Supplementary Material - ANX GWAS

Methods: Analysis Code

1. Filtering and aligning of GWAS summary statistics

1.1. Filtering & aligning

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_HDio8lC_MeWoAg2jlggrtU/edit?tab=t.0#heading=h.4008addvumol)
- HRC-reference file which contains the following information (or an equivalent reference in the same format):
  SNP(rs-number) A1 A2 CHR_BP_A1_A2 MAF

This script is arguably a bit difficult to follow (and you probably have to change it 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";el
```

```
se 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";el
se 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";el
se 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";el
se 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 ambiguous 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
```

Equivalent R-script:

```
# 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))
}
```

1.2. DENTIST (Detecting Errors in aNalyses of summary staTISTics)

We applied DENTIST individually to summary statistics for each cohort using commands option similar to their tutorials. As reference panel we used the European part of the Haplotype Reference Consortium. Below we present the batch file we used on lisa (saralisa) server to run DENTIST for one of the data sets:

```
#!/bin/bash
#Set job requirements
#SBATCH -n 16
#SBATCH -t 24:00:00

#Loading modules
#module load python
while getopts i: flag
do
    case "${flag}" in
        i) index=${OPTARG};;
    esac
done
```

```
#Copy input file to scratch
cp $HOME/DENTIST/input/daner_final_aligned_bioVUv2_regenie_ANX "$TMPDIR"
cp
$HOME/HRC_reference.r1-1/HRC.r1-1.EGA.GRCh37.chr"$SLURM_ARRAY_TASK_ID".impu
te.plink.*  "$TMPDIR"

#Create output directory on scratch
mkdir "$TMPDIR"/output_dir

#Execute a Python program located in $HOME, that takes an input file and output directory as
arguments.

$HOME/DENTIST/DENTIST_1.2.0.0  --gwas-summary
"$TMPDIR"/daner_final_aligned_bioVUv2_regenie_ANX --bfile
"$TMPDIR"/HRC.r1-1.EGA.GRCh37.chr"$SLURM_ARRAY_TASK_ID".impute.plink --maf 0.05
--out "$TMPDIR"/output_dir/bioVUv2_regenie_ANXlast."$SLURM_ARRAY_TASK_ID"

#Copy output directory from scratch to home
cp -r "$TMPDIR"/output_dir/* $HOME/DENTIST/output/

### Then  run it using
# sbatch -a 1-22 lisa.batch.ind.files.44.txt
```

2. GenomicSEM & LDSC

2.1. Heritability & genetic correlations

We followed this tutorial: https://github.com/bulik/ldsc/wiki/Heritability-and-Genetic-Correlation
And used the effective sample size for heritability estimation (then specifying the sample prevalence as 0.5). See:
https://github.com/GenomicSEM/GenomicSEM/wiki/2.1-Calculating-Sum-of-Effective-Sample-Size-and-Preparing-GWAS-Summary-Statistics

2.2. paLDSC

We followed this tutorial: https://rpubs.com/JaFuente/paLDSC

2.3. 1-factor Model: EFA & CFA

We followed this tutorial:
https://github.com/GenomicSEM/GenomicSEM/wiki/3.-Genome%E2%80%90wide-Models

3. Polygenic Risk-Score (PRS) analyses

3.1. Using MegaPRS in GenoPred for UKBB

The software and tutorials for conducting PRS analyses using MegaPRS within GenoPred are available here: https://opain.github.io/GenoPred/pipeline_technical.html

4. Characterization and functional annotation of GWAS SNPs

4.1. Variant fine mapping with FINEMAP

We used FINEMAP with default values for everything (see https://hongchengyao.github.io/fine-mapping_document/FINEMAP/). The LD matrices for regions encompassing significant signals were accurately precomputed using GAUSS R package (https://statsleelab.github.io/gauss/). Below are the first 2 lines of the file containing options used in running FINEMAP for the first such region:

```
z;ld;snp;config;cred;log;n_samples
/home/bacanusa/ANX/finemap/data.1.z;/home/bacanusa/ANX/finemap/data.1.ld;/home/bacanusa/ANX/finemap/data.1.snp;/home/bacanusa/ANX/finemap/data.1.config;/home/bacanusa/ANX/finemap/data.1.cred;/home/bacanusa/ANX/finemap/data.1.log;369286
```

4.2. Functional annotation (eQTL/HiC) with FUMA

We used FUMA v1.6.1 for several of our analyses. As part of these analyses we used third party datasets provided via the FUMA framework that we would like to list below. For details on the analyses please either see the tutorials on the FUMA website (SNP2GENE module: https://fuma.ctglab.nl/tutorial#snp2gene) or see our material and methods section in the manuscript.)

eQTL datasets:

eQTLcatalogue/BrainSeq_ge_brain.txt.gz,

PsychENCODE/PsychENCODE_eQTLs.txt.gz,

eQTLGen/eQTLGen_cis_eQTLs.txt.gz,

eQTLGen/eQTLGen_trans_eQTLs.txt.gz,

CMC/CMC_SVA_cis.txt.gz, CMC/CMC_SVA_trans.txt.gz, CMC/CMC_NoSVA_cis.txt.gz, CMC/CMC_NoSVA_trans.txt.gz,

BRAINEAC/CRBL.txt.gz, BRAINEAC/FCTX.txt.gz, BRAINEAC/HIPP.txt.gz, BRAINEAC/MEDU.txt.gz, BRAINEAC/OCTX.txt.gz, BRAINEAC/PUTM.txt.gz, BRAINEAC/SNIG.txt.gz, BRAINEAC/TCTX.txt.gz, BRAINEAC/THAL.txt.gz, BRAINEAC/WHMT.txt.gz, BRAINEAC/aveALL.txt.gz,

GTEx/v8/Brain_Amygdala.txt.gz, GTEx/v8/Brain_Anterior_cingulate_cortex_BA24.txt.gz, GTEx/v8/Brain_Caudate_basal_ganglia.txt.gz, GTEx/v8/Brain_Cerebellar_Hemisphere.txt.gz, GTEx/v8/Brain_Cerebellum.txt.gz, GTEx/v8/Brain_Cortex.txt.gz, GTEx/v8/Brain_Frontal_Cortex_BA9.txt.gz, GTEx/v8/Brain_Hippocampus.txt.gz, GTEx/v8/Brain_Hypothalamus.txt.gz, GTEx/v8/Brain_Nucleus_accumbens_basal_ganglia.txt.gz, GTEx/v8/Brain_Putamen_basal_ganglia.txt.gz, GTEx/v8/Brain_Spinal_cord_cervical_c-1.txt.gz, GTEx/v8/Brain_Substantia_nigra.txt.gz

PsychENCODE/enhancer.bed.gz, PsychENCODE/enhancer_high_conf.bed.gz, PsychENCODE/PFC_H3K27ac_peak.bed.gz, PsychENCODE/TC_H3K27ac_peak.bed.gz, PsychENCODE/CBC_H3K27ac_peak.bed.gz, PsychENCODE/TARs.bed.gz,

BOCA/DLPFC_neuron.bed.gz, BOCA/DLPFC_glia.bed.gz, BOCA/OFC_neuron.bed.gz, BOCA/OFC_glia.bed.gz, BOCA/VLPFC_neuron.bed.gz, BOCA/VLPFC_glia.bed.gz, BOCA/ACC_neuron.bed.gz, BOCA/ACC_glia.bed.gz, BOCA/STC_neuron.bed.gz, BOCA/STC_glia.bed.gz, BOCA/ITC_neuron.bed.gz, BOCA/ITC_glia.bed.gz, BOCA/PMC_neuron.bed.gz, BOCA/PMC_glia.bed.gz, BOCA/INS_neuron.bed.gz, BOCA/INS_glia.bed.gz, BOCA/PVC_neuron.bed.gz, BOCA/PVC_glia.bed.gz, BOCA/AMY_neuron.bed.gz, BOCA/AMY_glia.bed.gz, BOCA/HIPP_neuron.bed.gz, BOCA/HIPP_glia.bed.gz, BOCA/MDT_neuron.bed.gz, BOCA/MDT_glia.bed.gz, BOCA/NAC_neuron.bed.gz, BOCA/NAC_glia.bed.gz, BOCA/PUT_neuron.bed.gz, BOCA/PUT_glia.bed.gz

HiC datasets:

EP/PsychENCODE/EP_links_oneway.txt.gz,

HiC/PsychENCODE/Promoter_anchored_loops.txt.gz, HiC/Giusti-Rodriguez_et_al_2019/Adult_Cortex.txt.gz, HiC/Giusti-Rodriguez_et_al_2019/Fetal_Cortex.txt.gz, HiC/GSE87112/Dorsolateral_Prefrontal_Cortex.txt.gz,

HiC/GSE87112/Hippocampus.txt.gz,

HiC/GSE87112/Mesenchymal_Stem_Cell.txt.gz, HiC/GSE87112/Neural_Progenitor_Cell.txt.gz

4.3. Stratified linkage disequilibrium score regression (LDSC)

Stratified LDSC used standard procedures for conducting stratified heritability analyses available at: https://github.com/bulik/ldsc/wiki/Partitioned-Heritability. All of the functional and cell-type specific categories were downloaded from: https://console.cloud.google.com/storage/browser/broad-alkesgroup-public-requester-pays.

4.4. Cell type & tissue enrichment in MAGMA/FUMA

Cell-type and tissue enrichment analyses conducted with MAGMA/FUMA were implemented using standard procedures. A tutorial for conducting the analyses can be found here: https://fuma.ctglab.nl/

5. Gene-based associations and enrichment

5.1. Gene-based and gene-set analyses with MAGMA

Gene-based and gene-set analyses conducted with MAGMA were implemented using standard procedures. A tutorial for conducting the analyses can be found here: https://fuma.ctglab.nl/

5.2. Summary-based mendelian randomization: T-SMR, P-SMR, MSMR

We used SMR with default values for everything (see https://yanglab.westlake.edu.cn/software/smr/#SMR&HEIDIanalysis).

5.3. Gene-drug interaction analyses using DrugTargetor

We used DrugTargetor to integrate the results from our GWAS meta-analyses and drug bioactivity data to prioritize drugs and targets for a given phenotype. The software to conduct the analyses can be found here: https://drugtargetor.com/index_v1.21.html

6. Genetic overlap between ANX and other phenotypes

6.1. PheWAS: Figure 3a

```
# PheWAS plot

# Load packages
require(data.table)
require(qqman)
library(readxl)


# Function to add braces to the plot
CurlyBraces <- function(x, y, range, pos = 1, direction = 1 ) {
   a=c(1,2,3.4,42,44)    # set flexion point for spline
   b=c(0,.01,.02,.03,.04) # set depth for spline flexion point
```

```
    curve = spline(a, b, n = 50, method = "hyman")$y / 2 # natural
    curve = c(curve,rev(curve))
    a_sequence = rep(x,100)
    b_sequence = seq(y-range/2,y+range/2,length=100)
    # direction
    if(direction==1)
    a_sequence = a_sequence+curve
    if(direction==2)
    a_sequence = a_sequence-curve
    # pos
    if(pos==1)
    lines(a_sequence,b_sequence) # vertical
    if(pos==2)
    lines(b_sequence,a_sequence) # horizontal
    }

## Load Data
p <- read.table("p.txt", header = T)
phe2 <- read.table("phe2.txt", header = T)
anxRes <-
fread("daner_fullANX_woAS_150923_updatedUTAH_ESTBBcorrN_70percentfilter_v11.gz")
snps   <-as.data.frame( read_excel("../Supplementary Tables (11-24-23).xlsx", sheet = 3))

# define the names of the SNPs and Phenotypic Categories

SNPnames <- phe2$rsid[phe2$uni == F]
SNPnames <- data.frame(rsid = SNPnames, ord = length(SNPnames):1)

nam <- cbind(names(table(phe2$rsid)))
colnames(nam) <- "rsid"

snpsX <- merge(SNPnames , nam, by = "rsid")
snpsX <- snpsX[order(as.numeric(snpsX$ord)),]
snpNames <- snpsX$rsid


CATnames <- as.data.frame(cbind(c(
    "Psychiatric"
  , "Personality"
  , "Behavioral"
  , "Cognitive"
  , "Cardiometabolic"
  , "Hematological"
  , "Immunological"
  , "Anthropometric"
  , "Skeletal"
  , "Reproduction"
  , "Respiratory"
  , "Other" ), 12:1))
```

```
cats <- cbind(names(table(phe2$category)))
colnames(cats) <- "V1"

catsX <- merge(CATnames , cats, by = "V1")
catsX <- catsX[order(as.numeric(catsX$V2)),]

catNames <- catsX$V1

NAMES <- list(snpNames, catNames)

# Define the graphing parameters

alpha=.75
border=.2
hide = F
layer = F
blocks = T
axis_labels = c("SNP", "Category")

gap.width = 0.2
snp.cex= 4
cat.cex = 5
lab.cex = 7

xw=0.25
cw = 0.15
wider <- 8
right <- 21



np <- ncol(p) - 5
n <- nrow(p)

ordering <- NULL

if (!is.null(ordering)) {
    stopifnot(is.list(ordering))
    if (length(ordering) != np)
        stop("'ordering' argument should have ", np, " components, has ",
            length(ordering))
}

#if (missing(layer)) {
#    layer <- 1:n
#}
#layer = phe2$snpOrd == 0
#p$layer <- layer
```

```
d <- p[, 1:2, drop = FALSE]
p <- p[, -c(1:2), drop = FALSE]
p$freq <- with(p, p$freq/sum(p$freq))
col <- col2rgb(p$col, alpha = TRUE)
if (!identical(alpha, FALSE)) {
    col["alpha", ] <- p$alpha * 256
}
p$col <- apply(col, 2, function(x) do.call(rgb, c(as.list(x),  maxColorValue = 256)))
isch <- sapply(d, is.character)
d[isch] <- lapply(d[isch], as.factor)
if (length(blocks) == 1) {
    blocks <- if (!is.na(as.logical(blocks))) {
        rep(blocks, np)
    }
    else if (blocks == "bookends") {
        c(TRUE, rep(FALSE, np - 2), TRUE)
    }
}
if (is.null(axis_labels)) {
    axis_labels <- names(d)
}   else {
    if (length(axis_labels) != ncol(d))
        stop("`axis_labels` should have length ", names(d),
            ", has ", length(axis_labels))
}
getp <- function(i, d, f, w = gap.width) {
    a <- c(i, (1:ncol(d))[-i])
    if (is.null(ordering[[i]])) {
        o <- do.call(order, d[a])
    }
    else {
        d2 <- d
        d2[1] <- ordering[[i]]
        o <- do.call(order, d2[a])
    }
    x <- c(0, cumsum(f[o])) * (1 - w)
    x <- cbind(x[-length(x)], x[-1])
    gap <- cumsum(c(0L, diff(as.numeric(d[o, i])) != 0))
    mx <- max(gap)
    if (mx == 0)
        mx <- 1
    gap <- gap/mx * w
    (x + gap)[order(o), ]
}
dd <- lapply(seq_along(d), getp, d = d, f = p$freq)
rval <- list(endpoints = dd)

labCol <- cbind(phe2[,c("rsid", "uni")], p[,"col"])
colnames(labCol) <- c("rsid", "uni","col")
labCols <- labCol[labCol$uni ==F, c("col")]
```

```
##### defining the parameters for the manhattan plot section
m <- anxRes[, c("CHR", "SNP", "BP", "P")]    ### removed anxRes$P < .001 for full graph
m <- m[order(m$CHR, m$BP), ]
m$index <- rep.int(seq_along(unique(m$CHR)), times = tapply(m$SNP, m$CHR, length))
m$logp <- -log10(m$P)

lastbase = 0
ticks = NULL
m$pos <- NA

# generating unique positions for plotting
for (i in unique(m$CHR)) {
    if (i == 1) {
        m$pos[m$CHR == i] <- m$BP[m$CHR == i]
    } else {
        lastbase = lastbase + max(m$BP[m$CHR == (i - 1)])
        m$pos[m$CHR == i] <-  m$BP[m$CHR == i] + lastbase
    }
}

m$pos <- (max(m$pos)+1) - m$pos
ticks <- tapply(m$pos, m$CHR, quantile, probs = 0.5)
labs <- unique(m$CHR)
ymax = ceiling(max(m$pos) )
ymin = floor(max(m$pos) )
col <-  ifelse(   (m$CHR %% 2) == 0 , "gray10", "gray60" )
m$AdjPos <- m$pos/max(m$pos)

ticks01 <- ticks/max(m$pos)
chrLab <- as.character(1:22)


snpsList <- data.frame("SNP" = c(snps[snps$P<5e-8,"SNP"]), "all" = 1 )

# phe2 identifies snps from the PheWAS. Therefore, any SNP in there is *pleiotropic*

phe2$uni <- duplicated(phe2$rsid)
PLE <- phe2[phe2$uni==F, c("rsid", "cols")]
PLE$pleio <- 1


com <- merge(snpsList, PLE, by.x = "SNP", by.y = "rsid", all = T)
pleio <- com$SNP[!is.na(com$pleio)]
non  <- com$SNP[is.na(com$pleio)]



rgb2col = function(rgbmat){
```

```
  # function to apply to each column of input rgbmat
  ProcessColumn = function(col){
    rgb(rgbmat[1, col],
        rgbmat[2, col],
        rgbmat[3, col],
                    rgbmat[4, col],
        maxColorValue = 255)
  }
  # Apply the function
  sapply(1:ncol(rgbmat), ProcessColumn)
}

    m.pleio <- m[which(m$SNP %in% pleio), ]
          m.pleio <- merge(m.pleio,  com, by = "SNP")
          pre.col <- col2rgb(m.pleio$cols, alpha = TRUE)
          pre.col["alpha",] <- alpha*256
          m.pleio$colX <- rgb2col(pre.col)


    m.non = m[which(m$SNP %in% non), ]


          ax <- lapply(split(dd[[1]], d[, 1]), range)
          triangle <- cbind( m.pleio[ order(m.pleio$CHR, m.pleio$BP), ], matrix(unlist(ax), ncol =
2, byrow = TRUE))
          triangle$V1 <- rev(triangle$V1)
          triangle$V2 <- rev(triangle$V2)
          triangle$pos <- triangle$pos/max(m$pos)

          cur <- 2
          l <- 15
          wi<- 2.9

          ind <- which(!p$hide)[rev(order(p[!p$hide, ]$layer))]

# Adjust SNP labels to prevent overlap
          ADJ <- c(
          0 , # "rs13056300"
          0 , # "rs7290074"
          0 , # "rs2070865"
          0 , # "rs12624433"
          -.001 , # "rs4801024"
          0 , # "rs8091977"
          0 , # "rs2289590"
          0 , # "rs12588874"
          -.0035 , # "rs3007061"
          -.0025 , # "rs61990288"
          -.0015 , # "rs9556979"
          0 , # "rs36119415"
          .0015 , # "rs7997746"
```

```
        .0025 , # "rs9534593"
        0 , # "rs6539062"
        -.0005 , # "rs989657"
        0 , # "rs78120929"
        0 , # "rs73034295"
        0 , # "rs7110863"
        0 , # "rs174560"
        0 , # "rs7121169"
        0 , # "rs2071754"
        0 , # "rs11599236"
        0 , # "rs28474857"
        0 , # "rs10961649"
        0 , # "rs10959883"
        0 , # "rs4976976"
        0 , # "rs4395923"
        .0015 , # "rs2371365"
        0 , # "rs12699332"
        0 , # "rs9373363"
        0 , # "rs58825580"
        0 , # "rs10476497"
        0 , # "rs11241568"
        0 , # "rs288160"
        0 , # "rs77960"
        0 , # "rs72704544"
        0 , # "rs2710323"
        0 , # "rs9867083"
        0 , # "rs17407658"
        0 , # "rs5015511"
        0 , # "rs11580539"
        0 ) # "rs34579341"

##### Start of the plotting

pdf("Figure3a_pheWAS.pdf", width=60, height=90)

# General plot parameters
op <- par(mai = c(6, 5.5, 5, 4), xpd=TRUE)

plot(NULL, type = "n", xlim = c(0, 35), ##  c(1 - cw, np + cw),
ylim = c(-0, 1), xaxt = "n", yaxt = "n", xaxs = "i",
yaxs = "i", xlab = "", ylab = "", frame = FALSE)

#################################################################
##    Alluvial plot
#################################################################

for (i in ind) {
   for (j in 1:1) {
          xspline(
                 (right - 1)+ c(j, j, j + xw, j + wider - xw, j + wider, j + wider - xw, j + xw,
```

```
j) + rep(c(cw, -cw, cw), c(3, 4, 2)),
                    c(dd[[j]][i, c(1, 2, 2)], rev(dd[[j + 1]][i, c(1, 1, 2, 2)]), dd[[j]][i, c(1, 1)]),
              shape = c(0, 0, 1, 1, 0, 0, 1, 1, 0, 0), open = FALSE,
              col = p$col[i], border = p$border[i])
    }
}

# Adding the snp labels to the plot
  j <- 1
  ax <- lapply(split(dd[[j]], d[, j]), range)
  for (k in seq_along(ax)) {
      text(right - .3, mean(ax[[k]])+ ADJ[k], labels = NAMES[[j]][k] , cex = snp.cex, adj = 1)   #,
col = labCols[k]
   CurlyBraces(x =  right+.1, y = mean(ax[[k]]), range = (max(ax[[k]])- min(ax[[k]])), pos = 1,
direction = 2 )
  }

# Adding the category labels to the plot
  j <- 2
  ax <- lapply(split(dd[[j]], d[, j]), range)
  for (k in seq_along(ax)) {
      text(right + wider + .3, mean(ax[[k]]) , labels = NAMES[[j]][k] , cex = cat.cex, adj = 0) ###
   CurlyBraces(x =  right + wider -.1, y = mean(ax[[k]]), range = (max(ax[[k]])- min(ax[[k]])), pos
= 1, direction = 1 )
  }

# adding general labels
  axis(3, at = c(right - 1.5, right+wider+ 2), tick = FALSE, labels = c("Pleiotropic\nSNPs",
"Phenotypic\nCategory"),
              cex.axis = lab.cex)
##############################################################
##      Manhattan Plot
##############################################################
pdf("Figure3b_anx_psych_heat.pdf", width=3, height=6)
points(m$logp, m$pos/max(m$pos), las = 1, pch = 20, col = col)
dev.off()


    lines( c(5,5), c(0, 1.01), col = "blue", lwd = 1.5)                    # Suggestive
Significane
    lines( c(-log10(5e-8), -log10(5e-8)), c(0, 1.01), col = "red", lwd = 1.5)        # Genome wide
Significance

    lines( c(0,0), c( 0, 1.01), col = "black")

points(m$logp[1], m$pos[1]/max(m$pos), col = "red", pch = 18, cex = 10)

for (i in 1:dim(triangle)[1]) {
      xspline(
              c(triangle$logp[i], l + cur, l + wi, l + wi, l + cur , triangle$logp[i] ),
```

6.1. PheWAS: Figure 3b

```
require("pheatmap")
phe <- as.data.frame(read.csv("phe.csv"))
psych <- phe[phe$category == "Psychiatric", c("rsid", "trait")]
psych$cat <- psych$trait
pheatmap(table(psych$rsid, psych$cat), color = colorRampPalette(c("white", "firebrick3"))(50))
```

6.2. Bi-directional generalized summary-data based mendelian randomization (GSMR)

The bi-directional GSMR analysis was done as outlined here:
https://yanglab.westlake.edu.cn/software/gsmr/