels=c("Yes","No")
                    )

# take the top X rows before plotting
rownames(magma) <- 1:nrow(magma)
# magma <- magma[1:top_n_results, , drop=F]

# make sure order of categories stays constant
magma$CATEGORY <- factor(magma$CATEGORY,
                     levels=rev(magma$CATEGORY))

# make a ggplot with the top X results based on mean(log10p)
p<-ggplot(data=magma,
        aes(x=CATEGORY, y=LOG10P, fill=Significant)) +
  geom_bar(stat="identity") +
  xlab(DATASET_NAME) +
  ylab("-log10p")
```

```
# change font size, angle
p<-p+theme(axis.text=element_text(size=3))

# change to colorblind palette
p<-p+scale_fill_manual(values=PALETTE)

# add a line for log10 pvalue threshold
p<-p+geom_hline(yintercept=-log10(bonf_thresh))

# flip x/y coordinates
p<-p+coord_flip()

# save the object
ggsave(out.pdf, plot=p,
       units="in", width=8.5, height=11)

# read zeisel extended data table, do formatting on zeisel celltypes
zeisel.celltypes_info <- read.table(zeisel.celltypes_info.tsv,
                                    header=T, sep="\t",
                                   stringsAsFactors=F)
zeisel.celltypes_info <- zeisel.celltypes_info[,c("Symbol","Description","Region","Likely.location")]
zeisel.celltypes_info$Symbol <- gsub(" ","", zeisel.celltypes_info$Symbol)
colnames(zeisel.celltypes_info) <- c("CATEGORY","Description","Region","Likely.location")

# merged with full dataframe, order by decreasing log10p
magma <-
magma[,c('CATEGORY','NGENES','BETA','BETA_STD','SE','P','LOG10P','FDR_P','Significant')]
magma <- merge(magma, zeisel.celltypes_info, by='CATEGORY')
magma <- magma[rev(order(magma$LOG10P)), ]

# get counts per Description
magma$Description_n <- NA
magma$n <- rep(1, nrow(magma))
d.counts <- table(magma$Description)
for (d.i in names(d.counts)) {
  d.n <- d.counts[[d.i]]
}
magma$n <- rep(1, nrow(magma))
magma$Description_n <- rep(NA, nrow(magma))
for (i in rownames(magma)) {
       d.i <- magma[i, "Description"]
       d.n <- d.counts[[d.i]]
  magma[i, "n"] <- d.n
  magma[i,"Description_n"] <- paste0(d.i, "(n=",d.n, ")")
}

# get subset of table where n(Description) > 1
magma.n1 <- subset(magma, n>1)
d.rep <- unique(sort(magma.n1$Description))

# get mean -log10p per Description
out.magma <- data.frame(Description=character(),
                        log10p_mean=numeric(),
                        p=numeric()
                        )
```

```
  for (d.x in d.rep) {
    magma.x <- subset(magma, Description==d.x)
    log10p_mean=mean(magma.x$LOG10P)
            p_mean <- 10^(-(log10p_mean))
            out.magma <- rbind(out.magma,
                    data.frame(Description=d.x,
                        log10p_mean=log10p_mean,
                            p=p_mean
                            )
                    )
  }

  # apply fdr adjustment
  out.magma$fdr_p <- p.adjust(out.magma$p, method='fdr')
  out.magma$FDR_SIGNIF <- ifelse(out.magma$fdr_p < 0.05, "Yes", "No")
  out.magma.signif <- subset(out.magma, fdr_p < 0.05)

  # get significant Descr
  d.signif <- unique(out.magma.signif$Description)
  out.magma$SignificantGroup <- ifelse(out.magma$Description %in% d.signif, "Yes", "No")

  # print stats to stdout
  cat("N 'Description' classifications with at least 2 individual celltypes included :",
      length(d.rep), "\n")
  cat("N classifications that are significant (FDR-adjusted for ", length(d.rep),
      "classifications tested, p<0.05) :", length(d.signif), "\n")
  cat("Significant classifications :\n")
  for (d.i in d.signif) {cat(" ", d.i, "\n")}

  # sort out.magma by log10p_mean
  out.magma <- out.magma[order(out.magma$log10p_mean), ]

  # get subset of original table where Description has n > 1
  magma.rep <- subset(magma, Description %in% d.rep)
  magma.rep <- merge(magma.rep, out.magma[,c("Description", "log10p_mean", "fdr_p")],
by='Description')
  magma.rep <- magma.rep[rev(order(magma.rep$fdr_p)), ]

  # get significant Descr
  d.signif <- unique(out.magma.signif$Description)
  magma.rep$SignificantGroup <- ifelse(magma.rep$Description %in% d.signif, "Yes", "No")

  # write df for clustered results to file before plotting
  write.csv(magma.rep, file=out.clustered.csv,
            row.names=F, quote=T)

  # plot results
  magma.rep$SignificantGroup <- ifelse(magma.rep$Description %in% out.magma.signif$Description,
"Yes", "No")
  magma.rep$SignificantGroup <- factor(magma.rep$SignificantGroup, levels=c("Yes","No"))
  magma.rep$Significant <- factor(magma.rep$Significant, levels=c("Yes","No"))

  # order dataframe by log10p_mean, factorize Description
  magma.rep$Description=factor(magma.rep$Description)
  magma.rep <- magma.rep[order(magma.rep$log10p_mean), ]
```

```
  magma.rep$Description_n=factor(magma.rep$Description_n,
levels=unique(magma.rep$Description_n))

  # plot the clustered results
  gg <- ggplot(aes(y=Description_n, x=LOG10P, group=Description_n), data=magma.rep) +
       geom_jitter(aes(color=Significant), shape=16, position=position_jitter(0.01,0.01))
  gg <- gg + stat_summary(aes(color=SignificantGroup),fun=mean, geom="point", shape=73,size=5)
  gg <- gg + labs(x='-log10(p)')
  gg <- gg + theme(axis.title.y=element_blank(),
                        legend.position=c(0.75, 0.2))

  # change to colorblind palette
  gg<-gg+scale_color_manual(values=PALETTE)

  ggsave(gg, file=out.clustered.pdf)

}

Usage <- function(){
  cat("scRNA_MAGMA_Zeisel_lvl5_analysis.R",
      "<dataset.analysis.MAGMA.csv>",
      "<out.csv> <out.pdf> <zeisel.celltypes_info.tsv>",
      "<out.clustered.csv> <out.clustered.pdf>\n")
  q()
}

if (interactive() == F) {
  Main()
}
```

8. Author list

This R-script generates an author list from an excel table.

```
#####
# This script creates an authorlist with numbered affiliations from an excel sheet
# author: Nora Strom, date: Feb. 12th 2024
# Depending on the columns provided in the excel sheet either the columns or this script needs to be
adapted
# For the script relevant columns are named the following in the excel file. Adapt if your column names
are different.
# Name    Title   Given.Name      Middle.Initials Institution_1         Department_1   Division_1
City_1   State.Province_1          Country_1         Street.address_1 [...] Institution_5
Department_5   Division_5         City_5   State.Province_5          Country_5         Street.address_5
# It lists authors in the order they appear in the excel input sheet
#####

rm(list=ls())
library(openxlsx)
```

```
# Read the Excel file
df <- read.xlsx("authorlist.xlsx") #change to correct file name

# Initialize vectors to store formatted authors and their affiliations
formatted_authors <- character()
formatted_affiliations <- character()

# Function to format authors with multiple affiliations
format_author <- function(row) {
  # Format the name
  last_name <- row["Name"]
  #  first_name <- paste(substr(row["Given.Name"], 1, 1), ".", sep = "")
  first_name <- row["Given.Name"]
  middle_initial <- row["Middle.Initials"]
  if (!is.na(middle_initial)) {
    middle_initial <- paste(substr(middle_initial, 1, 1), ".", sep = "")
  } else {
    middle_initial <- ""
  }
  name <- paste(last_name, first_name, middle_initial, sep = ", ")
  # Remove the trailing comma if there is no middle initial
  if (middle_initial == "") {
    name <- sub(", $", "", name)
  }
  # Initialize vector to store affiliations
  affiliations <- character()
  # Loop through potential affiliations
  for (i in 1:5) {
    # Check if additional affiliation columns exist
    if (paste0("Institution_", i) %in% colnames(row)) {
      # Check if additional affiliation exists
      if (!is.na(row[paste0("Institution_", i)])) {
        # Get the affiliation number
        affiliation_components <- c(row[paste0("Department_", i)],
                                    ifelse(is.na(row[paste0("Division_", i)]), "", row[paste0("Division_", i)]),
                                    ifelse(is.na(row[paste0("Institution_", i)]), "", row[paste0("Institution_", i)]),
                                    ifelse(is.na(row[paste0("City_", i)]), "", row[paste0("City_", i)]),
                                    ifelse(is.na(row[paste0("State/Province_", i)]), "", row[paste0("State/Province_",
i)]),
                                    ifelse(is.na(row[paste0("Country_", i)]), "", row[paste0("Country_", i)]))
        # Remove any NA components and trailing commas
        affiliation_components <- affiliation_components[!is.na(affiliation_components) &
affiliation_components != ""]
        # Concatenate the affiliation components
        affiliation <- paste(affiliation_components, collapse = ", ")
        # Check if this affiliation has been seen before
        existing_index <- match(affiliation, formatted_affiliations)
        if (is.na(existing_index)) {
          # If not, add it to the list of affiliations
          formatted_affiliations <<- c(formatted_affiliations, affiliation)
          existing_index <- length(formatted_affiliations)
        }
        # Add affiliation number to vector
        affiliations <- c(affiliations, existing_index)
      }
```

```
    }
  }
  # Add the affiliations to the name
  if (length(affiliations) > 0) {
    #    name <- paste0(name, " ^", paste(affiliations, collapse = ", ^")) # with ^
    name <- paste0(name, " ", paste(affiliations, collapse = ", ")) # without ^
  }
  return(name)
}

# Apply the function to format authors
formatted_authors <- sapply(1:nrow(df), function(i) format_author(df[i, ]))

# version 1:
# Combine authors and affiliations (authors in one row each)
# output <- c(formatted_authors, paste(paste(1:length(formatted_affiliations)), ": ",
formatted_affiliations, sep = ""))
# Combine affiliations
#affiliation_output <- paste(paste(1:length(formatted_affiliations)), ": ", formatted_affiliations, sep = "")

# version 2:
# Create a vector with numbers for affiliations
affiliation_numbers <- paste(1:length(formatted_affiliations),":")

# Combine affiliations with numbers and semicolons as separators
affiliation_output <- paste(affiliation_numbers, formatted_affiliations, collapse = "; ")

# Combine authors and affiliations
output <- c(formatted_authors, affiliation_output)


# Write to a text file
writeLines(output, "authorlist_output.txt")
```