Genomic Sparse Index Pipeline - WSL Command Guide

This guide provides step-by-step commands to set up and run the genomic sparse index pipeline in WSL.

Prerequisites

- WSL (Windows Subsystem for Linux) with Ubuntu 20.04 or later
- Internet connection for downloading dependencies

Quick Start (Automated Setup)

1. Run the Setup Script

```bash
# Download and run the automated setup script
curl -sSL https://raw.githubusercontent.com/your-repo/genomic-pipeline/main/setup.sh | bash

# Or if you have the setup.sh file locally:
chmod +x setup.sh
./setup.sh
```

Manual Setup (Step by Step)

1. Update System and Install Dependencies

```bash
# Update package lists and upgrade system
sudo apt update && sudo apt upgrade -y

# Install essential development tools
sudo apt install -y curl wget git build-essential pkg-config libssl-dev python3 python3-pip def

# Verify installations
gcc --version
python3 --version
java -version
```

2. Install Rust

```bash
# Install Rust using rustup
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh -s -- -y

# Source the Rust environment
source ~/.cargo/env

# Verify Rust installation
rustc --version
cargo --version
```

3. Install Nextflow

```bash
# Download and install Nextflow
curl -s https://get.nextflow.io | bash

# Move to system path
sudo mv nextflow /usr/local/bin/
sudo chmod +x /usr/local/bin/nextflow

# Verify Nextflow installation
nextflow -version
```

4. Create Project Structure

```bash
# Create and enter project directory
mkdir genomic_sparse_index
cd genomic_sparse_index

# Create subdirectories
mkdir -p src data results scripts

# Initialize Rust project
cargo init --name genomic_sparse_index
```

5. Set Up Project Files

Copy the provided artifact files to the appropriate locations:

```bash
# Copy Cargo.toml to project root
# (Copy the Cargo.toml content from the artifacts)

# Copy main.rs to src directory
# (Copy the main.rs content from the artifacts)

# Copy Nextflow workflow
# (Copy the main.nf content from the artifacts)

# Copy data generator script
# (Copy the generate_sample_data.py content to scripts/)
chmod +x scripts/generate_sample_data.py
```

6. Generate Sample Data

```bash
# Generate small test dataset (fast)
python3 scripts/generate_sample_data.py -o data/genotypes_small.txt -n 1000 -m 100000

# Generate medium dataset (for testing)
python3 scripts/generate_sample_data.py -o data/genotypes_medium.txt -n 10000 -m 1000000

# Generate large dataset (for performance testing)
python3 scripts/generate_sample_data.py -o data/genotypes_large.txt -n 100000 -m 10000000

# View sample data
head -20 data/genotypes_small.txt
```

7. Build the Rust Project

```bash
# Build in debug mode (faster compilation, slower execution)
cargo build

# Build in release mode (slower compilation, faster execution - recommended)
cargo build --release

# Copy binary to project root (needed for Nextflow)
cp target/release/rust_sparse_index .
chmod +x rust_sparse_index

# Verify the binary works
./rust_sparse_index --help
```

Running the Pipeline

Method 1: Direct Rust Execution

bash

```bash
# Basic run with small dataset
./rust_sparse_index \
    --input data/genotypes_small.txt \
    --output results/small_results.txt \
    --max-index 100000 \
    --num-tasks 4

# Run with medium dataset and verbose logging
./rust_sparse_index \
    --input data/genotypes_medium.txt \
    --output results/medium_results.txt \
    --max-index 1000000 \
    --num-tasks 8 \
    --verbose

# Run with large dataset (high performance)
./rust_sparse_index \
    --input data/genotypes_large.txt \
    --output results/large_results.txt \
    --max-index 10000000 \
    --num-tasks 16 \
    --verbose
```

Method 2: Nextflow Pipeline

```bash
# Basic Nextflow run
nextflow run main.nf

# Run with custom parameters
nextflow run main.nf \
    --input data/genotypes_medium.txt \
    --output_dir results_nf \
    --max_index 1000000 \
    --num_tasks 8

# Run with work directory cleanup
nextflow run main.nf \
    --input data/genotypes_large.txt \
    --output_dir results_large \
    --max_index 10000000 \
    --num_tasks 16 \
    -with-report results_large/execution_report.html \
    -with-timeline results_large/timeline.html

# Resume a failed run
nextflow run main.nf -resume
```

Examining Results

View Results Files

```bash
# View processing results
cat results/index_results.txt

# View summary statistics
head -20 results/index_results.txt

# Check pipeline logs
cat results/rust_sparse_index.log

# View Nextflow execution report (if generated)
# Open results/execution_report.html in a browser
```

Performance Analysis

```bash
# Check processing time and memory usage
time ./rust_sparse_index \
    --input data/genotypes_medium.txt \
    --output results/perf_test.txt \
    --max-index 1000000 \
    --num-tasks 8 \
    --verbose

# Monitor resource usage during execution
top -p $(pgrep rust_sparse_index)
```

Troubleshooting

Common Issues and Solutions

1. Rust Build Errors

```bash
# Update Rust toolchain
rustup update

# Clean and rebuild
cargo clean
cargo build --release

# Check for missing dependencies
sudo apt install pkg-config libssl-dev
```

2. Memory Issues with Large Datasets

```bash
# Reduce max-index parameter
./rust_sparse_index \
    --input data/genotypes_large.txt \
    --output results/large_results.txt \
    --max-index 5000000 \
    --num-tasks 4

# Monitor memory usage
free -h
```

3. Nextflow Execution Issues

```bash
# Check Nextflow configuration
nextflow info

# Clean work directory
rm -rf work/

# Run with debug information
nextflow run main.nf -with-trace -with-report
```

4. File Permission Issues

```bash
# Fix executable permissions
chmod +x rust_sparse_index
chmod +x scripts/generate_sample_data.py

# Fix data file permissions
chmod 644 data/*.txt
```

Performance Optimization

Tuning Parameters

```bash
# For small datasets (< 10K variants)
./rust_sparse_index -i data/small.txt -o results/small.txt --max-index 100000 --num-tasks 2

# For medium datasets (10K - 100K variants)
./rust_sparse_index -i data/medium.txt -o results/medium.txt --max-index 1000000 --num-tasks 8

# For large datasets (> 100K variants)
./rust_sparse_index -i data/large.txt -o results/large.txt --max-index 10000000 --num-tasks 16
```

Resource Monitoring

```bash
# Monitor CPU and memory usage
htop

# Monitor disk I/O
iotop

# Check available disk space
df -h

# Monitor network usage (if applicable)
nethogs
```

Advanced Usage

Custom Data Generation

```bash
# Generate data with specific characteristics
python3 scripts/generate_sample_data.py \
    --output data/custom_genotypes.txt \
    --num-variants 25000 \
    --num-chromosomes 22 \
    --max-position 2000000 \
    --seed 123

# Generate data without allele frequencies
python3 scripts/generate_sample_data.py \
    --output data/no_freq_genotypes.txt \
    --num-variants 15000 \
    --no-freq
```

Batch Processing

```bash
# Process multiple files
for file in data/genotypes_*.txt; do
    output="results/$(basename "$file" .txt)_results.txt"
    echo "Processing $file -> $output"
    ./rust_sparse_index -i "$file" -o "$output" --num-tasks 8
done
```

Integration with Other Tools

bash

```bash
# Convert results to different formats
# (Add custom conversion scripts as needed)

# Pipe results to downstream analysis
./rust_sparse_index -i data/genotypes.txt | awk '{print $3}' | sort -n
```

Environment Variables

bash

```bash
# Set Rust log level for detailed debugging
export RUST_LOG=debug
./rust_sparse_index -i data/genotypes.txt -o results/debug.txt

# Set number of threads for Rayon (parallel processing)
export RAYON_NUM_THREADS=8

# Set Nextflow work directory
export NXF_WORK=/tmp/nextflow_work
```

Cleanup

bash

```bash
# Clean Rust build artifacts
cargo clean

# Remove Nextflow work directory
rm -rf work/ .nextflow/

# Remove generated data (be careful!)
rm -rf data/genotypes_*.txt

# Remove all results
rm -rf results/*
```

This guide should help you successfully set up and run the genomic sparse index pipeline in your WSL environment. Adjust parameters based on your specific data size and system capabilities.