QTL Analysis Pipeline - Complete Configuration Guide

Table of Contents

Introduction

The QTL Analysis Pipeline is a comprehensive tool for identifying genetic variants that influence molecular traits (expression, protein, splicing). This guide will help you configure the pipeline correctly for your specific dataset and research questions.

What is QTL Analysis?

- eQTL: Expression Quantitative Trait Loci - genetic variants affecting gene expression
- pQTL: Protein Quantitative Trait Loci - genetic variants affecting protein levels
- sQTL: Splicing Quantitative Trait Loci - genetic variants affecting RNA splicing

Quick Start Guide

For Beginners - Basic Configuration

1. Start with the minimal configuration:

```yaml
results_dir: "my_qtl_results"


input_files:

  genotypes: "data/genotypes.vcf.gz"

  covariates: "data/covariates.txt"

  annotations: "data/genes.bed"

  expression: "data/expression.txt"


analysis:

  qtl_types: "eqtl"

  qtl_mode: "cis"


performance:

  num_threads: 4

  memory_gb: 8
```

2. Run the pipeline:

```bash
python run_QTLPipeline.py --config config.yaml
```

Configuration File Structure

Complete Configuration Template

yaml

```yaml
# QTL Analysis Pipeline - Complete Configuration

# ==============================================


# MANDATORY: Results directory where all output will be stored

results_dir: "results"


# MANDATORY: Input files section

input_files:

  genotypes: "data/genotypes.vcf"

  covariates: "data/covariates.txt"

  annotations: "data/annotations.bed"

  expression: "data/expression.txt"

  protein: "data/protein.txt"

  splicing: "data/splicing.txt"


# Analysis type and mode

analysis:

  qtl_types: "all"  # "all", "eqtl", "pqtl", "sqtl"

  qtl_mode: "cis"   # "cis", "trans", "both"
  run_gwas: false

# Normalization methods
normalization:
  eqtl:
    method: "vst"
  pqtl:
    method: "log2"
  sqtl:
    method: "log2"

# Performance settings
performance:
  num_threads: 8
  memory_gb: 32
# [Additional sections...]
```

Mandatory Settings

1. Results Directory

```
yaml
results_dir: "path/to/your/results"
```

Description: Directory where all analysis outputs will be saved
Recommendations:

- Use an absolute path for clarity
- Ensure sufficient disk space (50GB+ for large datasets)
- Don't use system directories like `/tmp/`

2. Input Files Configuration

Genotype File

```yaml
genotypes: "data/genotypes.vcf.gz"
```

Supported Formats:

- VCF/VCF.GZ (Recommended): Standard variant call format
- BCF: Binary VCF, faster for large datasets
- PLINK BED: PLINK binary format

Format Recommendations:

- For datasets < 1GB: VCF
- For datasets 1GB-10GB: VCF.GZ
- For datasets > 10GB: BCF or PLINK BED

Covariates File

```yaml
covariates: "data/covariates.txt"
```

Format: Tab-separated file with samples as columns and covariates as rows

```text
       sample1    sample2    sample3
age     45         32         58
sex     1          2          1
pc1     0.12      -0.05       0.23
```

Required Covariates:

- Age, sex, genetic principal components (PCs)
- Batch effects, technical covariates
- Typically include 5-20 PCs for genetic background

Annotation File

```yaml
annotations: "data/annotations.bed"
```

Format: BED format (chromosome, start, end, gene_id)

```text
chr1    1000    5000    gene1
chr1    8000    12000   gene2
chr2    5000    9000    gene3
```

Phenotype Files

```yaml
expression: "data/expression.txt"

protein: "data/protein.txt"

splicing: "data/splicing.txt"
```

Format: Tab-separated with features as rows and samples as columns

```text
       sample1    sample2    sample3

gene1   10.5       8.2        12.1

gene2   5.1        6.8        4.9
```

Normalization Settings

eQTL Normalization

```yaml
eqtl:

  method: "vst"  # Options: "vst", "log2", "quantile", "tpm", "raw"

  vst_blind: true

  fit_type: "parametric"

  use_deseq2: true
```

Method Recommendations:

- VST (Recommended): Variance stabilizing transformation, handles count data well
- log2: Simple log transformation, good for normalized data

- quantile: Forces same distribution across samples
- tpm: Transcripts per million, for RNA-seq data
- raw: No transformation, use with caution

pQTL Normalization

```yaml
pqtl:
  method: "log2"  # Options: "log2", "quantile", "zscore", "raw"
  log2_pseudocount: 1
  remove_zeros: true
```

Method Recommendations:

- log2: Most common for protein data
- zscore: Standardization, good for comparing across proteins
- quantile: When distribution normalization is needed

sQTL Normalization

```yaml
sqtl:
  method: "log2"  # Options: "log2", "arcsinh", "zscore", "raw"
  log2_pseudocount: 1
  psi_range: [0, 1]  # PSI values range
```

Method Recommendations:

- log2: For PSI (percent spliced in) values
- arcsinh: For data with many zeros
- zscore: Standardized effect sizes

Analysis Configuration

QTL Types

```yaml
qtl_types: "all"  # "all", "eqtl", "pqtl", "sqtl" or comma-separated list
```

Recommendations:

- Start with one QTL type to test the pipeline
- Use `"eqtl"` for gene expression
- Use `"pqtl"` for protein data
- Use `"sqtl"` for splicing data
- Use `"all"` for comprehensive analysis

QTL Mapping Mode

```yaml
qtl_mode: "cis"  # "cis", "trans", "both"
```

Mode Explanations:

- cis-QTL: Variants near the gene (< 1Mb), faster, more power
- trans-QTL: Variants anywhere in genome, slower, requires more samples
- both: Run both analyses

Sample Size Recommendations:

- cis-QTL: 100+ samples
- trans-QTL: 500+ samples

Statistical Parameters

```yaml
```

```
tensorqtl:
  cis_window: 1000000  # 1Mb window around each gene

  maf_threshold: 0.05  # Minor allele frequency threshold

  min_maf: 0.01        # Minimum MAF

  fdr_threshold: 0.05  # False discovery rate threshold

  num_permutations: 1000  # Number of permutations for FDR
```

Parameter Guidelines:

- `cis_window`: 100kb-1Mb (100000-1000000)
- `maf_threshold`: 0.01-0.05 (1-5%)
- `num_permutations`: 1000-10000 (higher = more accurate FDR)

Performance Optimization

Basic Performance Settings

```yaml
performance:

  num_threads: 8       # Number of CPU threads

  memory_gb: 32        # Memory allocation in GB

  temp_dir: "temp"     # Temporary directory

  cleanup_temp: true   # Clean up temporary files
```

Hardware Recommendations

Dataset Size	CPU Threads	Memory	Disk Space	Recommended Use
Small (<1GB)	4-8	8-16GB	20GB	Testing, small studies
Medium (1-10GB)	8-16	16-32GB	50GB	Standard eQTL studies
Large (10-50GB)	16-32	32-64GB	100GB	Multi-omics studies
Very Large (>50GB)	32+	64-128GB	200GB+	Large consortia

Large Dataset Configuration

```yaml
large_data:

  min_memory_gb: 16

  min_disk_gb: 50

  process_by_chromosome: true

  force_plink: true

  monitor_resources: true
```

Quality Control Settings

Enhanced QC Configuration

```yaml
enhanced_qc:

  enable: true

  sample_missingness_threshold: 0.1    # 10% missing samples

  variant_missingness_threshold: 0.1   # 10% missing variants

  hwe_threshold: 1e-6                  # Hardy-Weinberg equilibrium

  maf_threshold: 0.01                  # 1% MAF filter

  run_pca: true

  num_pcs: 10                          # Number of principal components
```

QC Threshold Recommendations

Metric	Strict	Moderate	Liberal	Description
Sample missingness	0.02	0.05	0.10	Remove samples with high missingness
Variant missingness	0.02	0.05	0.10	Remove variants with high missingness
MAF threshold	0.05	0.01	0.005	Minimum minor allele frequency
HWE threshold	1e-6	1e-4	1e-3	Hardy-Weinberg equilibrium p-value

Advanced Features

Interaction Analysis

```
yaml
```

```yaml
interaction_analysis:

  enable: false

  interaction_covariates: ["age", "sex", "bmi"]

  fdr_threshold: 0.1

  method: "linear"
```

Use Cases:

- Test if QTL effects differ by age, sex, or other traits
- Requires larger sample sizes (>500 samples)
- Higher FDR threshold often needed

Fine-mapping

```
yaml
```

```yaml
fine_mapping:

  enable: false

  method: "susie"  # "susie", "finemap"

  credible_set_threshold: 0.95

  max_causal_variants: 5
```

Use Cases:

- Identify causal variants in associated regions
- Requires high-quality genotype data

- Works best in European ancestry populations

GWAS Analysis

```
yaml
```

```yaml
analysis:

  run_gwas: true

  gwas_phenotype: "data/gwas_phenotype.txt"



gwas:

  method: "linear"  # "linear", "logistic"

  covariates: true

  maf_threshold: 0.01
```

Output Configuration

Output Settings

```
yaml
```

```yaml
output:

  compression: true

  remove_intermediate: false

  generate_report: true

  report_format: "html"

  save_plots: true
```

Output Files Structure

```text
results/
├── QTL_results/           # Main QTL results
├── GWAS_results/          # GWAS results (if enabled)
├── plots/                 # All generated plots
├── reports/               # HTML and text reports
├── QC_reports/            # Quality control reports
├── interaction_results/   # Interaction analysis
├── fine_mapping_results/  # Fine-mapping results
└── logs/                  # Pipeline logs
```

Example Configurations

Basic eQTL Analysis

```yaml
results_dir: "eqtl_results"

input_files:
  genotypes: "data/genotypes.vcf.gz"
  covariates: "data/covariates.txt"
  annotations: "data/genes.bed"
  expression: "data/expression.txt"

analysis:
  qtl_types: "eqtl"
  qtl_mode: "cis"

normalization:
  eqtl:
    method: "vst"

performance:
  num_threads: 4
  memory_gb: 16

output:
  generate_report: true
  save_plots: true
```

Multi-omics Analysis

```
yaml
```

```
results_dir: "multi_omics_results"
```

```yaml
input_files:
  genotypes: "data/genotypes.bcf"
  covariates: "data/covariates.txt"
  annotations: "data/genes.bed"
  expression: "data/expression.txt"
  protein: "data/protein.txt"
  splicing: "data/splicing.txt"

analysis:
  qtl_types: "all"
  qtl_mode: "both"

normalization:
  eqtl:
    method: "vst"
  pqtl:
    method: "log2"
  sqtl:
    method: "log2"

performance:
  num_threads: 16
  memory_gb: 64

enhanced_qc:
  enable: true
  run_pca: true
```

Large Consortium Analysis

```yaml
results_dir: "large_study_results"

input_files:
  genotypes: "data/genotypes.bcf"
  covariates: "data/covariates.txt"
  annotations: "data/genes.bed"
  expression: "data/expression.txt"

analysis:
  qtl_types: "eqtl"
  qtl_mode: "both"

large_data:
  min_memory_gb: 32
  min_disk_gb: 100

  process_by_chromosome: true

  force_plink: true


performance:

  num_threads: 32

  memory_gb: 128

  use_gpu: false


tensorqtl:

  batch_size: 10000

  chunk_size: 100
```

Troubleshooting Guide

Common Issues and Solutions

1. Memory Errors

Symptoms: Pipeline crashes with memory errors
Solutions:

- Increase `memory_gb` in performance section
- Enable `process_by_chromosome: true`
- Use PLINK format with `force_plink: true`

2. Long Runtime

Symptoms: Analysis takes too long
Solutions:

- Increase `num_threads`
- Use `qtl_mode: "cis"` instead of "both"
- Reduce `num_permutations` to 1000
- Use BCF or PLINK format instead of VCF

3. No Significant Results

Symptoms: Pipeline runs but finds no significant QTLs
Solutions:

- Check sample size (need 100+ for cis, 500+ for trans)
- Verify phenotype normalization
- Check covariate inclusion
- Reduce `maf_threshold` to 0.01

4. File Format Errors

Symptoms: Validation fails on input files
Solutions:

- Ensure VCF files are properly formatted
- Check that all files are tab-separated
- Verify sample IDs match across files
- Check chromosome naming consistency

Validation Checklist

Before running the pipeline, verify:

- All input files exist and are readable
- Sample IDs match across all files
- Chromosome naming is consistent (chr1 vs 1)
- Sufficient disk space in results directory
- Adequate memory and CPU resources
- File formats are correct (VCF, BED, tab-separated)

Debug Mode

Enable debug mode for detailed logging:

```bash
python run_QTLPipeline.py --config config.yaml --debug
```

Getting Help

1. Check the log files in `results/logs/`
2. Run validation only mode:

```bash
python run_QTLPipeline.py --config config.yaml --validate-only
```

3. Check the HTML report for detailed error information

Best Practices

Data Preparation

1. Genotype Data:
   - Use imputed genotypes for better variant coverage
   - Filter for MAF > 1% and call rate > 95%
   - Use GRCh38 reference genome when possible
2. Expression Data:
   - Use normalized counts for RNA-seq
   - Remove lowly expressed genes
   - Correct for batch effects
3. Covariates:
   - Include age, sex, and genetic PCs
   - Include technical covariates (RIN, sequencing depth)
   - Consider including known confounding factors

Computational Resources

1. Start Small: Test with chromosome 22 first
2. Monitor Resources: Use `monitor_resources: true`
3. Use Temporary Storage: Ensure `/tmp` has sufficient space
4. Parallelize: Use multiple threads for faster analysis

Quality Control

1. Check Sample Concordance: Ensure samples match across datasets
2. Review QC Plots: Check for batch effects and outliers
3. Validate Normalization: Use normalization comparison plots
4. Check Lambda GC: Should be close to 1.0 for well-controlled data