README.md

```
markdown
```

```
# QTL Analysis Pipeline

A comprehensive, production-ready pipeline for QTL (eQTL, pQTL, sQTL) and GWAS analysis using QTLTools and PLINK.

## 🚀 Features

- **Multiple QTL Types**: Support for eQTL (expression), pQTL (protein), and sQTL (splicing) analysis
- **GWAS Integration**: Optional GWAS analysis with PLINK integration
- **Flexible Input**: Handles VCF/VCF.GZ genotypes and various phenotype formats
- **Professional Plotting**: Manhattan plots, QQ plots, volcano plots, and summary visualizations
- **Comprehensive Reports**: HTML and text reports with results and plots
- **Robust Error Handling**: Comprehensive logging and validation
- **Modular Design**: Easy to extend and customize

## 📁 Project Structure
```

```
QTL_ANALYSIS/
├── config/ # Configuration files
│ └── config.yaml # Main configuration file
├── data/ # Example input data
├── scripts/ # Pipeline code
│ ├── main.py # Main pipeline orchestrator
│ └── utils/ # Utility modules
├── run_QTLPipeline.py # Main runner script (use this!)
├── shellScript/ # Shell script wrappers
├── README_QTL_ANALYSIS.pdf # This file
```

```
text
```

```
## 🛠️ Quick Start
```

```
### 1. Installation
```bash
# Clone or download the pipeline

# Ensure you have the required dependencies (see INSTALL file)
```

2. Configuration

Edit `config/config.yaml` with your file paths:

```yaml
results_dir: "my_analysis_results"

input_files:
  genotypes: "path/to/your/genotypes.vcf"
  covariates: "path/to/your/covariates.txt"
  annotations: "path/to/your/annotations.bed"
  expression: "path/to/your/expression.txt"   # For eQTL
  protein: "path/to/your/protein.txt"         # For pQTL
  splicing: "path/to/your/splicing.txt"       # For sQTL

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

  run_gwas: false   # Set to true for GWAS analysis
```

3. Run the Pipeline

```bash
# Run complete analysis
python run_QTLPipeline.py --config config/config.yaml

# Run specific QTL types only
python run_QTLPipeline.py --config config/config.yaml --analysis-types
eqtl,pqtl

# Run with GWAS analysis
python run_QTLPipeline.py --config config/config.yaml --run-gwas
```
```

```
# Validate inputs only
python run_QTLPipeline.py --config config/config.yaml --validate-only
```

📊 Output Structure

The pipeline creates this organized output structure:

```
text
results/
├── qtl_results/          # QTL analysis results
│   ├── eqtl_significant.txt
│   ├── pqtl_significant.txt
│   └── sqtl_significant.txt
├── gwas_results/         # GWAS analysis results
│   └── gwas_combined_results.txt
├── plots/                # Generated visualizations
│   ├── eqtl_manhattan.png
│   ├── eqtl_qq.png
│   ├── gwas_manhattan.png
│   └── analysis_summary.png
├── reports/              # Comprehensive reports
│   ├── analysis_report.html
│   └── pipeline_summary.txt
├── logs/                 # Detailed execution logs
│   └── pipeline_*.log
└── temp/                 # Temporary files (optional)
```

⚙️ Configuration Guide

Mandatory Settings

- `results_dir`: Output directory for all results
- `input_files`: Paths to required input files

Optional Settings

- `analysis.qtl_types`: "all", "eqtl", "pqtl", "sqtl" or comma-separated list
- `analysis.run_gwas`: true/false to enable GWAS analysis
- `plotting.enabled`: true/false to generate plots
- `output.generate_report`: true/false to generate HTML reports

Input File Requirements

Genotypes: VCF or VCF.GZ format with sample genotypes
Covariates: Tab-separated file with samples in columns
Annotations: BED format with feature coordinates
Phenotypes: Tab-separated with features in rows, samples in columns

🎯 Examples

Basic eQTL Analysis

```
bash

python run_QTLPipeline.py --config config/config.yaml --analysis-types eqtl
```

Complete Multi-omics Analysis

```
bash

python run_QTLPipeline.py --config config/config.yaml --analysis-types all
--run-gwas
```

Custom Analysis with Specific Parameters

```
bash

# Edit config.yaml first, then run:

python run_QTLPipeline.py --config config/custom_config.yaml
```

🔧 Advanced Usage

Custom Plotting

Modify `config.yaml` plotting section:

```yaml
plotting:
  enabled: true
  dpi: 300
  format: "png"
  plot_types: ["manhattan", "qq", "volcano", "distribution", "summary"]
```

GWAS Configuration

```yaml
analysis:
  run_gwas: true
  gwas_phenotype: "path/to/gwas_phenotype.txt"

gwas:
  method: "linear"  # or "logistic"
  covariates: true
  maf_threshold: 0.01
```

Quality Control Settings

```yaml
qc:
  check_sample_concordance: true
  filter_low_expressed: true
  expression_threshold: 0.1
  normalize: true
```

🐛 Troubleshooting

Common Issues

1. File not found errors
    - Check all file paths in config.yaml
    - Use absolute paths for clarity
2. Tool not found errors
    - Ensure QTLTools, PLINK, bcftools are in PATH
    - See INSTALL file for installation instructions
3. Memory issues
    - Reduce number of permutations in config
    - Use subset of samples/features for testing
4. Plotting errors
    - Ensure matplotlib and seaborn are installed
    - Check write permissions in results directory

Getting Help

Check the log files in `results/logs/` for detailed error information. The pipeline provides comprehensive logging at every step.

📄 License

[Add your license information here]

🤝 Contributing

[Add contribution guidelines here]

📚 Citation

If you use this pipeline in your research, please cite:

[Add citation information]

```text
## INSTALL

```bash
# QTL Analysis Pipeline - Installation Guide
# ==========================================

## System Requirements
- Linux or macOS
- Python 3.7+
- 8GB+ RAM (16GB+ recommended)
- 20GB+ free disk space

## 1. Software Dependencies

### Required Tools
```bash
# QTLTools (for QTL analysis)
# Download from: https://qtltools.github.io/qtltools/
# Or install via conda:
conda install -c bioconda qtltools

# PLINK (for GWAS analysis)
# Download from: https://www.cog-genomics.org/plink/
# Or install via conda:
conda install -c bioconda plink

# BCFtools and HTSlib (for VCF processing)
conda install -c bioconda bcftools

conda install -c bioconda htslib
```

Verify Installation

```bash
qtltools --version
plink --version
bcftools --version
```

```
bgzip --version

tabix --version
```

2. Python Dependencies

```
bash

# Install required Python packages
pip install pandas numpy matplotlib seaborn scipy pyyaml

# Or using conda:

conda install pandas numpy matplotlib seaborn scipy pyyaml
```

Verify Python Dependencies

```
bash

python -c "import pandas, numpy, matplotlib, seaborn, scipy, yaml; print('All Python dependencies installed successfully')"
```

3. Pipeline Setup

Option A: Download from Repository

```
bash

# Clone or download the pipeline files

# Ensure the directory structure is maintained
```

Option B: Manual Setup

```
bash

# Create the directory structure
```

```
mkdir -p QTL_ANALYSIS/{config,data,scripts/utils,shellScript}

# Place all Python scripts in their respective locations

# Ensure run_QTLPipeline.py is in the root directory
```

4. Test Installation

Using Example Data

```
bash

# Navigate to the pipeline directory
cd QTL_ANALYSIS

# Run with example configuration
python run_QTLPipeline.py --config config/config.yaml --validate-only

# If validation passes, run a small test

python run_QTLPipeline.py --config config/config.yaml --analysis-types eqtl
```

5. Configuration

Edit Configuration File

```
bash

# Copy the example config
cp config/config.yaml config/my_analysis.yaml

# Edit with your file paths

nano config/my_analysis.yaml
```

Key Configuration Sections

- `results_dir`: Where to store output
- `input_files`: Paths to your data files
- `analysis`: Which analyses to run
- `parameters`: Analysis parameters
- `plotting`: Visualization settings

6. Input File Preparation

Genotype Files

- Format: VCF or VCF.GZ
- Should include all samples used in analysis
- Can be whole genome or targeted sequencing

Phenotype Files

- Tab-separated format
- Features in rows, samples in columns
- First column: feature IDs
- Supported: expression, protein, splicing data

Covariate Files

- Tab-separated format
- Covariates in rows, samples in columns
- Include PCs, sex, age, batch effects, etc.

Annotation Files

- BED format
- Required for QTL analysis
- Maps features to genomic positions

7. Troubleshooting

Common Installation Issues

QTLTools not found:

```bash
# Add to your PATH or install via conda
export PATH=/path/to/qtltools:$PATH
```

Python package errors:

```bash
# Update pip and retry
pip install --upgrade pip

pip install --force-reinstall pandas numpy matplotlib seaborn scipy pyyaml
```

Permission errors:

```bash
# Make scripts executable
chmod +x run_QTLPipeline.py
chmod +x scripts/*.py

chmod +x scripts/utils/*.py
```

Memory errors:

- Reduce dataset size for testing
- Increase system RAM
- Use fewer permutations in config

Getting Help

1. Check the log files in `results/logs/`
2. Run with `--validate-only` to check inputs
3. Ensure all file paths in config are correct
4. Verify all required tools are in PATH

8. Performance Tips

- Use VCF.GZ instead of VCF for faster I/O
- Enable compression in config for intermediate files
- Adjust number of permutations based on needs
- Use subset of data for testing before full run

9. Updating

To update the pipeline, download the latest version and replace the script files while preserving your configuration and data.

For additional support, please check the documentation in README.md or open an issue in the project repository.

````
text
## Updated Shell Script (`shellScript/run_qtl_pipeline.sh`)

```bash
#!/bin/bash

# QTL Analysis Pipeline - Shell Wrapper
# Usage: ./run_qtl_pipeline.sh [options]

set -euo pipefail

# Color codes for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

# Logging functions
log_info() { echo -e "${BLUE}[INFO]${NC} $1"; }
log_warn() { echo -e "${YELLOW}[WARN]${NC} $1"; }
log_error() { echo -e "${RED}[ERROR]${NC} $1"; }
log_success() { echo -e "${GREEN}[SUCCESS]${NC} $1"; }
````
````

```
# Default values
CONFIG_FILE="config/config.yaml"
ANALYSIS_TYPES=""
RUN_GWAS=false
VALIDATE_ONLY=false

# Usage function
usage() {
    cat << EOF
Usage: $0 [OPTIONS]

QTL Analysis Pipeline - Shell Wrapper

OPTIONS:
    -c, --config FILE        Configuration file (default: config/config.yaml)
    -a, --analysis-types TYPES Analysis types (eqtl,pqtl,sqtl or all)
    -g, --run-gwas           Enable GWAS analysis
    -v, --validate-only      Only validate inputs, don't run analysis
    -h, --help               Show this help message

EXAMPLES:
    $0 -c config/config.yaml -a eqtl,pqtl
    $0 --config my_config.yaml --run-gwas
    $0 --validate-only
EOF
}

# Parse command line arguments
while [[ $# -gt 0 ]]; do
    case $1 in
        -c|--config)
            CONFIG_FILE="$2"
            shift 2
            ;;
        -a|--analysis-types)
            ANALYSIS_TYPES="$2"
            shift 2
            ;;
        -g|--run-gwas)
            RUN_GWAS=true
            shift
```

```
            ;;
        -v|--validate-only)
            VALIDATE_ONLY=true
            shift
            ;;
        -h|--help)
            usage
            exit 0
            ;;
        *)
            log_error "Unknown option: $1"
            usage
            exit 1
            ;;
    esac
done

# Check if config file exists
if [[ ! -f "$CONFIG_FILE" ]]; then
    log_error "Config file not found: $CONFIG_FILE"
    exit 1
fi

# Build command
CMD="python run_QTLPipeline.py --config $CONFIG_FILE"

if [[ -n "$ANALYSIS_TYPES" ]]; then
    CMD="$CMD --analysis-types $ANALYSIS_TYPES"
fi

if [[ "$RUN_GWAS" == true ]]; then
    CMD="$CMD --run-gwas"
fi

if [[ "$VALIDATE_ONLY" == true ]]; then
    CMD="$CMD --validate-only"
fi

# Execute
log_info "Starting QTL Analysis Pipeline..."
log_info "Command: $CMD"
```

```
eval $CMD

if [[ $? -eq 0 ]]; then
    log_success "Pipeline completed successfully!"
else
    log_error "Pipeline failed!"
    exit 1
fi
```

Usage Examples

```bash
# Make scripts executable
chmod +x run_QTLPipeline.py
chmod +x shellScript/run_qtl_pipeline.sh

# Run complete analysis
python run_QTLPipeline.py --config config/config.yaml

# Or use the shell wrapper
./shellScript/run_qtl_pipeline.sh -c config/config.yaml

# Run specific analyses
python run_QTLPipeline.py --config config/config.yaml --analysis-types
eqtl,pqtl --run-gwas

# Validate inputs only
python run_QTLPipeline.py --config config/config.yaml --validate-only
```