Module Execution Commands

1. Data Preparation (No dependencies)

```bash
python run_QTLPipeline.py --modules data_preparation
```

2. Genotype Processing (Depends on Data Preparation)

```bash
# If data_preparation already completed:
python run_QTLPipeline.py --modules genotype_processing

# If data_preparation not run, auto-resolve dependencies:
python run_QTLPipeline.py --modules genotype_processing --auto-deps

# Force run (skip dependency check):
python run_QTLPipeline.py --modules genotype_processing --force
```

3. Expression Processing (Depends on Data Preparation)

```bash
# If data_preparation already completed:
python run_QTLPipeline.py --modules expression_processing

# Auto-resolve dependencies:
python run_QTLPipeline.py --modules expression_processing --auto-deps

# Force run:
python run_QTLPipeline.py --modules expression_processing --force
```

4. Quality Control (Depends on Genotype & Expression Processing)

```bash
# If dependencies completed:
python run_QTLPipeline.py --modules quality_control

# Auto-resolve all dependencies:
python run_QTLPipeline.py --modules quality_control --auto-deps

# Force run:
python run_QTLPipeline.py --modules quality_control --force
```

5. QTL Mapping (Depends on Quality Control)

```bash
# If quality_control completed:
python run_QTLPipeline.py --modules qtl_mapping

# Auto-resolve all dependencies:
python run_QTLPipeline.py --modules qtl_mapping --auto-deps

# Force run:
python run_QTLPipeline.py --modules qtl_mapping --force
```

6. Fine Mapping (Depends on QTL Mapping)

```bash
# If qtl_mapping completed:
python run_QTLPipeline.py --modules fine_mapping

# Auto-resolve dependencies:
python run_QTLPipeline.py --modules fine_mapping --auto-deps

# Force run:
python run_QTLPipeline.py --modules fine_mapping --force
```

7. Interaction Analysis (Depends on QTL Mapping)

```bash
# If qtl_mapping completed:
python run_QTLPipeline.py --modules interaction_analysis

# Auto-resolve dependencies:
python run_QTLPipeline.py --modules interaction_analysis --auto-deps

# Force run:
python run_QTLPipeline.py --modules interaction_analysis --force
```

8. Visualization (Depends on QTL Mapping & Fine Mapping)

```bash
# If dependencies completed:
python run_QTLPipeline.py --modules visualization

# Auto-resolve all dependencies:
python run_QTLPipeline.py --modules visualization --auto-deps

# Force run:
python run_QTLPipeline.py --modules visualization --force
```

9. Report Generation (Depends on QTL Mapping, Fine Mapping & Visualization)

```bash
# If dependencies completed:
python run_QTLPipeline.py --modules report_generation

# Auto-resolve all dependencies:
python run_QTLPipeline.py --modules report_generation --auto-deps

# Force run:
```

```
python run_QTLPipeline.py --modules report_generation --force
```

Common Module Combinations

Data Processing Pipeline

```
bash
python run_QTLPipeline.py --modules data_preparation genotype_processing
expression_processing quality_control
```

QTL Analysis Pipeline

```
bash
python run_QTLPipeline.py --modules qtl_mapping fine_mapping
interaction_analysis
```

Visualization & Reporting

```
bash
python run_QTLPipeline.py --modules visualization report_generation
```

Complete Analysis (All Modules)

```
bash
python run_QTLPipeline.py --all
```

Module Dependency Flow Chart

```
text
```

```
data_preparation
    ├── genotype_processing
    ├── expression_processing
    └── quality_control
        └── qtl_mapping
            ├── fine_mapping
            ├── interaction_analysis
            └── visualization
                └── report_generation
```

Smart Dependency Resolution

Option 1: Manual Dependency Management

```bash
# Run in dependency order
python run_QTLPipeline.py --modules data_preparation
python run_QTLPipeline.py --modules genotype_processing expression_processing
python run_QTLPipeline.py --modules quality_control
python run_QTLPipeline.py --modules qtl_mapping
python run_QTLPipeline.py --modules fine_mapping interaction_analysis
python run_QTLPipeline.py --modules visualization
python run_QTLPipeline.py --modules report_generation
```

Option 2: Auto-Dependency Resolution

```bash
# Let the pipeline figure out dependencies automatically
python run_QTLPipeline.py --modules qtl_mapping --auto-deps
python run_QTLPipeline.py --modules visualization --auto-deps
python run_QTLPipeline.py --modules report_generation --auto-deps
```

Option 3: Force Run (Use with Caution)

```bash
# Skip dependency checks (only if you're sure dependencies are met)
python run_QTLPipeline.py --modules qtl_mapping --force
```

Checking Module Status

List All Available Modules

```bash
python run_QTLPipeline.py --list
```

Check What Will Run

```bash
# See what modules will be executed with auto-deps
python run_QTLPipeline.py --modules visualization --auto-deps --list
```

Practical Execution Examples

Example 1: Step-by-Step Execution

```bash
# Day 1: Data preparation
python run_QTLPipeline.py --modules data_preparation genotype_processing
expression_processing

# Day 2: QC and QTL mapping
python run_QTLPipeline.py --modules quality_control qtl_mapping

# Day 3: Advanced analysis
python run_QTLPipeline.py --modules fine_mapping interaction_analysis
```

```
# Day 4: Results and reports
python run_QTLPipeline.py --modules visualization report_generation
```

Example 2: Restart After Failure

```
bash
# Suppose qtl_mapping failed, restart from there
python run_QTLPipeline.py --modules qtl_mapping fine_mapping visualization
report_generation --auto-deps
```

Example 3: Only Generate Reports

```
bash
# If analysis already completed, just generate reports
python run_QTLPipeline.py --modules visualization report_generation --force
```

Key Points:

1. Use `--auto-deps` when you're not sure about dependencies
2. Use `--force` only when you're certain dependencies are satisfied
3. Check with `--list` first to see module dependencies
4. Modules can be combined in a single command for efficiency
5. The system tracks completed modules within a session

QTL Pipeline Modules Overview

Module Name	Main Scripts Used	Dependent Modules	Description	Key Output Files	Command Example
data_preparation	enhanced_qc.py, validation.py	None	Prepare and validate input data	input_validation_report.txt, sample_mapping.txt	python run_QTLPipeline.py --modules data_preparation
genotype_processing	genotype_processing.py	data_preparation	Process genotype data (VCF/PLINK formats)	final_genotypes.vcf.gz, plink_files/	python run_QTLPipeline.py --modules genotype_processing --auto-deps
expression_processing	qtl_analysis.py	data_preparation	Process expression data (normalization, QC)	expression_processed.txt, normalized_expression/	python run_QTLPipeline.py --modules expression_processing --auto-deps
quality_control	enhanced_qc.py	genotype_processing, expression_processing	Perform comprehensive quality control	qc_report.html, sample_QC_stats.txt	python run_QTLPipeline.py --modules quality_control --auto-deps
qtl_mapping	qtl_analysis.py, main.py	quality_control	Perform cis/trans QTL mapping analysis	cis_qtl.txt, trans_qtl.txt, nominal_results/	python run_QTLPipeline.py --modules qtl_mapping --auto-deps
fine_mapping	fine_mapping.py	qtl_mapping	Fine mapping of QTL signals	fine_mapped_variants.txt, credible_sets/	python run_QTLPipeline.py --modules fine_mapping --auto-deps
interaction_analysis	interaction_analysis.py	qtl_mapping	Covariate interaction analysis	interaction_results.txt, covariate_effects/	python run_QTLPipeline.py --modules interaction_analysis --auto-deps
visualization	plotting.py, advanced_plotting.py, main.py	qtl_mapping, fine_mapping	Generate plots and visualizations	manhattan_plot.png, qq_plot.png, interactive_plots/	python run_QTLPipeline.py --modules visualization --auto-deps
report_generation	report_generator.py, main.py	qtl_mapping, fine_mapping, visualization	Generate comprehensive HTML reports	final_report.html, pipeline_summary.txt	python run_QTLPipeline.py --modules report_generation --auto-deps

Quick Reference Commands

Individual Modules (with auto-dependencies):

```bash
# Data processing
python run_QTLPipeline.py --modules data_preparation
python run_QTLPipeline.py --modules genotype_processing --auto-deps
python run_QTLPipeline.py --modules expression_processing --auto-deps
python run_QTLPipeline.py --modules quality_control --auto-deps

# Analysis
python run_QTLPipeline.py --modules qtl_mapping --auto-deps
python run_QTLPipeline.py --modules fine_mapping --auto-deps
python run_QTLPipeline.py --modules interaction_analysis --auto-deps

# Outputs
python run_QTLPipeline.py --modules visualization --auto-deps

python run_QTLPipeline.py --modules report_generation --auto-deps
```

Popular Combinations:

```bash
# Complete analysis in one command
python run_QTLPipeline.py --all

# Just the analysis (assuming data is ready)
python run_QTLPipeline.py --modules qtl_mapping fine_mapping
interaction_analysis --force

# Regenerate reports only

python run_QTLPipeline.py --modules visualization report_generation --force
```

Module Execution Flow Table

Execution Order	Module	Can Run Alone?	Typical Runtime	Memory Requirements
1	data_preparation	✅ Yes	Fast (1-10 min)	Low (< 4GB)
2	genotype_processing	❌ Requires data_preparation	Medium to Long (10min-2h)	High (8-32GB)
3	expression_processing	❌ Requires data_preparation	Medium (5-30 min)	Medium (4-16GB)
4	quality_control	❌ Requires genotype+expression	Fast (1-15 min)	Medium (4-8GB)
5	qtl_mapping	❌ Requires quality_control	Long to Very Long (1h-24h+)	High to Very High (16-64GB+)
6	fine_mapping	❌ Requires qtl_mapping	Medium (15min-2h)	Medium (8-16GB)
7	interaction_analysis	❌ Requires qtl_mapping	Medium (10min-1h)	Medium (8-16GB)

8	visualization	❌ Requires qtl_mapping+fine_mapping	Fast (1-10 min)	Low (< 4GB)
9	report_generation	❌ Requires visualization	Fast (1-5 min)	Low (< 2GB)

Common Module Combinations

Combination Name	Modules Included	Use Case	Command
Data Processing	data_preparation, genotype_processing, expression_processing, quality_control	Prepare data for analysis	`python run_QTLPipeline.py --modules data_preparation genotype_processing expression_processing quality_control`
QTL Analysis	qtl_mapping, fine_mapping, interaction_analysis	Complete QTL discovery	`python run_QTLPipeline.py --modules qtl_mapping fine_mapping interaction_analysis --auto-deps`

Results & Reports	visualization, report_generation	Generate final outputs	`python run_QTLPipeline.py --modules visualization report_generation --auto-deps`
Quick QC	data_preparation, quality_control	Data quality check	`python run_QTLPipeline.py --modules data_preparation quality_control --auto-deps`
Full Pipeline	All modules	Complete analysis	`python run_QTLPipeline.py --all`

Module Configuration Requirements

Module	Required Config Sections	Optional Features
data_preparation	`input_files`, `enhanced_qc`	Sample filtering, normalization
genotype_processing	`genotype_processing`, `large_data`	PLINK conversion, chunking
expression_processing	`input_files.expression`, `qtl`	Normalization methods

quality_control	`enhanced_qc`, `qc`	Sample concordance, PCA
qtl_mapping	`analysis`, `qtl`, `performance`	cis/trans modes, permutations
fine_mapping	`fine_mapping`	Credible sets, posterior probabilities
interaction_analysis	`interaction_analysis`	Covariate interactions
visualization	`plotting`	Interactive plots, multi-panel
report_generation	`output`	HTML reports, summaries

Troubleshooting Common Module Issues

Module	Common Issues	Solutions
genotype_processing	Large file handling	Use `--auto-deps`, ensure disk space
qtl_mapping	Memory errors	Increase `performance.memory_gb` in config
visualization	Missing results	Run `qtl_mapping` first or use `--auto-deps`
quality_control	Sample mismatches	Check sample IDs in `data_preparation`