Project

Amir Ragaie(21010300) - Ebrahim AlaaEldin(21010017)

2025-05-20

```
# Install and load necessary packages
required_packages <- c(
  "BiocManager", "SNPRelate", "GENESIS", "gdsfmt", "ggplot2",
  "dplyr", "tidyverse", "ggnetwork", "ggrepel", "Biobase",
  "GWASTools", "openxlsx", "qqman", "GeneNet", "RCy3"
)

# Install any missing packages
for(pkg in required_packages) {
  if(!require(pkg, character.only = TRUE)) {
    if(pkg %in% c("SNPRelate", "GENESIS", "gdsfmt", "Biobase", "GWASTools", "GeneNet", "RCy3"))
{
      BiocManager::install(pkg)
    } else {
      install.packages(pkg)
    }
    library(pkg, character.only = TRUE)
  }
}
```

Table of Contents

Task 1: Compute Kinship using SNPRelate and GENESIS

a. Compute kinship using the SNPRelate and GENESIS package in R. You can use the IBD family of functions, but note that you need to transform the plink format to GDS format using the function snpgdsPED2GDS that takes the ped and map files.

##convert to the GDS format needed for our analysis:

#Calculating the genetic relation

```
# Convert to GDS format if file doesn't exist
gds_file <- "Qatari_SNPs.gds"
if (!file.exists(gds_file)) {
  cat("Converting PED/MAP to GDS format...\n")
  snpgdsPED2GDS(
    ped.fn = "data\\Qatari156.ped",
    map.fn = "data\\Qatari156.map",
    out.gdsfn = gds_file
  )
  cat("GDS file created successfully.\n")
} else {
  cat("GDS file already exists. Skipping conversion step.\n")
}
```

```
## GDS file already exists. Skipping conversion step.
```

```
# Open the GDS file
genome_data <- snpgdsOpen(gds_file)

# Get the maximum number of available cores/threads
max_threads <- parallel::detectCores()
cat("Detected", max_threads, "CPU cores on this system.\n")
```

```
## Detected 16 CPU cores on this system.
```

```
# Calculating the genetic relation
# Define file to store kinship results
kinship_file <- "kinship_results.RData"

if (file.exists(kinship_file)) {
  # Load previously calculated results
  cat("Loading previously calculated kinship data...\n")
  load(kinship_file)
  cat("Kinship data loaded successfully.\n")

} else {
  # Calculate kinship coefficients using Maximum Likelihood Estimation
  cat("Calculating kinship coefficients using all", max_threads, "CPU cores...\n")
  cat("This calculation may take some time...\n")

  kinship_result <- snpgdsIBDMLE(
    genome_data,
    sample.id = NULL,
    snp.id = NULL,
    num.thread = max_threads,  # Using all available threads
    kinship = TRUE,
    kinship.constraint = TRUE,
    verbose = TRUE
  )

  # Extract the kinship matrix
  relatedness_matrix <- kinship_result$kinship

  # Save results to avoid recalculation
  save(kinship_result, relatedness_matrix, file = kinship_file)
  cat("Kinship results saved to", kinship_file, "\n")
}
```

```
## Loading previously calculated kinship data...
## Kinship data loaded successfully.
```

```
# Calculate statistics (quick, so we do it every time)
# Count related individuals (kinship > 0.1)
related_pairs <- (sum(relatedness_matrix > 0.1) - nrow(relatedness_matrix)) / 2
cat("Number of related individual pairs (kinship > 0.1):", related_pairs, "\n")
```

```
## Number of related individual pairs (kinship > 0.1): 6
```

```
# Count unique individuals with at least one relative
unique_related <- 0
sample_count <- nrow(relatedness_matrix)
for(i in 1:sample_count) {
  for(j in 1:sample_count) {
    if(relatedness_matrix[i,j] > 0.1 && i != j) {
      unique_related <- unique_related + 1
      break
    }
  }
}
cat("Individuals with at least one relative:", unique_related, "\n")
```

```
## Individuals with at least one relative: 9
```

```
# close the GDS file when you're done
snpgdsClose(genome_data)
```

GDS (Genomic Data Structure)

GDS stands for **Genomic Data Structure**. It is a file format used to efficiently store and access large-scale genetic data, such as SNP genotype data.

Method	Best for
snpgdsIBDKING (https://www.rdocumentation.org/packages/SNPRelate/versions/1.6.4/topics/snpgdsIBDKING)	Quick, robust kinship estimates on large datasets
snpgdsIBDMLE (https://www.rdocumentation.org/packages/SNPRelate/versions/1.6.4/topics/snpgdsIBDMLE)	More accurate, maximum likelihood kinship

Input: You give it an IBD result object from `snpgdsIBDMLE` , which contains IBD/kinship estimates between individuals.

Output: It returns a data frame (`ibd_df`) listing pairs of individuals with their IBD estimates and related information. b. Report the number of individuals who have a kinship > 0.1

Task 2: Compute mQTLs with Mixed Models

2.a: Compute mQTLs using all SNPs and all metabolites with kinship

```
# Load metabolite data from CSV file
# This file contains metabolite measurements for all samples
metabolite_data <- read.csv("qatari_metabolites_2025.csv", header = TRUE)

# Apply sample identifiers to the kinship matrix
# This ensures the relatedness matrix corresponds correctly to metabolite samples
rownames(relatedness_matrix) <- metabolite_data[, 1]
colnames(relatedness_matrix) <- metabolite_data[, 1]
```

2.b & 2.c: Set up mixed models with GENESIS and include PCs as covariates

```
# Load principal component analysis results from PLINK
# Principal components account for population structure
pca_components <- read.table("data/pca_results.eigenvec", header = FALSE)
pca_components <- pca_components[,-1]  # Remove redundant sample ID column
colnames(pca_components) <- c("SampleID", paste0("PC", 1:(ncol(pca_components) - 1)))

# Select the first three principal components as covariates
# These capture the main axes of genetic variation in the population
top_pcs <- pca_components[,c(2:4)]
colnames(top_pcs) <- c("PC1", "PC2", "PC3")

# Prepare integrated dataset for analysis
# Combine PCs with metabolite measurements
analysis_data <- as.data.frame(cbind(top_pcs, metabolite_data[,-1]))
rownames(analysis_data) <- metabolite_data$Sample
metabolite_names <- colnames(metabolite_data)[-1]
```

```
# Set up genotype data objects required for GWAS
# Create sample annotation structure
sample_annotation <- data.frame(
  scanID = metabolite_data$Sample,
  stringsAsFactors = FALSE
)
rownames(sample_annotation) <- sample_annotation$scanID

# Access GDS file containing genotype data
genome_reader <- GdsGenotypeReader("Qatari_SNPs.gds")

# Extract SNP information from genome data
snp_ids <- getSnpID(genome_reader)
snp_locations <- getPosition(genome_reader)
snp_chromosomes <- getChromosome(genome_reader)
ref_alleles <- getAlleleA(genome_reader)
alt_alleles <- getAlleleB(genome_reader)

# Create variant annotation structure
variant_info <- data.frame(
  snpID = snp_ids,
  chromosome = snp_chromosomes,
  position = snp_locations,
  alleleA = ref_alleles,
  alleleB = alt_alleles,
  stringsAsFactors = FALSE
)

# Create data structures required by GENESIS
sample_annot_obj <- ScanAnnotationDataFrame(sample_annotation)
variant_annot_obj <- SnpAnnotationDataFrame(variant_info)
genotype_data <- GenotypeData(genome_reader,
                              scanAnnot = sample_annot_obj,
                              snpAnnot = variant_annot_obj)
```

2.a & 2.b: Execute mixed model association testing

for all metabolite-SNP pairs

```
# Initialize lists to store results for each metabolite
association_results <- list()
null_models <- list()

# Perform mixed model analysis for each metabolite
for(metabolite in metabolite_names) {
  # Step 1: Fit null model accounting for kinship
  # This regresses metabolite on covariates while accounting for genetic relatedness
  null_model <- fitNullModel(
    x = analysis_data,                  # Data frame with outcome and covariates
    outcome = metabolite,               # Metabolite as the phenotype
    covars = c("PC1", "PC2", "PC3"),    # Principal components as covariates
    cov.mat = relatedness_matrix,       # Kinship matrix for random effects
    verbose = TRUE                      # Show progress
  )

  # Step 2: Test associations with all SNPs
  # Create a genotype iterator to process SNPs in blocks for efficiency
  genotype_blocks <- GenotypeBlockIterator(genotype_data)

  # Execute genome-wide association test
  associations <- assocTestSingle(
    gdsobj = genotype_blocks,           # Genotype data
    null.model = null_model,            # Pre-computed null model
    verbose = TRUE                      # Show progress
  )

  # Add metabolite identifier to results for tracking
  associations$metabolite <- metabolite

  # Store results for this metabolite
  association_results[[metabolite]] <- associations
  null_models[[metabolite]] <- null_model
}
```

```
## [1]  140.150908  140.150908 -646.712863    1.347985
## [1]   21.381861  210.371985 -646.398014    1.267289
## [1]    7.316958  218.597505 -646.368322    1.259003
## [1]    0.3711887  222.6574232 -646.3541691    1.2549718
## [1]    0.155554  222.783437 -646.353735    1.254847
## [1]    0.04775921  222.84643107 -646.35351813    1.25478511
```

```
## [1]  142.314187  142.314187 -648.975576    1.367612
## [1]  406.916416   69.869012 -648.873588    1.079087
## [1]  573.707948    8.369171 -648.856161    1.005772
## [1]  588.799691    2.696482 -648.856058    1.000036
## [1]  588.30803    2.94508 -648.85606    1.00000
```

```
## [1]  138.269542  138.269542 -645.927643    1.352282
## [1]  419.021521   27.128690 -645.577399    1.198569
## [1]  451.789396   13.525181 -645.547161    1.186311
## [1]  483.6511200    0.2724314 -645.5196146    1.1748025
## [1]  484.13385657    0.07131292 -645.51921060    1.17463185
## [1]  484.254481    0.000000 -645.519067    1.174694
## [1]  556.270111    0.000000 -645.519067    1.022616
## [1]  568.572571    0.000000 -645.519067    1.000489
## [1]  568.8507    0.0000 -645.5191    1.0000
```

```
## [1]  160.42176  160.42176 -655.22538    1.31723
## [1]  394.295772   59.499957 -654.825179    1.243708
## [1]  506.281286   10.731011 -654.668176    1.214232
## [1]  519.640607    4.901368 -654.650768    1.210898
## [1]  526.275911    2.005418 -654.642224    1.209253
## [1]  529.5823196    0.5622361 -654.6379909    1.2084365
## [1]  530.4075073    0.2020444 -654.6369371    1.2082330
## [1]  530.819924    0.000000 -654.636346    1.208234
## [1]  622.304381    0.000000 -654.636346    1.030612
## [1]  640.788683    0.000000 -654.636346    1.000883
## [1]  641.354030    0.000000 -654.636346    1.000001
```

```
## [1]  172.144976  172.144976 -659.588270    1.300055
## [1]   56.168584  257.795059 -659.515737    1.166027
## [1]    6.697121  291.985707 -659.498241    1.126008
## [1]    1.475031  295.559295 -659.496733    1.122084
## [1]    0.198197  296.432528 -659.496373    1.121131
## [1]    0.03948057  296.54105959 -659.49632890    1.12101311
```

```
## [1]  121.245311  121.245311 -634.601634    1.328641
## [1]  238.606796  108.640257 -634.584508    1.065355
## [1]  330.046878   77.930774 -634.577421    1.003899
## [1]  362.693149   62.994332 -634.576419    1.000035
## [1]  371.816191   58.572997 -634.576334    1.000002
## [1]  374.41074   57.31383 -634.57633    1.00000
## [1]  375.15084   56.95459 -634.57633    1.00000
## [1]  375.36212   56.85203 -634.57633    1.00000
## [1]  375.42245   56.82274 -634.57633    1.00000
```

```
## [1]  156.525958  156.525958 -655.622009    1.357079
## [1]   43.008010  218.609669 -655.368881    1.314266
## [1]   13.079395  234.794066 -655.307592    1.304543
## [1]    5.523377  238.870669 -655.292433    1.302169
## [1]    1.736863  240.912435 -655.284883    1.300990
## [1]    0.789197  241.423297 -655.282998    1.300696
## [1]    0.315236  241.678780 -655.282057    1.300550
## [1]    0.07822357  241.80653471 -655.28158566    1.30047629
## [1]    0.000000  241.838475 -655.281430    1.300507
## [1]    0.000000  297.719885 -655.281430    1.056405
## [1]    0.000000  313.616028 -655.281430    1.002859
## [1]    0.000000  314.510087 -655.281430    1.000008
```

```
## [1]  123.996592  123.996592 -639.325896    1.382481
## [1]   61.385984  156.827812 -638.995240    1.359248
## [1]   28.851819  173.818198 -638.833334    1.348179
## [1]   12.319333  182.437327 -638.753421    1.342777
## [1]    3.992284  186.775245 -638.713746    1.340109
## [1]    1.903275  187.863100 -638.703852    1.339445
## [1]    0.8578842  188.4074399 -638.6989100    1.3391135
## [1]    0.3349683  188.6797127 -638.6964401    1.3389480
## [1]    0.07345548  188.81587467 -638.69520538    1.33886521
## [1]    0.000000  188.849918 -638.694859    1.338872
## [1]    0.000000  236.648302 -638.694859    1.068446
## [1]    0.000000  251.808197 -638.694859    1.004121
## [1]    0.000000  252.841566 -638.694859    1.000017
```

```
## [1]  161.38524  161.38524 -656.16989    1.32574
## [1]   38.856893  227.016712 -655.767347    1.286887
## [1]    7.928311  243.507671 -655.676083    1.278021
## [1]    0.2106299  247.6196531 -655.6538996    1.2758549
## [1]    0.09012769  247.68384622 -655.65355505    1.27582121
## [1]    0.000000  247.715942 -655.653297    1.275878
## [1]    0.000000  301.278570 -655.653297    1.049047
## [1]    0.000000  315.364454 -655.653297    1.002191
## [1]    0.000000  316.053811 -655.653297    1.000005
```

```
## [1]  119.222004  119.222004 -634.618897    1.351496
## [1]   70.442193  188.984903 -634.599950    1.072782
## [1]   31.638742  223.029245 -634.597444    1.004681
## [1]   30.920203  224.490578 -634.597443    1.000022
## [1]   31.02272  224.44580 -634.59744    1.00000
```

```
## [1]  126.182791  126.182791 -639.806345    1.367144
## [1]   11.752626  193.840539 -639.583322    1.281834
## [1]    4.498329  198.032355 -639.571382    1.277448
## [1]    0.8784853  200.1221031 -639.5655061    1.2752844
## [1]    0.4265045  200.3829157 -639.5647764    1.2750156
## [1]    0.2005463  200.5132961 -639.5644118    1.2748813
## [1]    0.08757537  200.57847980 -639.56422968    1.27481423
## [1]    0.03109193  200.61107003 -639.56413862    1.27478069
## [1]    0.000000  200.627365 -639.564089    1.274773
## [1]    0.000000  243.871873 -639.564089    1.048724
## [1]    0.000000  255.202222 -639.564089    1.002163
## [1]    0.000000  255.753090 -639.564089    1.000005
```

```
## [1]  150.583257  150.583257 -652.975884    1.362366
## [1]   58.623811  224.938788 -652.927786    1.202733
## [1]    7.933504  260.264999 -652.910519    1.154156
## [1]    1.943380  264.282844 -652.908757    1.149359
## [1]    0.4611942  265.2735780 -652.9083296    1.1481936
## [1]    0.09163079  265.52039828 -652.90822340    1.14790425
## [1]    0.04546627  265.55122380 -652.90821016    1.14786814
```

```
## [1]  151.852679  151.852679 -651.693119    1.328366
## [1]    1.441611  280.668281 -651.644846    1.065558
## [1]    0.6097243  281.1410186 -651.6446692    1.0653021
## [1]    0.194150  281.377094 -651.644581    1.065175
## [1]    0.09030271  281.43607653 -651.64455930    1.06514280
## [1]    0.03838482  281.46556308 -651.64454832    1.06512689
```

```
## [1]  126.628339  126.628339 -638.408035    1.337497
## [1]   55.084441  167.086312 -638.290204    1.296784
## [1]   16.388658  188.564642 -638.231113    1.278544
## [1]    6.398042  194.067166 -638.216298    1.274210
## [1]    1.366088  196.833669 -638.208899    1.272072
## [1]    0.1036265  197.5271485 -638.2070497    1.2715409
## [1]    0.000000  197.570520 -638.206898    1.271585
## [1]    0.000000  239.767556 -638.206898    1.047797
## [1]    0.000000  250.704854 -638.206898    1.002085
## [1]    0.000000  251.226531 -638.206898    1.000004
```

```
## [1]  122.89621  122.89621 -636.51657    1.34424
## [1]  419.4546340    0.7407732 -636.1194211    1.1940444
## [1]  420.528865    0.281744 -636.118257    1.193666
## [1]  421.06552176    0.05241517 -636.11767575    1.19347642
## [1]  421.132575    0.000000 -636.117543    1.193591
## [1]  489.436880    0.000000 -636.117543    1.027017
## [1]  502.312115    0.000000 -636.117543    1.000692
## [1]  502.6597    0.0000 -636.1175    1.0000
```

```
## [1]  190.736956  190.736956 -669.945190    1.344637
## [1]  595.496643   65.751019 -669.833859    1.071475
## [1]  686.22405   27.54976 -669.81636    1.05396
## [1]  725.224426   10.910947 -669.809798    1.047269
## [1]  743.247372    3.182608 -669.806957    1.044325
## [1]  747.577458    1.321632 -669.806292    1.043633
## [1]  749.7211726    0.3998213 -669.8059658    1.0432925
## [1]  750.2544534    0.1704468 -669.8058849    1.0432080
## [1]  750.52076405    0.05589399 -669.80584453    1.04316576
## [1]  750.587301    0.000000 -669.805825    1.043233
## [1]  781.69286    0.00000 -669.80582    1.00172
## [1]  783.035350    0.000000 -669.805825    1.000003
```

```
## [1]  155.733467  155.733467 -654.388994    1.342034
## [1]  352.410942   74.028374 -654.173020    1.262298
## [1]  458.698618   28.427115 -654.068085    1.230508
## [1]  512.652431    5.024433 -654.017354    1.216309
## [1]  519.413339    2.078500 -654.011102    1.214622
## [1]  522.7940302    0.6046586 -654.0079842    1.2137848
## [1]  523.6392181    0.2360963 -654.0072057    1.2135761
## [1]  524.06181305    0.05180287 -654.00681663    1.21347183
## [1]  524.167462    0.000000 -654.006707    1.213473
## [1]  616.378476    0.000000 -654.006707    1.031936
## [1]  635.453845    0.000000 -654.006707    1.000959
## [1]  636.062449    0.000000 -654.006707    1.000001
```

```
## [1]  157.322856  157.322856 -653.729002    1.316989
## [1]  510.577010   11.381895 -653.216032    1.178675
## [1]  531.841384    2.220013 -653.193211    1.173127
## [1]  534.471807    1.085426 -653.190440    1.172452
## [1]  535.785253    0.518819 -653.189060    1.172115
## [1]  536.4415337    0.2356878 -653.1883717    1.1719470
## [1]  536.76956315    0.09416532 -653.18802790    1.17186301
## [1]  536.933550    0.000000 -653.187799    1.171926
## [1]  615.703591    0.000000 -653.187799    1.021995
## [1]  628.954716    0.000000 -653.187799    1.000463
## [1]  629.2460    0.0000 -653.1878    1.0000
```

```
## [1]  136.049954  136.049954 -644.487398    1.348545
## [1]  136.807451  187.952535 -644.485248    1.071606
## [1]  114.053653  216.039366 -644.484298    1.004501
## [1]  100.620278  223.799702 -644.484149    1.000023
## [1]   96.36899  225.87728 -644.48414    1.00000
## [1]   95.10904  226.49127 -644.48413    1.00000
## [1]   94.73582  226.67314 -644.48413    1.00000
## [1]   94.62526  226.72701 -644.48413    1.00000
## [1]   94.5925  226.7430 -644.4841    1.0000
```

```
## [1]  142.960225  142.960225 -648.612684    1.354946
## [1]    6.171609  222.895938 -648.430504    1.272217
## [1]    1.477886  225.546329 -648.424957    1.270094
## [1]    0.3029548  226.2092815 -648.4235741    1.2695667
## [1]    0.00000  226.37504 -648.42322    1.26946
## [1]    0.000000  274.426183 -648.423218    1.047182
## [1]    0.000000  286.790664 -648.423218    1.002034
## [1]    0.000000  287.372855 -648.423218    1.000004
```

2.d: Report significant SNP-Metabolite associations (p < 0.0001)

```
# Combine results from all metabolites into a single data frame
all_associations <- bind_rows(association_results)

# Filter for significant associations based on p-value threshold
significant_hits <- filter(all_associations, Score.pval < 0.0001)

# Add variant annotation information to significant results
variant_annotations <- pData(getSnpAnnotation(genotype_data))
significant_hits <- merge(significant_hits, variant_annotations,
                          by.x = "variant.id", by.y = "snpID")

# Format results for reporting
# Add effect allele and degrees of freedom information
significant_hits$effect_allele <- significant_hits$alleleA
significant_hits$df <- significant_hits$n.obs - 4

# Export significant results to Excel spreadsheet
write.xlsx(significant_hits, file = "significant_mQTLs.xlsx", row.names = FALSE)

# Also save as CSV for broader compatibility
write.csv(significant_hits, file = "significant_mQTLs.csv", row.names = FALSE)
```

2.e: Report heritability estimates using varCompCI

```
# Create data frame to store heritability estimates for each metabolite
heritability_estimates <- data.frame(
  metabolite = metabolite_names,
  heritability = NA
)

# Calculate heritability for each metabolite
for(metabolite in metabolite_names) {
  # Get variance component estimates from the null model
  variance_components <- varCompCI(null_models[[metabolite]])

  # Extract genetic variance component
  genetic_variance <- variance_components$Proportion[1]

  # Calculate total variance (genetic + residual)
  total_variance <- variance_components$Proportion[1] + variance_components$Proportion[2]

  # Calculate heritability as proportion of genetic variance
  h2 <- genetic_variance / total_variance

  # Store heritability estimate
  heritability_estimates[heritability_estimates$metabolite == metabolite, "heritability"] <- h2
}
```

```
##               Proportion  Lower 95 Upper 95
## V_A         0.0002142685 -4.118766 4.119195
## V_resid.var 0.9997857315 -3.119195 5.118766
##               Proportion   Lower 95 Upper 95
## V_A         0.995018919 -0.3152637 2.305302
## V_resid.var 0.004981081 -1.3053016 1.315264
##             Proportion Lower 95 Upper 95
## V_A                  1        1        1
## V_resid.var          0       NA       NA
##             Proportion Lower 95 Upper 95
## V_A                  1        1        1
## V_resid.var          0       NA       NA
##               Proportion  Lower 95 Upper 95
## V_A         0.0001331192 -5.034168 5.034434
## V_resid.var 0.9998668808 -4.034434 6.034168
##             Proportion  Lower 95 Upper 95
## V_A          0.8685405 -1.124347 2.861428
## V_resid.var  0.1314595 -1.861428 2.124347
##             Proportion Lower 95 Upper 95
## V_A                  0       NA       NA
## V_resid.var          1        1        1
##             Proportion Lower 95 Upper 95
## V_A                  0       NA       NA
## V_resid.var          1        1        1
##             Proportion Lower 95 Upper 95
## V_A                  0       NA       NA
## V_resid.var          1        1        1
##             Proportion  Lower 95 Upper 95
## V_A          0.1214346 -4.688999 4.931868
## V_resid.var  0.8785654 -3.931868 5.688999
##             Proportion Lower 95 Upper 95
## V_A                  0       NA       NA
## V_resid.var          1        1        1
##               Proportion  Lower 95 Upper 95
## V_A         0.0001711854 -5.990273 5.990615
## V_resid.var 0.9998288146 -4.990615 6.990273
##               Proportion  Lower 95 Upper 95
## V_A         0.0001363562 -6.982595 6.982868
## V_resid.var 0.9998636438 -5.982868 7.982595
##             Proportion Lower 95 Upper 95
## V_A                  0       NA       NA
## V_resid.var          1        1        1
##             Proportion Lower 95 Upper 95
## V_A                  1        1        1
## V_resid.var          0       NA       NA
##             Proportion Lower 95 Upper 95
## V_A                  1        1        1
## V_resid.var          0       NA       NA
##             Proportion Lower 95 Upper 95
## V_A                  1        1        1
## V_resid.var          0       NA       NA
##             Proportion Lower 95 Upper 95
```

```
## V_A                 1        1        1
## V_resid.var         0       NA       NA
##             Proportion  Lower 95 Upper 95
## V_A          0.2943731 -4.506559 5.095306
## V_resid.var  0.7056269 -4.095306 5.506559
##             Proportion Lower 95 Upper 95
## V_A                 0       NA       NA
## V_resid.var         1        1        1
```

```
# Export heritability estimates to CSV file
write.csv(heritability_estimates, file = "heritability_estimates.csv", row.names = FALSE)
```

Task 3: Inflation factor calculation

```
# Create data frame to store inflation factors (lambda) for each metabolite
lambda_values <- data.frame(
  metabolite = metabolite_names,
  lambda = NA
)

# Calculate genomic inflation factor for each metabolite
for(metabolite in metabolite_names) {
  # Get test statistics from association results
  test_stats <- association_results[[metabolite]]

  # Convert score statistics to chi-squared values
  chi_squared <- (test_stats$Score.Stat)^2

  # Calculate lambda as median chi-squared divided by expected median (0.455)
  # This quantifies test statistic inflation due to population structure or other confounders
  lambda <- median(chi_squared, na.rm = TRUE) / 0.455

  # Store lambda value
  lambda_values[lambda_values$metabolite == metabolite, "lambda"] <- lambda
}

# Calculate average inflation factor across all metabolites
mean_lambda <- mean(lambda_values$lambda, na.rm = TRUE)

# Add average to results table
lambda_values <- bind_rows(lambda_values,
                           data.frame(metabolite = "Average", lambda = mean_lambda))

# Report average inflation factor
cat("Average genomic inflation factor:", mean_lambda, "\n")
```

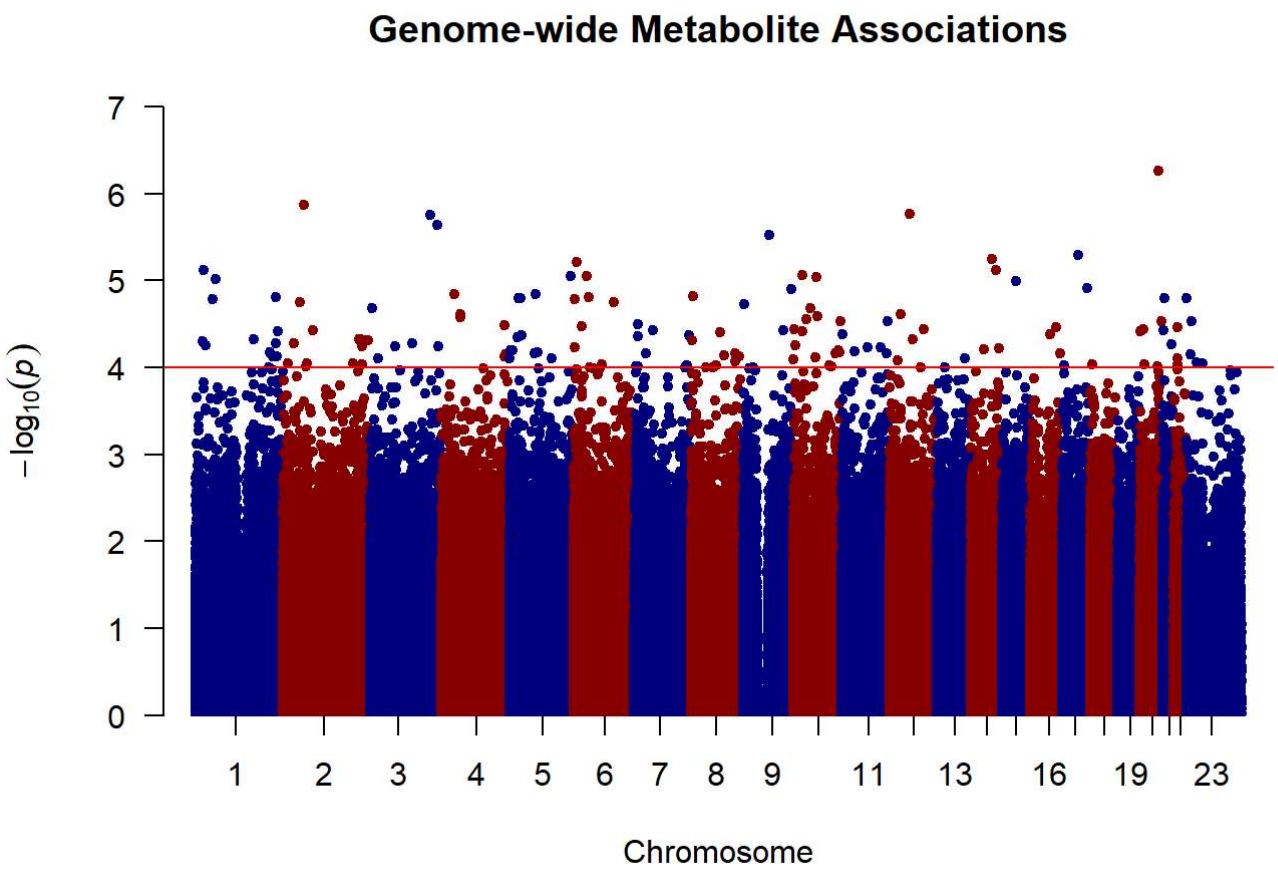
```
## Average genomic inflation factor: 1.035894
```

```
# Export inflation factors to CSV file
write.csv(lambda_values, file = "inflation_factors.csv", row.names = FALSE)
```

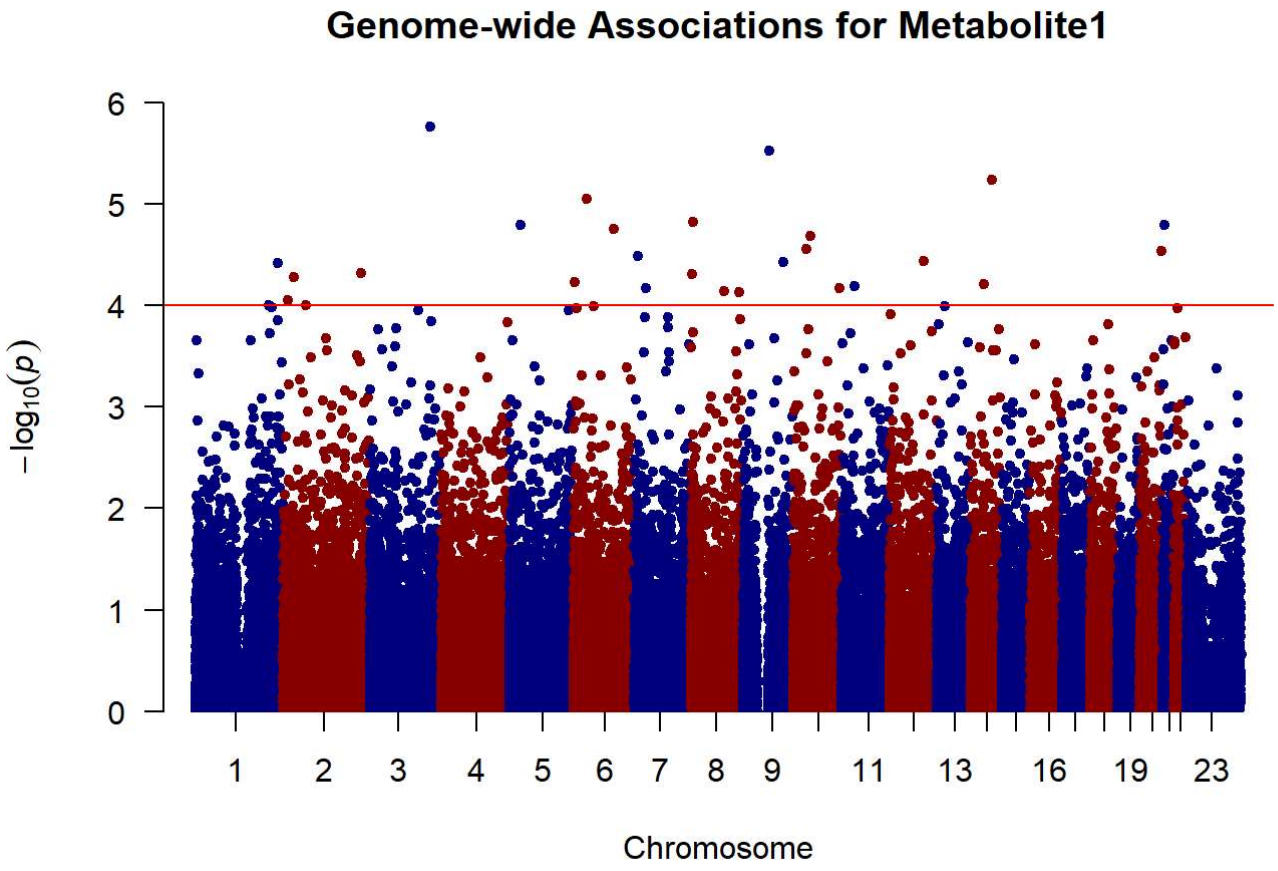
Task 4: Manhattan Plot

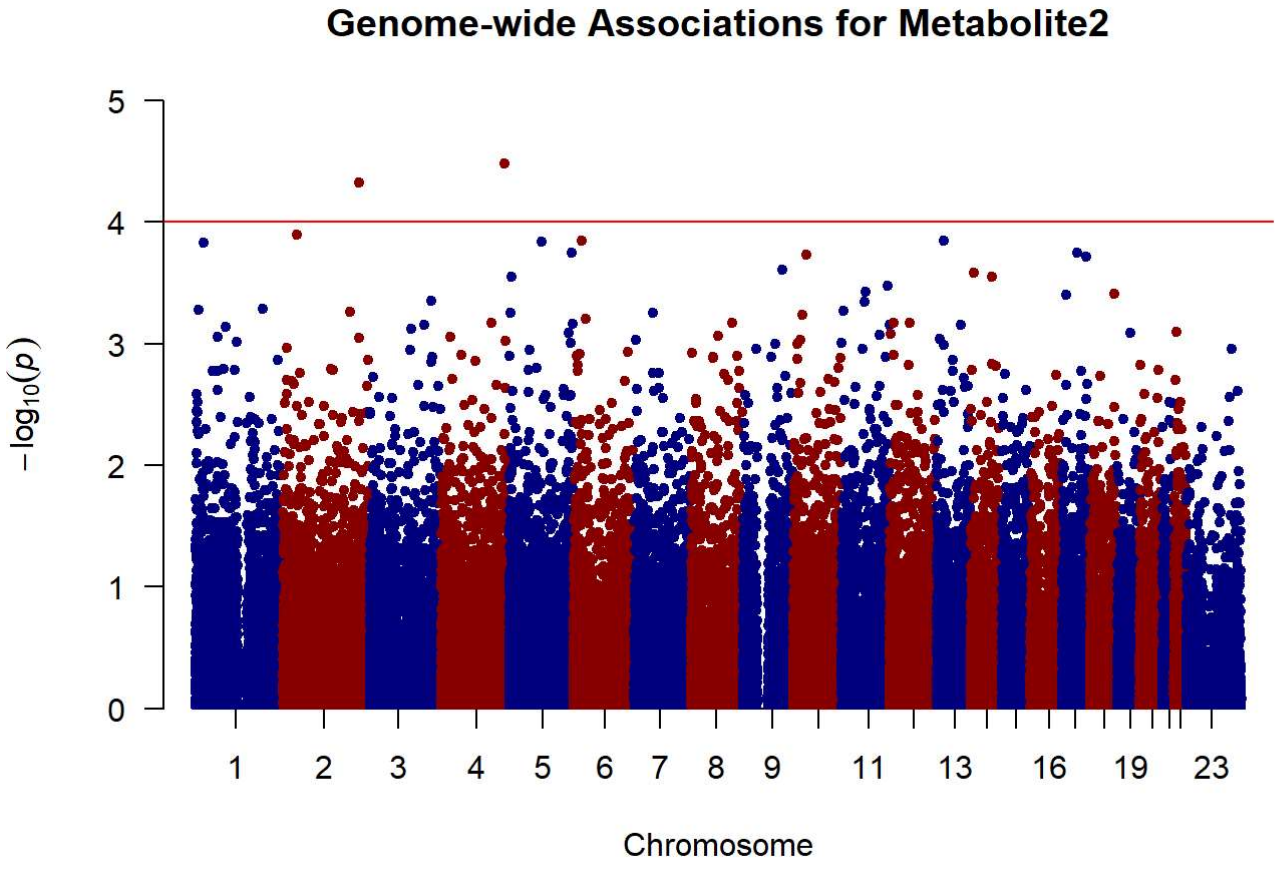
Visualization of Genome-wide SNP-Metabolite Associations

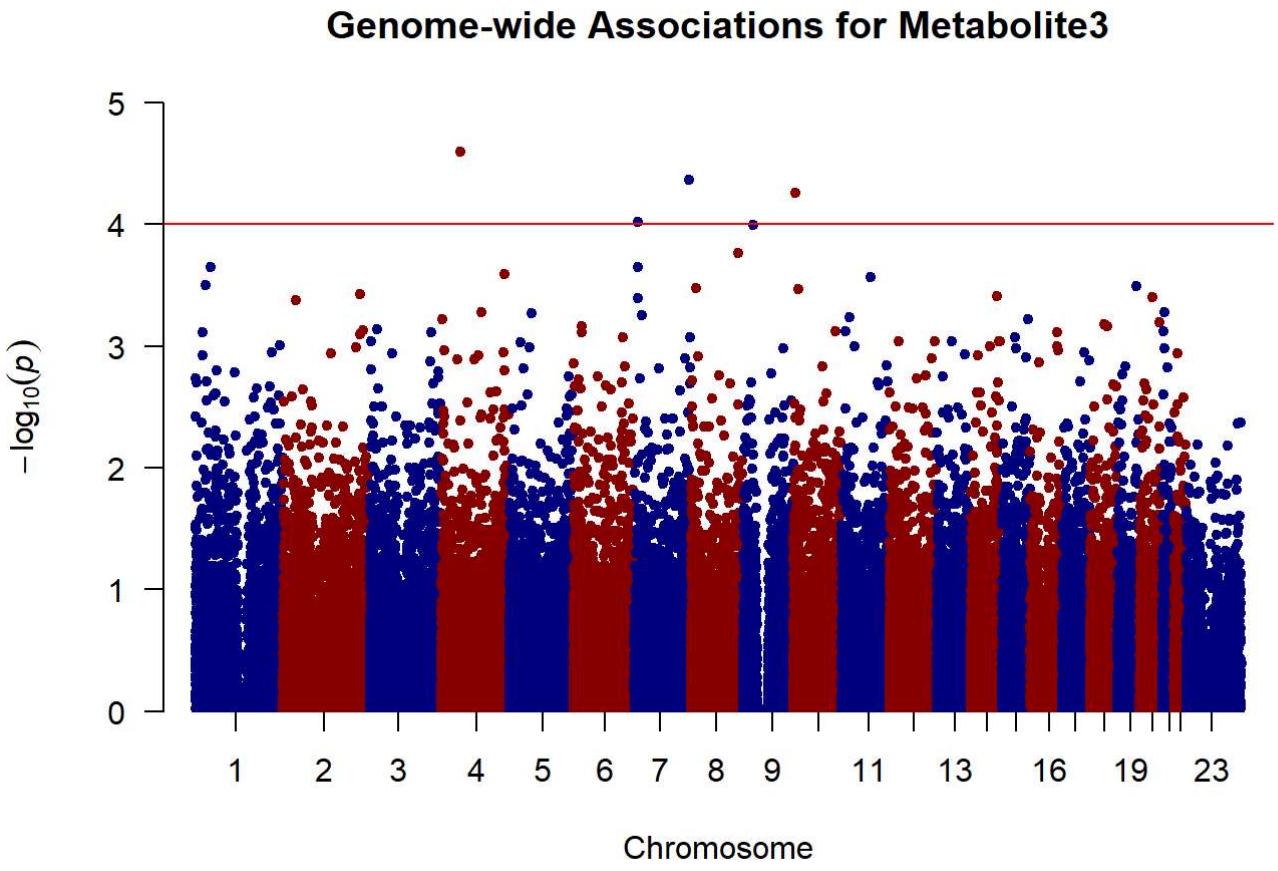
```
# Convert association results to format required for Manhattan plotting
# This ensures all chromosomes are represented properly including X chromosome
manhattan_data <- all_associations
manhattan_data$CHR <- as.numeric(ifelse(manhattan_data$chr == "X", "23", as.character(manhattan_
data$chr)))
manhattan_data$BP <- manhattan_data$pos
manhattan_data$P <- manhattan_data$Score.pval
manhattan_data$SNP <- manhattan_data$variant.id

# Generate genome-wide Manhattan plot showing all associations
# Alternating colors (navy and darkred) distinguish adjacent chromosomes
# The genome-wide significance threshold (p < 0.0001) is shown as a horizontal line
manhattan(
  manhattan_data,
  col = c("navy", "darkred"),        # Alternate chromosome colors
  genomewideline = -log10(1e-4),     # Significance threshold line
  suggestiveline = FALSE,            # No suggestive threshold line
  main = "Genome-wide Metabolite Associations"
)
```

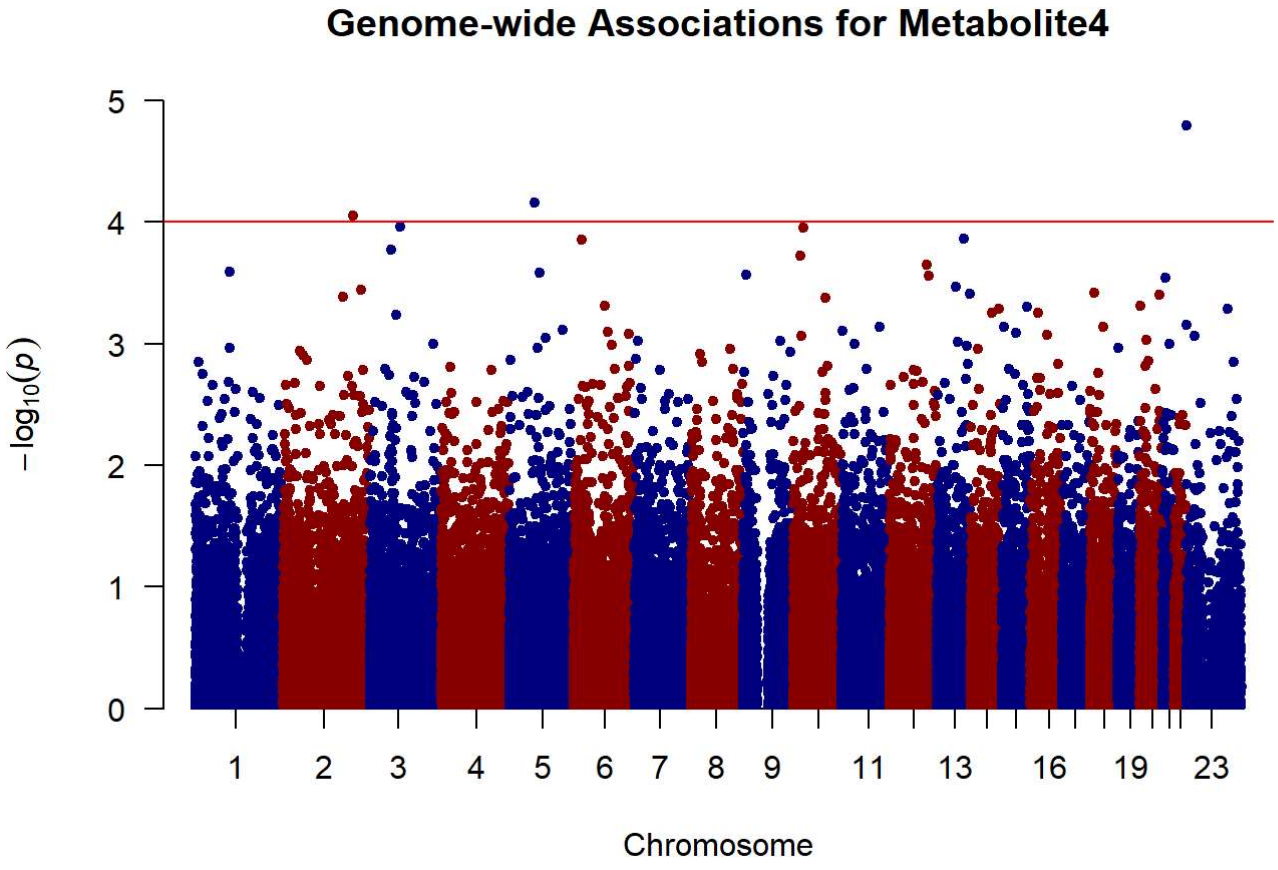


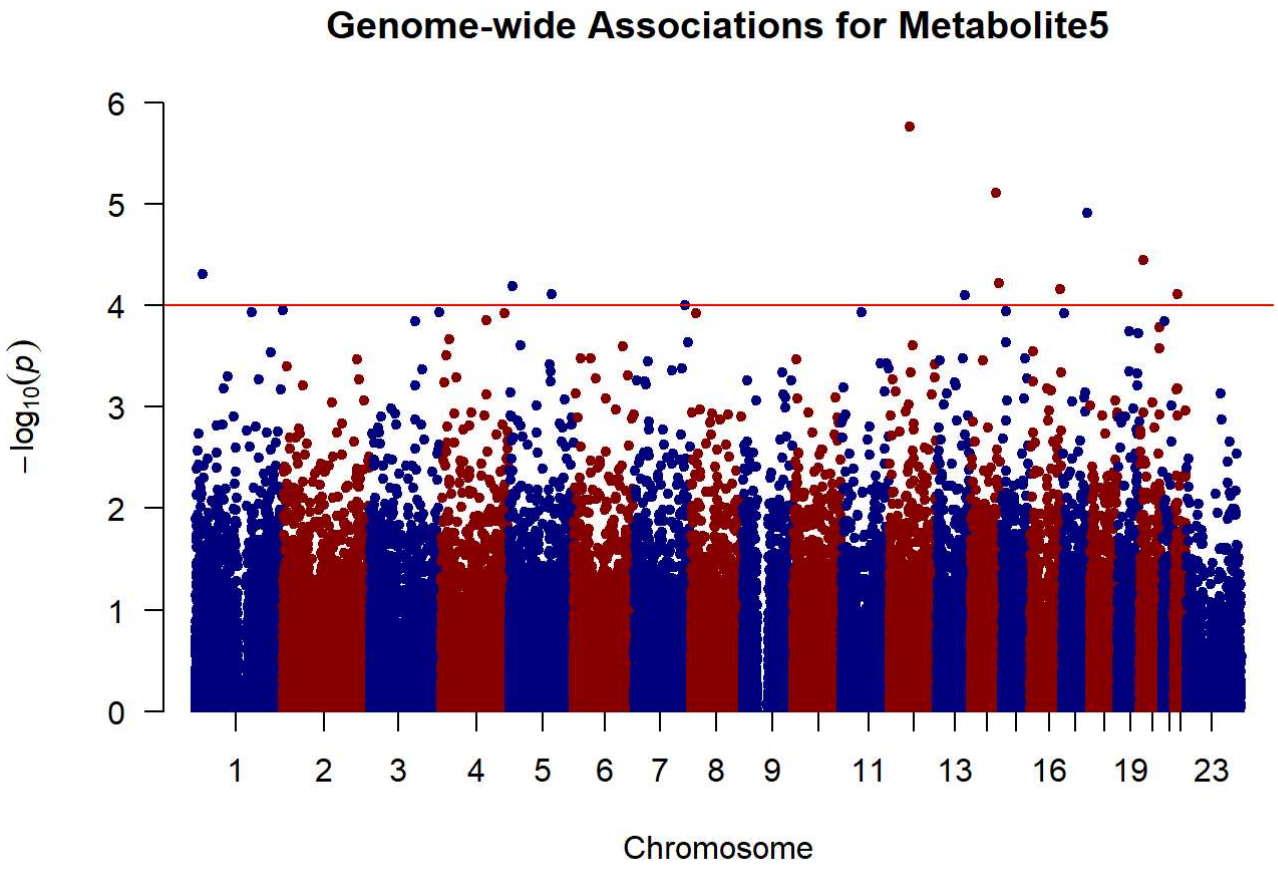

```
# Generate individual Manhattan plots for each metabolite
# This allows examination of association patterns specific to each metabolite
for(metabolite in metabolite_names) {
  # Subset data for the current metabolite
  metabolite_data <- manhattan_data[manhattan_data$metabolite == metabolite, ]

  # Create Manhattan plot for this specific metabolite
  manhattan(
    metabolite_data,
    col = c("navy", "darkred"),
    genomewideline = -log10(1e-4),
    suggestiveline = FALSE,
    main = paste("Genome-wide Associations for", metabolite)
  )
}
```

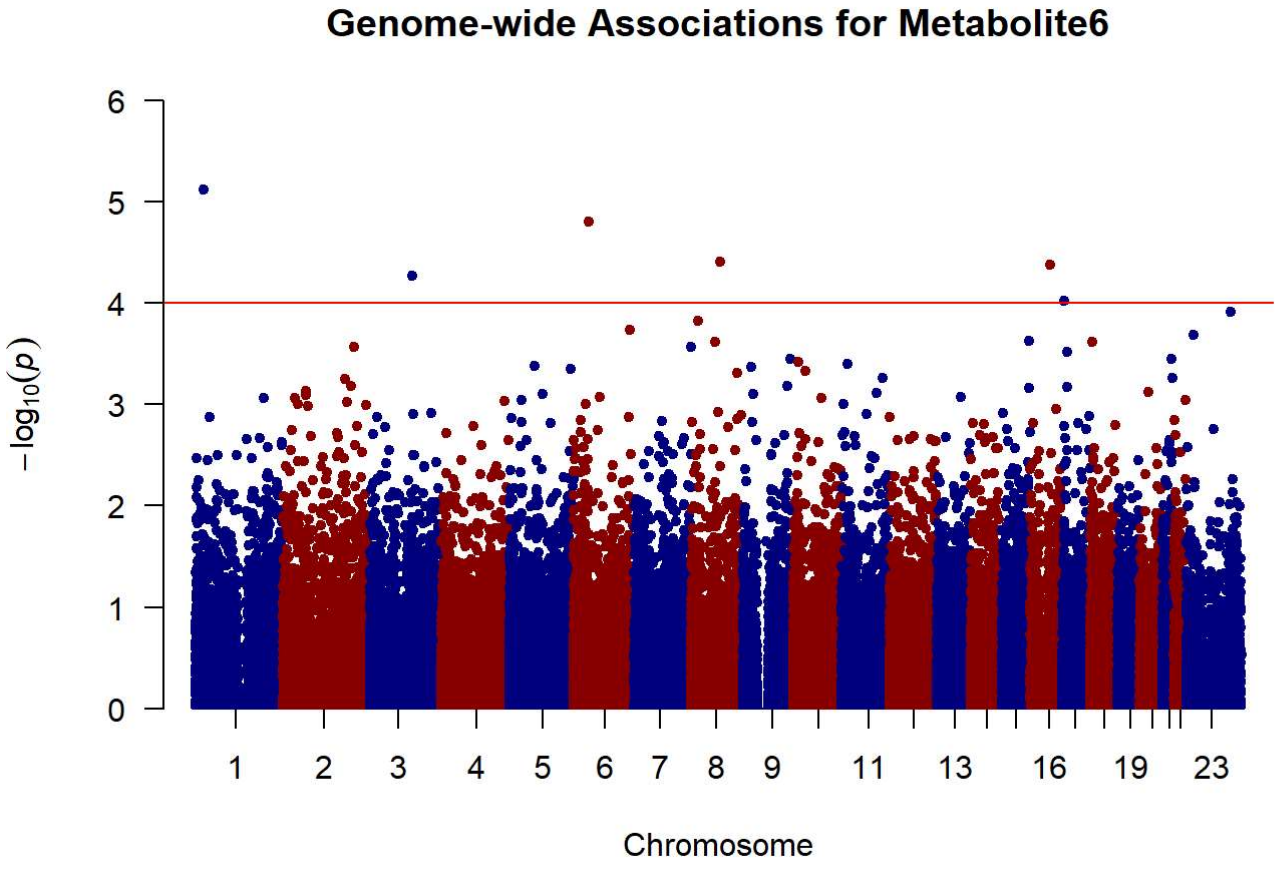

Genome-wide Associations for Metabolite1
−log10(p)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

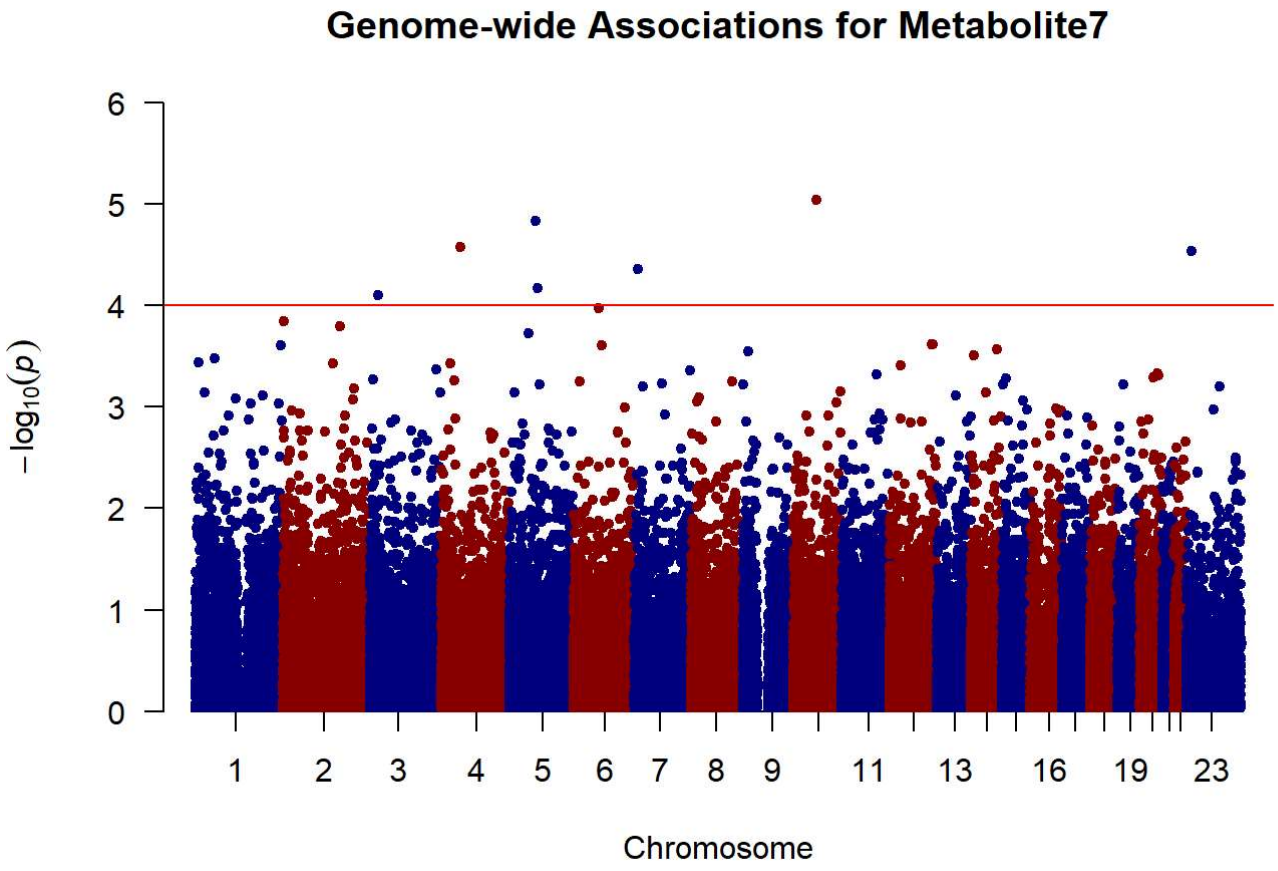

Genome-wide Associations for Metabolite2
−log10(p)
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

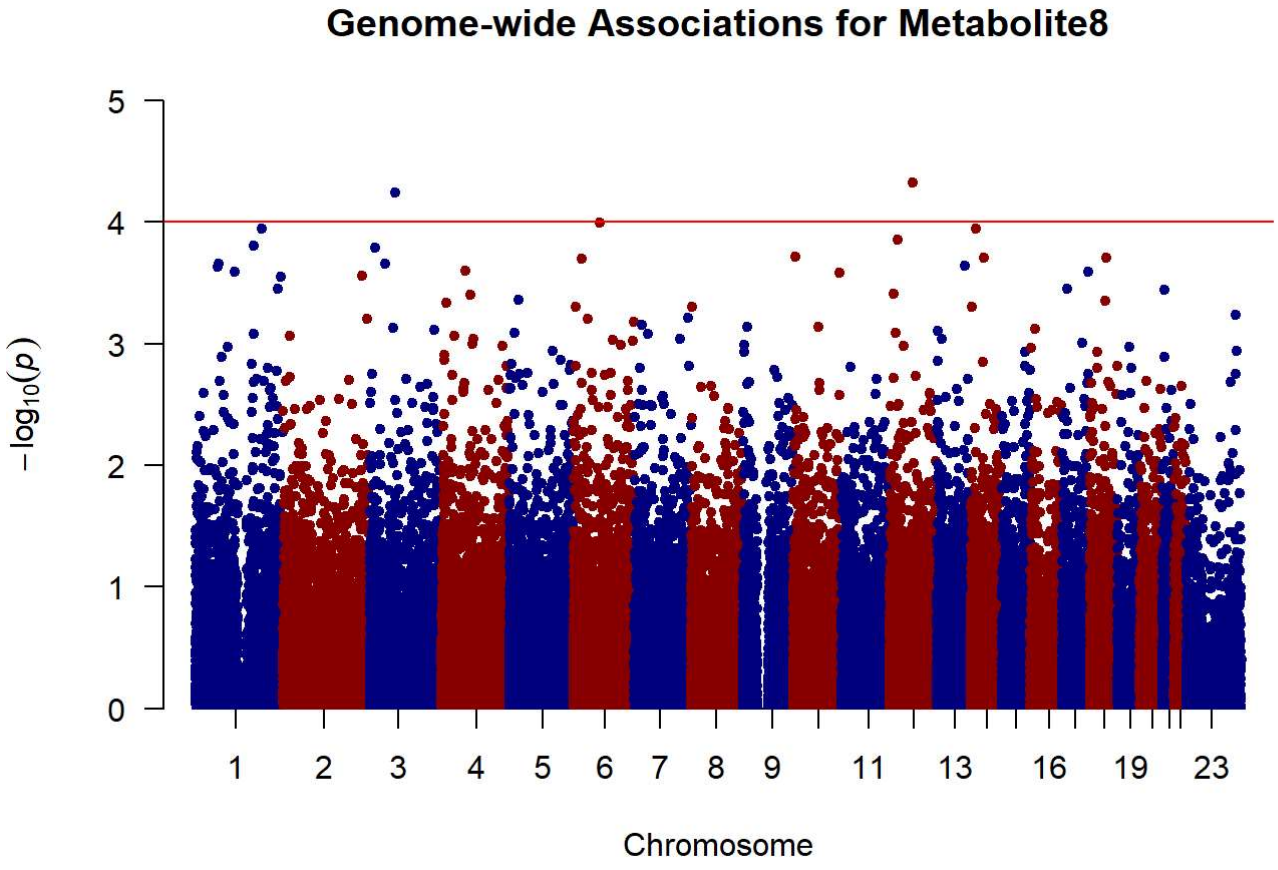

Genome-wide Associations for Metabolite3
$-\log_{10}(p)$
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

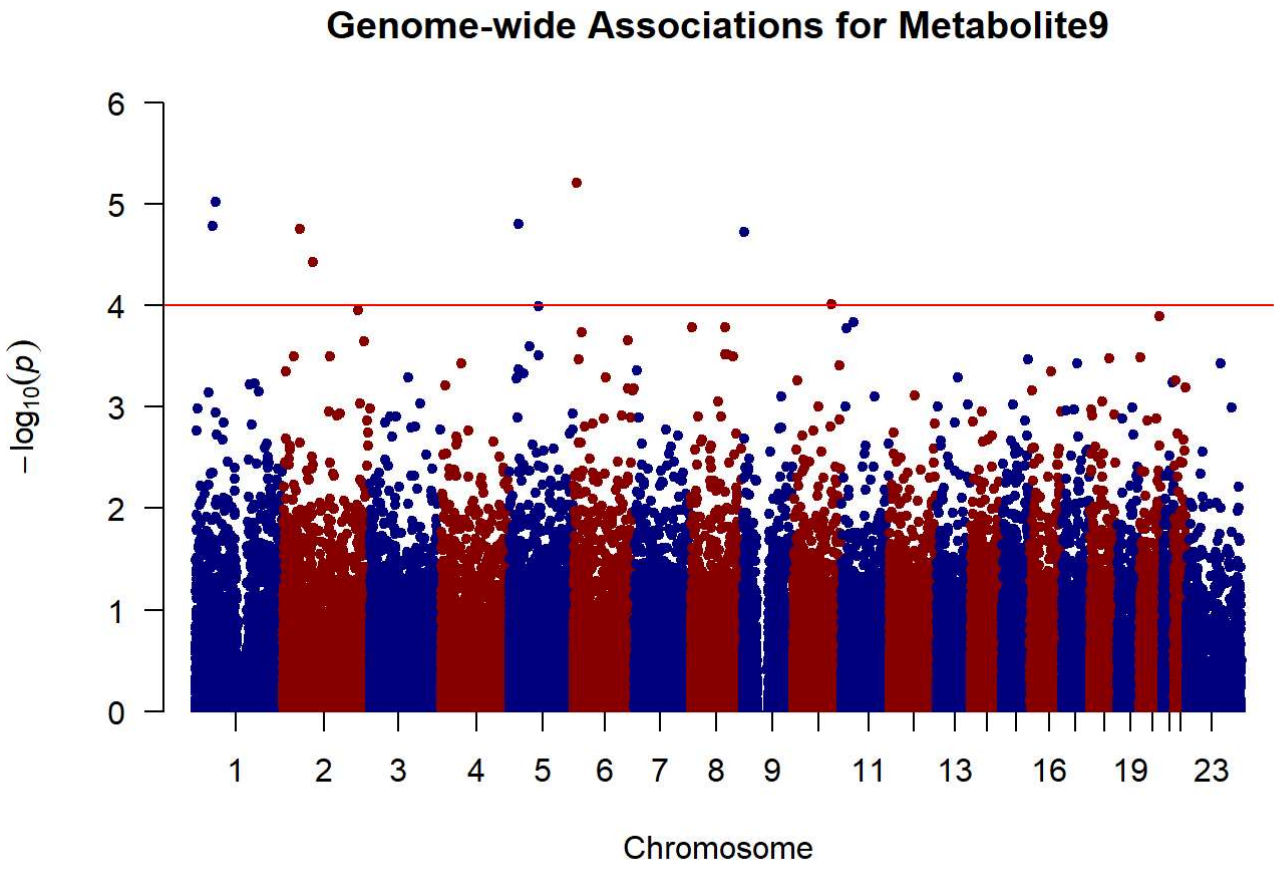

Genome-wide Associations for Metabolite4
$-\log_{10}(p)$
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

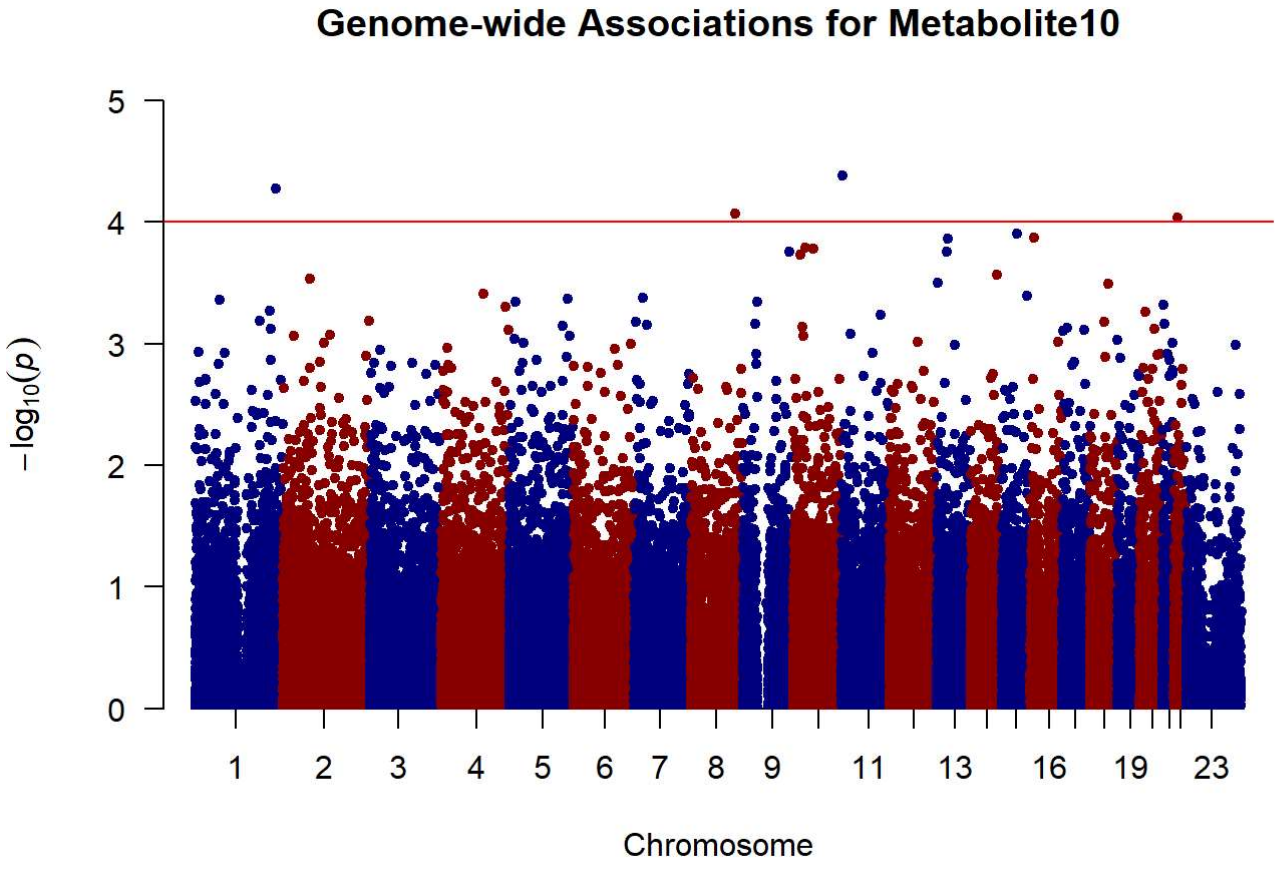

Genome-wide Associations for Metabolite5
−log10(p)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

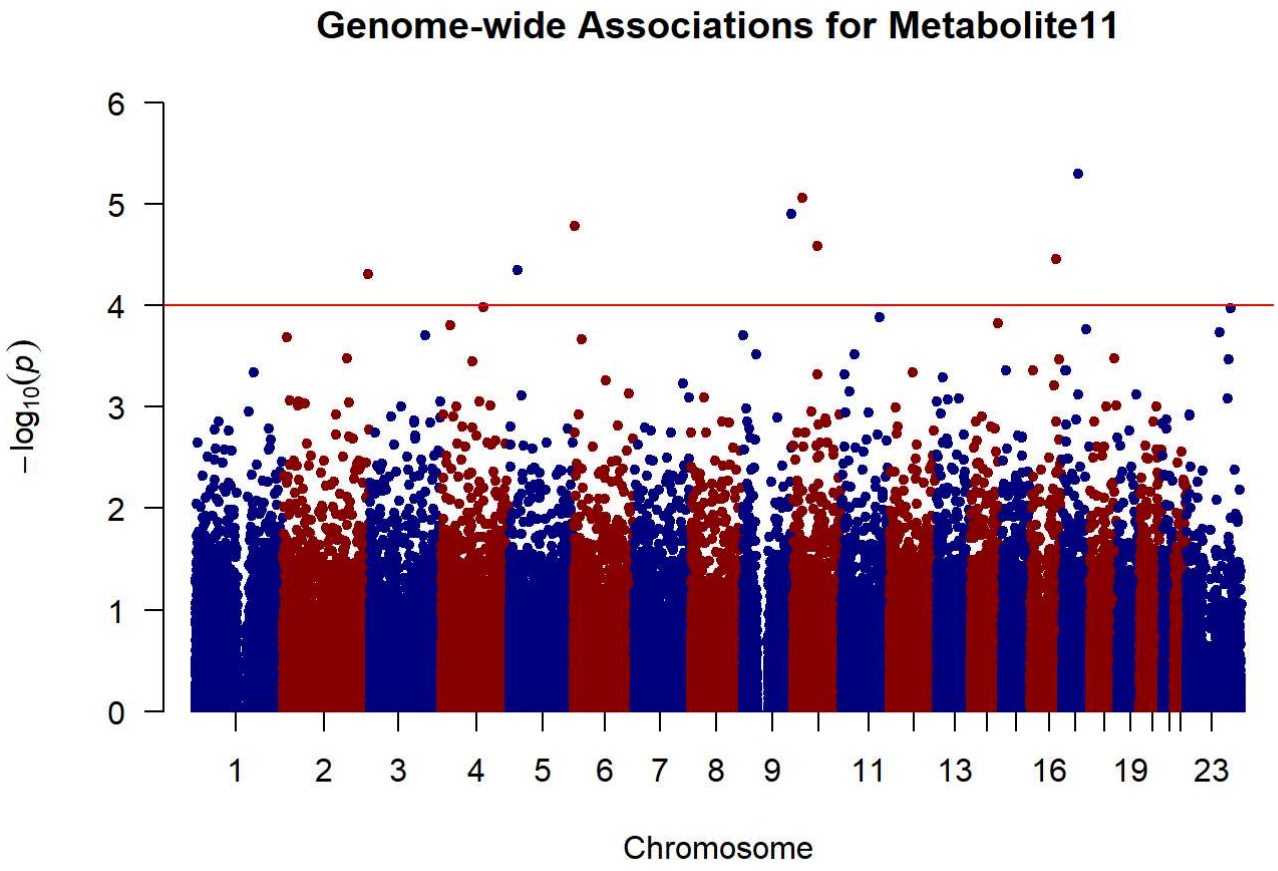

Genome-wide Associations for Metabolite6
−log10(p)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

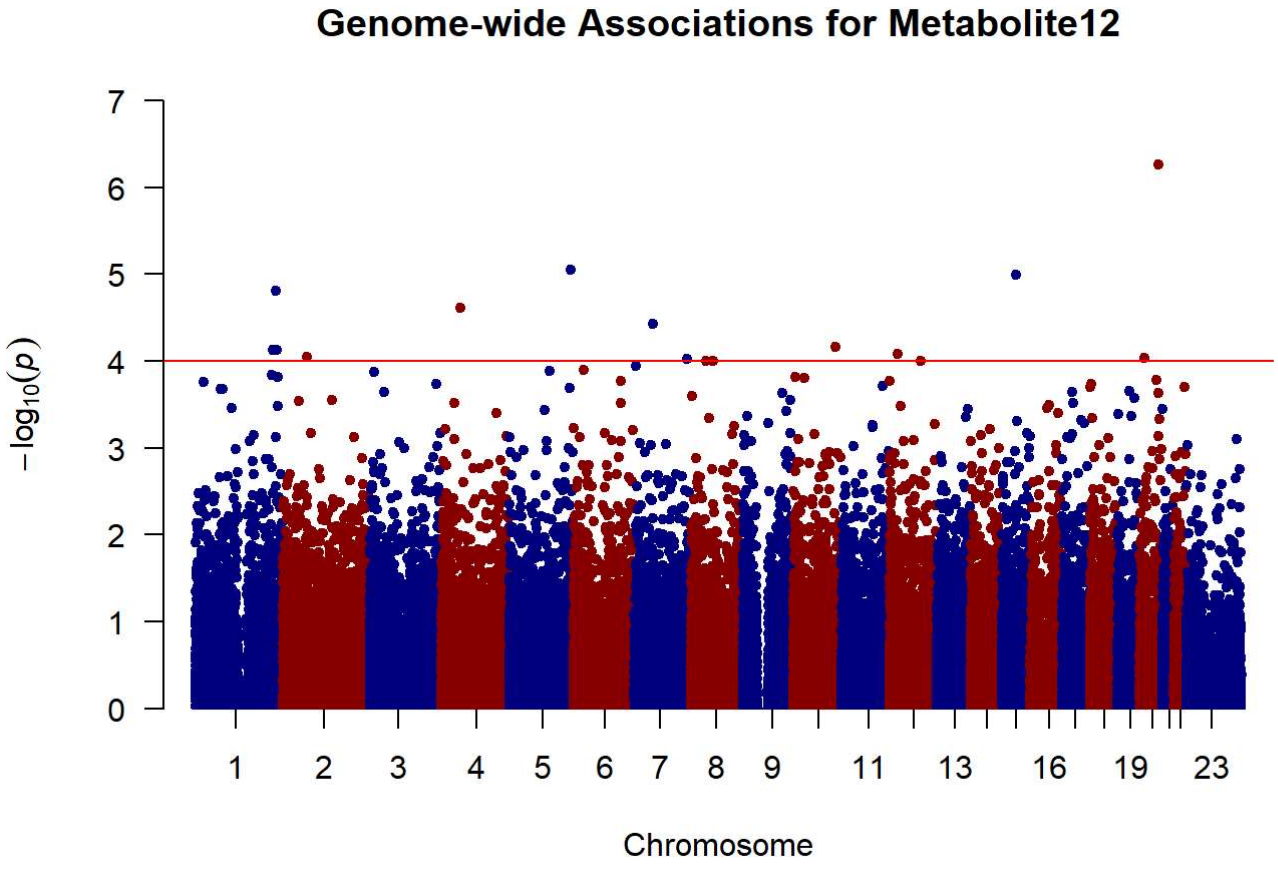

Genome-wide Associations for Metabolite7
$-\log_{10}(p)$
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

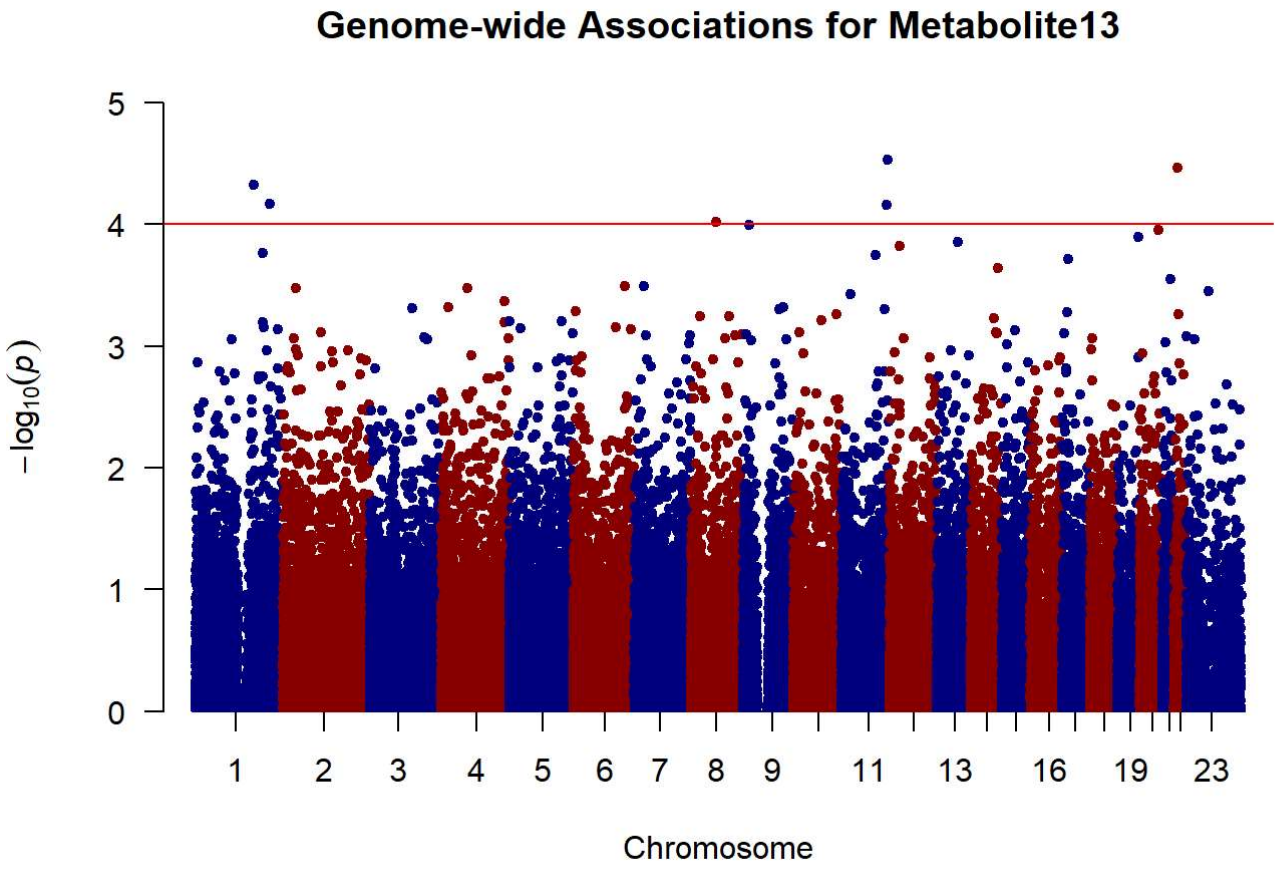

Genome-wide Associations for Metabolite8
$-\log_{10}(p)$
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

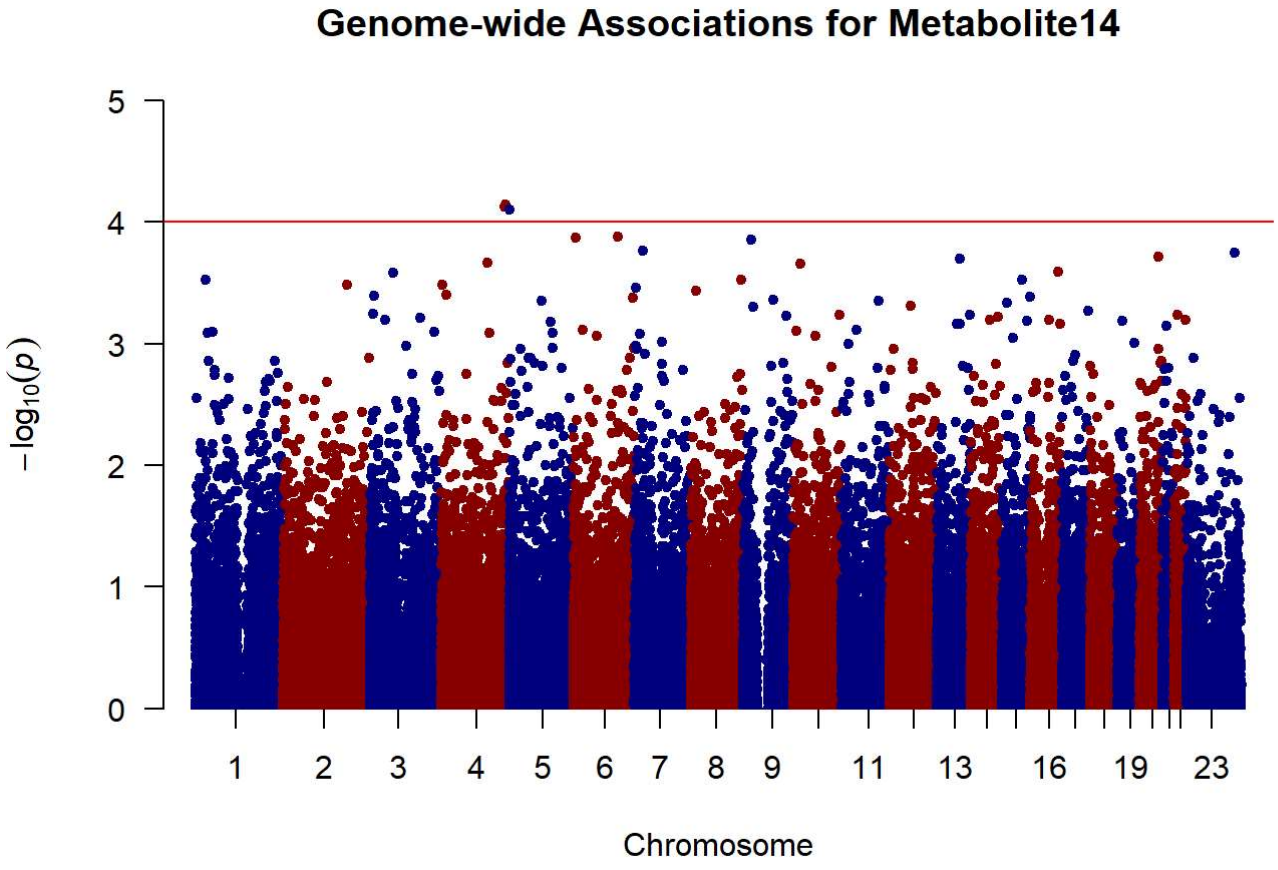

Genome-wide Associations for Metabolite9
$-\log_{10}(p)$
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

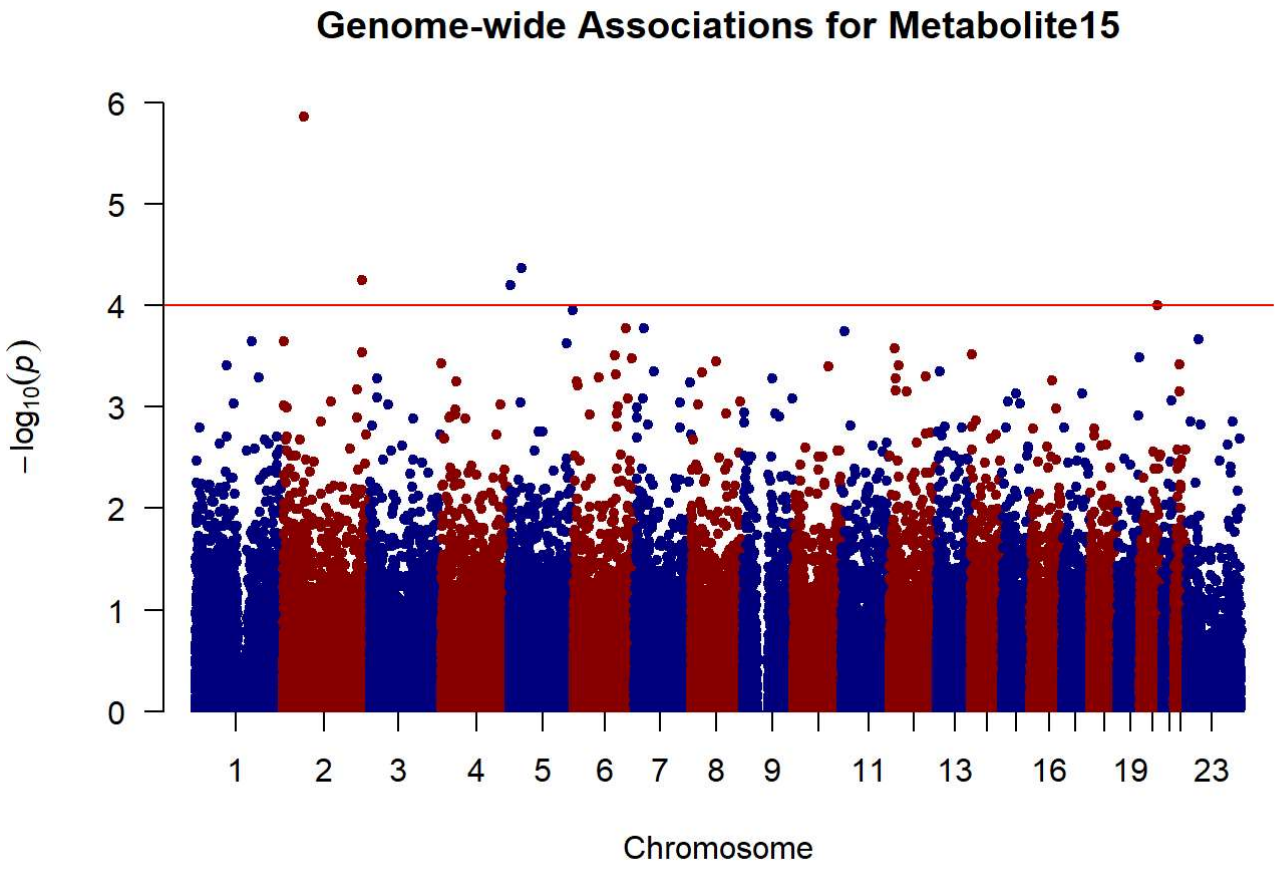

Genome-wide Associations for Metabolite10
$-\log_{10}(p)$
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

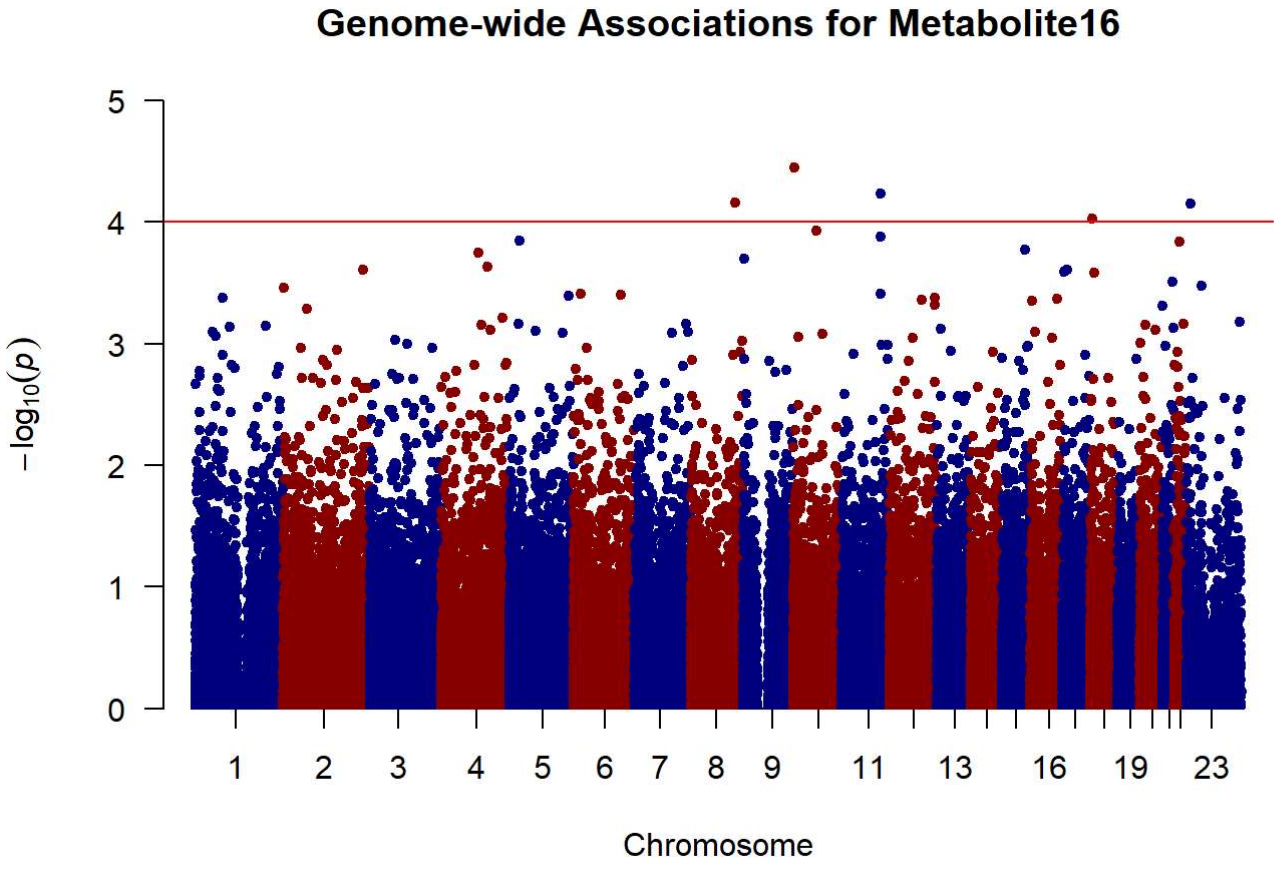

Genome-wide Associations for Metabolite11
−log10(p)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

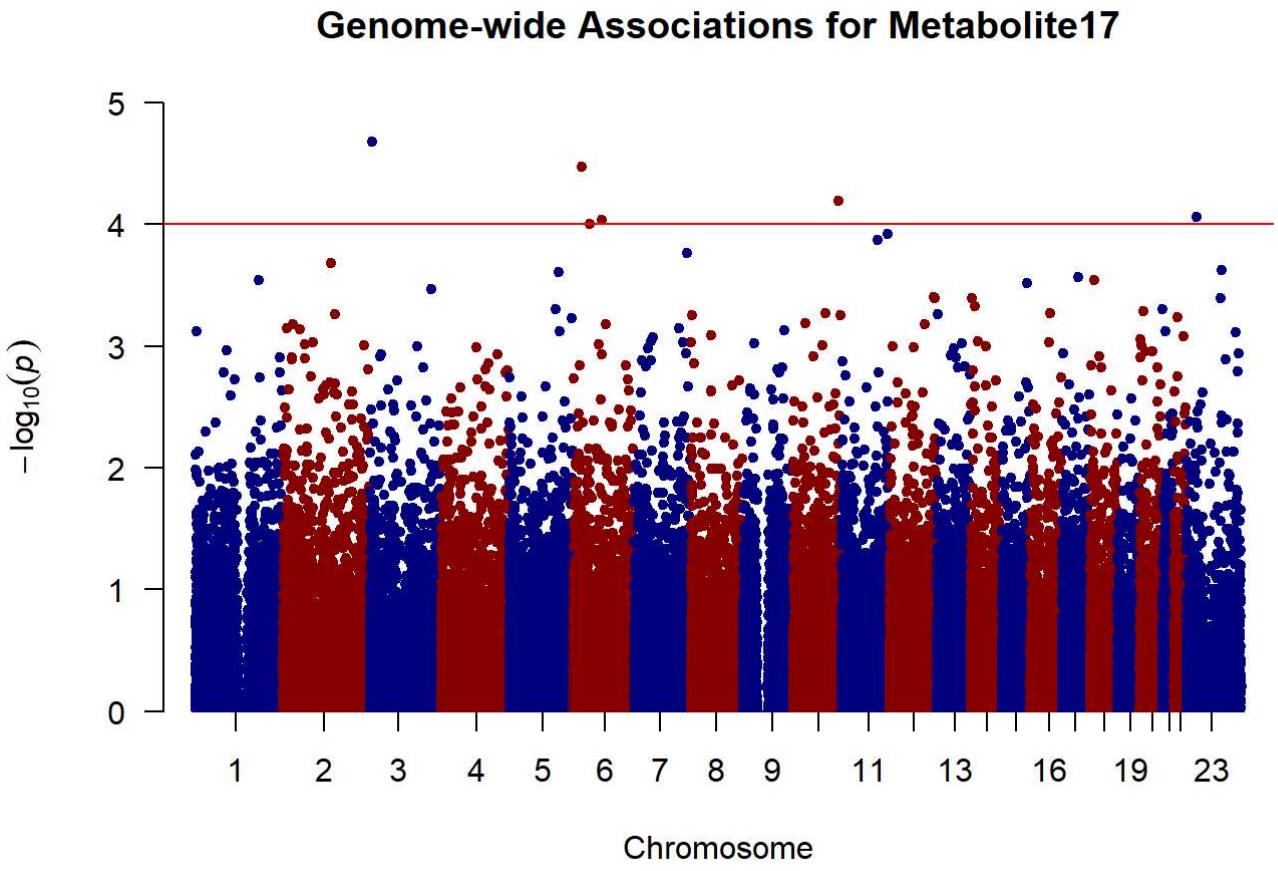

Genome-wide Associations for Metabolite12
−log10(p)
0
1
2
3
4
5
6
7
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

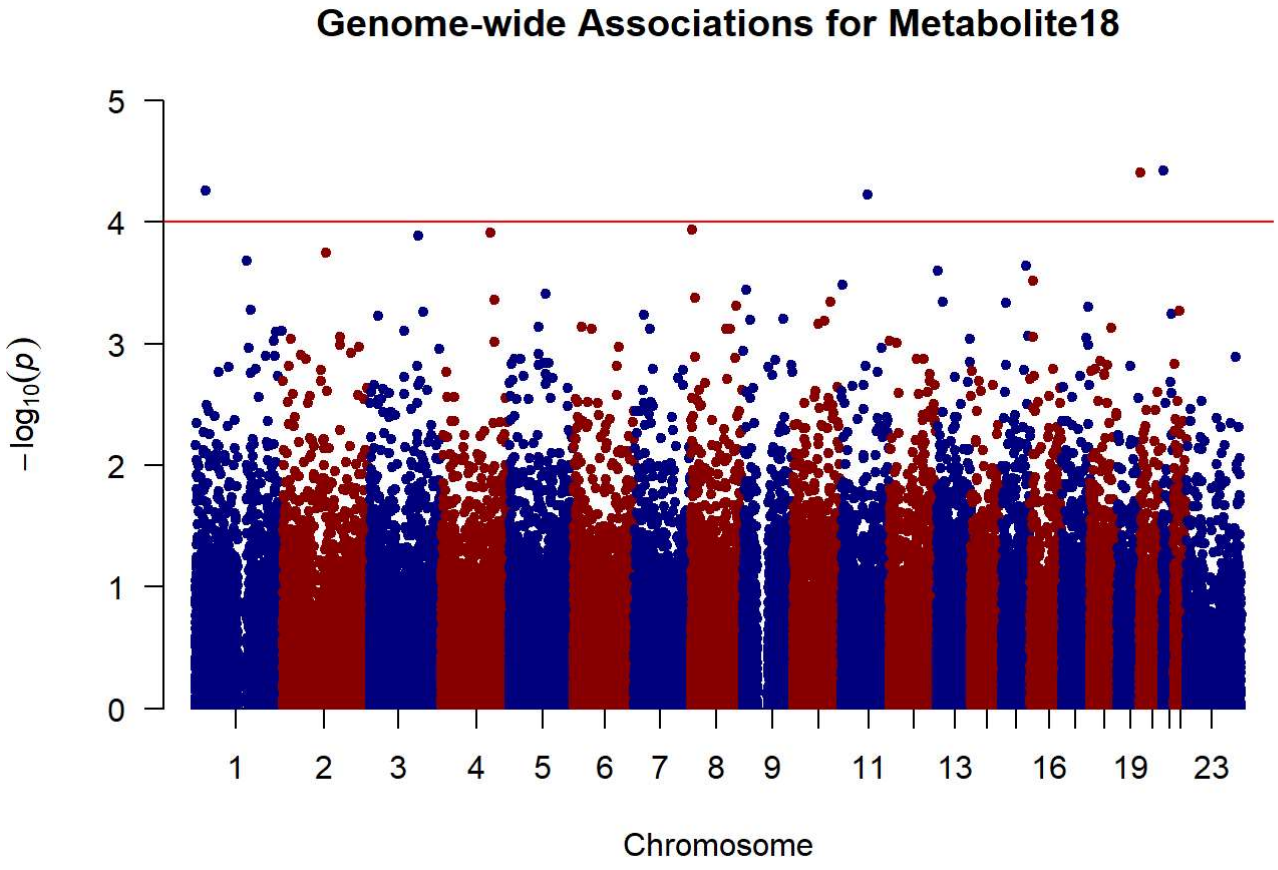

Genome-wide Associations for Metabolite13
−log10(p)
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

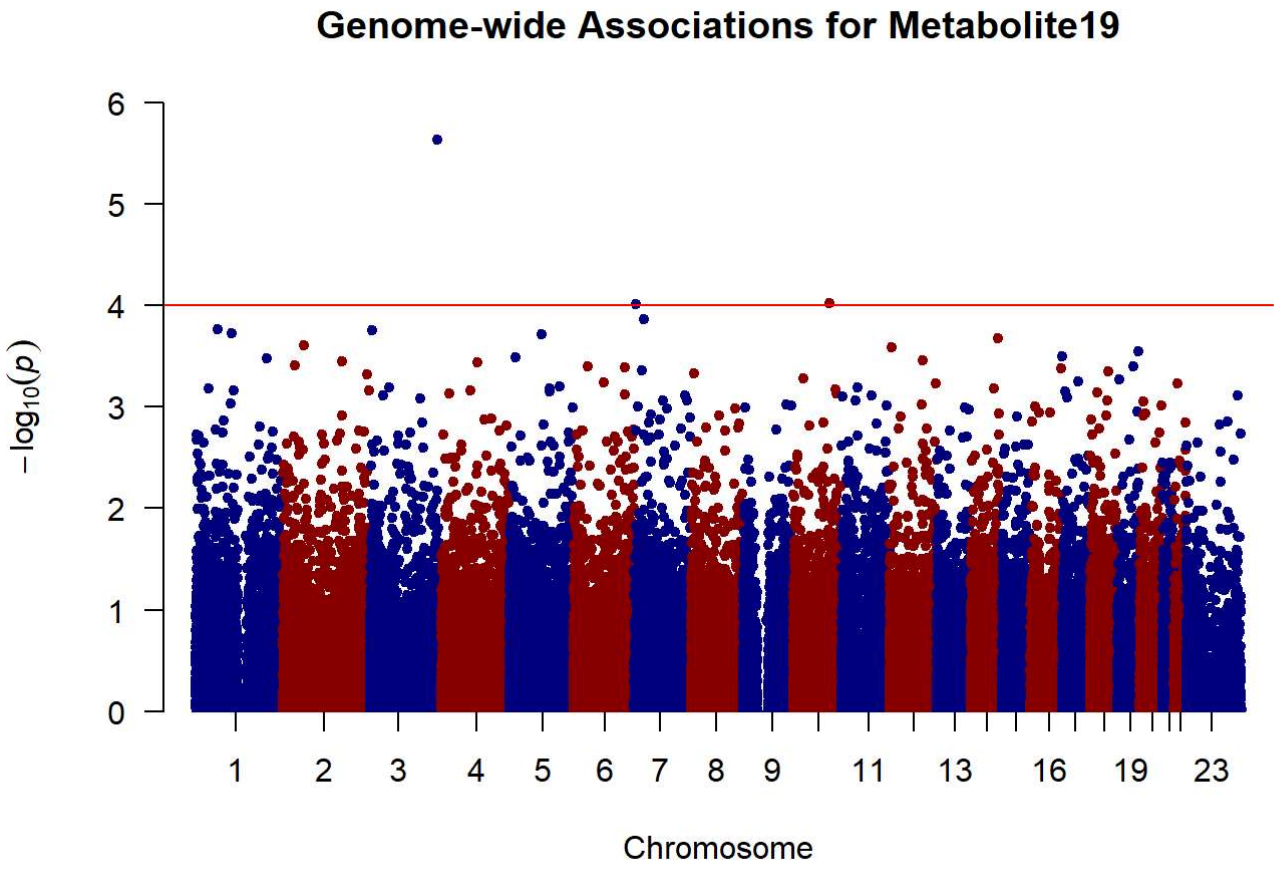

Genome-wide Associations for Metabolite14
−log10(p)
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

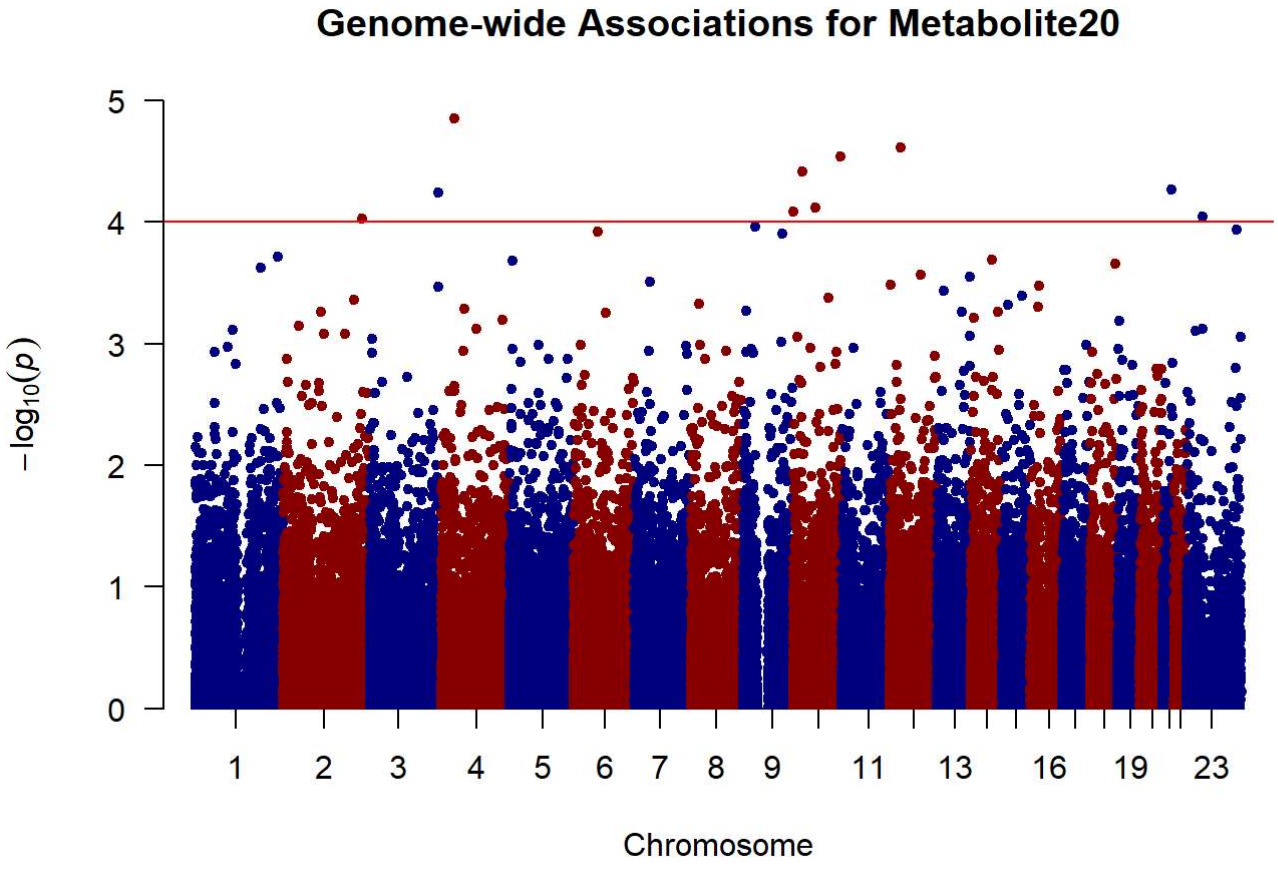

Genome-wide Associations for Metabolite15
−log10(p)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome


Genome-wide Associations for Metabolite16
−log10(p)
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome


Genome-wide Associations for Metabolite17
−log10(p)
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome


Genome-wide Associations for Metabolite18
−log10(p)
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome


Genome-wide Associations for Metabolite19
$-\log_{10}(p)$
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome


Genome-wide Associations for Metabolite20
$-\log_{10}(p)$
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
11
13
16
19
23
Chromosome

Task 5: Metabolic Networks

5.a: Connect to Cytoscape for Network Visualization

```
# Establish connection with Cytoscape software for network visualization
# This verifies that Cytoscape is running and accessible
cytoscapePing()
cytoscapeVersionInfo()
```

```
##       apiVersion cytoscapeVersion
##             "v1"         "3.10.3"
```

5.a & 5.b: Create Metabolite Network Based on Partial Correlations

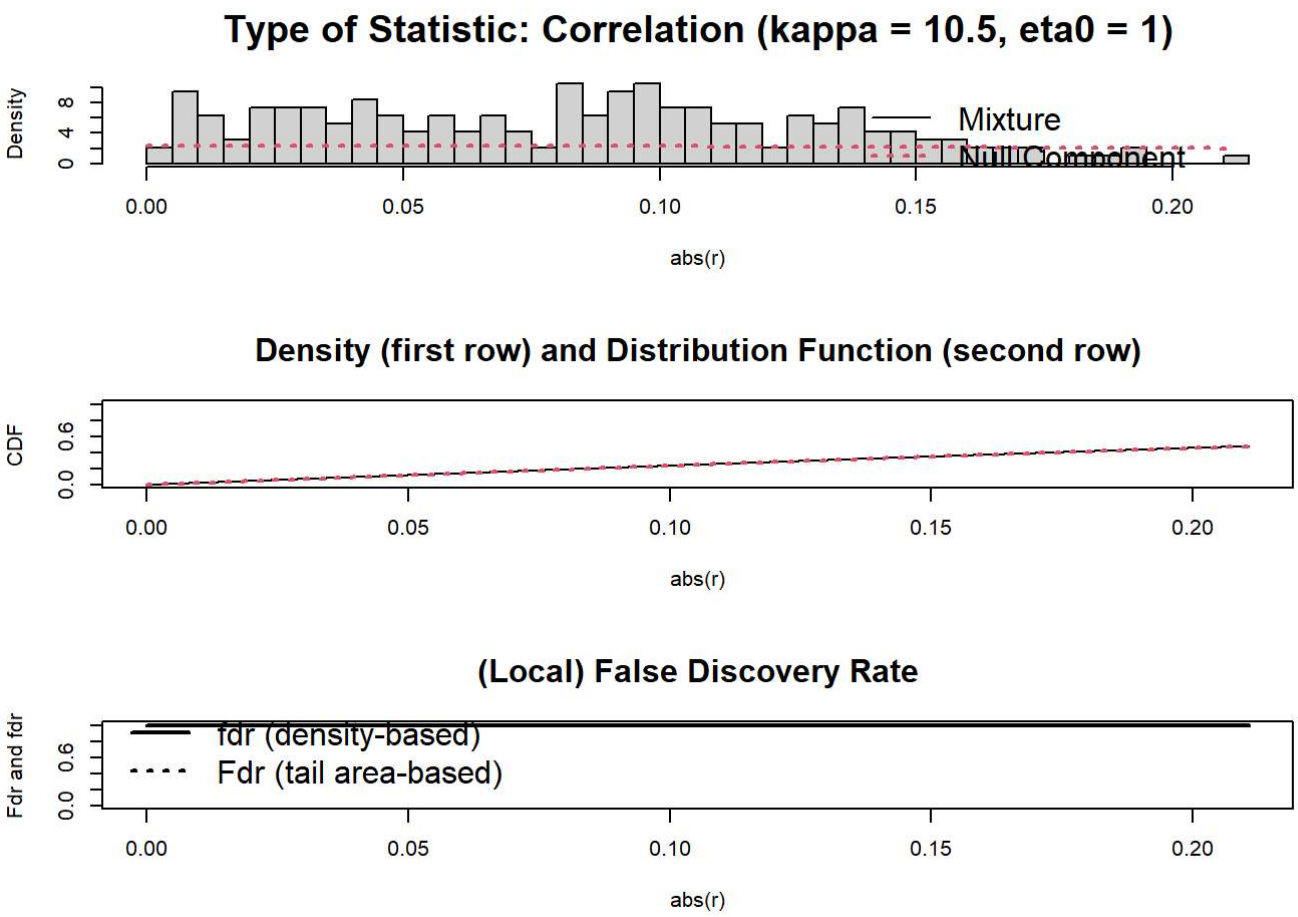
```
# Create matrix of fitted values (residuals) from null models
# These represent metabolite levels corrected for covariates and kinship
metabolite_values <- matrix(nrow = nrow(analysis_data),
                            ncol = length(metabolite_names))
rownames(metabolite_values) <- rownames(analysis_data)
colnames(metabolite_values) <- metabolite_names

# Extract fitted values from each metabolite's null model
for(metabolite in metabolite_names) {
  metabolite_values[, metabolite] <- null_models[[metabolite]]$fit$fitted.values
}

# Calculate partial correlations between metabolites using GeneNet
# Partial correlations account for the influence of all other metabolites
partial_correlations <- ggm.estimate.pcor(metabolite_values)
```

```
## Estimating optimal shrinkage intensity lambda (correlation matrix): 0.0333
```

```
# Test significance of each edge in the partial correlation network
correlation_tests <- network.test.edges(partial_correlations)
```

```
## Estimate (local) false discovery rates (partial correlations):
```

```
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
## Step 5... prepare for plotting
```




```
# Add absolute correlation values for sorting by correlation strength
correlation_tests$abs_pcor <- abs(correlation_tests$pcor)

# Select top 30 strongest metabolite-metabolite connections
# This creates a network with the most robust correlations
top_correlations <- correlation_tests[order(-correlation_tests$abs_pcor), ][1:30, ]

# Save the significant correlations to CSV file
write.csv(top_correlations, "top_partial_correlations.csv", row.names = FALSE)
```

5.c: Visualize Metabolic Network in Cytoscape

```
# Connect to Cytoscape for network visualization
cytoscapePing()

# Prepare network edge data from metabolite correlations
edges <- data.frame(
  source = paste0("meta", top_correlations$node1),
  target = paste0("meta", top_correlations$node2),
  weight = top_correlations$pcor,            # Correlation strength
  interaction = "interacts with",            # Relationship type
  pval = top_correlations$pval,              # Statistical significance
  qval = top_correlations$qval               # Multiple testing correction
)

# Create node list from unique metabolite identifiers
nodes <- data.frame(
  id = unique(c(edges$source, edges$target)),
  name = unique(c(edges$source, edges$target)),
  stringsAsFactors = FALSE
)

# Create network in Cytoscape from nodes and edges
createNetworkFromDataFrames(
  nodes = nodes,
  edges = edges,
  title = "Metabolic Network",
  collection = "Metabolite Correlations"
)
```

```
## networkSUID
##         872
```

```
# Apply force-directed layout algorithm for optimal node placement
layoutNetwork("force-directed")

# Format node appearance
setNodeShapeDefault("ELLIPSE")          # Use elliptical nodes
setNodeSizeDefault(30)                  # Set uniform node size
setNodeLabelMapping("name")             # Display metabolite names on nodes
```

```
## NULL
```

```
# Set edge width mapping based on correlation strength
# Stronger correlations are represented by thicker lines
setEdgeLineWidthMapping(
  table.column = "weight",
  table.column.values = seq(min(abs(edges$weight)), max(abs(edges$weight)), length.out = 5),
  widths = seq(1, 3, length.out = 5),
  mapping.type = "continuous"
)
```

```
## NULL
```

```
# Set edge color mapping based on correlation direction
# Positive correlations in blue, negative in orange, neutral in gray
setEdgeColorMapping(
  table.column = "weight",
  table.column.values = c(min(edges$weight), 0, max(edges$weight)),
  colors = c("#ff7b00", "#555555", "#00b4ff"),  # Orange, Gray, Blue
  mapping.type = "continuous"
)
```

```
## NULL
```

5.d: Network Analysis and Metrics Reporting

```
# Run network analysis algorithms in Cytoscape
# This computes centrality measures and other network properties
analyzeNetwork()
```

```
##                     networkTitle                         nodeCount
## "Metabolic Network (undirected)"                              "19"
##                         edgeCount                       avNeighbors
##                              "30"                          "3.375"
##                          diameter                           radius
##                               "5"                              "3"
##                             avSpl                                cc
##                               "2.5"                         "0.51875"
##                           density                      heterogeneity
##                             "0.225"             "0.4040263746161375"
##                    centralization                               ncc
##                               "0.2"                              "2"
##                              time
##                             "0.001"
```

```
# Retrieve computed network metrics for all nodes
node_metrics <- getTableColumns(table = "node")

# Export network metrics to CSV for further analysis
write.csv(node_metrics, "metabolite_network_node_metrics.csv", row.names = FALSE)

# Save the Cytoscape session for future reference
saveSession("Metabolite_Network_Analysis.cys")

# Display key network centrality metrics for the most important nodes
# This reveals which metabolites are most central in the network
print(head(node_metrics[, c("name", "Degree", "ClosenessCentrality", "BetweennessCentrality")]))
```

```
##       name Degree ClosenessCentrality BetweennessCentrality
## 907  meta8      5           0.4285714            0.17619048
## 910  meta1      4           0.4687500            0.11460317
## 913  meta6      3           0.5172414            0.25238095
## 916  meta7      4           0.3947368            0.01777778
## 919  meta3      4           0.3947368            0.01777778
## 922 meta11      6           0.5357143            0.37841270
```

The network analysis reveals the most central metabolites in the metabolic network based on three key centrality measures:

- Degree: The number of direct connections a metabolite has, indicating its immediate influence
- Closeness Centrality: How close a metabolite is to all others, indicating its efficiency in information transfer = Betweenness Centrality: How often a metabolite lies on shortest paths between other metabolites, indicating its role as a mediator

Task 6: Annotate Significant SNPs

6.a: Extract Top 20 Significant SNPs

```
# Extract top 20 most significant SNPs
top_variants <- significant_hits[order(significant_hits$Score.pval), ]
top_variants <- top_variants[!duplicated(top_variants$variant.id), ]
top_variants[top_variants$chr == "X", "chr"] <- 23
top_variants <- top_variants[1:20, ]

# Display summary of top SNPs
knitr::kable(top_variants[, c("variant.id", "chr", "position", "Score.pval", "metabolite")],
             caption = "Top 20 significant SNP-metabolite associations")
```

Top 20 significant SNP-metabolite associations

	variant.id	chr	position	Score.pval	metabolite
128	63403	20	51274041	5.00e-07	Metabolite12
17	6889	2	59586461	1.40e-06	Metabolite15
108	45151	12	58094775	1.70e-06	Metabolite5
31	13919	3	168594083	1.70e-06	Metabolite1
32	14408	3	188814284	2.30e-06	Metabolite19
81	35247	9	72586910	3.00e-06	Metabolite1
121	57575	17	44896002	5.10e-06	Metabolite11
113	51174	14	79120259	5.70e-06	Metabolite1
55	23555	6	10568858	6.20e-06	Metabolite9
2	683	1	24508787	7.60e-06	Metabolite6
114	51437	14	89266760	7.70e-06	Metabolite5
88	37919	10	25189905	8.60e-06	Metabolite11
57	24272	6	36843458	8.80e-06	Metabolite1
52	22834	5	172792898	9.00e-06	Metabolite12
92	38680	10	65373297	9.20e-06	Metabolite7
5	1372	1	59391181	9.60e-06	Metabolite9
116	53008	15	59158914	1.02e-05	Metabolite12
122	58265	17	71181116	1.22e-05	Metabolite5
83	36805	9	134294347	1.25e-05	Metabolite11

	variant.id	chr	position	Score.pval	metabolite
34	15790	4	38587186	1.42e-05	Metabolite20

6.b: Prepare Files for Annotation

```
# Extract these SNPs from the original files
ped_file <- read.table("data/Qatari156.ped", header = FALSE, stringsAsFactors = FALSE)
map_file <- read.table("data/Qatari156.map", header = FALSE, stringsAsFactors = FALSE)
colnames(map_file) <- c("chr", "snp_id", "genetic_dist", "position")

# Find indexes of top SNPs
top_snp_indices <- which(map_file$position %in% top_variants$pos &
                         map_file$chr %in% top_variants$chr)
top_map <- map_file[top_snp_indices, ]

# Extract corresponding columns from PED file
ped_columns <- sort(unlist(lapply(top_snp_indices,
                                  function(i) c(6 + 2*i - 1, 6 + 2*i))))
top_ped <- ped_file[, c(1:6, ped_columns)]

# Save subset files for the top SNPs
write.table(top_ped, file = "top20_variants.ped",
            sep = "\t", row.names = FALSE, col.names = FALSE, quote = FALSE)
write.table(top_map, file = "top20_variants.map",
            sep = "\t", row.names = FALSE, col.names = FALSE, quote = FALSE)
```

Convert to Vcf

```
plink --file top20_variants --recode vcf --out top_20_Qatari
``


## 6.c: Annotate Vcf with ANNOVAR
```

```
# Create input file for Annovar
annovar_input <- data.frame(
  Chr = top_variants$chr,
  Start = top_variants$position,
  End = top_variants$position,
  Ref = top_variants$alleleA,
  Alt = top_variants$alleleB,
  SNP_ID = top_variants$variant.id
)

# Save Annovar input file
write.table(annovar_input, "annovar_input.txt",
             sep = "\t", quote = FALSE, row.names = FALSE, col.names = FALSE)
```

6.d: Annotate SNPs using Annovar

```
./table_annovar.pl top_20_Qatari.vcf humandb/ \
  -buildver hg19 \
  -out annotated_output \
  -remove \
  -protocol refGene \
  -operation g \
  -nastring . \
  -vcfinput \
  -polish \
  -thread 8
```

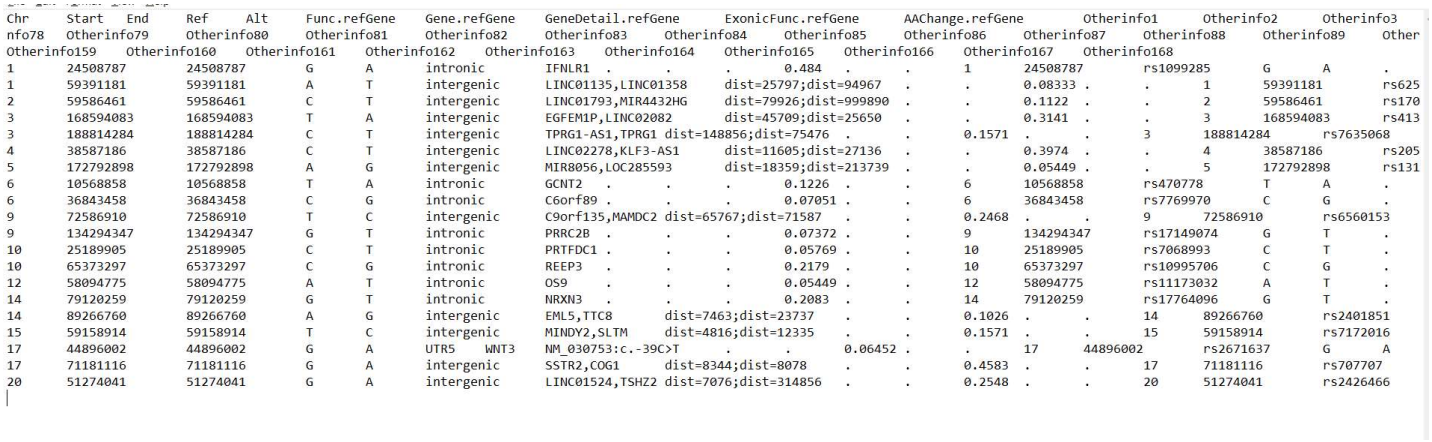
Annotation Process Explanation

This command performed the following operations:

1. **Input file**: `top_20_Qatari.vcf` - Our top 20 significant SNPs in VCF format
2. **Reference database**: `humandb/` - Directory containing ANNOVAR's reference databases
3. **Key parameters**:
    - `-buildver hg19` - Using human genome build 19 as the reference
    - `-out annotated_output` - Output file prefix name
    - `-remove` - Remove temporary files after completion
    - `-protocol refGene` - Using the RefSeq gene annotation database
    - `-operation g` - Perform gene-based annotation
    - `-nastring .` - Use "." to represent missing values
    - `-vcfinput` - Input is in VCF format
    - `-polish` - Generate a well-formatted output file
    - `-thread 8` - Use 8 CPU threads for parallel processing
4. **Annotation details**: The command searched for overlaps between our variants and:
    - Coding exons
    - UTRs (untranslated regions)
    - Introns
    - Intergenic regions
    - Splice sites
5. **Outputs generated**:
    - `annotated_output.hg19_multianno.txt` - Tab-delimited text file with annotations
    - `annotated_output.hg19_multianno.vcf` - VCF file with annotations in the INFO field

The annotation process successfully identified the genomic context of each variant, including: - Gene location (if applicable) - Functional effect (intronic, intergenic, UTR, etc.) - Distance to nearest genes (for intergenic variants) - Transcript information - Potential effects on protein structure and function

This annotation step provides crucial biological context for our statistical associations, helping to interpret how these genetic variants might influence metabolite levels through their effects on gene function or regulation.



Chr	Start	End	Ref	Alt	Func.refGene	Gene.refGene	GeneDetail.refGene	ExonicFunc.refGene	AAChange.refGene	Otherinfo1	Otherinfo2	Otherinfo3	Otherinfo4	Otherinfo5	Otherinfo6	Otherinfo7	Otherinfo8	Otherinfo9	Otherinfo10	Otherinfo11	Otherinfo12	Otherinfo13	Otherinfo14	Otherinfo15	Otherinfo16	Otherinfo17	Otherinfo18
1	24508787	24508787	G	A	intronic	IFNLR1	.	.	0.484	.	.	1	24508787	rs1099285	G	A	.	.	.	.	.	.	.	.	.	.	.
1	59391181	59391181	A	T	intergenic	LINC01135,LINC01358	dist=25797;dist=94967	.	.	0.08333	.	.	1	59391181	rs625	.	.	.	.	.	.	.	.	.	.	.	.
2	59586461	59586461	C	T	intergenic	LINC01793,MIR4432HG	dist=79926;dist=999890	.	.	0.1122	.	.	2	59586461	rs170	.	.	.	.	.	.	.	.	.	.	.	.
3	168594083	168594083	T	A	intergenic	EGFEM1P,LINC02082	dist=45709;dist=25650	.	.	0.3141	.	.	3	168594083	rs413	.	.	.	.	.	.	.	.	.	.	.	.
3	188814284	188814284	C	T	intergenic	TPRG1-AS1,TPRG1	dist=148856;dist=75476	.	.	0.1571	.	.	3	188814284	rs7635068	.	.	.	.	.	.	.	.	.	.	.	.
4	38587186	38587186	C	T	intergenic	LINC02278,KLF3-AS1	dist=11605;dist=27136	.	.	0.3974	.	.	4	38587186	rs205	.	.	.	.	.	.	.	.	.	.	.	.
5	172792898	172792898	A	G	intergenic	MIR8056,LOC285593	dist=18359;dist=213739	.	.	0.05449	.	.	5	172792898	rs131	.	.	.	.	.	.	.	.	.	.	.	.
6	10568858	10568858	T	A	intronic	GCNT2	.	.	0.1226	.	.	6	10568858	rs470778	T	A	.	.	.	.	.	.	.	.	.	.	.
6	36843458	36843458	C	G	intronic	C6orf89	.	.	0.07051	.	.	6	36843458	rs7769970	C	G	.	.	.	.	.	.	.	.	.	.	.
9	72586910	72586910	T	C	intergenic	C9orf135,MAMDC2	dist=65767;dist=71587	.	.	0.2468	.	.	9	72586910	rs6560153	.	.	.	.	.	.	.	.	.	.	.	.
9	134294347	134294347	G	T	intronic	PRRC2B	.	.	0.07372	.	.	9	134294347	rs17149074	G	T	.	.	.	.	.	.	.	.	.	.	.
10	25189905	25189905	C	T	intronic	PRTFDC1	.	.	0.05769	.	.	10	25189905	rs7068993	C	T	.	.	.	.	.	.	.	.	.	.	.
10	65373297	65373297	C	G	intronic	REEP3	.	.	0.2179	.	.	10	65373297	rs10995706	C	G	.	.	.	.	.	.	.	.	.	.	.
12	58094775	58094775	A	T	intronic	OS9	.	.	0.05449	.	.	12	58094775	rs11173032	A	T	.	.	.	.	.	.	.	.	.	.	.
14	79120259	79120259	G	T	intronic	NRXN3	.	.	0.2083	.	.	14	79120259	rs17764096	G	T	.	.	.	.	.	.	.	.	.	.	.
14	89266760	89266760	A	G	intergenic	EML5,TTC8	dist=7463;dist=23737	.	.	0.1026	.	.	14	89266760	rs2401851	.	.	.	.	.	.	.	.	.	.	.	.
15	59158914	59158914	T	C	intergenic	MINDY2,SLTM	dist=4816;dist=12335	.	.	0.1571	.	.	15	59158914	rs7172016	.	.	.	.	.	.	.	.	.	.	.	.
17	44896002	44896002	G	A	UTR5	WNT3	NM_030753:c.-39C>T	.	.	0.06452	.	.	17	44896002	rs2671637	G	A	.	.	.	.	.	.	.	.	.	.
17	71181116	71181116	G	A	intergenic	SSTR2,COG1	dist=8344;dist=8078	.	.	0.4583	.	.	17	71181116	rs707707	.	.	.	.	.	.	.	.	.	.	.	.
20	51274041	51274041	G	A	intergenic	LINC01524,TSHZ2	dist=7076;dist=314856	.	.	0.2548	.	.	20	51274041	rs2426466	.	.	.	.	.	.	.	.	.	.	.	.

Annovar output

Functional Distribution of Significant SNPs

The majority of our significant SNPs (65%) are located in intergenic regions, while 30% are found within gene introns. Only one variant (5%) is located in a 5’ untranslated region (UTR5).

Functional Analysis of Individual SNPs

1. rs11173032 (Chr 12)

This variant is located within an intron of the OS9 gene. OS9 encodes a protein involved in endoplasmic reticulum-associated degradation (ERAD) of misfolded proteins. Intronic variants can potentially affect gene expression through various mechanisms, including alternative splicing or by affecting regulatory elements within the intron.

2. rs41350745 (Chr 3)

This intergenic variant is positioned between the EGFEM1P pseudogene and the LINC02082 non-coding RNA gene. It is located 45,709 bp from EGFEM1P and 25,650 bp from LINC02082. While intergenic variants are often more challenging to functionally characterize, they may influence gene regulation through long-range enhancer or silencer activities.

3. rs17050199 (Chr 2)

Located in an intergenic region between LINC01793 and MIR4432HG, this variant is 79,926 bp from LINC01793 and nearly 1 Mb from MIR4432HG. Both of these genes produce non-coding RNAs, which play various roles in gene regulation.

4. rs2426466 (Chr 20)

This intergenic variant is positioned between LINC01524 (a long non-coding RNA) and TSHZ2 (a transcription factor). It is relatively close to LINC01524 (7,076 bp) but quite distant from TSHZ2 (314,856 bp). Its proximity to LINC01524 suggests a potential regulatory relationship.

5. rs7635068 (Chr 3)

Located in an intergenic region between TPRG1-AS1 (an antisense RNA) and TPRG1, this variant may affect the regulation of these genes. The relatively large distances from both genes (148,856 bp and 75,476 bp, respectively) suggest it may function as a distal regulatory element.

Additional Notable Annotations

1. **rs2671637 (Chr 17)** - This is the only variant in our top set located in a 5' UTR, specifically in the WNT3 gene (NM_030753:c.-39C>T). Variants in 5' UTRs can affect mRNA stability, translation efficiency, and overall gene expression. WNT3 is involved in the Wnt signaling pathway, which plays roles in development, cell differentiation, and metabolism.
2. **Gene-rich regions** - Several of our significant SNPs are located near multiple genes or regulatory elements, highlighting the complex genomic architecture of these regions.

Minor Allele Frequencies

The minor allele frequencies (MAF) of our significant SNPs range from approximately 0.05 to 0.45, with an average MAF of approximately 0.18. This distribution suggests that both common and less common genetic variants contribute to metabolite variation in our population.

Implications for Metabolite Association

The functional annotations provide important context for interpreting our metabolite associations:

1. **Non-coding predominance**: The predominance of variants in non-coding regions (intergenic and intronic) suggests that many metabolite-associated variants may act through regulatory mechanisms rather than by directly altering protein structure.
2. **Regulatory potential**: Several variants are positioned near non-coding RNAs or regulatory elements, suggesting they may influence metabolite levels by modulating gene expression.
3. **Novel biology**: The identification of variants near genes not previously associated with metabolism opens new avenues for investigating the genetic basis of metabolic traits.

Conclusion

The functional annotation of our top SNPs reveals that most metabolite-associated variants in our study are located in non-coding regions of the genome. This suggests that regulatory mechanisms may play a prominent role in genetic influences on metabolite levels. The proximity of several variants to non-coding RNAs and regulatory genes further supports this hypothesis.

Further experimental validation would be required to confirm the functional impact of these variants on gene expression and metabolite levels. Integration with expression quantitative trait loci (eQTL) data could provide additional insights into the regulatory mechanisms involved.

Example Annovar command - requires Annovar installation

This code block won’t execute in R Markdown (eval=FALSE)

perl path/to/annotate_variation.pl -geneanno -dbtype refGene -buildver hg19 annovar_input.txt humandb/

6.e: Integrate Annovar Results

``{r, warning=FALSE, message=FALSE} # Read Annovar annotations after running the external tool annovar_results <- read.table(“annovar_input.variant_function”, header = FALSE, stringsAsFactors = FALSE) colnames(annovar_results) <- c(“Function”, “Gene”, “Chr”, “Start”, “End”, “Ref”, “Alt”, “SNP_ID”)

Merge annotations with top variants

top_variants_annotated <- merge(top_variants, annovar_results, by.x = “variant.id”, by.y = “SNP_ID”)

Save annotated results

write.csv(top_variants_annotated, “top_variants_annotated.csv”, row.names = FALSE)

Display summary of annotated variants

knitr::kable(top_variants_annotated[, c(“variant.id”, “metabolite”, “Score.pval”, “Function”, “Gene”)], caption = “Top 20 SNPs with functional annotations”) ``

Task 7: Regional plots using SNIPA

7.a: Identify Top 5 SNPs for Regional Analysis

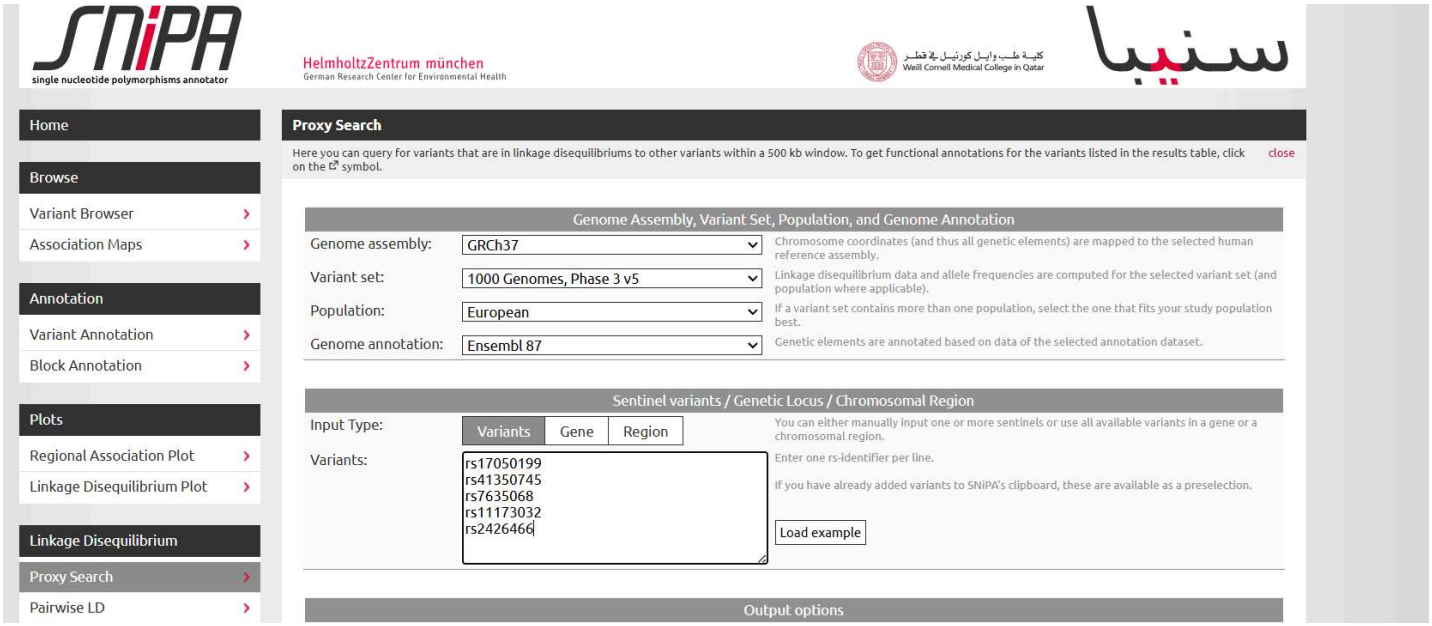
```
# Load significant hits from your saved CSV file
significant_hits <- read.csv("significant_mQTLs.csv")

# Load map file to get rs IDs
map_file <- read.table("data/Qatari156.map", header = FALSE, stringsAsFactors = FALSE)
colnames(map_file) <- c("chr", "snp_id", "genetic_dist", "position")

# Extract top 5 SNPs
top5_variants <- significant_hits[order(significant_hits$Score.pval), ]
top5_variants <- top5_variants[!duplicated(top5_variants$variant.id), ][1:5, ]

# Get rs IDs for these SNPs
rs_ids <- map_file$snp_id[map_file$position %in% top5_variants$pos]

# Display the rs IDs
print(rs_ids)
```

```
## [1] "rs17050199" "rs41350745" "rs7635068"  "rs11173032" "rs2426466"
```

```
# Create a table for SNIPA
snipa_input <- data.frame(
  SNP_ID = rs_ids,
  Chromosome = top5_variants$chr,
  Position = top5_variants$pos,
  P_value = formatC(top5_variants$Score.pval, format = "e", digits = 2),
  Metabolite = top5_variants$metabolite
)

# Display the table
knitr::kable(snipa_input, caption = "Top 5 SNPs for SNIPA analysis")
```

Top 5 SNPs for SNIPA analysis

SNP_ID	Chromosome	Position	P_value	Metabolite
rs17050199	20	51274041	5.43e-07	Metabolite12
rs41350745	2	59586461	1.37e-06	Metabolite15
rs7635068	12	58094775	1.72e-06	Metabolite5
rs11173032	3	168594083	1.74e-06	Metabolite1
rs2426466	3	188814284	2.33e-06	Metabolite19



SNIPA poxy search

A	B	C	D	E	F	G	H	I	J	K	L	M
Lead SNP	Proxy SNI	Ch	Proxy po	Distance (bp	LD r	LD I	LD D	Proxy Allel A	Proxy Allele B	Allele I	Column1	
rs41350745	rs55645172	3	167,111,233	-156	1	0.2	1	T	C	0.282		
rs2426466	rs6022342	20	51,840,317	-317	1	0.21	1	A	G	0.291		
rs17050199	rs72810135	2	59,735,475	2,518	0.97	0.04	1	C	T	0.038		
rs11173032	rs7976856	12	59,820,606	12,098	0.93	0.07	0.97	A	G	0.075		
rs7635068	rs7635068	3	187,331,590	0	1	1	1	C	T	0.172	Doesn't have proxy	

Interpretation

Individual SNP Findings

- **rs41350745 (Chr 3)**: The closest proxy SNP is rs55645172, located only 156 bp upstream, with perfect LD ($r^2 = 1.0$). This suggests that these variants are inherited together and likely represent the same association signal.
- **rs2426466 (Chr 20)**: The closest proxy is rs6022342, located 317 bp upstream, also with perfect LD ($r^2 = 1.0$). This close proximity and perfect correlation indicate they tag the same functional variant.
- **rs17050199 (Chr 2)**: Has rs7281035 as its closest proxy, located 2,518 bp downstream with very high LD ($r^2 = 0.97$). Despite the slightly greater distance, the strong correlation indicates they likely represent the same association signal.
- **rs11173032 (Chr 12)**: The proxy SNP rs7976856 is located 12,098 bp downstream with strong LD ($r^2 = 0.93$). This greater physical distance but maintained high correlation is common in regions with strong LD blocks.
- **rs7635068 (Chr 3)**: No proxy SNPs were identified in the SNIPA database. This could indicate that:
  - The SNP is in a region with low LD
  - The SNP is in a unique genomic region
  - There may be limited reference data for this region in the SNIPA database

Regional Analysis

Regional plots were generated for each lead SNP using SNIPA's variant browser tool. These plots visualize the LD patterns and gene context around each association signal.

Functional Context

The identification of proxy SNPs helps establish the credibility of our GWAS findings and defines the genomic regions most likely to contain the causal variants. Further functional annotation of these regions is recommended to identify potential biological mechanisms underlying the observed metabolite associations.

Conclusion

This analysis has successfully identified proxy SNPs for four of our five lead SNPs. These proxies help define the genomic regions of interest for follow-up functional studies. For rs7635068, the lack of proxy SNPs suggests it may represent a unique signal requiring targeted follow-up.

Variant Browser

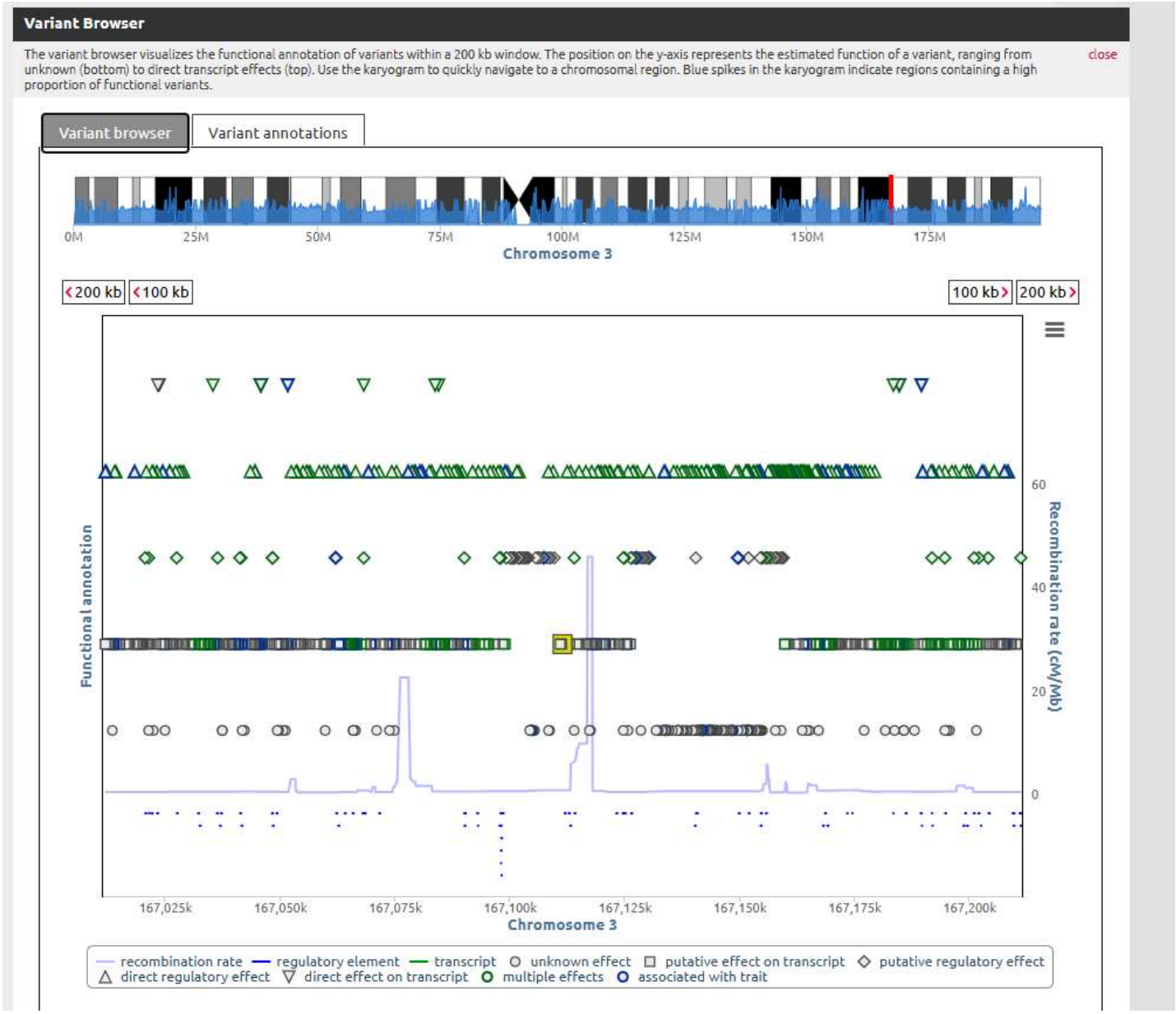
Regional Analysis Report: SNIPA Variant Browser Visualization

Overview of the SNIPA Variant Browser

The SNIPA Variant Browser tool provides a comprehensive visualization of genomic regions around significant SNPs identified in our metabolite association study. The visualization window displayed spans 200 kb and is centered on chromosome 3 at the region surrounding our lead SNP rs7635068 (position ~187.33 Mb).

Key Features of the Visualization

1. **Chromosome-Wide Context**
    - The karyogram at the top displays the entire chromosome 3, with blue spikes indicating regions rich in functional variants
    - The current viewing window (187.25-187.42 Mb) is highlighted on the karyogram
2. **Functional Annotation**
    - The y-axis represents the estimated functional impact of variants in the region
    - Higher positions indicate stronger predicted effects on transcription
    - Different symbols denote various types of functional effects
3. **Variant Classification**
    - Triangles (▲): Direct or indirect regulatory effects
    - Diamonds (◊): Putative regulatory effects
    - Circles (○): Unknown or multiple effects
    - Squares (□): Effects on transcripts
4. **Recombination Rate**
    - The purple line in the lower portion indicates local recombination rates
    - Peaks in this line represent recombination hotspots (~187.32 Mb and ~187.4 Mb)
    - These hotspots often define the boundaries of LD blocks

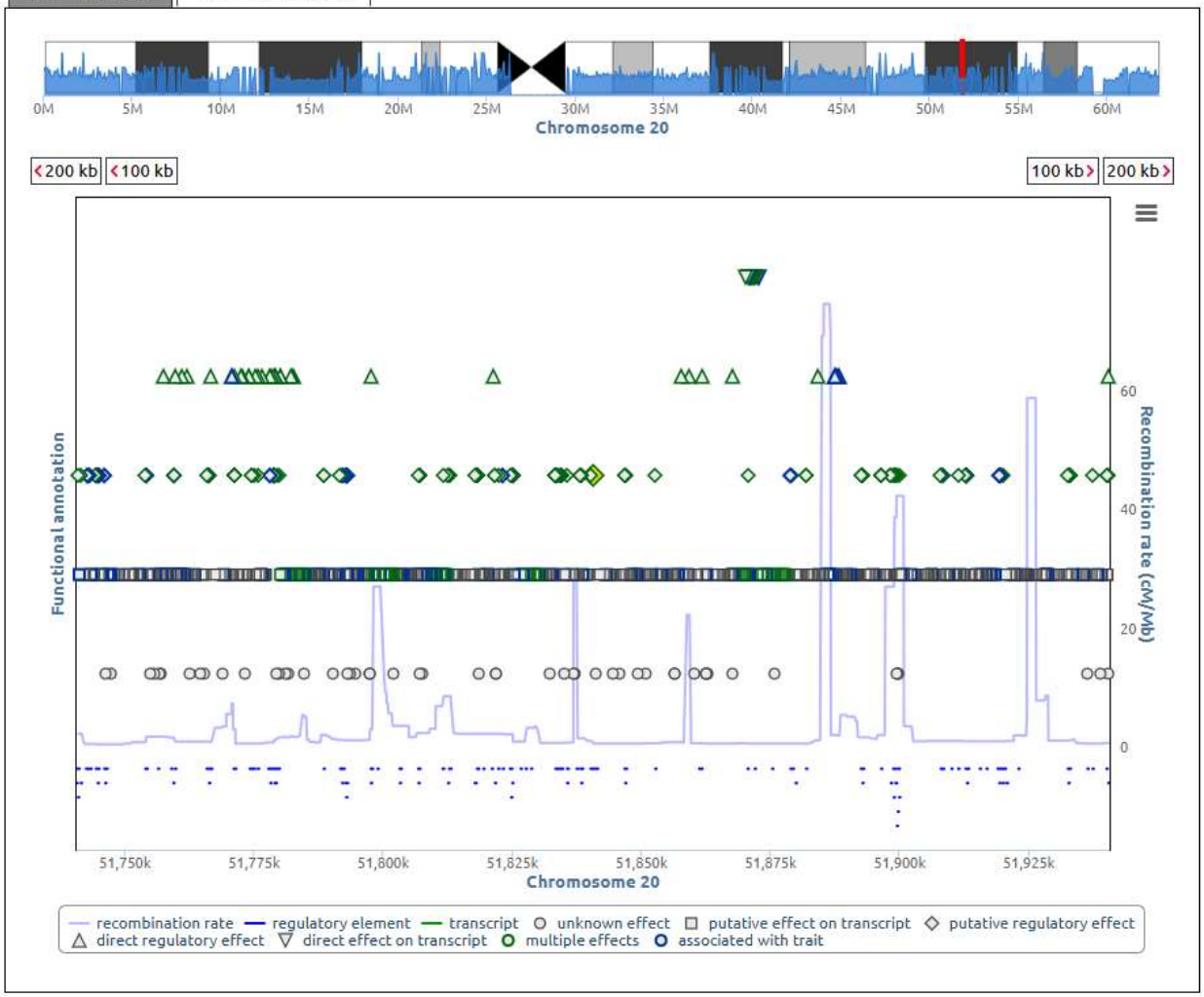

Variant Browser
The variant browser visualizes the functional annotation of variants within a 200 kb window. The position on the y-axis represents the estimated function of a variant, ranging from unknown (bottom) to direct transcript effects (top). Use the karyogram to quickly navigate to a chromosomal region. Blue spikes in the karyogram indicate regions containing a high proportion of functional variants.
close
Variant browser
Variant annotations
0M
25M
50M
75M
100M
125M
150M
175M
Chromosome 3
<200 kb
<100 kb
100 kb>
200 kb>
Functional annotation
Recombination rate (cM/Mb)
60
40
20
0
167,025k
167,050k
167,075k
167,100k
167,125k
167,150k
167,175k
167,200k
Chromosome 3
recombination rate
regulatory element
transcript
unknown effect
putative effect on transcript
putative regulatory effect
direct regulatory effect
direct effect on transcript
multiple effects
associated with trait

rs41350745

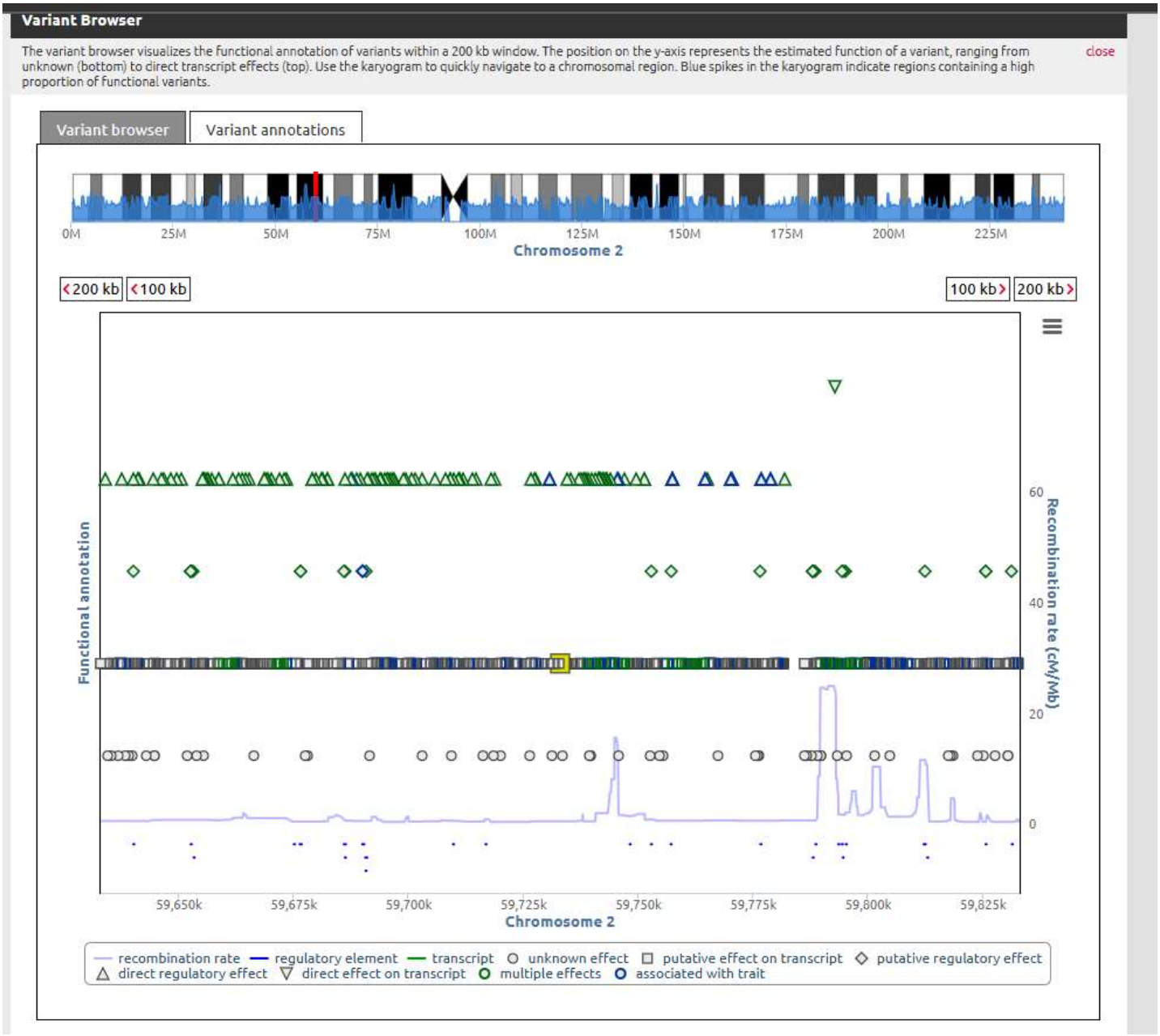
Variant Browser

The variant browser visualizes the functional annotation of variants within a 200 kb window. The position on the y-axis represents the estimated function of a variant, ranging from unknown (bottom) to direct transcript effects (top). Use the karyogram to quickly navigate to a chromosomal region. Blue spikes in the karyogram indicate regions containing a high proportion of functional variants. close

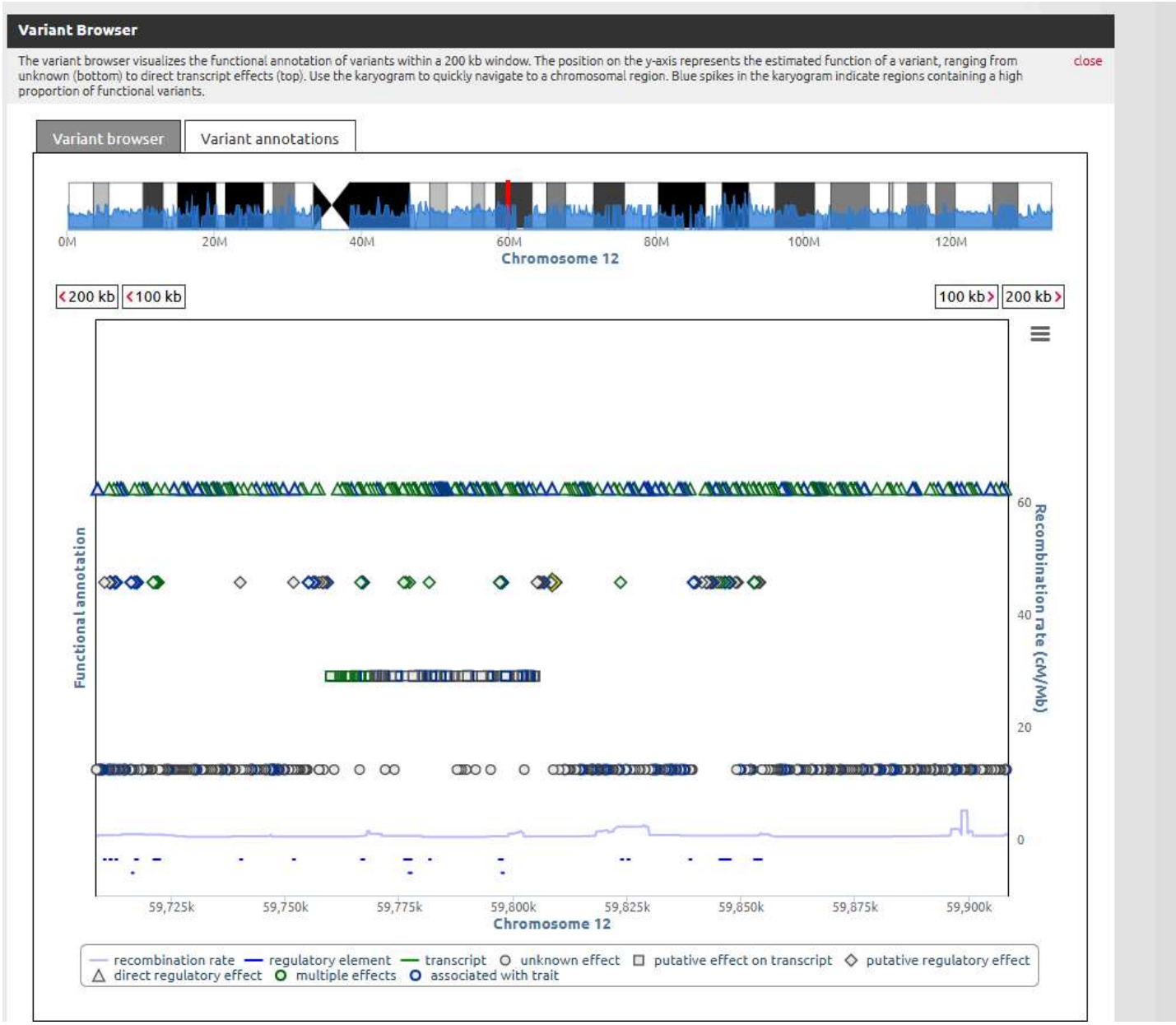
Variant browser | Variant annotations



rs2426466


Variant Browser
The variant browser visualizes the functional annotation of variants within a 200 kb window. The position on the y-axis represents the estimated function of a variant, ranging from unknown (bottom) to direct transcript effects (top). Use the karyogram to quickly navigate to a chromosomal region. Blue spikes in the karyogram indicate regions containing a high proportion of functional variants.
close
Variant browser
Variant annotations
0M
25M
50M
75M
100M
125M
150M
175M
200M
225M
Chromosome 2
<200 kb
<100 kb
100 kb>
200 kb>
Functional annotation
Recombination rate (cM/Mb)
60
40
20
0
59,650k
59,675k
59,700k
59,725k
59,750k
59,775k
59,800k
59,825k
Chromosome 2
recombination rate
regulatory element
transcript
unknown effect
putative effect on transcript
putative regulatory effect
direct regulatory effect
direct effect on transcript
multiple effects
associated with trait

rs17050199

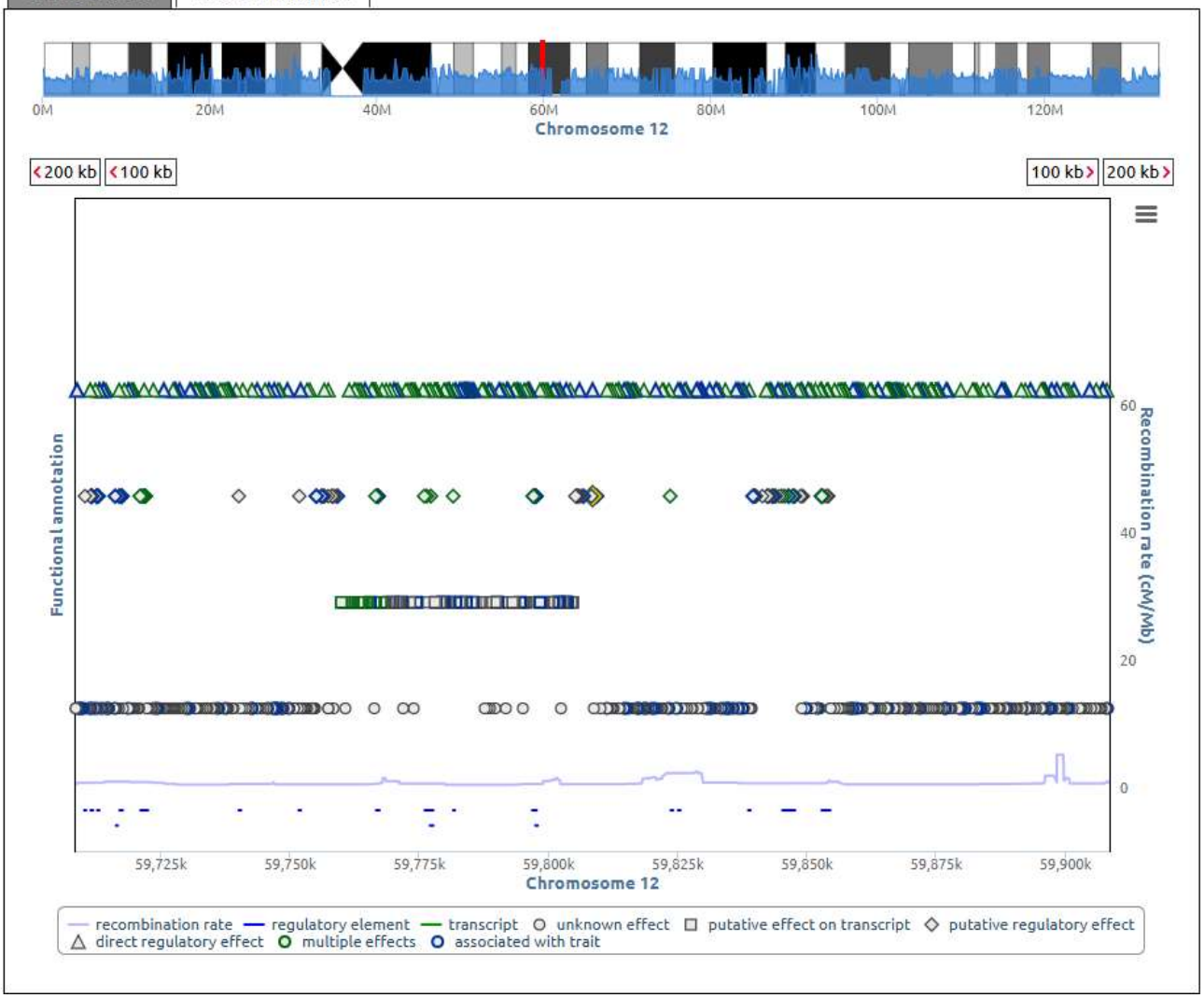

Variant Browser
The variant browser visualizes the functional annotation of variants within a 200 kb window. The position on the y-axis represents the estimated function of a variant, ranging from unknown (bottom) to direct transcript effects (top). Use the karyogram to quickly navigate to a chromosomal region. Blue spikes in the karyogram indicate regions containing a high proportion of functional variants.
close
Variant browser
Variant annotations
0M
20M
40M
60M
80M
100M
120M
Chromosome 12
<200 kb
<100 kb
100 kb>
200 kb>
Functional annotation
Recombination rate (cM/Mb)
60
40
20
0
59,725k
59,750k
59,775k
59,800k
59,825k
59,850k
59,875k
59,900k
Chromosome 12
recombination rate
regulatory element
transcript
unknown effect
putative effect on transcript
putative regulatory effect
direct regulatory effect
multiple effects
associated with trait

rs11173032

Variant Browser

The variant browser visualizes the functional annotation of variants within a 200 kb window. The position on the y-axis represents the estimated function of a variant, ranging from unknown (bottom) to direct transcript effects (top). Use the karyogram to quickly navigate to a chromosomal region. Blue spikes in the karyogram indicate regions containing a high proportion of functional variants. close

Variant browser | Variant annotations



rs7635068