# SU-PBL GWAS Pipeline

Code generated and/or used by the Stellenbosch University Plant Breeding Laboratory.

### Require:

* Python 3.X
* R
* PLINK
* STRUCTURE
* TASSEL

### Note:

Many of the following pipelines and scripts have been developed for the Axiom 35K Wheat Breeders Array data obtained from CerealsDB (<https://www.cerealsdb.uk.net/cerealgenomics/CerealsDB/axiom_download.php>).

## Folder Directory Structure

You can create the entire directory structure using the following bash command (run from the root of your GWAS project folder):

mkdir -p \

raw\_data \

scripts \

results/filtered \

results/IBS \

results/pruned\_data \

results/ld\_decay \

results/STRUCTURE \

results/GWAS \

plink\_format \

tmp

## Microarray Data Preparation and Quality Control

### Objective:

Prepare a high-quality SNP dataset by removing low-quality markers and individuals before downstream population structure and GWAS analysis.

#### Input:

* raw\_data/raw\_genotypes.csv

Genotype data file (Axiom 35K Wheat Breeders Array): SNPs in rows, samples in columns

Genotype codes: 0 = AA, 1 = AB, 2 = BB, 9 or blank = missing

* raw\_data/35k\_probe\_set\_IWGSCv1.xlsx

SNP annotation file with chromosome and physical position

### Step 1.1: Filter SNPs by MAF and missingness

python scripts/01\_filter\_maf\_missing.py \

raw\_data/raw\_genotypes.csv \

results/filtered/filtered\_genotypes.csv \

5 5

* Filters SNPs with MAF < 5% and missing data ≥ 5%
* Missing values coded as '9' or blank are excluded from calculations

#### Outputs:

* results/filtered/filtered\_genotypes.csv
* results/filtered/filtered\_snp\_list.txt

### Step 1.2: Remove SNPs not mapped to the reference genome

python scripts/02\_filter\_snps\_mapped.py \

results/filtered/filtered\_genotypes.csv \

raw\_data/35k\_probe\_set\_IWGSCv1.xlsx \

results/filtered/filtered\_genotypes\_mapped.csv

#### Output:

* results/filtered/filtered\_genotypes\_mapped.csv

### Step 1.3: Remove individuals with ≥ 5% missing data

python scripts/03\_filter\_individuals\_missingness.py \

results/filtered/filtered\_genotypes\_mapped.csv \

results/filtered/filtered\_genotypes\_strict.csv

Output:

* results/filtered/filtered\_genotypes\_strict.csv
* results/filtered/filtered\_genotypes\_strict\_removed\_individuals.txt

### Step 1.4: Transpose genotype matrix

python scripts/04\_transpose\_genotypes.py \

results/filtered/filtered\_genotypes\_strict.csv \

results/IBS/filtered\_genotypes\_strict\_T.csv

* Converts SNPs-in-rows format to SNPs-in-columns

#### Output:

* results/IBS/filtered\_genotypes\_strict\_T.csv

### Step 1.5: Calculate IBS matrix (Python)

python scripts/05\_calculate\_ibs\_matrix.py \

results/IBS/filtered\_genotypes\_strict\_T.csv \

results/IBS/python\_ibs/ibs\_matrix.csv

#### Output:

* results/IBS/python\_ibs/ibs\_matrix.csv

### Output Summary:

* filtered\_genotypes.csv – SNPs passing MAF/missingness filters
* filtered\_genotypes\_mapped.csv – SNPs mapped to genome
* filtered\_genotypes\_strict.csv – High-quality individuals retained
* filtered\_genotypes\_strict\_T.csv – Transposed genotype matrix
* ibs\_matrix.csv – Pairwise sample IBS similarity scores

### Rationale:

Filtering based on MAF and missingness ensures data reliability. Removing unmapped SNPs and low-quality individuals prevents bias. Transposing and computing IBS enables individual-level similarity checks used in population structure and QC.

## 2. PLINK Formatting and IBS/LD Decay Analysis

### Objective:

Convert the filtered SNP dataset into PLINK-compatible format and compute identity-by-state (IBS) matrices and linkage disequilibrium (LD) decay to inform SNP pruning.

#### Input:

* results/filtered/filtered\_genotypes\_strict.csv

Final high-quality SNP dataset (after MAF, mapping, and missingness filters)

* raw\_data/35k\_probe\_set\_IWGSCv1\_cleaned\_tab.map

Tab-delimited file with SNP ID, chromosome, and physical position (converted from original .xlsx)

### Step 2.1: Generate PLINK .map file

python scripts/06\_create\_plink\_map.py

#### Output:

* plink\_format/filtered\_genotypes\_strict.map

### Step 2.2: Generate PLINK .tfam file

plink --file plink\_format/filtered\_genotypes\_strict \

--recode \

--allow-extra-chr \

--out plink\_format/filtered\_genotypes\_strict

#### Output:

* plink\_format/filtered\_genotypes\_strict.tfam

### Step 2.3: Generate .tped file from transposed matrix

python scripts/07\_transpose\_Tcsv\_to\_tped.py \

results/IBS/filtered\_genotypes\_strict\_T.csv \

plink\_format/filtered\_genotypes\_strict.tped \

plink\_format/filtered\_genotypes\_strict.map

#### Output:

* plink\_format/filtered\_genotypes\_strict.tped

### Step 2.4: Compute IBS matrix using PLINK

plink --file plink\_format/filtered\_genotypes\_strict \

--allow-extra-chr \

--cluster \

--matrix \

--out results/IBS/plink\_ibs/ibs\_matrix\_strict\_qc

#### Outputs:

* ibs\_matrix\_strict\_qc.mibs: IBS similarity matrix (range 0–1)
* ibs\_matrix\_strict\_qc.mibs.id: Sample identifiers
* ibs\_matrix\_strict\_qc.log: Log file

### (Optional) Compare with the Python IBS matrix:

* Python IBS: results/IBS/ibs\_matrix.csv
* PLINK IBS: results/IBS/plink\_ibs/ibs\_matrix\_strict\_qc.mibs

Comparison of matrices (e.g., via clustering or correlation) can validate consistency.

## LD Decay Analysis:

### Step 2.5: Compute pairwise r² values using PLINK

plink --file plink\_format/filtered\_genotypes\_strict \

--r2 \

--allow-extra-chr \

--ld-window-r2 0 \

--ld-window 999999 \

--ld-window-kb 600000 \

--out results/ld\_decay/filtered\_genotypes\_strict

#### Output:

* results/ld\_decay/filtered\_genotypes\_strict.ld

### Step 2.6: Summarize LD decay by distance

cat results/ld\_decay/filtered\_genotypes\_strict.ld | \

sed 1,1d | \

awk -F "\t" 'function abs(v){return v < 0 ? -v : v} BEGIN{OFS="\t"}{print abs($5-$2), $7}' | \

sort -k1,1n > results/ld\_decay/filtered\_genotypes\_strict.ld.summary

#### Output:

* results/ld\_decay/filtered\_genotypes\_strict.ld.summary

### Step 2.7: Plot LD decay (R)

Rscript scripts/16\_ld\_decay\_plot.R

#### Output:

* results/ld\_decay/LD\_decay\_plot.png

### Output Summary

* .map, .tped, .tfam PLINK files for filtered dataset
* Python and PLINK-derived IBS matrices
* LD decay summary and visual plot

### Rationale:

Creating PLINK-compatible files is necessary for pruning and downstream association tools. Computing LD decay informs pruning thresholds, while IBS matrices support population structure inference and outlier detection.

## 3. LD Pruning and Generation of Pruned Genotype Files

### Objective:

Remove redundant, highly correlated SNPs by pruning based on linkage disequilibrium (LD), resulting in a set of unlinked markers suitable for population structure and GWAS analyses.

#### Inputs:

* plink\_format/filtered\_genotypes\_strict.tped
* plink\_format/filtered\_genotypes\_strict.tfam
* plink\_format/filtered\_genotypes\_strict.map
* results/filtered/filtered\_genotypes\_strict.csv – Full genotype matrix

These files represent the high-quality, mapped, and transposed SNP genotype dataset from previous steps.

### Step 3.1: LD pruning in PLINK

plink --tfile plink\_format/filtered\_genotypes\_strict \

--allow-extra-chr \

--indep-pairwise 50 5 0.2 \

--out results/pruned\_data/ld\_pruned\_genotypes\_strict

#### Parameters:

* 50 = window size (in SNPs)
* 5 = step size
* 0.2 = r² threshold (from LD-decay)

#### Outputs:

* ld\_pruned\_genotypes\_strict.prune.in: SNPs retained
* ld\_pruned\_genotypes\_strict.prune.out: SNPs removed
* ld\_pruned\_genotypes\_strict.log: PLINK log

### Step 3.2: Extract LD-pruned SNP subset (Python)

python scripts/08\_extract\_snp\_subset.py \

results/pruned\_data/ld\_pruned\_genotypes\_strict.prune.in \

results/filtered/filtered\_genotypes\_strict.csv \

results/pruned\_data/ld\_pruned\_genotypes\_strict.csv

#### Output:

* ld\_pruned\_genotypes\_strict.csv: Final LD-pruned SNP matrix (for Python-based IBS or PCoA)

### Step 3.3: Transpose pruned genotype matrix (for Python-based analysis)

python scripts/04\_transpose\_genotypes.py \

results/pruned\_data/ld\_pruned\_genotypes\_strict.csv \

results/pruned\_data/ld\_pruned\_genotypes\_strict\_T.csv

#### Output:

* ld\_pruned\_genotypes\_strict\_T.csv: Transposed matrix of unlinked SNPs

### Output Summary:

* ld\_pruned\_genotypes\_strict.prune.in: List of retained unlinked SNPs
* ld\_pruned\_genotypes\_strict.csv: LD-pruned genotype matrix
* ld\_pruned\_genotypes\_strict\_T.csv: Transposed version for STRUCTURE, IBS, or PCoA

### Rationale:

LD pruning reduces SNP redundancy and population structure bias. This streamlined SNP set improves computational efficiency and statistical robustness for downstream dimensionality reduction, STRUCTURE, and GWAS.

## 4. PLINK Binary Conversion and LD-Pruned IBS Matrix Generation

### Objective:

Convert the LD-pruned genotype matrix into PLINK binary format for efficient processing, and recompute the IBS and dissimilarity matrices using the reduced SNP set.

#### Inputs:

* results/pruned\_data/ld\_pruned\_genotypes\_strict\_T.csv

Transposed matrix of LD-pruned SNPs

* plink\_format/filtered\_genotypes\_strict.tfam

Sample metadata from previous PLINK step

* plink\_format/filtered\_genotypes\_strict.map

SNP coordinate file

### Step 4.1: Generate .tped file from pruned transposed matrix

python scripts/07\_transpose\_Tcsv\_to\_tped.py \

results/pruned\_data/ld\_pruned\_genotypes\_strict\_T.csv \

plink\_format/filtered\_genotypes\_strict.tped \

plink\_format/filtered\_genotypes\_strict.tfam

#### Output:

* plink\_format/filtered\_genotypes\_strict.tped

### Step 4.2: Convert .tped/.tfam to PLINK binary format

plink --tfile plink\_format/filtered\_genotypes\_strict \

--allow-extra-chr \

--make-bed \

--out results/pruned\_data/filtered\_genotypes\_strict\_qc

#### Outputs:

* filtered\_genotypes\_strict\_qc.bed
* filtered\_genotypes\_strict\_qc.bim
* filtered\_genotypes\_strict\_qc.fam

## Recalculate IBS and Dissimilarity Matrix (Python)

### Step 4.3: Transpose LD-pruned genotype matrix

python scripts/04\_transpose\_genotypes.py \

results/pruned\_data/ld\_pruned\_genotypes\_strict.csv \

results/pruned\_data/ld\_pruned\_genotypes\_strict\_T.csv

#### Output:

* ld\_pruned\_genotypes\_strict\_T.csv (already generated in Step 3.3, reused here)

### Step 4.4: Calculate IBS matrix on LD-pruned data

python scripts/05\_calculate\_ibs\_matrix.py \

results/pruned\_data/ld\_pruned\_genotypes\_strict\_T.csv \

results/IBS/python\_ibs/ibs\_matrix\_ldpruned.csv

#### Output:

* ibs\_matrix\_ldpruned.csv: Pairwise identity-by-state matrix

### Step 4.5: Convert IBS to dissimilarity matrix

python scripts/09\_convert\_psm\_to\_dissimilarity.py \

results/IBS/python\_ibs/ibs\_matrix\_ldpruned.csv \

results/IBS/python\_ibs/dissimilarity\_matrix\_ldpruned.csv

#### Output:

* dissimilarity\_matrix\_ldpruned.csv: Distance matrix (1 - IBS)

### Output Summary:

* PLINK binary files: .bed, .bim, .fam
* LD-pruned IBS matrix: ibs\_matrix\_ldpruned.csv
* Dissimilarity matrix: dissimilarity\_matrix\_ldpruned.csv

### Rationale:

Binary PLINK files are required for efficient memory handling in tools like STRUCTURE and TASSEL. Recalculating IBS on pruned data avoids correlation inflation and ensures accuracy in structure and kinship modelling.

### 5. Dimensionality Reduction via Principal Coordinates Analysis (PCoA)

### Objective:

Reduce the high-dimensional dissimilarity matrix into a small number of axes (principal coordinates) to visualize genetic structure and guide downstream clustering or association analysis.

#### Input:

* results/IBS/python\_ibs/dissimilarity\_matrix\_ldpruned.csv

Pairwise dissimilarity matrix (1 - IBS)

* (Optional): results/IBS/python\_ibs/genotype\_labels.txt

Sample metadata file (for coloring samples by group/population in the plot)

### Step 5.1: Run PCoA in R

Rscript scripts/10\_dimensionality\_reduction\_pco.R

### This script:

* Performs classical multidimensional scaling (PCoA)
* Plots the first two principal coordinate axes (PC1 vs PC2)
* Supports optional coloring by sample group

#### Outputs:

* results/IBS/python\_ibs/pcoa\_coordinates.csv: PC1, PC2, PC3 for each sample
* results/plots/pcoa\_plot.png: 2D PCoA scatterplot

### Output Summary:

* pcoa\_coordinates.csv: Principal coordinate values
* pcoa\_plot.png: Visual representation of genetic structure

### Rationale:

PCoA condenses complex genetic relationships into key dimensions, enabling detection of population structure, stratification, and outlier samples. It also provides useful input for STRUCTURE interpretation and GWAS covariates.

## 6. STRUCTURE Analysis Setup and Execution

### Objective:

Detect population structure among genotypes by inferring genetic clusters using the STRUCTURE software and a diploid genotype matrix formatted to its specification.

#### Input:

* results/pruned\_data/ld\_pruned\_genotypes\_strict\_T.csv

Transposed genotype matrix of LD-pruned SNPs (genotypes in rows, SNPs in columns)

### Step 6.1: Extract individual labels (no header)

python scripts/11\_extract\_labels.py \

results/pruned\_data/ld\_pruned\_genotypes\_strict\_T.csv \

results/STRUCTURE/ld\_pruned\_T\_noheader.txt

#### Output:

* ld\_pruned\_T\_noheader.txt: List of sample names (no header)

### Step 6.2: Convert to STRUCTURE diploid format

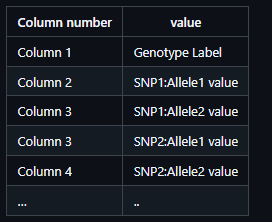
python scripts/12\_to\_structure.py \

results/STRUCTURE/ld\_pruned\_T\_noheader.txt \

results/STRUCTURE/structure\_input.txt

#### Output:

* structure\_input.txt: STRUCTURE-compatible file in diploid encoding



### Step 6.3: Run STRUCTURE (external software)

Run STRUCTURE with the following parameters:

|  |  |
| --- | --- |
| Parameter | Value |
| Admixture model | ON |
| Burn-in period | 10000 |
| MCMC reps after burn-in | 10000 |
| K (clusters) tested | 1 to 10 |
| Replicates per K | 10 |

Be sure to prepare mainparams and extraparams files.

Input file: structure\_input.txt

### Output Summary:

* structure\_input.txt: STRUCTURE-formatted diploid genotype matrix
* STRUCTURE output files (1–10 clusters × 10 reps each)

### Rationale:

STRUCTURE uses genotype likelihoods to infer hidden population groups and admixture proportions. Results guide downstream correction for population structure and biological interpretation.

## 7. Optimal K Inference and STRUCTURE Result Alignment

### Objective:

Identify the best number of population clusters (K) using the Evanno method and align cluster assignments across STRUCTURE replicates using CLUMPAK.

#### Input:

* STRUCTURE result directories:
* Located in:
* results/STRUCTURE/STRUCTURE\_HARVESTER/harvester\_input/
* Example contents:
* INEW\_GWAS\_MUDAU\_FINAL\_P\_run\_1\_f/
* INEW\_GWAS\_MUDAU\_FINAL\_P\_run\_2\_f/
* ...
* INEW\_GWAS\_MUDAU\_FINAL\_P\_run\_50\_f/

### Step 7.1: Run STRUCTURE HARVESTER

Use the script provided by the STRUCTURE HARVESTER tool to determine the best K using the Evanno method:

python structureHarvester\_scripts/structureHarvester.py \

--dir results/STRUCTURE/STRUCTURE\_HARVESTER/harvester\_input \

--out results/STRUCTURE/STRUCTURE\_HARVESTER/harvester\_output \

--evanno

#### Outputs:

* evanno.txt: Evanno method ΔK scores for K=1 to 10
* summary.txt: Summary statistics for each tested K

Location:

* results/STRUCTURE/STRUCTURE\_HARVESTER/harvester\_output/

### Step 7.2: Align clusters using CLUMPAK

**Manually prepare a zip file for CLUMPAK.**

1. Create zipped sets of STRUCTURE runs for each K:

* K1.zip → contains all 10 runs for K=1
* K2.zip → contains all 10 runs for K=2
* ...
* K10.zip

1. Combine into a master archive:

structure\_results.zip

├── K1.zip

├── K2.zip

...

└── K10.zip

1. Submit structure\_results.zip to CLUMPAK online:

* <https://clumpak.tau.ac.il/index.html>

#### Output (from CLUMPAK):

* Cluster alignment summary plots
* Barplots of individual assignment probabilities
* Consensus clustering outputs

### Output Summary:

* evanno.txt, summary.txt: Optimal K determination
* CLUMPAK output: aligned cluster labels and visualisations

### Rationale:

STRUCTURE HARVESTER helps choose the most likely number of genetic groups (K) based on the ΔK method. CLUMPAK resolves label switching between replicates, enabling coherent clustering results across STRUCTURE runs.

## 8. Prepare GWAS Input Files for TASSEL

### Objective:

Extract LD-pruned SNPs from the full genotype matrix, convert them to PLINK binary and PED/MAP formats, and prepare input files for TASSEL-based GWAS.

#### Input:

* results/pruned\_data/filtered\_genotypes\_strict\_qc.bed
* results/pruned\_data/filtered\_genotypes\_strict\_qc.bim
* results/pruned\_data/filtered\_genotypes\_strict\_qc.fam
* results/pruned\_data/ld\_pruned\_genotypes\_strict.prune.in

→ List of retained SNPs after LD pruning

### Step 8.1: Extract LD-pruned SNPs in binary format

plink --file results/pruned\_data/filtered\_genotypes\_strict\_qc \

--extract results/pruned\_data/ld\_pruned\_genotypes\_strict.prune.in \

--make-bed \

--allow-extra-chr \

--out results/pruned\_data/pruned\_data

#### Outputs:

* pruned\_data.bed
* pruned\_data.bim
* pruned\_data.fam

### Step 8.2: Convert BED to PED/MAP for TASSEL

plink --bfile results/pruned\_data/pruned\_data \

--recode \

--tab \

--allow-extra-chr \

--out results/pruned\_data/pruned2

#### Outputs:

* pruned2.ped or pruned2\_fixed.ped (used in TASSEL)
* pruned2.map

Note: You may rename or manually correct pruned2.ped to pruned2\_fixed.ped if any format adjustments are needed (e.g. allele naming).

### Output Summary:

* pruned\_data.bed/.bim/.fam: Binary files for efficient storage and access
* pruned2.ped/.map: PLINK PED/MAP files for TASSEL

### Rationale:

TASSEL requires input in PED/MAP format. These files represent a clean, LD-pruned set of high-confidence markers, enabling accurate and efficient GWAS in the TASSEL GUI environment.

## 9. GWAS Analysis in TASSEL

(https://avikarn.com/2019-07-22-GWAS/)

### Objective:

Perform GWAS using the TASSEL GUI, incorporating population structure (via MDS/PCA) and relatedness (via kinship matrix), using the MLM (mixed linear model) and GLM (general linear model) approaches.

#### Input:

* results/pruned\_data/pruned2.ped
* results/pruned\_data/pruned2.map
* Phenotypic data file in TASSEL-compatible format:

A screenshot of a computer

AI-generated content may be incorrect.

### TASSEL GUI Steps

### Step 9.1: Load genotype data

1. File → Open As → PLINK PED
2. Select: pruned2.ped and pruned2.map

### Step 9.2: Genomic Summary Statistics

1. Sequence View → Data → Geno Summary

* Inspect minor allele frequency distribution
* Check missing genotype rates
* Assess heterozygosity for quality and selfing

### Step 9.3: Multidimensional Scaling (MDS)

1. Sequence View → Analysis → Relatedness → Distance Matrix
2. Select pruned2
3. Analysis → Relatedness → MDS

#### Outputs:

* MDS\_Eigenvalues\_Matrix
* MDS\_PCs\_Matrix

(These serve as covariates in MLM model)

### Step 9.4: Kinship Matrix (Centered IBS)

1. Sequence View → Analysis → Relatedness → Kinship
2. Select: Centered IBS method

#### Output:

* Centered\_IBS\_pruned2 (under Matrix)

### Step 9.5: Load phenotypic data

1. File → Open As → Phenotype
2. Select your phenotype file (formatted as shown above)

### Step 9.6: Join genotype, phenotype, and PCA tables

1. Data → Intersect Join
2. Select genotype (pruned2), phenotype file, and MDS\_PCs\_Matrix

### Step 9.7: Run Association Models

1. Mixed Linear Model (MLM: PCA + Kinship)
2. Analysis → Associations → MLM
3. Use joined table and kinship matrix as input

#### Review:

* Q-Q plot
* Manhattan plot
* Export SNP statistics

### Output Summary:

* Q-Q plots and Manhattan plots per trait
* SNP-level association statistics
* Joined matrix combining genotype, phenotype, and structure

### Rationale

TASSEL enables user-friendly GWAS with support for correcting confounding from structure (PCA) and relatedness (kinship). The MLM model is the standard approach for identifying robust trait–marker associations.

## 10. Post-GWAS Processing and Visualization

### Objective:

Split MLM output into trait-specific result files, apply multiple testing correction, and visualize significant associations across traits.

#### Input:

* results/GWAS/mlm\_stats.txt

MLM result file from TASSEL (tab-delimited; contains all traits)

### Step 10.1: Split MLM results by trait

Rscript scripts/14\_extract\_traits\_only.R

#### Input:

* mlm\_stats.txt

#### Output:

* One CSV file per trait, e.g.:
* results/GWAS/mlm\_stats\_tiller\_height.csv
* results/GWAS/mlm\_stats\_grain\_weight.csv
* ...

### Step 10.2: Adjust p-values and identify significant SNPs

Rscript scripts/15\_adjust\_pvalues\_plot.R

#### Inputs:

* All trait-specific MLM files (from Step 10.1)

#### Outputs:

* adj\_p\_<trait>.csv: P-values adjusted via Bonferroni and FDR
* top\_10\_<trait>.csv: Top 10 SNPs per trait
* significant\_snps\_all\_traits.csv: Summary of significant associations

### Output Summary:

* Trait-specific adjusted results (adj\_p\_\*.csv)5
* Top hits per trait (top\_10\_\*.csv)
* Cross-trait summary (significant\_snps\_all\_traits.csv)
* Plots: Manhattan and Q-Q plots per trait (download directly from TASSEL)

### Rationale:

Post-GWAS filtering ensures that reported SNP–trait associations are statistically significant after correcting for multiple testing. Clear plots and summaries facilitate interpretation, reporting, and candidate gene prioritization.

## Step 11: Candidate gene identification

#### What You Need:

For each significant SNP:

* SNP name (e.g. AX-94629608)
* Chromosome and position (e.g. chr2A:14,327,950)
* Flanking sequence (e.g. 250–500 bp)

### Step 11.1. BLAST the SNP sequence (NCBI)

#### Use NCBI BLAST:

* Paste the SNP flanking sequence
* Tool: BLASTn
* Organism: Triticum aestivum
* Get the top hit:
  + Gene/protein name
  + Accession number
  + % identity and E-value

### Step 11.2. Check gene location (Ensembl Plants)

#### Go to Ensembl Plants:

* Search for the gene using the accession number or name from NCBI
* Note the chromosome and start–end position

### Step 11.3. Confirm if gene is within ±1 Mbp of the SNP

#### Calculate SNP LD window:

* SNP position window (from LD-decay) - eg 1,000,000 to SNP position + 1,000,000
* If gene start or end falls in that window → ✅ Candidate gene