Julius-Maximilians-Universität Würzburg

Master Biosciences (Master of Science)

Genome-Wide Association Studies to Infer Signs of Selection in *Arabidopsis thaliana*

Master Thesis

Leyla Sırkıntı (2656019)

Supervisor PD Arthur Korte

Written at the

Center for Computational and Theoretical Biology

21.06.2022

Supervisor PD Arthur Korte

Second reviewer Dr. Shishir Kumar Gupta

Date of Submission, office stamp

Zusammenfassung

Genomweite Assoziationsstudien (GWAS) sind eine Methode zur Verknüpfung genetischer Variationen mit Phänotypen, die in den letzten zwei Jahrzehnten an Popularität gewonnen hat [1]. Die GWAS begannen mit der Forschung am Menschen, um Risikogene zu ermitteln, die Krankheiten verursachen könnten. Später wurde sie aufgrund der wachsenden Bevölkerung auch als Methode zur Gewinnung wirtschaftlich effizienterer und ertragreicherer Pflanzenarten eingesetzt. Eine weitere Anwendung von GWAS besteht darin, die evolutionäre Bedeutung natürlicher Variationen von Arten zu verstehen und dadurch deren Evolution zu ergründen.

Arabidopsis thaliana spielt eine wichtige Rolle in der Evolution und die Auswirkungen genetischer Variationen auf die evolutionäre Genomik wurden in dieser Studie mit Hilfe von Mutationen, die durch GWAS identifiziert wurden, genauer untersucht. Es gibt eine Vielzahl von Mutationen die in allen Lebewesen auftreten. Unter diesen Mutationen ist das vorzeitige Stoppcodon (Non-Sense-Mutation) eine gefährliche Mutation, die zu einem unvollständigen Protein führt [2]. Frühere Studien haben gezeigt, dass durch das vorzeitige Stoppcodon in derselben Akzession von *A. thaliana* übermäßig viele Genpaare entstanden sind. Es wurde vermutet, dass ein regulatorisches Gen die Bildung der vorzeitigen Stoppcodons in denselben Akzessionen beeinflusst. Diese regulatorischen Gene werden in dieser Arbeit als trans-Gene bezeichnet. In dieser Studie wurden aus 1135 Akzessionen von *Arabidopsis thaliana* 1128 signifikant simultan auftretende Genpaare verwendet. Insgesamt wurden 32 signifikante Gene (trans-Gene) identifiziert, die bei der Bildung dieser Assoziationen eine Rolle spielen. Gemeinsame trans-Gene in den Inversionsversionen dieser simultan auftretenden Genpaare wurden zur Bestätigung bei der Auswahl der signifikanten trans-Gene herangezogen. Diese 32 trans-Gene werden im Gennetzwerk so dargestellt, dass die Wechselwirkungen von trans-Genen auf simultan auftretende Genpaare eingehender untersucht werden können. Die Auswirkungen signifikanter trans-Gene auf die vorzeitige Stoppcodon-Bildung von Genpaaren in derselben Akzession wurden durch diese Studie deutlich. Obwohl der, für den nicht normal verteilten Phänotyp in den GWAS-Ergebnissen, verwendete Schwellenwert angemessen war, wurde festgestellt, dass sowohl mit dem Manhattan-Plot als auch mit dem Quantil-Quantil-Plot mehr falsch-positive Ergebnisse erkannt wurden als erwartet. Es wurde versucht, mit Hilfe der Ergebnisse der GWAS, kausale Varianten zu identifizieren und die Bedeutung dieser Gene für die Bildung des vorzeitigen Stoppcodons zu bestimmen. Der Nachweis von kausalen Varianten und die Identifizierung von SNPs, die eine vorzeitige Stoppcodonbildung verursachen können, erwies sich jedoch als sehr schwierig. Die Faktoren, die zu dieser Situation führen, werden noch erforscht.

Außerdem wird nach alternativen Methoden gesucht, um die Ursachen für diese Verzerrung aufzulösen.
Zusammenfassend lässt sich sagen, wurden in der Studie die Gene AT2G21830, AT1G08840 und AT2G18735 als wichtige Regulatorgene identifiziert. Dass es zwar immer noch zu einer Verzerrung der SNP-Ergebnisse kommen kann, dass aber die Bildung vorzeitiger Stoppcodon-Genpaare von trans-Genen in denselben Akzessionen die evolutionäre Bedeutung bei künftigen Bestätigungstests aufzeigen soll.

Summary

Genome-Wide Association Studies (GWAS) are methods of associating genetic variations with phenotypes that have gained popularity in the last two decades [1]. The GWAS began with human research to identify risk genes that could cause diseases. Later it was also used as a method of obtaining more economically efficient and yielding plant species due to the increasing population. Another application of GWAS is to comprehend the evolutionary significance of natural variations in species. *Arabidopsis thaliana* plays an important role as model plant in evolution, and the effect of genetic variations on evolutionary genomics was investigated using mutations identified through GWAS in this study.

Various mutations occur in all living organisms. Among these mutations, the premature stop codon (nonsense mutations) is a dangerous mutation and causes protein truncation [2]. Previous research showed that over co-occurrence gene pairs were formed as a result of premature stop codons in the same *A. thaliana* accessions. It was hypothesized that there is a regulatory genes effect on the formation of the premature stop codons in the same accessions. These regulatory genes called as trans genes in this thesis. A total of 1128 *A. thaliana* significant over co-occurrence gene pairs from 1135 accessions were used in this study and, 32 significant regulatory genes (trans genes) that played a role in the formation of these associations were identified. The over co-occurrence gene pairs cause the formation of 32 trans genes and these trans genes are shown in the same gene network, allowing the interactions of trans genes on over co-occurrence gene pairs to be explored in greater depth. Permutation based GWAS was used in accordance with the threshold used for the non-normally distributed phenotype in the GWAS results. However, both the Manhattan plots and the Quantile-Quantile plots detected more false positive SNPs than expected. It was attempted to determine causal variants from the GWAS results and the importance of the respective genes in the formation of the premature stop codon. However, detection of causal variants and the identification of SNPs that can cause premature stop codon formation proved to be very difficult. The factors that will cause this situation are still being researched and alternative methods are being sought to eliminate the causes of false positive SNPs.

In conclusion, the AT2G21830, AT1G08840, and AT2G18735 genes were identified as important regulatory genes in the study. It aims to demonstrate the impact of these important regulatory genes on the formation of premature stop codons in the same accessions, with future validation tests.

Table of Contents

1. Introduction

1.1. GWAS Overview

Genome-wide association studies (GWAS) are a widely used methods for linking genetic variants associated with different phenotypes on a population scale [3]. GWAS focuses on single nucleotide polymorphisms (SNPs) to explain the phenotypic association with genotypic variations. SNPs are single nucleotide changes in DNA sequences that are caused by genetic variations in the genome and are used as a genetic marker in GWAS to identify genetic variants. Most individuals in the population have the same nucleotide at the same position in the genome, but a few individuals have a different nucleotide at the same position, which is known as an SNP. Non-synonymous SNPs occur in gene coding regions and alter protein synthesis or inhibit protein synthesis. However, not all SNPs need to have an effect. Synonymous SNPs could alter protein synthesis, but could also just be neutral. As a result of these differences, each individual becomes unique in the population [4]. The vast majority of SNPs have a minor impact on the genome, but the result of mRNA changes, amino acid changes, and other transcription-related changes have a direct impact on genomic stability. SNPs also occur in non-coding regions of the genome, and it is thought that there is a significant association between changes in the non-coding region of the genome and increased risk for certain traits and diseases.

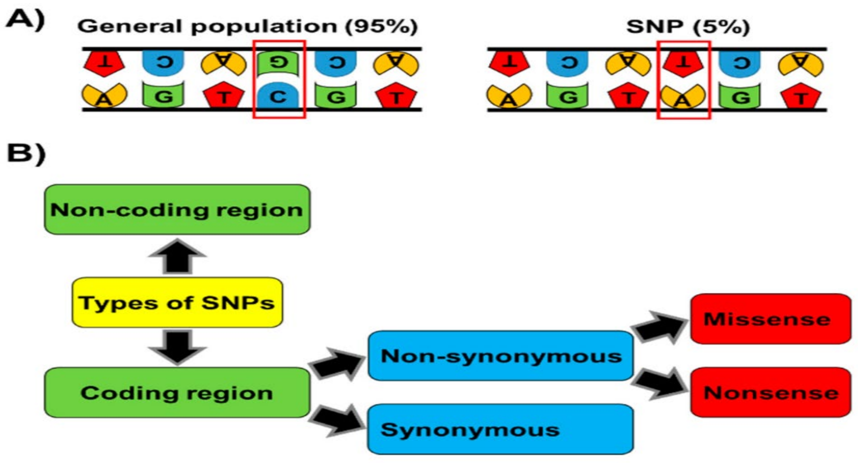


Figure 1. 1 Single Nucleotide Polymorphism (SNP) Representation and Types

(A) The majority of the population (95%) have the same nucleotide at the same position in the genome, while a few individuals have a different nucleotide at the same position, which is called SNPs. (B) SNPs are classified according to whether they occurred in non-coding or coding regions. SNPs occurred at the coding region categorized if this mutation causes an amino acid change that is called non-synonymous, if not called synonymous. Non-synonymous SNPs are categorized as missense (producing different proteins) and nonsense (producing truncated protein) (Figure from [5]).

The possible explanation for this reason, is that SNPs changes occur at the gene expression level, not protein function or there might be linkage disequilibrium with causative SNPs, and these are just tacking it [6]. SNPs are classified into several types (Figure 1), and this thesis focuses on Nonsense SNPs. Nonsense SNPs are a non-synonymous subtype (change in the protein due to a nucleotide change in the amino acid) that occurs in the coding region of the genome, causing the formation of stop codons in the gene, resulting in the formation of incomplete proteins that lead to protein truncation. This type of stop codon is known as a premature stop codon [5].

SNPs are used in GWAS as genomic markers to show which specific regions of the genome are responsible for phenotypic variations [7]. Certain SNPs are usually found in a specific population and associated with specific phenotypes. These SNPs are either causative or associated with causative mutations. GWAS provides results in explaining the relationship of genetic variants (SNPs) with traits after population and phenotype selection [8]. Millions of genetic variants in the genome of numerous individuals are measured in a given population to relate genotype to phenotype. One of the most important applications of GWAS is to identify some of the genes that were not previously associated with the traits or diseases of interest, so the analysis of GWAS reveals some of these unknown functions of the gene relationships. Identification of novel variants associated with traits can facilitate a deeper understanding of biological mechanisms.

1.2. GWAS Applications and Usage: Medicine and Plant Breeding

The Human Haplotype Map Project is the lead project of the GWAS study. Following this study, diseases began to be associated with changes in the genome [9]. GWAS analysis has had a significant impact on genetic disease research over the last decade because it is primarily used to identify genetic variants associated with human genetic diseases. The GWAS strategy is a screening of many different human genomes using genetic markers to predict the presence of disease. When genetic markers are defined, they are used to understand how genes are associated and affect disease formation, which helps in disease prevention or the development of new treatment strategies [10].

Despite initially focusing on human studies, GWAS strategies have been applied to other organisms and phenotypic variants for a variety of purposes. The primary goal of analyzing genomic variants was gene identification. However, it also resulted in the reconstruction of population history, the probability of SNP heritability, the genetic correlation between traits, and quality control for next-generation sequencing validation of new analytic strategies [11]. The world's growing population causes to increase in the demand for raw materials therefore GWAS studies in plant breeding gained importance. As a result, the GWAS study on agricultural plants was initiated and is still ongoing. The genome of plants has been associated with traits such as height, growth time and, seed weight. Genetic variations of maize, rice, wheat, and agriculturally important plant are being studied in particular to develop the most economically suitable traits with high yields [12].

1.3. GWAS Data Obtaining Methods

The most important step in GWAS is to select samples with specific traits or diseases that will be used to analyze these samples in the next steps. GWAS data are obtained from numerous samples in various populations that carry specific diseases or have specific traits for further research [8]. However, for inbred species such as Arabidopsis, we can rely on existing genomic data and don't need to isolate new DNA every time. The next step is to isolate DNA from samples in order to genotype them to detect genetic variants, which are obtained using array-based technologies to detect SNPs (Figure 1. 2). Variants (SNPs) are identified by comparing them to reference genomes [10].

Whole-genome sequencing was an expensive method of analyzing genetic variants; however, thanks to technological advancements, it is now relatively inexpensive if not deeply sequenced. As an alternative to whole-genome sequencing, a microarray/next-generation sequencing-based technology that detects only SNPs on the genome has been developed [12]. SNPs array, a type of DNA microarray, is a method for detecting single nucleotide polymorphisms in populations. The use of SNP-based arrays in the GWAS study is more effective in detecting genetic variations but mainly used for non-model species. It is less expensive, more diverse, and easier to use than other methods. SNPs arrays contain immobilized short synthetic DNA fragments. The fluorescently dyed nucleic acid sequences to be analyzed are added to this SNPs array, and genetic variations are detected [13].

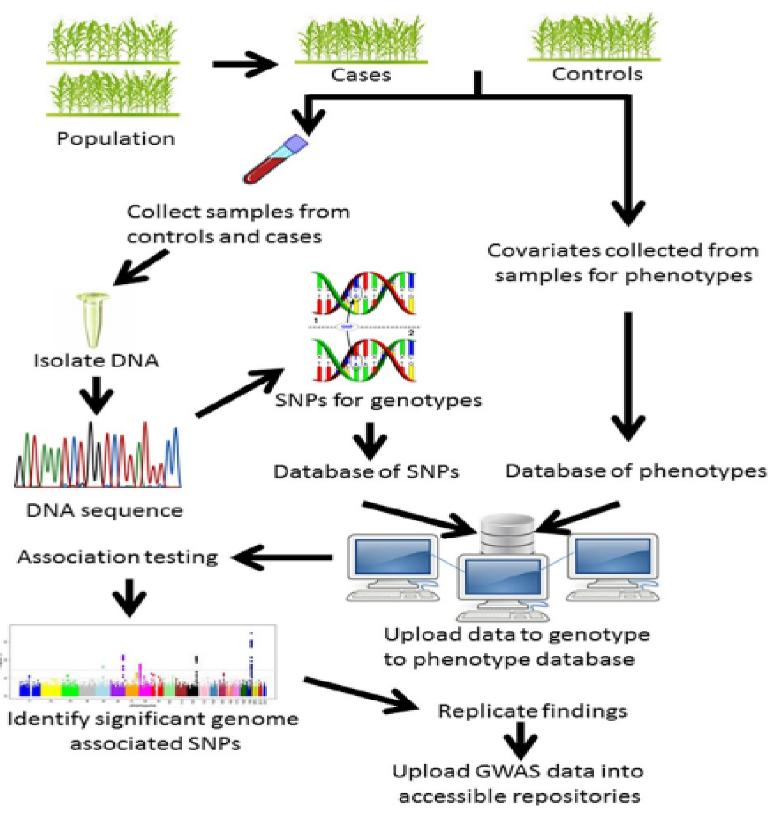

Population
Cases
Controls
Collect samples from controls and cases
Covariates collected from samples for phenotypes
Isolate DNA
SNPs for genotypes
DNA sequence
Database of SNPs
Database of phenotypes
Association testing
Upload data to genotype to phenotype database
Identify significant genome associated SNPs
Replicate findings
Upload GWAS data into accessible repositories

Figure 1. 2 Genome-Wide Association Studies (GWAS) Data Obtaining Process
SNPs occurring in the DNA sequences obtained from the population are added to the database as genomic data, and the phenotypic features obtained from the same population are added to the database in the form of binary code. With these data obtained, GWAS is analyzed and a relation is established between genotype and phenotype (Figure from [13]).

Quality control is required to proceed with the analysis to eliminate errors and increase the significance of the results. These data filtering steps include removing false positive and false negative errors caused by genotyping errors, sample duplication, and improbable sample relationships. Phenotype characteristics are collected from both the case and control groups, which is a concept used in GWAS disease research. This concept is not required for non-disease research. Different phenotype traits are collected and, data information must be generated for the phenotype. A relationship is established between the SNPs information in the created genome and the phenotype information. The methods used to collect data are tailored to the study's objectives. According to the results, the locations of significant SNPs in the genome are determined and their importance is investigated [14].

1.4. Filtering of GWAS Data and Interpretation of the Results

Identifying significant SNPs is essential in GWAS analysis. It is ensured that the obtained results are not the result of random chance, but are within a certain significance value, and the study proceeds by these findings. To establish a true relationship between the identified SNPs and the phenotype, it must be determined that the statistical study's reliability is within the confidence interval. As a result, non-significant SNPs must be filtered out of the GWAS results to improve reliability [15].

Minor allele frequency (MAF) reflects the frequency of the rare allele at each position. Studies have shown that rare allele has an effect on heredity [16]. Human studies [15] have shown that usually the minor allele has a greater impact on disease occurrence than the major allele. This is explained by negative natural selection, which means that minor alleles influence the occurrence of risk genes. Negative natural selection removes risk alleles that cause Mendelian diseases because it has negative fitness effects that cause disease formation. However, according to the studies (from [17],[18]), the speed of negative natural selection does not have enough time to remove minor alleles that affect environmental and lifestyle-dependent diseases. Another explanation is that major alleles work to prevent disease occurrence. The detection of risk alleles is required in the GWAS study to associate that trait with the genome. The hypothetical views are generally considered according to classical Mendelian genetics. Because some major alleles are associated with risk formation in non-Mendelian studies, it is thought that this is due to genetic drift. Some harmful alleles may have a chance to spread and be the major allele cause of genetic drift [19].

The minor allele frequency filter is a step in GWAS analysis that removes rare alleles from the analysis to reduce the error rate. It is unclear whether these rare variants have a different genotype or are the result of incorrect genetic calling. The work must go through filtering processes to obtain meaningful genetic variants and continue. The minor allele count (MAC) is the second most common allele in a population to use a cut off value in GWAS [20]. For GWAS analysis, the general threshold for MAC > 5 (5%) or MAF > 0.05 is acceptable [21]. MAF < 0.05 is considered cannot reliable to test because it is unclear whether these SNPs occurred by random chance or by true significance. GWAS provides estimates of the likelihood of genetic disease in humans as well as yield rate and flowering times on plants. To interpret this information, both genotypic and phenotypic trait knowledge is required [22]. GWAS is expected to answer the question of whether rare or common variants have a significant impact on the phenotype. While doing this, it is critical to distinguish between true and false associations. This issue should be considered when interpreting. Because only a few alleles are truly associated with the trait, only those alleles should be associated.

1.5. Limitations of GWAS and Alternatives to Overcome These Obstacles

The challenges of GWAS analysis are to identify true causative genes and to link the strong association SNPs with these phenotypes to the chromosome and position they are on. A major challenge is distinguishing false positive SNPs from true positive. False-positive SNP is an error in GWAS results in which SNPs result incorrectly indicate the significance of a condition; such as when the SNPs are not associated with traits but are detected as significant. False-negative SNPs are the opposite error, where the GWAS result incorrectly indicates the significance of a condition when it is true significant SNPs. Increasing the sample control size that confirms significantly associated variants reduces this error [1]. Multiple testing correction is important in terms of the significance of this study because it eliminates non-significant samples from population clusters. Although the multiple testing corrections vary depending on the study, population, or phenotype under consideration, there are some common tests. The most commonly used test to help reduce false-positive results is the Bonferroni correction. The Bonferroni correction is used to keep the false positive rate under 5% [23]. The Bonferroni threshold is calculated by dividing the significance level (usually 0.05) by the number of tests. SNPs with p-values less than this threshold are chosen. In this way, it obtains significant associated SNPs with traits. This method is the most commonly used, and the error rate in rejecting the false null hypothesis is still quite high. Increasing the sample size could be one solution to these problems. Increasing sample size is not always possible for certain unique traits, populations, or diseases, so GWAS research for specific traits will remain a challenge. Another method for lowering the error rate is to repeat the GWAS analysis to ensure there is a true relationship. For this purpose, it is expected to be performed on a similar population with the same trait. These replications do not always produce precise and accurate results, and it is preferable to choose a larger population and sample group to produce a more reliable GWAS result; however, as previously stated, this is not always possible [24]. Because GWAS analyses require a large number of genetic markers to produce a significant result, it is critical to reducing error rates. Even if the error rate decreases in this case, the method is very expensive and time-consuming [25]. To prevent family wise error, the permutation-based threshold (the threshold used in permutation based GWAS [26]) was developed as an alternative method to the Bonferroni correction test. The use of permutation-based thresholds reduces Type I error by calculating all possible false-positive results [27]. The disadvantages of this method are that it takes a long time and requires a lot of storage. Each test has advantages as well as disadvantages. As a result, depending on the analysis and population type to be performed, one of these multiple testing methods may be preferred. When the thresholds used are more stringent, the false-positive rate decreases significantly, and the variance value increases statistically because the number of genes decreases, which is also significant. These SNPs, statistically cannot be correlated with using more stringent thresholds. The genes with the strongest relationships are not always chosen as candidates; however, they are more likely to have a more interesting function or effect than the genes that are chosen [23].

1.6. Selection of Loss-of-function in *Arabidopsis thaliana* Variants

Arabidopsis thaliana is an important model organism for genomic research and gene function studies. It enables the comprehension of different genes and their functions in plants. *A. thaliana* is the first whole-genome sequenced plant, allowing us to identify conserved genes and regulations in other plants, and the comparison of other eukaryotic organisms enlightens similar processes to demonstrate the evolutionary relationship and allow improving crop yields. This plant was chosen as a model organism for several reasons, including its ability to reproduce quickly, produce a large number of offspring, have a relatively small genome sequence, and provide an inbreeding opportunity. All of these are the scientific reasons for studying this plant. This plant is significant not only in plant studies, but also in evolutionary biology, genomics research, and medicine [28]. Duplication events in the *A. thaliana* genome provide information about the species' ancestors and play an important role in polyploid plant evolution. In this sense, GWAS contributes to evolutionary studies by helping in the identification of genetic variations. Working with inbred species allows researchers to examine multiple phenotypic features using a single sequenced genome, which reduces costs. As a result, *A. thaliana* is an excellent candidate for GWAS analysis because it is a well-known genotype that conserves genotypic information during inbreeding. Self-fertilization is playing a role for *A. thaliana* use as a model organism for GWAS research. It allows for the use of the same genotypic information for association mapping on different phenotypes [24].

Natural variations at the genomic scale are explained by GWAS, allowing us to approach the evolutionary history of species at the population level. These natural variants may be advantageous in adjusting to new environmental conditions and preserving their lineage [25]. It is critical to determine which SNPs cause new type adaptation in response to changing environmental conditions. The natural history of *A. thaliana* accessions helps to explain the plant's evolutionary process, from which region they are exposed to what climatic conditions, and which adaptation mechanisms result in the phenotypic diversity of these conditions.

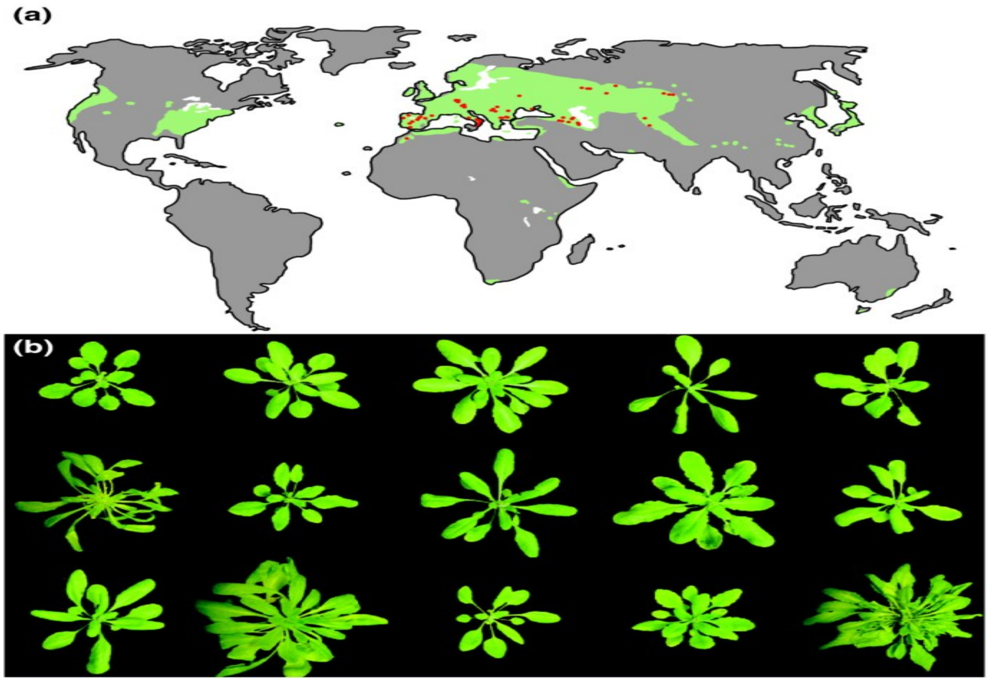


Figure 1. 3 Distribution and variations of the *Arabidopsis thaliana* throughout the world
(A)The Worldwide geographical distribution of *Arabidopsis thaliana* is shown for the 1001 genome project. (B) Variants of *Arabidopsis thaliana* from different regions are shown for the 1001 genome project (Figure from [29]).

In the *Arabidopsis thaliana* 1001 Genome project, it was possible to explain more easily how this evolutionary process occurred, especially with the help of GWAS, to explain the relationship of genetic variations with phenotype. In 2016, the *Arabidopsis thaliana* 1001 Genome project was completed, and 1135 genomes were thoroughly analyzed [30].
The "selection study on loss-of-function variants" study, which was a continuation of the 1135 genomes in the 1001 Genome project, discovered that nonsense SNPs, were found together in the same accessions. These discovered nonsense SNPs result in premature stop codon formation. The study found that some premature stop codons occur significantly in two different genes in the same accession.
This thesis is a continuation of previous research ("selection study on loss-of-function

variants"). This thesis attempted to comprehend the reasons for the co-occurrence of premature stop codons in the same accessions. Two different methods were used during this thesis: r-based GWAS and permutation-based GWAS. The study was continued with permutation-based GWAS due to the non-normally distributed phenotypic data used. This method was used to detect genetic variants (SNPs) that cause premature stop codons to occur in the same accession. In the study, significant genetic variants (SNPs) were identified using various statistical data analyses. The study found that 32 significant genes outside the expected region cause premature stop codons to occur in the same accession. The interactions of these genes with over co-occurrence gene pairs are shown with the network. The reason why these genes cause premature stop codon co-occurred in the same accessions, as well as its possible effect on *Arabidopsis thaliana* in natural selection, was detailed and attempted to be understood using GO enrichment analysis. Moreover, genetic variants detected by the Manhattan plot and quantile-quantile plot were found to be non-normally distributed. This study is still ongoing, and the circumstances that contribute to the bias in the results are being investigated.

2. Material

2.1. Computational Infrastructure

This study required high computational nodes to investigate high resources demanded unfiltered data. All the computational analysis was done at the Center for Computational and Theoretical Biology (CCTB) department at the University of Würzburg. CCTB has its high-performance computer cluster system with 7 compute nodes and 360 cores which contain its storage systems 116TB+232TB that run Linux (Debian) and SLURM scheduler. Individual nodes properties are Jupiter (Intel Xeon E7 4870, 2.4GHz, 80 cores, 1024 GB RAM), Saturn1 (Intel Xeon E7 4850, 2.0 GHz, 80 cores, 512 GB RAM), Saturn2 (Intel Xeon E7 4870, 2.4 GHz, 80 cores, 512 GB RAM), Uranus[1-3] (AMD Opteron 6274, 2.2 GHz, 32 cores, 192 GB RAM), Neptun1 (AMD Opteron 6172, 2.1 GHz, 24 cores, 192 GB RAM), and The GPU node saturn2 additionally includes two graphics processing units for GPU computing, one Nvidia Pascal GP104 GPU (8 GB RAM, 2560 cores) and one Nvidia Pascal GP107 GPU (4 GB RAM, 768 cores), both running under CUDA 8.0. Genome-Wide Associations Studies need more resources thus, mostly these computing nodes are used during analysis via the SLURM scheduler. During this study, two programming languages were used to complete this master thesis these are Python version 3.8.8 and R Studio version 4.1.2. The obtained results were visualized using Cytoscape version 3.9.1.

Two different GWAS strategies were used to complete the analysis. The first was that GWAS is working on the R studio environment, which was created by Arthur Korte and contributors. [31]. Functions used for GWAS are gwas.r (script to run GWAS with a mixed model), emma.r (script to update the p-values of the top SNPs), and plot_gwas (script for plotting the results of GWAS). The availability of the GWAS functions was stored at https://github.com/arthurkorte/GWAS.git. Additionally, f95_gwas.r and GWAS_f952.r were required for running GWAS functions that are shown in Appendix Section and, can be found https://github.com/leylasrknt/GWAS_Premature_Stop_Codons.git. The second was the permutation based GWAS, an open-source Python tool written by John Maura and contributors [26]. All the necessary instructions available for the permutation based GWAS tools were stored at https://github.com/grimmlab/permGWAS.git. All of the other codes required to complete the GWAS analysis were stored at https://github.com/leylasrknt/GWAS_Premature_Stop_Codons.git also can be found in the

Appendix part of this thesis. GO Enrichment Analysis was performed at https://www.arabidopsis.org/tools/go_term_enrichment.jsp.

2.2. *Arabidopsis thaliana* Genomic and Phenotypic Datasets

The genotypic and phenotypic data used in this master's thesis were obtained from the *Arabidopsis thaliana* 1001 Genome project. A total of 1,135 *A. thaliana* accessions were collected from various geographic regions around the world. The accessions were collected in a hierarchical order based on their geographic distance from the nearest neighbor accession. Accessions were genotyped with 250 K SNPs markers to prevent sequencing of the same samples [29]. As a result of the additional 727 transcriptome data provided in the 1001 project, the transcriptome and genomic data did not match. The study continued with 665 of these 1135 accessions. Consequently, the differences in gene expression, 7124 premature stop codons differ significantly in their gene expression. The study was continued with stop codons, which caused a decrease in gene expression levels. 216 stop codons were found to decrease gene expression using the Bonferroni correction.

Hypergeometric test was used to show whether the co-occurrences of gene pairs in the same accessions statistically deviated from the expected distribution. The hypergeometric test is a statistical distribution that is used when events are not independent of one another. The hypergeometric test was used on 665 accessions that formed premature stop codons. The number of draws was chosen as the number of the first premature stop codon. The presence of a second early stop codon determines the significance of these draws. It was determined if the presence of both the second and first premature stop codons in the same accession was distributed significantly. The study continued with statistically significant results. Premature stop codons were found to be significant in 1128 gene pairs. As a result, 1128 gene pairs were generated and used as phenotype data in GWAS. The method section goes into more detail about how these phenotype values were coded. The reference genome used in GWAS is based on the *A. thaliana* col-0 ecotype. In the permutation based GWAS, which was used as the second method, there was no need for an external kinship matrix, and the functions create their kinship. As a result, the GWAS data to be used in both different methods were prepared for the analysis. The first method, r-based GWAS, calculated the kinship matrix using the Col-0 genotype, and data was generated. In the second method, permutation-based GWAS, the functions themselves provide the kinship during the analysis. Finally, the GWAS data to be used in methods were prepared for analysis.

3. Methods

The analysis performed in light of the required materials, as well as the various methods used, were explained in detail in this section. This section also covered statistical and computational methods, as well as their variants. In this study, two different GWAS methods were used. The first method, R-based GWAS, was used for the analyses. The second preferred method was permutation based GWAS. Details of these methods were described below.

3.1. Phenotype Files Prefiltering for GWAS Analysis

In this section, the GWAS phenotype files were filtered to prepare for analysis. Premature stop codons had been used to generate phenotype files. The phenotype files were created in response to the significant over-cooccurrence of the two premature stop codons by Laura Steinmann (You can find the details of this work at https://github.com/laurasteinmann/Premature_Stop_Codons). The first premature stop codon was found in all accessions, and the phenotype value was determined by the status of the second stop codon. The phenotypic value was coded as 1 if the first and second premature stop codons formed at the same accessions, and 0 if only the first premature stop codon formed. The first and second premature stop codon genes were separated by an underscore, as shown in the results section (Table 4.1). These names were assigned based on which chromosome and which location this premature stop codon occurred. For example, in the filename AT3G32920_AT2G16380.csv, the first premature stop codon occurred in 32920 regions on chromosome 3, and the second premature stop codon occurred in 16380 regions on chromosome 2. In this AT3G32920_AT2G16380.csv phenotype file, 88 first premature stop codon AT3G32920 were found, but only 16 second premature stop codon AT2G16380 were found. This means that there was a significant over co-occurrence of premature stop codons in 16 accessions. The phenotype file names were put into the first column of the data frame (Table 4.1). The number of rows of file (phenotype length), the number of KO (allele count 1, it means number of over co-occurrence gene pairs in the same accession), and the number of non KO (allele count 0) were counted for each phenotype file separately and added to this data frame as new columns. The first filtering step was to remove phenotype files with MPC (Minor phenotype count, second-most found phenotype counted) less than 5. In a second step, phenotype files length with less than 50 accessions (the length of 50) were filtered.

3.2. GWAS Analysis Run on the R Environment

The first method of this GWAS done in the R environment. f95_gwas (f95_gwas.r) was a required function to load genotypic files, and kinship files, to run GWAS functions (gwas.r, emma.r, plots_gwas.r can be downloaded from https://github.com/arthurkorte/GWAS), saves output GWAS files and plots output of the GWAS was written by Arthur Korte. Phenotype files were run in parallel on the SLURM scheduler to reduce time costs. After finishing the analysis, it was renamed all the filenames and GWAS results were saved as phenotypic file names.

3.2.1. Minor Allele Count (MAC) Filter

In this part, MAC (Minor Allele Count) value less than 5 was discarded from inside of all GWAS result files and saved. The rest of the steps were continued with these new filtered files.

3.2.2. Stringent Bonferroni Correction

Bonferroni correction is a GWAS analysis threshold for associating true positive SNPs with traits and preventing false positive (Type I error) type error [27]. Stringent Bonferroni corrections were performed in this thesis with a p-value less than the result of the number of SNPs multiplied by the number of files to be divided by 0.05. The result of the stringent Bonferroni filter was saved.

3.2.3. Count Number of Hits

SNPs that were within the range of the start and stop codons of a gene were called cis hits. SNPs that were not within the range of the start and stop codons of a gene were called trans hits. In this section, it was counted both cis and trans genes to continue our analysis with only trans genes. The Ara11 file, which contains all *A. thaliana* genes, was used for this analysis. Before beginning the analysis, 50 kb were subtracted from the start and 50 kb were added to the stop positions of the genes in the Ara11 file, and the analysis was performed using the new gene positions. This was because of the linkage disequilibrium. To count cis and trans hits, stringent Bonferroni correction files were used. The second gene name was taken from the file names. Each SNP position in these files was assigned a cis or trans value based on the Ara11 gene location file's Start and Stop positions. If these SNPs were located between the Start and Stop positions, they were called cis hits; otherwise, they were called trans hits. The number of trans hits was calculated by subtracting the number of cis hits from the total number of SNPs (hits) in each file. The next step was to continue with the files with trans gene, and only files with cis hit or non-hit were discarded. Because the gene AT2G15930 was not found in the Ara11 notation file, it was impossible to count all analyses as trans, even if some of them are cis hits. It was contained within the Tair10 file, but all of our analyses were performed on Ara11, so the file containing a second gene "AT2G15930" was discarded to continue this study. In the last step of this part, the cis hits were removed from trans files. The other steps of working with files containing only trans hits were continued.

3.2.4. Count the Number of Trans Genes

SNPs in files containing trans hits were added to the trans gene list by determining which genes they belong to. This was accomplished by determining which gene the SNPs in the trans files were located in the ara11_10 file, which contains all of the SNPs. In this way, if there is a gene corresponding to all SNPs, those genes were added to the list. It was also found in the genes in the 10 kb window and added to the list.

3.3. Permutation Based GWAS

Permutation based GWAS is a method that provides true association for results that do not have a normal distribution, and a different threshold is used for each result [32]. In this study, the results before repeating the prefiltering, which was done before the GWAS analysis in the previous method, were used. In this way, permutation based GWAS analysis was performed with significant 738 phenotype files. The phenotype and genotype files used are the same as the r-based GWAS and permutation-based GWAS does not require a kinship file. The permutation based GWAS used in this study is an open-source tool, downloaded from https://github.com/grimmlab/permGWAS.git and detailed information can be found here. After downloading the necessary tools, the phenotype and genotype file locations were specified, the appropriate permutation number (perm was set to 100 for this study), and the python script was run in the batch environment. Permutation based GWAS analysis was performed in this manner.

3.3.1. Permutation Based Threshold for GWAS

In this section, permutation-based threshold was applied to the significant GWAS results obtained. Separately new thresholds were found for each of the permutation based GWAS results. Every min p-value file only contains the first 100 p-values. The min p values are reordered based on their p-values. After that cut-off value is decided to 5th of the p-value

(0.05. these p values are decided by each of the PermGWAS results separately). Each threshold is used to filter PermGWAS results, and the results were saved.

3.3.2. Minor Allele Frequency (MAF) Filter

In this part, MAF (Minor Allele Frequency) value was less than 0.05 is discarded from inside of all permutation based GWAS files and saved. The rest of the steps were to continue with these new filtered files.

3.3.3. Permutation Based GWAS: Count Number of Hits

In this section, it counted both cis and trans genes to continue our analysis with only trans genes. The Ara 11 +/- 50 kb file used in the previous method (Method section 3.2.3) was also used when counting cis hits in permutation based GWAS results.

To count cis and trans hits, permutation based GWAS threshold filtered files were used. Each filtered permutation based GWAS threshold file was used. The second gene name was taken from the file names. Each SNP position in these files is assigned a cis or trans value based on the Ara11 gene location file's Start and Stop positions. If these SNPs were located between the Start and Stop positions, they were called cis hits; otherwise, they were called trans hits. The number of trans hits was calculated by subtracting the number of cis hits from the total number of SNPs (hits) in each file. The next step was to continue with the files with trans gene, and only files with cis hit or non-hit were discarded. Because the gene AT2G15930 was not found in the Ara11 notation file, it was impossible to count all analyses as trans, even if some of them are cis hits. It was contained within the Tair10 file, but all of our analyses were performed on Ara11, so the file containing a second gene "AT2G15930" was discarded in order to continue this study. In the last step of this part, the cis hits were removed from trans files. The other steps of working with files containing only trans hits were continued.

In addition, it was also examined whether or not SNPs formed in the second gene's stop codon. The second gene's premature stop codon location was determined using the Ara11 file. It was controlled whether or not the SNPs from the GWAS result were there. The output was saved in a new file.

3.3.4. Permutation Based GWAS: Count the Number of Trans Genes

SNPs in files containing trans hits were added to the trans gene list by determining which genes they belong to. The Ara11_10 file was used to determine which gene generated the SNPs in permutation-based GWAS results with trans hits. There was information in the

Ara11 10 file about which genes belong to all SNPs found in *A. thaliana*. The trans gene was detected when the SNPs in the files containing trans hits and the SNPs in Ara11_10 were the same. When a gene corresponded to an SNP in the trans hits files, it was added to the list. It was also found in the genes in the 10 kb window and added to the list. As a result of GWAS, duplicated gene pairs (files with gene1_gene2 and gene2_gene were used as duplicated gene pairs) from files containing trans gene gene1_gene2 (example: AT2G16365_AT1G13430.csv) and gene2_gene1 (example: AT1G13430_AT2G16365.csv) had been added to a list reciprocally. Only trans hits containing duplicated gene pairs were found in 240 of the 646 files with trans hits. Because these 240 files contain 120 gene pairs, the filenames were appended to the first column as gene1 gene2 and the second column as gene2 gene1. Then it was determined which trans hits belonged to which gene. Trans genes that share both gene1 gene2 and gene2 gene1 had been added as a new column to the list of duplicated gene pairs. The number of trans genes in each GWAS result file was then counted and sorted. The number of trans genes in each file was determined, and their distribution was calculated. It was determined how many trans genes were present in each ratio. The study continued to include the 32 most repetitive trans genes. To demonstrate that our 32 most frequently repeated trans genes were not chosen at random, confidence intervals and thresholds were calculated. Random genes were selected from the Ara_11 file, equal to the number of trans genes generated by each GWAS result file. This random gene selection was repeated 100 times. Consequently, 3200 genes were randomly selected. The number of the most repetitive genes among these randomly selected genes was recorded in separate files for each. The numbers of these randomly repeating genes were summed and divided by 3200 to find the mean of this null hypothesis. The mean of this null hypothesis was 3,03, with a standard deviation of 0.24. The z-value associated with the 95% confidence level was used. From here, values were placed in the formula CI $=\bar{X} \pm (Z \times s/\sqrt{n})$ for the confidence interval, and the calculation was made (X is the mean, Z is the chosen Z-value (1.96 for 95%), s is the standard error, n is the sample to you) [23]. The confidence interval of randomly selected trans genes was accepted as a null hypothesis (Ho). The average of 32 most trans genes obtained as a result of permutation-based GWAS was found to be 6.18 and it was accepted as an alternative hypothesis (Ha).

3.3.5. Gene Network

The GWAS files corresponding to the 32 most significant trans genes were created in response to the phenotype files. This method was used to determine which phenotypes were caused by each trans gene. The interaction with trans genes corresponding to these phenotype files was visualized using Cytoscape.

3.3.6. GO Enrichment Analysis

It was critical to understand what functions the obtained trans genes have and whether there was any commonality between these genes in terms of making associations with the obtained results. GO enrichment analysis, which was developed for this purpose, enables us to make sense of which biological, molecular or cellular functions these genes have from the given gene data set [28]. GO enrichment analysis was performed on 32 trans genes obtained. Obtained results were interpreted in the result section.

3.4. Interpretation of the GWAS Results

The results were interpreted in a more meaningful way by creating a Gen network, Manhattan graphs, and distribution plots. Gene networks for 32 trans genes were created separately. Each network was built with a trans gene and the genes that cause it (knock out gene pairs, phenotypic genes). To begin with, a network was built by interacting with over cooccurrence pair genes. Second, a network was built by interacting with trans genes and cooccurrence pair genes. Finally, these two separate gene networks were merged. The final network was built using 32 significant trans genes and 95 over cooccurrence genes that cause these genes. A network was created by interacting with 95 over cooccurrence gene pairs, and the network was created by interacting with over cooccurrence pair genes in 32 significant trans genes, and both networks were shown on the same network.

Manhattan plots were created based on the GWAS results of over cooccurrence gene pairs with the trans genes "AT2G21830," "AT1G08840," and "AT2G18735". The Manhattan plots also showed the common trans genes that these over cooccurrence gene pairs share.

SNPs that caused trans gene formation have their genotype files created. Only the genotype files corresponding to the accessions in the phenotype files were retrieved from these. The genotype file was created for the same accessions as the phenotype file. Genotype files with the same accession versus over cooccurrence gene pairs having trans gene "AT2G21830" “AT1G08840” and, “AT2G18735” were prepared. The genotype files corresponding to these phenotype files were used to generate distribution plots.

4. Results

The study was conducted with the 1135 accession of *Arabidopsis thaliana*, the effect of the previously obtained over cooccurrence significant gene pairs on the phenotype, and the connection of this study with other genes in terms of evolution was established. The association of genes that cause premature stop codons observed in phenotype was analyzed by the permutation based GWAS method. In conclusion in this study, it was discovered that over cooccurrence gene pairs that occur together frequently formed premature stop codons in the same accessions. Statistical analyses were used to identify genes that caused premature stop codons in the same accessions Gene networks, Manhattan graphs, and distribution plots were presented as a result of this study. Moreover, it was attempted to correlate in detail with GO enrichment analysis, which features of these trans genes and which may cause these over cooccurrence gene pairs to have premature stop codons together in the same accession.

4.1. Phenotype Files Prefiltering for GWAS Analysis

It was critical to continue with significant phenotype files before beginning GWAS analysis. For this, phenotype length and allele count were filtered using given thresholds. The study excluded pairs of over cooccurrence genes with less than 50 accessions from the phenotype data (length of the phenotype file) before the GWAS analysis. It was possible that the GWAS analysis will produce more errors than expected in data with a small number of accessions. Another important filtering requirement was that the number of minor phenotypes of over cooccurrence gene pairs is greater than 5. This was because, in the absence of a sufficient sample majority, it was impossible to determine whether the genes that cause the SNPs are indeed the genes of interest or whether the genes that cause the actual SNPs will be the majority. To reduce the error rate, over cooccurrence gene pairs with less than 5 minor allele phenotypes were removed before the GWAS study. The lengths of the gene pairs, how many times they created a premature stop codon together, and how frequently only the first gene creates a premature stop codon alone are shown (Table 4.1). Those with a length less than 50 or minor alleles less than 5 of the phenotype were marked with "1" in the files. These files were not included in the GWAS analysis, according to the last column of the table (Table 4.1). As a result of this filtering, a total of 738 over-occurrence gene pairs remained for GWAS analysis. Different methods were used to obtain GWAS results with these 738 over co-occurrence gene pairs.

The study was further continued with significant genes under the selected thresholds.

Table 4. 1 Over Cooccurrence Gene Pair Distribution

Over cooccurrence gene pair cause a premature stop codon together files shown in this Table 1. Phenotype lengths represent accessions number and they discarded less than 50 accessions. Second filters occurred for minor phenotype count (MPC) less than 5, second most common phenotype selected according to ko (knock out, gene pairs have together premature stop codons in same accessions) or non_ko (non-knock out, gene pairs do not have together premature stop codons in same accessions) are selected that occur less than each other.

File_Name	Phe_Length	KO	non_KO	MPC	MPC<5	Phe_Length<50	MPC<5Phe|Leng<50
AT3G32920_AT2G16380.csv	88	16	72	16.0	0	0	0.0
AT1G42680_AT5G63630.csv	135	32	103	32.0	0	0	0.0
AT4G00970_AT3G28610.csv	50	37	13	13.0	0	0	0.0
AT5G33280_AT5G03830.csv	97	70	27	27.0	0	0	0.0
AT1G42680_AT3G45740.csv	135	24	111	24.0	0	0	0.0
AT3G26240_AT4G00970.csv	88	27	61	27.0	0	0	0.0
AT2G41710_AT4G15950.csv	44	22	22	22.0	0	1	1.0
AT1G01695_AT5G20450.csv	43	36	7	7.0	0	1	1.0
AT5G20450_AT5G03830.csv	120	84	36	36.0	0	0	0.0
AT2G16380_AT1G29710.csv	21	17	4	4.0	1	1	1.0
…	…	…	…	…	…	…	…
AT5G35930_AT3G58050.csv	295	232	63	63.0	0	0	0.0
AT2G16380_AT2G41470.csv	21	18	3	3.0	1	1	1.0
AT3G45740_AT2G16380.csv	47	17	30	17.0	0	1	1.0
AT5G45150_AT4G04220.csv	86	24	62	24.0	0	0	0.0
AT5G51795_AT2G04845.csv	36	16	20	16.0	0	1	1.0
AT3G23610_AT2G42730.csv	207	56	151	56.0	0	0	0.0
AT2G15930_AT4G15950.csv	311	25	286	25.0	0	0	0.0
AT1G01695_AT2G15930.csv	43	3	40	3.0	1	1	1.0
AT5G03830_AT5G33280.csv	331	70	261	70.0	0	0	0.0
AT2G16380_AT3G32920.csv	21	16	5	5.0	0	1	1.0
AT1G13430_AT2G41710.csv	57	16	41	16.0	0	0	0.0

4.2. R-Based GWAS Analyzing

This study's first method was R-based GWAS. The study's findings revealed that more genes than expected were responsible for this premature stop codon pair. Because the results did not have a normal distribution, the study was not completed using the reliability of the Bonferroni correction results as the standard. Instead, a separate threshold was applied to each result that did not have a normal distribution when using permutation based GWAS.

The position of the SNPs on which chromosome, p-values, minor allele count values, and minor allele frequency values were given in the GWAS results in files. SNPs from these GWAS results with minor allele counts less than 5 were removed from these files. This was because it is unclear whether the number of these SNPs is significant to the phenotype. Minor

Allele Count is the second most common allele in a population, and studies have shown that it has a significant impact on traits. The Appendix section described this process in more detail. Because non-significant SNPs may still exist because of errors in these filtered GWAS files, a stringent Bonferroni correction was applied, reducing the possibility of Type I error.

In this way, the number of SNPs in the files in the GWAS results with significant SNPs was counted. The position of the second gene in the genome from the over cooccurrence gene pair that causes the formation of these SNPs was known. This position was specified in the Ara11 file. These SNPs were called "cis hits" if they were located between the Start and Stop codons of the second gene. This was considered a confirmation of the work done. Because SNPs in the second gene were expected to cause a premature stop codon. The SNPs that were not in the range of the second gene's Start and Stop positions were what make this study interesting. Although the resulting SNPs were not on the second gene, they still cause this gene pair to have premature stop codons together. These SNPs that occur in a different region of the genome were referred to as "trans hits" (Supplementary Table 1).

According to this result, the Venn diagram (Figure 4.2) showed how many files are cis hit, trans hit, and non-hit in total.

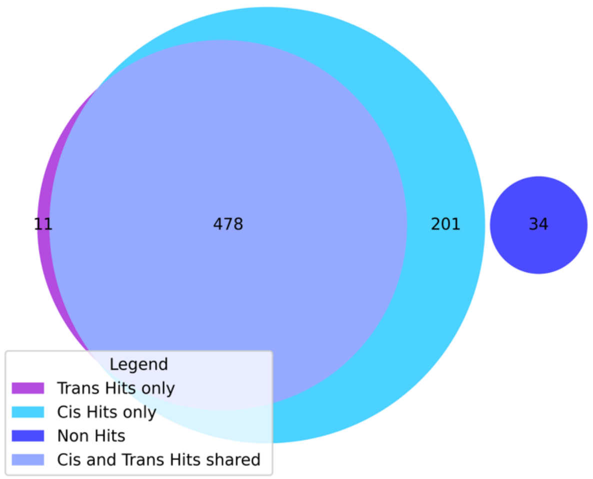


Figure 4. 1 Venn Diagram Number of the Hit Distribution

Venn Diagram 1, shows a number of files have how many SNPs, these SNPs are named as cis if it is inside the Start and Stop range inside that gene, if not that SNPs are named as trans. As seen 34 files do not have any significant SNPs related to phenotype.

In the study, the study was continued with the files with trans hits to understand which genes were the genes that cause trans hits and the effect on over cooccurrence gene pair. The SNPs 2029 file contains 10709466 SNPs that belong to the gene to which these SNPs belong. As a result, all trans hits were associated with trans genes. However, the presence of more trans genes than expected was not considered appropriate for the reliability of the study. It was critical to understand how pairs of over cooccurrence have significantly premature stop codons together by focusing only on the significant SNPs in the study. When significant genes could not be identified using R-based GWAS analysis, the study was left here. In the second method that was used in this study, permutation based GWAS, significant trans genes were found and these genes were analyzed in detail (Result 4.3).

4.3. Permutation Based GWAS Analysis

Data from 738 significant phenotypes were used in the permutation based GWAS analysis. The threshold in this method was determined by the p-value value in each permutation based GWAS result rather than the Bonferroni threshold that is normally used. This was because only the Bonferroni corrected filtering was filtered based on the overall result, not each result, so the likelihood of false-negative results is higher. As a result, the study continued using only significant SNPs. The position of the SNPs on which chromosome, p-values, and minor allele frequency values were shown in the GWAS results in files. To continue with these SNPs, it was decided to filter maf > 0.05 values rather than MAC > 5. However, it was more accurate to have confirmation that significant SNPs have a MAC greater than 5.

SNPs from these GWAS results with a minor allele frequency of less than 0.05 were removed from these files. It was unclear whether the number of these SNPs could be significant to the phenotype. The Appendix section describes this process in more detail.

It was explained in the R-Based GWAS result section which regions of the genome had trans and cis hits and how they were affected (Result 4.3. permutation based GWAS analysis). permutation based GWAS results showed how many cis and trans hits there are (Figure 4.2, Supplementary Table 2). The study was continued files with only trans hits to understand which genes cause trans hits and the effect on over cooccurrence gene pairs. A total of 646 files have only trans hits left to analyze which genes cause these SNPs and what effects the occurrence of premature stop codons in the same accessions has on these genes.

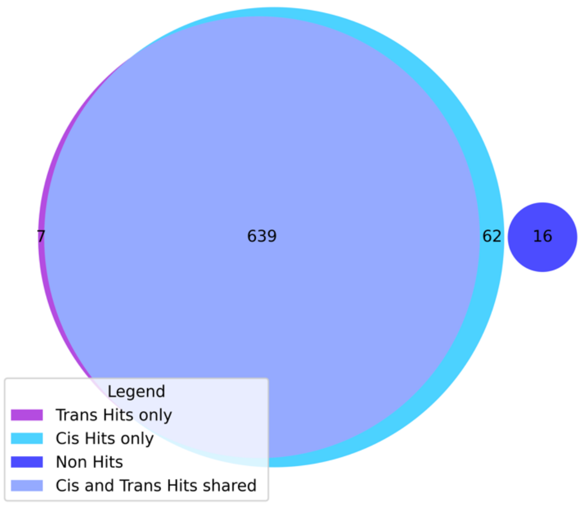


Figure 4. 2 Venn Diagram Number of the Hit Distribution
Venn Diagram 2, shows a number of files have how many SNPs, these SNPs are named as cis if it is inside the Start and Stop range inside that gene, if not those SNPs are named as trans. As it seen 16 files do not have any significant SNPs related to phenotype.

It was critical to understand how pairs of over cooccurrence have significantly premature stop codons together by focusing only on the significant SNPs in the study. As a result, trans genes corresponding to all trans hits were identified from the SNPs 2029 file, which contains 10709466 SNPs and the gene to which these SNPs belong. Because of the premature stop codons, 7182 trans genes associated with trans hits (SNPs) occurred. Common trans genes were identified in duplicated gene pairs containing a premature stop codon. The gene pairs were identified, in which the trans genes were not chosen at random, having a significant effect on revealing the premature stop codon. Thus, confirmation had been obtained. Continuing the study, it was discovered that only 50 of the 120 gene pairs in the files had a common trans gene. Significant trans genes were identified in these 50 gene pairs.

It was critical to reduce the number of genes associated with traits, focusing only on significant genes to find a pattern. Among these genes, constantly repeating genes were found. The fact that repetition of these genes may have a greater effect on premature stop codon formation than other genes, and the properties of these genes were examined in more detail later in the study. In summary, 6546 trans unique genes were identified.

Understanding that these genes were not chosen at random is critical for the study's reliability. The reliability of this test has increased as a result of the fact that the trans genes selected as a result of GWAS were not chosen at random, and that this could affect the formation of significant over cooccurrence gene pairs. The obtained results are considered significant because the mean of randomly chosen genes is only 3.03. An alternative hypothesis is that the mean of the 32 most repetitive genes identified through GWAS is 6.18. The alternative hypothesis was that these trans genes were not chosen at random and were chosen because they were thought to have a greater effect on the premature stop codon. Because 32 GWAS-identified genes were found to be more frequently repeated than expected. These genes were thought to have a bigger impact on the formation of premature stop codons within the same accession.

Understanding which over-cooccurrence gene pairs result in these 32 significant trans genes was one of the findings that will help to explain why premature stop codons occur in the same accession. Moreover, gene networks with pairs of over cooccurrence genes that cause SNPs in these significant trans genes were created. These interactions were thus more easily seen in the gene network. In this way, it had been demonstrated that the accuracy of the results was significant and not random. In this case, it was accepted that the alternative hypothesis was not chosen by chance. These trans genes were considered significant. The resulting gene networks and trans gene interactions with over co-occurrence gene pairs were visualized. The interaction of 32 trans genes with over cooccurrence gene pairs was shown. A comprehensive gene network had been constructed using all 32 trans genes and all gene pairs in which they played a role in over co-occurrence. Small gene networks were also shown with arrows in this gene network to help understand the interactions of trans genes (Figure 4.3).

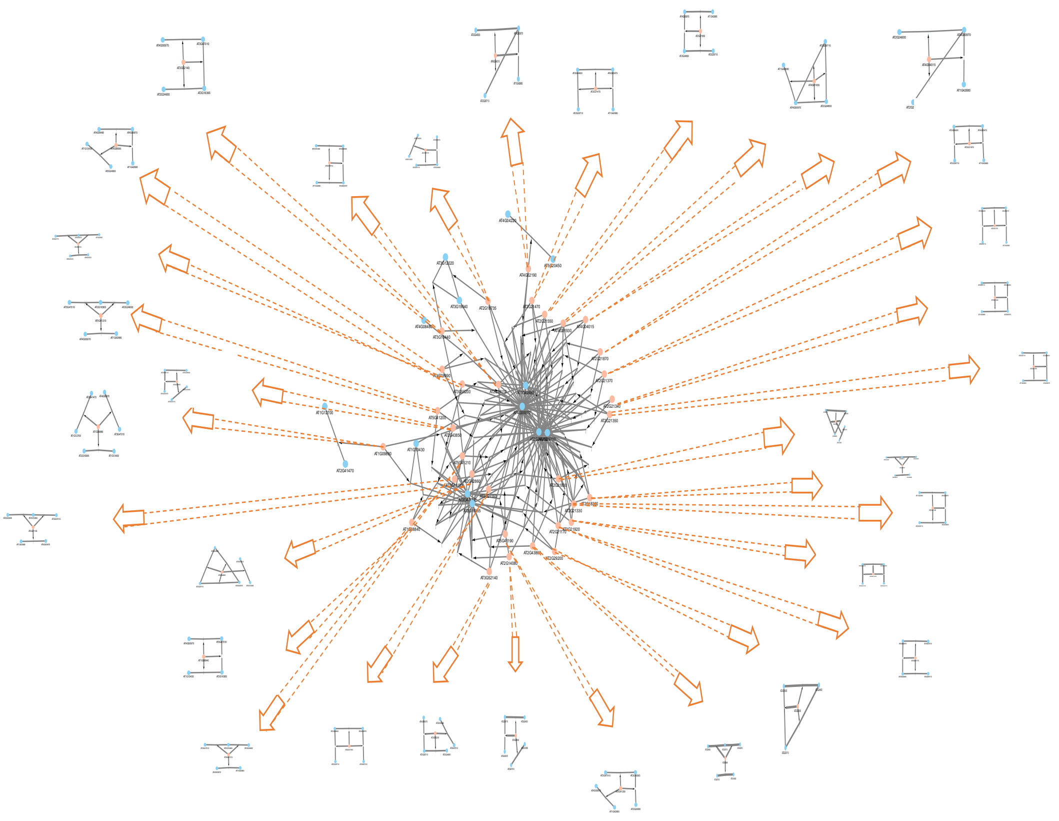


Figure 4. 3 Significant Trans Genes Networks with Over Cooccurrence Gene Pairs
Gene networks are established with 32 significant trans genes. These trans genes are separately created in each separate gene network with over cooccurrence gene pairs, a total of 32. Additionally, all these trans genes are connected in the same gene network to show their interaction with each other.

The "AT2G21830" trans gene was thought to have a significant effect on over-occurrence gene formation. Manhattan graphs were created using the permutation based GWAS results (maf > 0.05 filtered) of the over cooccurrence gene pairs responsible for the formation of the AT2G21830 trans gene. The common SNPs of the over cooccurrence gene pairs were marked on these graphs. Cis hits (SNPs) were shown in blue in Manhattan graphs, and the regions where these cis hits were found are highlighted with red arrows. These cis SNPs were found in regions formed by a premature stop codon formed by the second pair of over-cooccurrence genes (Figure 4.4).

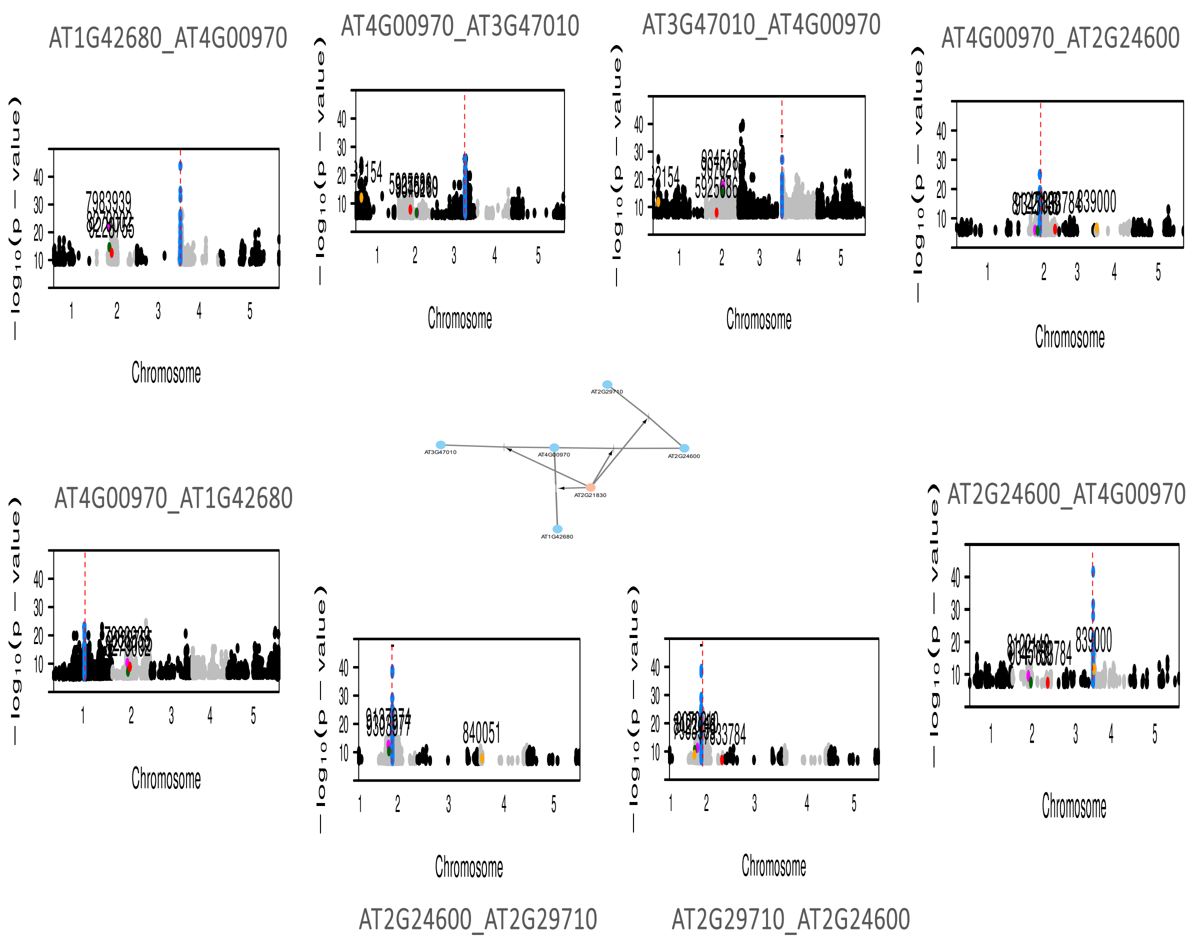


Figure 4. 4 Over Cooccurrence pair Manhattan Plots Cause to Trans Gene AT2G21830
Manhattan plots of GWAS results for over-occurrence gene pairs causing trans gene "AT2G21830" shown. SNPs causing the most significant trans genes in 8 over-occurrence gene pairs, which are 4 duplicated versions of each other, are shown in the Manhattan plot.

The scatter plots showed the association between the alleles that cause the over-cooccurrence genes and the phenotype. Over cooccurrence gene pairs resulting from the AT2G21830 trans gene were chosen. SNPs have been identified that cause the formation of this trans gene from these over cooccurrence gene pairs. Scatter plots were created for alleles versus phenotypes (Figure 4.5). As seen in these graphs, when the allele count (SNPs) is “1”, the phenotype was “1” as well (premature stop codon in same accessions). When the allele is "0," the phenotype was also expected to be "0". Other SNPs here were thought to be responsible for the phenotype being "1" when the allele is "0". While the resultant allele value is 1, the absence of a phenotypic effect was not expected. The reasons for this will also be investigated during the study's ongoing processes.

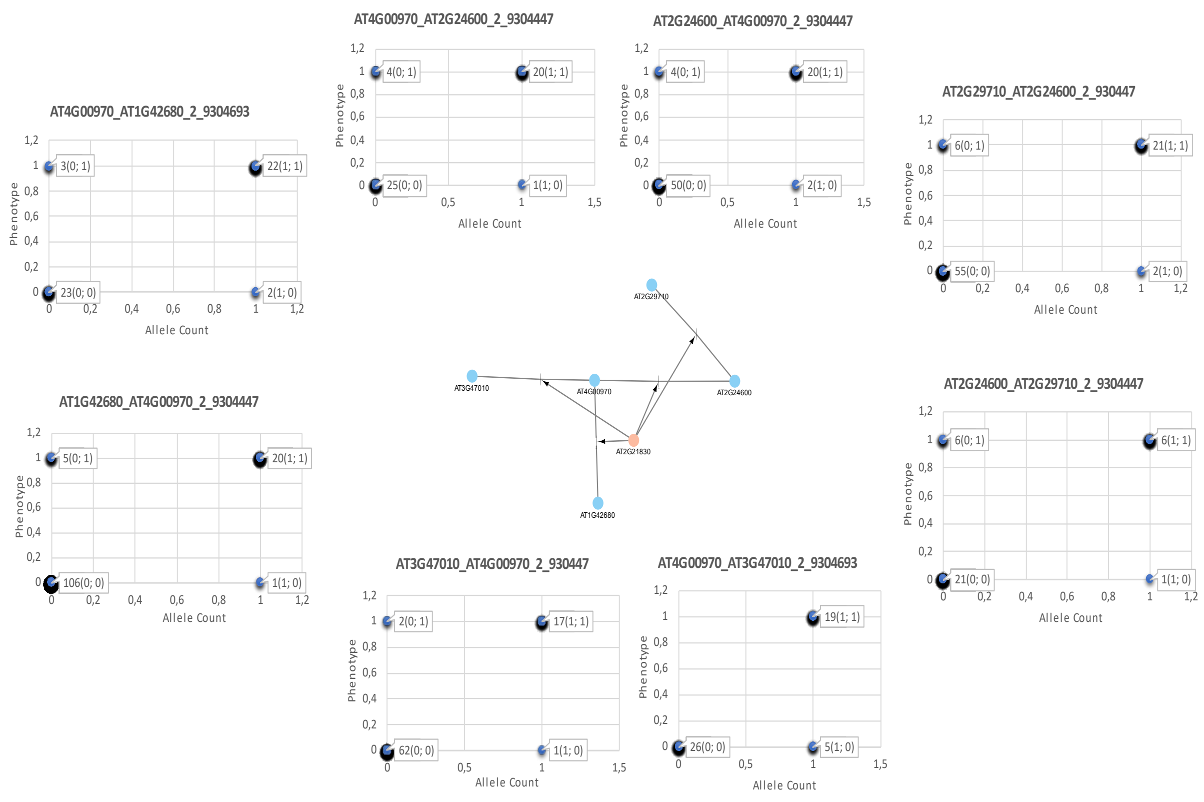


Figure 4. 5 Over Cooccurrence Pair Distribution Plots Cause to Trans Gene AT2G21830
Scatter plots of GWAS results for over-occurrence gene pairs causing trans gene "AT2G21830" shown. AT2G21830 trans gene occurred by 8 over-occurrence gene pairs, which are 4 duplicated versions of each other, are shown in the scatter plot. The scatter plots were drawn for alleles against the phenotype

The permutation based GWAS results were also demonstrated with Q-Q (Quantile-Quantile). The Q-Q plot results showed that our GWAS results do not distribute normally (Figure 4.6). Furthermore, it has already been detected on Manhattan plots that there are too many SNPs responsible for over co-occurrence gene pairs (Figure 4.4). The genomic inflation values in the Q-Q plots are also far from 1. The permutation based GWAS was intended to control data that was not normally distributed. However, the findings showed that even permutation based GWAS results were insufficient to reduce the number of significant SNPs.

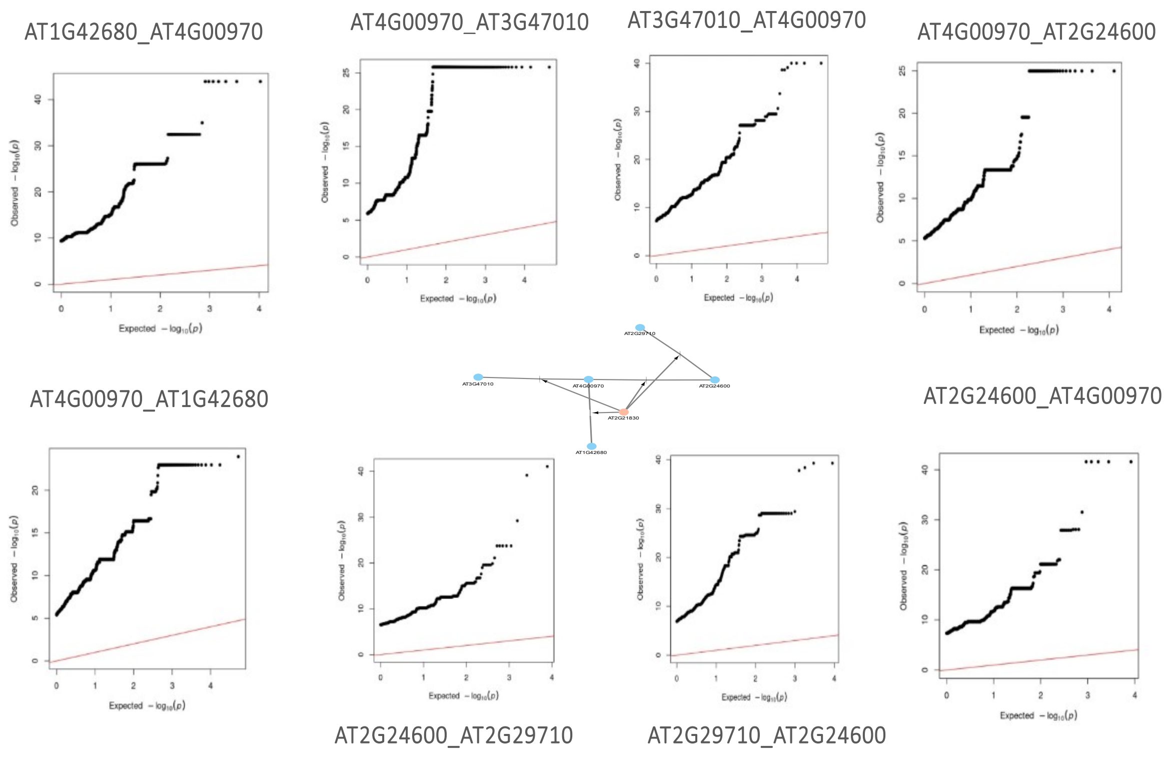


Figure 4. 6 Over Cooccurrence Pair Q-Q Plots Cause to Trans Gene AT2G21830
Q-Q (Quantile-Quantile) plots of GWAS results for over-occurrence gene pairs causing trans gene "AT2G21830" shown. AT2G21830 trans gene occurred by 8 over-occurrence gene pairs, which are 4 duplicated versions of each other, are shown in the scatter plot. The scatter plots were drawn for alleles against the phenotype

4.4 Summary of the GWAS to Infer Signs of Selection in *Arabidopsis thaliana*

This thesis began with 1128 co-occurrence gene pairs, 738 of which were analyzed using permutation based GWAS. Only 724 of the 738 permutation-based GWAS results identified produced statistically significant results. Only 646 trans hits were found among the 724 significant permutation-based GWAS results. Only 240 duplicated versions were found among the results with trans hits. That's 120 duplicated permutation-based GWAS results out of 240. Only 50 of the 120 duplicated permutation based GWAS results shared common trans genes. The study concluded with the identification of 32 most significant trans genes in these 50 duplicated permutation-based GWAS results. This study attempted to explain the effect of these 32 trans genes on over co-occurrence gene pair formation.

5. Discussion

The GWAS study concluded that 32 regulatory genes have a significant impact on gene pairs to form a premature stop codon in the same accessions. This study began with premature stop codons, and 1128 gene pairs had significant interactions. The study concluded with only 50 gene pairs that were statistically significant as a result of the permutation based GWAS analysis.

The phenotype files used in the study were generated using two genes. The effect was seen in the phenotype when the second of these genes had also a premature stop codon. Therefore, premature stop codons in the second gene, cis hits (SNPs) have occurred. Significant SNPs were formed in the gene region resulting in the premature stop codon in the second gene. These cis hits were also shown with the Manhattan plot (Figure 4.4).

There are several possibilities for premature stop codons, which also occur in the second gene in the same accessions. The premature stop codon formed due to SNPs in the first gene, and changes in the signal pathway could have resulted in premature stop codon formation in the second gene. If the proteins from gene one and gene two are physically interacting, the proteins produced by the first gene may have an effect on the protein mechanism in the second gene. According to this thesis findings, no SNP occurred directly in the premature stop codon of the second gene in the GWAS results. This increases the likelihood that trans genes played a role in the formation of premature stop codons in the same accession. It strengthens the hypothesis that regulatory genes (trans genes) may be the reason why premature stop codon still does occur in these genes even when both gene pairs do not have SNPs. For this reason, regulatory genes (trans genes) make this study interesting. It is thought that the mechanism of natural selection can be used to adapt to that environment by how these regulating genes (trans genes) affect natural variation, while causing gene loss in the same accession in both genes.

These are still hypothesis, but once these regulatory genes are thoroughly studied, they will be true hypotheses. Among regulatory genes (32 significant trans genes found result of this study), the "AT2G21830" gene belongs to the cysteine/histidine-rich DC1 domain protein family. It was shown from [33], the DC1 gene found in plants was named CaDC1 gene. The CaDC1 gene activates *Arabidopsis thaliana*'s defense mechanism during pathogen attack and provides ectopic CaDC1 expression. CaDC1 overexpression has a negative effect on the plant; it reduces the expression of the SA-responsive gene AtPR1, while increasing the expression of the JA-responsive gene AtPDF1.2. Because the SA-responsive gene activates and protects the plant's defense mechanism, its absence can result in plant cell death [33]. As a result, the AT2G21830 gene belongs to the CaDC1 gene family and plays a role in plant defense. In case of overexpression, it can trigger the SA-responsive gene or the downstream signaling mechanism of this gene, which can cause a loss of function or premature stop codon in those gene pairs. Only the AT2G21830 gene has been thoroughly studied out of the 32 trans genes. Due to the limited time of this thesis, the other 31 regulatory trans genes could not be analyzed in detail. In the future study, these genes will be thoroughly investigated.

Briefly, 32 significant trans genes were interpreted. However, additional research is required to demonstrate the reliability of the results. Because, while significant genes were identified in the results, it was detected that more SNPs than expected had an effect on the formation of over co-occurrence when the permutation based GWAS results were examined (Figure 4.4 and Figure 4.6). Previous research has also attempted to understand that there could be several possible causes for this situation. Strong linkage disequilibrium (LD) between SNPs, a strong relationship between SNP and traits, and bias can all cause genetic inflation [34]. Permutation-based GWAS was also used to reduce the error rate in non-normal distribution results, but this was insufficient to provide results with appropriate thresholds. The population structure results were non-normally distributed, making it difficult to distinguish between false positive and true positive SNPs. As a result of this situation, the genomic inflation values were quite high. Another reason could be that strong linkage disequilibrium (LD) between SNPs caused LD to form SNPs in neighboring regions. All of this is still just hypothesis, and research is still being conducted to identify genes that cause true association, which causes over co-occurrence gene pairs to produce premature stop codons in the same accessions.

Although these are still hypothesis, the study on *A. thaliana* (from [35]) suggested that the mutations in the genome were not as random as expected. It has been demonstrated that more variants are formed in non-synonymous and premature stop codons than in random mutations [34]. Nonsynonymous mutations are also thought to be important in plant adaptation. In addition to being harmful to the plant, premature stop codon formation might be an advantage such as making it easier to adapt to new environments and facilitating heredity in natural selection within the population. In the future, it should be investigated in which accessions the significant SNPs that cause the formation of premature stop codons occur. The geographic structure of the ecosystem from which these accessions are taken should also be considered. Thus, it can be deduced whether the cause is adaptation or the formation of a premature stop codon for another reason. In conclusion, understanding genetic variants at the population level is important for contributing to species origin and evolutionary studies on this subject, and more research and studies in this field should be carried out.

Supplementary Materials

Supplementary Table 1 Count the Number of Hits
Number of cis and trans hits are showed result of the R based GWAS. In the last column, it is shown separately for 738 files and in which files both trans and cis trans are shown.

	file_name	second_gene_name	total_hit	cis_hits	trans_hits	cis_1	trans_1	cis_trans_1
1	AT1G04380_AT2G27340.csv	AT2G27340	31	6	25	1	1	1.0
2	AT1G04380_AT4G01925.csv	AT4G01925	18	18	0	1	0	0.0
3	AT1G04380_AT4G04220.csv	AT4G04220	125	117	8	1	1	1.0
4	AT1G04380_AT5G51795.csv	AT5G51795	175	41	134	1	1	1.0
5	AT1G11180_AT1G29710.csv	AT1G29710	28	27	1	1	1	1.0
6	AT1G11180_AT1G65110.csv	AT1G65110	7992	63	7929	1	1	1.0
7	AT1G11180_AT2G15930.csv	AT2G15930	195	0	195	0	1	0.0
8	AT1G11180_AT2G16365.csv	AT2G16365	172	167	5	1	1	1.0
9	AT1G11180_AT2G16380.csv	AT2G16380	4648	20	4628	1	1	1.0
10	AT1G11180_AT2G42730.csv	AT2G42730	41	41	0	1	0	0.0
..	..	…	…	…	..	..	..	..
728	AT5G63630_AT5G20450.csv	AT5G20450	162	148	14	1	1	1.0
729	AT5G63630_AT5G51795.csv	AT5G51795	2110	80	2030	1	1	1.0
730	AT5G66630_AT1G29710.csv	AT1G29710	154	101	53	1	1	1.0
731	AT5G66630_AT2G04845.csv	AT2G04845	459	155	304	1	1	1.0
732	AT5G66630_AT3G50420.csv	AT3G50420	90	90	0	1	0	0.0
733	AT5G66630_AT4G34460.csv	AT4G34460	328	272	56	1	1	1.0
734	AT5G66630_AT5G20450.csv	AT5G20450	139	101	38	1	1	1.0
735	AT5G66630_AT5G27110.csv	AT5G27110	250	250	0	1	0	0.0
736	AT5G66630_AT5G45150.csv	AT5G45150	385	115	270	1	1	1.0
737	AT5G66630_AT5G48350.csv	AT5G48350	56	29	27	1	1	1.0
738	AT5G66630_AT5G54020.csv	AT5G54020	0	0	0	0	0	0.0

Supplementary Table 2 Count the Number of Hits: Permutation Based GWAS
Number of cis and trans hits are showed result of the permutation based GWAS. In the last column, it is shown separately for 738 files and in which files both trans and cis trans are shown.

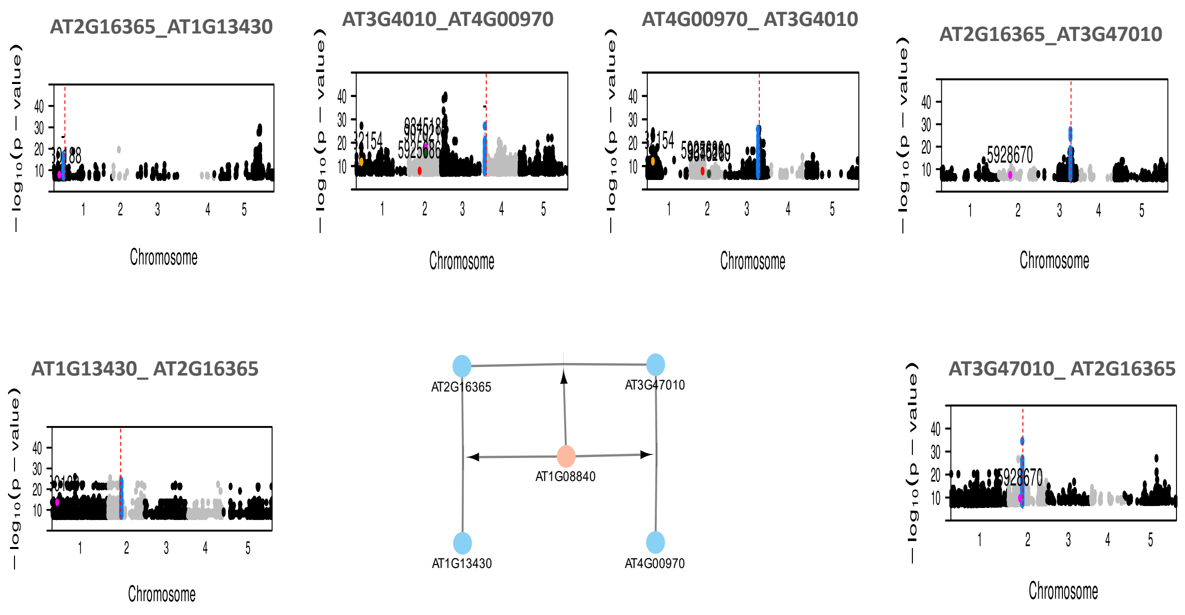
	file_name	second_gene_name	total_hit	cis_hits	trans_hits	cis_1	trans_1	cis_trans_1
1	AT1G04380_AT2G27340.csv	AT2G27340	782	71	711	1	1	1.0
2	AT1G04380_AT4G01925.csv	AT4G01925	234	84	150	1	1	1.0
3	AT1G04380_AT4G04220.csv	AT4G04220	806	189	617	1	1	1.0
4	AT1G04380_AT5G51795.csv	AT5G51795	2067	70	1997	1	1	1.0
5	AT1G11180_AT1G29710.csv	AT1G29710	116	79	37	1	1	1.0
6	AT1G11180_AT1G65110.csv	AT1G65110	25228	132	25096	1	1	1.0
7	AT1G11180_AT2G15930.csv	AT2G15930	251	0	251	0	1	0.0
8	AT1G11180_AT2G16365.csv	AT2G16365	4373	359	4014	1	1	1.0
9	AT1G11180_AT2G16380.csv	AT2G16380	18869	35	18834	1	1	1.0
10	AT1G11180_AT2G42730.csv	AT2G42730	107	60	47	1	1	1.0
..	…	…	…	…	…	…	…	…
728	AT5G63630_AT5G20450.csv	AT5G20450	451	315	136	1	1	1.0
729	AT5G63630_AT5G51795.csv	AT5G51795	1258	71	1187	1	1	1.0
730	AT5G66630_AT1G29710.csv	AT1G29710	719	121	598	1	1	1.0
731	AT5G66630_AT2G04845.csv	AT2G04845	0	0	0	0	0	0.0
732	AT5G66630_AT3G50420.csv	AT3G50420	327	275	52	1	1	1.0
733	AT5G66630_AT4G34460.csv	AT4G34460	864	374	490	1	1	1.0
734	AT5G66630_AT5G20450.csv	AT5G20450	26	26	0	1	0	0.0
735	AT5G66630_AT5G27110.csv	AT5G27110	387	386	1	1	1	1.0
736	AT5G66630_AT5G45150.csv	AT5G45150	9	9	0	1	0	0.0
737	AT5G66630_AT5G48350.csv	AT5G48350	26	10	16	1	1	1.0
738	AT5G66630_AT5G54020.csv	AT5G54020	136	135	1	1	1	1.0

Supplementary Table 3 Duplicated Files and Corresponding Trans Genes
The 32 most significant trans genes (column gens_trans) formed as a result of permutation based GWAS and the phenotype files that caused the formation of these trans genes are shown.

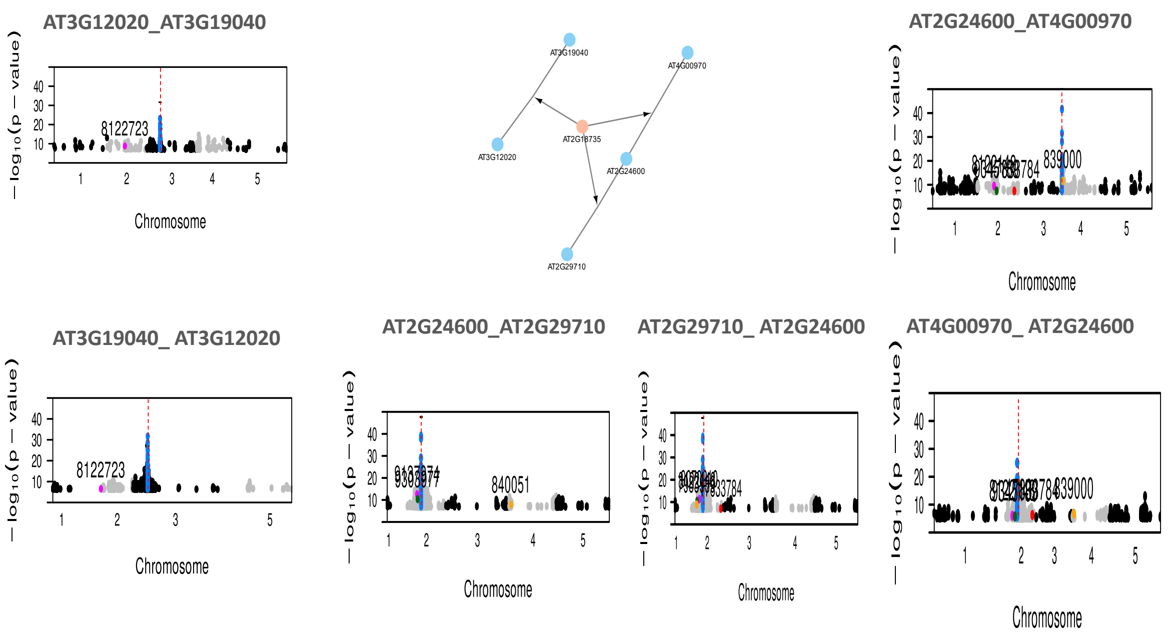
file_1	file_2	gens_trans
AT2G16365_AT1G13430.csv	AT1G13430_AT2G16365.csv	AT1G08840
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT1G08840
AT2G16365_AT3G47010.csv	AT3G47010_AT2G16365.csv	AT1G08840
AT2G41470_AT1G12700.csv	AT1G12700_AT2G41470.csv	AT1G09880
AT2G16365_AT1G13430.csv	AT1G13430_AT2G16365.csv	AT1G09880
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT1G09880
AT2G16365_AT3G47010.csv	AT3G47010_AT2G16365.csv	AT2G14080
AT4G00970_AT3G47010.csv	AT3G47010_AT4G00970.csv	AT2G14080
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G18380
AT2G29710_AT2G24600.csv	AT2G24600_AT2G29710.csv	AT2G18380
AT2G29710_AT2G16365.csv	AT2G16365_AT2G29710.csv	AT2G18380
AT3G12020_AT3G19040.csv	AT3G19040_AT3G12020.csv	AT2G18735
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G18735
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G18735
AT4G00970_AT1G42680.csv	AT1G42680_AT4G00970.csv	AT2G19110
AT2G16365_AT2G24600.csv	AT2G24600_AT2G16365.csv	AT2G19110
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G19110
AT2G29710_AT2G24600.csv	AT2G24600_AT2G29710.csv	AT2G21160
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G21160
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT2G21160
AT4G00970_AT1G42680.csv	AT1G42680_AT4G00970.csv	AT2G21830
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G21830
AT4G00970_AT3G47010.csv	AT3G47010_AT4G00970.csv	AT2G21830
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21830
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G42860
AT3G47010_AT2G16365.csv	AT2G16365_AT3G47010.csv	AT2G42860
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G42860
AT2G29710_AT4G00970.csv	AT4G00970_AT2G29710.csv	AT2G42860
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT4G01930
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT4G01930
AT4G00970_AT2G29710.csv	AT2G29710_AT4G00970.csv	AT4G01930
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT4G01930
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21170
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21170
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT2G21170
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21330
AT4G00970_AT1G42680.csv	AT1G42680_AT4G00970.csv	AT2G21330
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21330
AT2G29710_AT2G24600.csv	AT2G24600_AT2G29710.csv	AT2G21340
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G21340
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G21340
AT2G29710_AT2G24600.csv	AT2G24600_AT2G29710.csv	AT2G21350

AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G21350
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21350
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21370
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21370
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G21370
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21470
AT2G29710_AT2G24600.csv	AT2G24600_AT2G29710.csv	AT2G21470
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G21470
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21550
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21550
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G21550
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21870
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21870
AT4G00970_AT1G42680.csv	AT1G42680_AT4G00970.csv	AT2G21870
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G21920
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21920
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT2G21920
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G29200
AT2G16365_AT2G24600.csv	AT2G24600_AT2G16365.csv	AT2G29200
AT2G29710_AT2G16365.csv	AT2G16365_AT2G29710.csv	AT2G29200
AT2G16365_AT1G13430.csv	AT1G13430_AT2G16365.csv	AT1G08840
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT1G08840
AT2G16365_AT3G47010.csv	AT3G47010_AT2G16365.csv	AT1G08840
AT2G41470_AT1G12700.csv	AT1G12700_AT2G41470.csv	AT1G09880
AT2G16365_AT1G13430.csv	AT1G13430_AT2G16365.csv	AT1G09880
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT1G09880
AT4G00970_AT3G47010.csv	AT3G47010_AT4G00970.csv	AT2G14080
AT2G16365_AT3G47010.csv	AT3G47010_AT2G16365.csv	AT2G14080
AT3G47010_AT2G16365.csv	AT2G16365_AT3G47010.csv	AT2G14080
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G18380
AT4G00970_AT1G42680.csv	AT1G42680_AT4G00970.csv	AT2G18380
AT2G16365_AT2G29710.csv	AT2G29710_AT2G16365.csv	AT2G18380
AT3G12020_AT3G19040.csv	AT3G19040_AT3G12020.csv	AT2G18735
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G18735
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G18735
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G19110
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G19110
AT2G24600_AT2G16365.csv	AT2G16365_AT2G24600.csv	AT2G19110
AT2G29710_AT2G24600.csv	AT2G24600_AT2G29710.csv	AT2G21160
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G21160
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT2G21160
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G21830
AT3G47010_AT4G00970.csv	AT4G00970_AT3G47010.csv	AT2G21830
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT2G21830
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT2G21830
AT3G47010_AT2G16365.csv	AT2G16365_AT3G47010.csv	AT2G42860

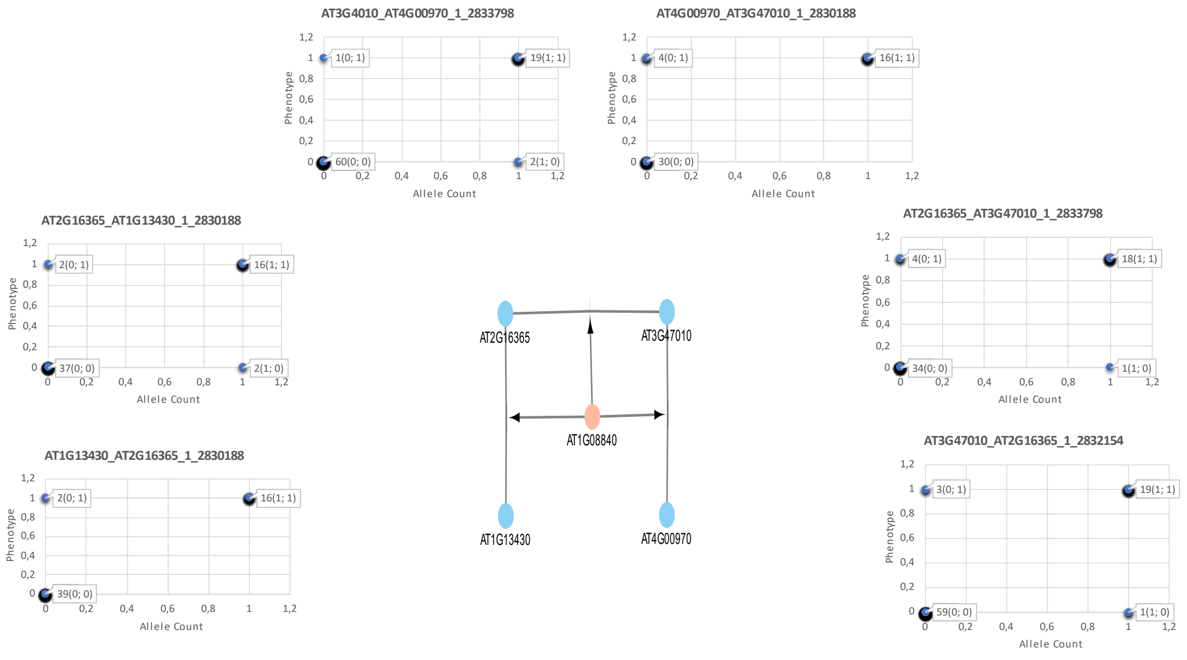
AT2G24600_AT4G00970.csv	AT4G00970_AT2G24600.csv	AT2G42860
AT4G00970_AT2G29710.csv	AT2G29710_AT4G00970.csv	AT2G42860
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT2G42860
AT2G29710_AT4G00970.csv	AT4G00970_AT2G29710.csv	AT4G01930
AT4G00970_AT2G24600.csv	AT2G24600_AT4G00970.csv	AT4G01930
AT2G24600_AT2G29710.csv	AT2G29710_AT2G24600.csv	AT4G01930
AT1G42680_AT4G00970.csv	AT4G00970_AT1G42680.csv	AT4G01930



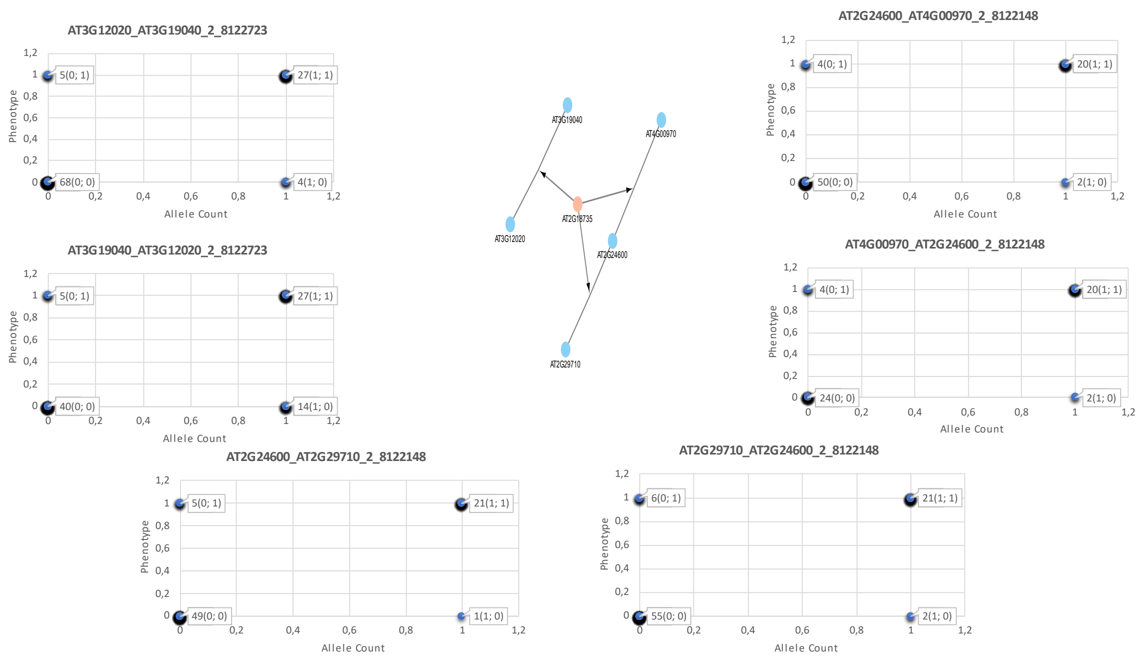
Supplementary Figure 1 Over Cooccurrence Pair Manhattan Plots Cause to Trans Gene AT1G08840
Manhattan plots of GWAS results for over-occurrence gene pairs causing trans gene " AT1G08840" shown. SNPs causing the most significant trans genes in 6 over-occurrence gene pairs, which are 3 duplicated versions of each other, are shown in red in the Manhattan plot.



Supplementary Figure 2 Over Cooccurrence Pair Manhattan Plots Cause to Trans Gene AT2G18735
Manhattan plots of GWAS results for over-occurrence gene pairs causing trans gene " AT2G18735" shown. SNPs causing the most significant trans genes in 6 over-occurrence gene pairs, which are 3 duplicated versions of each other, are shown in red in the Manhattan plot.



Supplementary Figure 3 Over Cooccurrence Pair Distribution Plots Cause to Trans Gene AT1G08840
Scatter plots of GWAS results for over-occurrence gene pairs causing trans gene " AT1G08840" shown. AT1G08840 trans gene occurred by 6 over-occurrence gene pairs, which are 3 duplicated versions of each other, are shown in the scatter plot. The scatter plots were drawn for alleles against the phenotype



Supplementary Figure 4 Over Cooccurrence Pair Distribution Plots Cause to Trans Gene AT2G18735
Scatter plots of GWAS results for over-occurrence gene pairs causing trans gene " AT2G18735" shown. AT2G18735 trans gene occurred by 6 over-occurrence gene pairs, which are 3 duplicated versions of each other, are shown in the scatter plot. The scatter plots were drawn for alleles against the phenotype

Appendix

The codes used in the results are shown in this section. The codes can be found at https://github.com/leylasrknt/GWAS_Premature_Stop_Codons. Only the code versions of the sections given in the Method section are shown in the rest of this section, and the Method section should be consulted as needed. Analyzes made in R studio are shown as "%%R". Analysis with Python is also indicated as "%% Python".

GWAS Data Analysis Part (R Based GWAS)

```
%%R
f95_gwas<-function(Y,n=2,X.folder='~/lastversion_gwas_analysis/F_95',K.file='~/
,→lastversion_gwas_analysis/F_95/K_F95.rda',incl.lm=FALSE,incl.
,→beta=FALSE,update.snps=FALSE,save.output=T,out.format='rda',generate.
,→plot=T,pre=colnames(Y)[n]) {
load(K.file)
for ( u in 1:6) {
1
filename<-paste(X.folder,'/X_f95_',u,'.rda',sep='')
load(filename)
if (u==1) {
results<-amm_gwas(Y,X,K,m=n,include.lm=incl.lm,calculate.effect.size=incl.
,→beta,update.top_snps=update.snps)
rm(X)
} else {
output<-amm_gwas(Y,X,K,m=n,include.lm=incl.lm,calculate.effect.size=incl.
,→beta,update.top_snps=update.snps)
results<-rbind(results,output)
rm(X)
}
}
#return(results)
if (save.output==T) {
auto<-0
if (('rda'%in%out.format)==T) {
name1<-paste(colnames(Y)[n],pre,'gwasf95.rda',sep='_')
save(results,file=name1)
auto<-1
}
if (('csv'%in%out.format)==T) {
name1b<-paste(colnames(Y)[n],pre,'gwasf95.csv',sep='_')
write.csv(results,file=name1b,row.names=FALSE)
auto<-1
}
if(auto==0){
cat('wrong output format to save the data specified','\n')}
}
if (generate.plot==T) {
name2<-paste(colnames(Y)[n],pre,'gwasf95.pdf',sep='_')
name3<-paste(colnames(Y)[n],pre,sep='_')
pdf(file=name2)
plot_gwas(results,maf_or_mac=2,mac=5,name=name3)
dev.off()
}}
```

GWAS Functions are Run

```
%%R
i <- as.numeric(Sys.getenv("SLURM_ARRAY_TASK_ID"))
##Load libraries and sources that are needed for reading files and creating a␣
,→list
library(tidyverse)
library(fs)
setwd("~/lastversion_gwas_analysis/F_95/")
source("gwas.r")
source("emma.r")
source("plots_gwas.r")
source('~/lastversion_gwas_analysis/f95_gwas.r')
##Creating an empty list and put all phenotype file in this list
setwd("~/lastversion_gwas_analysis/Over_Occ/Over_Coocurrence/") #phenotype␣
,→files folder
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
print(csvfiles[i])
Y[[i]] <- read.csv(file = csvfiles[i])
Y[[i]][, 1] <- as.numeric(Y[[i]][, 1])
Y[[i]][, 2] <- as.numeric(Y[[i]][, 2])
##Rewrite the same filename
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
# Get the start of filename prefix
prefix = gsub("_.*", "", x)
# Get the suffix number
suffix = gsub(".csv*", "", x)
colnames(file) <- paste(colnames(file), suffix, sep='_')
return(file)
})
##Gwas is running for each phenotype file with "f95_gwas" fuction
f95_gwas(Y[[i]],pre="")
```

GWAS_f952.r

```
%%R
###Load libraries that are needed for .csv file reading and creating list
##Load libraries and sources that are needed for reading files and creating a␣
,→list
library(tidyverse)
library(fs)
setwd("~/lastversion_gwas_analysis/F_95/")
source("gwas.r")
source("emma.r")
source("plots_gwas.r")
source('~/lastversion_gwas_analysis/f95_gwas.r')
csvfiles <-
dir(path = "~/lastversion_gwas_analysis/Over_Occ/Over_Coocurrence/",
pattern = ".csv",
full.names = TRUE)
##Creating an empty list and put all phenotype file in this list
Y <- list()
for (i in seq_along(csvfiles)) {
Y[[i]] <- read.csv(file = csvfiles[i])
Y[[i]][, 1] <- as.numeric(Y[[i]][, 1])
Y[[i]][, 2] <- as.numeric(Y[[i]][, 2])
}
Y<-set_names(Y,csvfiles )
##Gwas is running for each phenotype file with "f95_gwas" fuction
for (i in seq_along(csvfiles)) {
f95_gwas(Y[[i]],pre=i)
}
saveRDS(csvfiles, file="phenotype_list.Rds") #save the list
```

Filter Length of the Phenotype<50 and MPC<5

```
%% Python
import pandas as pd
import numpy as np
import os
import sys
import glob
os.chdir('/home/s417377/lastversion_gwas_analysis/Over_Occ/Over_Coocurrence/')
df = pd.DataFrame(columns=('File_name', 'Phenotype_length'))
for index,i in enumerate(os.listdir('.')):
df.loc[index] = [i,len(pd.read_csv(i).index)]
array=[]
for file_to_read in glob.glob("*.csv"):
df2 = pd.read_csv(file_to_read)
a=df2["Phenotype"].sum()
array.append(a)
b=pd.DataFrame(array)
df["ko"]=b
###add new column as non_SNPs,
df["non_ko"]=df["Phenotype_length"]-df["ko"]
#f
##add new column min SNPs or non_SNPs
df['MPC'] = df[['ko','non_ko']].min(axis=1)
#f
##add new column min_value<5 or phenotype_length<50
df["MPC<5"]=np.where(df['MPC'] <5, 1, 0)
df["Phe_length<50"]=np.where(df['Phenotype_length'] <50, 1, 0)
df.loc[(df['MPC'] < 5) | (df['Phenotype_length'] <50),␣
,→'MPC<5_or_Phe_length<50'] = 1
df.loc[(df['MPC'] >= 5) & (df['Phenotype_length'] >=50),␣
,→"MPC<5_or_Phe_length<50"] = 0
df
##save
```

Permutation Based GWAS

```
#!/bin/bash
FILES=(AT*.csv) FILE=${FILES[$SLURM_ARRAY_TASK_ID]}
singularity exec --bind /storage/full-share/permGWAS permgwas.sing python3␣
↪permGWAS/permGWAS.py -x F95.h5 -y ${FILE}          --out_dir outfile_${FILE} --perm␣
↪100
```

Permutation Based Threshold for GWAS

```
%%R
##permgwas:find pvalue_threshold each phenotype separetly then apply this␣
,→threshold value for filtering step
library(tidyverse)
library(fs)
setwd("~/lastversion_gwas_analysis/PermGWAS_Scripts/")

list_of_files <- list.files(path = "~/PermGWAS/",
recursive = TRUE,
pattern = "*min_",
full.names = TRUE)
Y <- lapply(list_of_files, function(x) {
file <- read.csv(x)
return(file)
})
###find perm_threshold for each phenotype
new_list<-vector()
for (i in 1:length((list_of_files))){
A<-Y[[i]][order(Y[[i]][,2]),]
perm_thres<-A[5,2]
new_list<-c(new_list,perm_thres)
new_list<-as.matrix(new_list, sep=",")
new<-as.data.frame(new_list)
}
####use this threshold and filter perm_gwas result
csv_files <- list.files(path = "~/PermGWAS/",
recursive = TRUE,
pattern = "*gwasp_",
full.names = TRUE)
Y_ <- lapply(csv_files, function(x) {
file <- read.csv(x)
return(file)
})
a<-read.csv("~/lastversion_gwas_analysis/names.csv")
for (i in 1:length((csv_files))){
R<-subset(Y_[[i]],Y_[[i]][["p_value"]]<(new[i,]))
write.csv(R, file=paste(a$x[i] , sep=""))
}
```

Maf Filter

```
%%R
#maf filtered >0.05
library(tidyverse)
library(fs)
setwd("~/lastversion_gwas_analysis/PermGWAS_Scripts/permgwas_filtered_pvalue/")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((list_of_files))){
R<-subset(Y[[i]],Y[[i]][["maf"]]>0.05)
write.csv(R, file=paste( "filtered_",csvfiles[i],".csv", sep=""))
}
```

Count Hits

```
%%R
##count cis hits
library(tidyverse)
library(fs)
setwd("~/lastversion_gwas_analysis/Over_Occ/stringent_correction_738/")
3
ara11<-read.csv("~/lastversion_gwas_analysis/ara11_50.csv")
small_gene<-read.csv("~/lastversion_gwas_analysis/Over_Occ/Small_gene.csv")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
new_list<-list()
for (i in 1:length((csvfiles))){
b<-ara11[which(ara11$Gene==small_gene$second_gene_name[i]),]
c<-sum(b$Start<Y[[i]][["Pos"]] & Y[[i]][["Pos"]]<b$Stop)
new_list<-c(new_list,c)
new_list<-as.matrix(new_list, sep=",")
write.csv( new_list, "small_gene_cis.csv")
}
####change the folder name########
folder = "~/lastversion_gwas_analysis/PermGWAS_Scripts/deneme2//"
```

```
files <- list.files(folder,pattern = "trans_.*.csv",full.names = T)
sapply(files,FUN=function(eachPath){
file.rename(from=eachPath,to= sub(pattern="trans_", paste0(""),eachPath))
})
```

Count Number of Cis and Trans Hits

```
%% Python
import pandas as pd
import numpy as np
import os
import sys
import glob
```

```
##add total SNPs hits, cis hits and trans hits as a new cloumns into␣
,→result(Small_gene.csv) then sum only trans only cis and non hits
os.chdir('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/maf_filtered/
,→')
total_hits = pd.DataFrame(columns=('file_name', 'total_hits'))
for index,i in enumerate(os.listdir('.')):
total_hits.loc[index] = [i,len(pd.read_csv(i).index)]
total_hits=total_hits.sort_values("file_name")
total_hits
total_hits.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→total_hit.csv', index=False)
result=pd.read_csv("/home/s417377/lastversion_gwas_analysis/Over_Occ/Small_gene.
,→csv")
total_hits=pd.read_csv("/home/s417377/lastversion_gwas_analysis/
,→PermGWAS_Scripts/total_hit.csv")
total_hits=total_hits["total_hits"]
result["total_hit"]=total_hits
cis_hits=pd.read_csv("/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→small_gene_cis.csv")
cis_hits=cis_hits["V1"]
cis_hits
result["cis_hits"]=cis_hits
result
result["trans_hits"]=result["total_hit"]-result["cis_hits"]
result
##add new column if cis_hits>0 and trans_hits>0:
result["cis_1"]=np.where(result['cis_hits'] >0, 1, 0)
result["trans_1"]=np.where(result['trans_hits'] >0, 1, 0)
result.loc[(result['cis_hits'] > 0) & (result['trans_hits'] >0),␣
,→'cis_trans_1'] = 1
result.loc[(result['cis_hits'] <= 0) | (result['trans_hits'] <=0),␣
,→"cis_trans_1"] = 0
result.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→cis_trans_hit.csv', index=False)
result
```

Filter

```
%% Python
##files does not contain gene "AT2G15930" filtered to continue next steps
#filter="AT2G15930"
filtered_files=(result.loc[result['second_gene_name'] != "AT2G15930"])
filtered_files=filtered_files.sort_values("file_name")
filtered_files.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→cis_trans_filtered_hit.csv', index=False)
filtered_files
```

Count Hits

```
%% Python
import pandas as pd
from tabulate import tabulate
result=pd.read_csv("/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→cis_trans_filtered_hit.csv")
hit_sum_result=[]
cis_sum=result.iloc[1:724, 5].sum()
trans_sum=result.iloc[1:724, 6].sum()
cis_trans_sum=result.iloc[1:724, 7].sum()
only_cis=cis_sum-cis_trans_sum
only_trans=trans_sum-cis_trans_sum
non_hit=724-only_cis-only_trans-cis_trans_sum
hit_sum_result=["cis_only",only_cis],["trans_only",only_trans],["cis_trans_sum",cis_trans_sum],["hit_sum_result
col_names=["hits","hit_sums"]
print(tabulate(hit_sum_result, headers=col_names))
```

Filter Trans Hits

```
%% Python
df=pd.read_csv("/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→cis_trans_filtered_hit.csv")
df
###filter trans files
trans_files=(result.loc[result['trans_1'] == 1])
trans_files.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→trans_filtered_hit.csv', index=False)
trans_files
```

```
%% Python
###filter non_trans files
non_files=(result.loc[result['trans_1'] != 1])
non_files.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→non_filtered_hit.csv', index=False)
non_files=non_files["file_name"]
non_files
non_files.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→discard_list.csv', index=False, header=False)
```

Cis Hits Removed from Files

```
%%R
####filter stringent result files, only contain trans hits
####filter stringent result files, only contain trans hits
setwd("~/lastversion_gwas_analysis/Over_Occ/")
small_gene<-read.csv("~/lastversion_gwas_analysis/Over_Occ/trans_filtered_hit.
,→csv")
ara11<-read.csv("~/lastversion_gwas_analysis/ara11_50.csv")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
8
b<-ara11[which(ara11$Gene==S$second_gene_name[i]),]
trials<-subset(Y[[i]],!(b$Start<Y[[i]][["Pos"]] & Y[[i]][["Pos"]]<b$Stop))
write.csv(trials, file=paste( "filtered_",csvfiles[i],".csv", sep=""))
}
```

Count Number of Trans Genes

```
%%R
##filtering overlapping genes in 10 kb window
library(tidyverse)
SNPs<-read.csv('~/lastversion_gwas_analysis/SNPs_11_10.csv')
ara11<-read.csv("~/lastversion_gwas_analysis/ara11_50.csv")
small_gene<-pd.read_csv("~/lastversion_gwas_analysis/Over_Occ/Small_gene.csv")
A<-read.csv('~/lastversion_gwas_analysis/PermGWAS_Scripts/trans_filtered_hit.
,→csv')
S<-A[,1:5]
S$gens_trans<-NA
S$gens_trans_10kb<-NA
####apply all files see if it is any pattern
setwd("~/lastversion_gwas_analysis/Over_Occ/trans_hits/")
Trans <- list()
csvfiles<-list.files(pattern = "*filtered_")
Trans <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Z<-merge(Trans[[i]][,c(4,9)],SNPs[,c(5,17,18)],by='SNP')
S[i,6]<-paste(as.character(na.omit(unique(Z$gene_ara11))),collapse=',')
S[i,7]<-paste(unique(unlist(strsplit(as.character(na.
,→omit(unique(Z$genes_10kb__ara11))),split=','))),collapse=',')
cat(i,'\n')
}
a<-nchar(gsub('[^,]','',S$gens_trans))+1
df<-as.data.frame(a)
S['gene_count']<-NA
S['gene_count']<-df
write.csv(S, "~/lastversion_gwas_analysis/overlapping_result/
,→perm_overlapping_genes.csv")
b<-S[duplicated(S$gens_trans),]
c<-overlapping_genes[duplicated(overlapping_genes$gens_trans_10kb),]
d<-overlapping_genes[duplicated(overlapping_genes$gens_trans) &␣
,→duplicated(overlapping_genes$gens_trans_10kb),]
```

Discard Files

```
%% Python
import os, glob
###discard files from non_filetered_hits.csv
path = ('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/maf_filtered/
,→')
os.chdir(path)
with open('non_list.csv', "r") as list_file:
_list = list_file.read().splitlines()
[os.remove(os.path.join(path,f)) for f in _list]
```

```
os.chdir('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/MAC_filtered/
,→')
```

```
##files does not contain gene "AT2G15930" filtered to continue next steps
#filter="AT2G15930"
filtered_files=(result.loc[result['second_gene_name'] == "AT2G15930"])
filtered_files=filtered_files.sort_values("file_name")
filtered_files.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→filtered_hit.csv', index=False)
filtered_files
filtered_files=filtered_files["file_name"]
filtered_files
filtered_files.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→non_list.csv', index=False, header=False)
filtered_files
```

Allele Count

```
%% Python
os.chdir('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/')
import os
import pandas as pd
path = ('/home/s417377/PermGWAS/PermGWAS_OverCo/')
files = [file for file in os.listdir(path) if file.endswith(".csv")]
output = pd.Series(name="Rows", dtype=int)
for file in files:
df = pd.read_csv(os.path.join(path, file))
output.at[file.replace(".csv", "")] = df.shape[0]
#output.at["Total"] = output.sum()
output.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→Allele_list.csv', header=True)
result=pd.read_csv("/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→Allele_list.csv")
result
result.columns=["file_name", "allele_count"]
result=result.sort_values("file_name")
result.to_csv('/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→Allele_count.csv', index=False ,header=True)
result
```

Fake Genes

```
%%R
#####duplicated_genes
library(fs)
library(tidyverse)
library(dplyr)
duplicated_genes<-read.csv("/storage/full-share/evolgen/leyla/
,→overlapping_result/duplicated_overlapping_genes.csv")
S<-duplicated_genes[which(!duplicated_genes$duplicated==""),]
a<-nchar(gsub('[^,]',",S$duplicated))+1
df<-as.data.frame(a)
S['gene_count']<-NA
S['gene_count']<-df
S['fake_genes']<-NA
w1<-unlist(strsplit(S$duplicated, ","))
W1<-table(w1)
W1<-sort(W1, decreasing = TRUE)
W_1<-W1[1:32]
setwd("~/lastversion_gwas_analysis/PermGWAS_Scripts/fake_genes_duplicated/")
load("/storage/full-share/evolgen/results_110522.rda")
a<-1
#d:file contain the file names, trans genes, number of trans genes, column 5␣
,→has information number of trans genes, column 13
#put fake genes according to column 12: number of rel trans genes,
for(a in 1:100){
u<-1
for (u in 1:nrow(S)){
i<-sample(1:33056, S[u,5])
S[u,6]<-as.character(paste(ara11[i,5],collapse =","))
}
w<-unlist(strsplit(S$fake_genes, ","))
W<-table(w)
W<-sort(W, decreasing = TRUE)
W[1:32]
W_<-W[1:32]
W_<-W[1:32]
W_<-as.matrix(W_, sep=",")
new<-as.data.frame(W_)
write.csv(new, file=paste( "filtered_",a,".csv", sep=""))
13
}
```

```
data_all <- list.files(path = "~/lastversion_gwas_analysis/PermGWAS_Scripts/
,→fake_genes_duplicated/", # Identify all CSV files
pattern = "*.csv", full.names = TRUE) %>%
lapply(read_csv) %>% # Store all files in list
bind_rows # Combine data sets into␣
,→one data set
data_all
S_<-S[,2:6]
write.csv(a_, "~/lastversion_gwas_analysis/overlapping_result/ger_1_2.csv", row.
,→names = FALSE)
mean(data_all$W1_1)
#6.1875
sd(data_all$W_1)
#0.5922892
#fake data results
mean(data_all$V1)
#3.036875
sd(data_all$V1)
# 0.2473042
```

Gene list for phenotype

```
import pandas as pd
c=pd.read_csv("/home/s417377/lastversion_gwas_analysis/PermGWAS_Scripts/
,→duplicated_file_names.csv")
c=c.explode('gens_trans')
c
c_=(c.loc[(c["gens_trans"]== "AT2G21830") | (c["gens_trans"]== "AT2G42860")|␣
,→(c["gens_trans"]== "AT4G01930")| (c["gens_trans"]== "AT1G08840")|␣
,→(c["gens_trans"]== "AT1G09880")| (c["gens_trans"]== "AT2G14080")|␣
,→(c["gens_trans"]== "AT2G18380")| (c["gens_trans"]== "AT2G18735")|␣
,→(c["gens_trans"]== "AT2G19110")| (c["gens_trans"]== "AT2G21160")])
#b_=(b.loc[(b["gens_trans"]== "AT3G44630") | (b["gens_trans"]== "AT4G16920")])
#result.loc[(result['cis_hits'] > 0) | (result['trans_hits'] >0),␣
,→'cis_trans_1'] = 1
c_=c_.sort_values("gens_trans")
c_.to_csv('/home/s417377/lastversion_gwas_analysis/overlapping_result/
,→gene1_2_trans_result_genes.csv', index=False)
c_1=(c.loc[(c["gens_trans"]== "AT2G21170 ") | (c["gens_trans"]== "AT2G21330")|␣
,→(c["gens_trans"]== "AT2G21340")| (c["gens_trans"]== "AT2G21350")|␣
,→(c["gens_trans"]== "AT2G21370")| (c["gens_trans"]== "AT2G21470")|␣
,→(c["gens_trans"]== "AT2G21550")| (c["gens_trans"]== "AT2G21870")|␣
,→(c["gens_trans"]== "AT2G21920")| (c["gens_trans"]==␣
,→"AT2G29200")|(c["gens_trans"]=="AT2G21170")])
#b_=(b.loc[(b["gens_trans"]== "AT3G44630") | (b["gens_trans"]== "AT4G16920")])
#result.loc[(result['cis_hits'] > 0) | (result['trans_hits'] >0),␣
,→'cis_trans_1'] = 1
c_1=c_1.sort_values("gens_trans")
c_1.to_csv('/home/s417377/lastversion_gwas_analysis/overlapping_result/
,→gene1_2_trans_result_genes2.csv', index=False)
c_2=(c.loc[(c["gens_trans"]== "AT2G43850") | (c["gens_trans"]== "AT2G43860")|␣
,→(c["gens_trans"]== "AT3G10440")| (c["gens_trans"]== "AT3G52140")|␣
,→(c["gens_trans"]== "AT3G55200")| (c["gens_trans"]== "AT4G02190")|␣
,→(c["gens_trans"]== "AT4G04015")| (c["gens_trans"]== "AT4G08990")|␣
,→(c["gens_trans"]== "AT5G41180")| (c["gens_trans"]== "AT5G41190")|␣
,→(c["gens_trans"]== "AT5G41200")| (c["gens_trans"]== "AT5G41210")])
#b_=(b.loc[(b["gens_trans"]== "AT3G44630") | (b["gens_trans"]== "AT4G16920")])
#result.loc[(result['cis_hits'] > 0) | (result['trans_hits'] >0),␣
,→'cis_trans_1'] = 1
c_2=c_2.sort_values("gens_trans")
c_2.to_csv('/home/s417377/lastversion_gwas_analysis/overlapping_result/
,→gene1_2_trans_result_genes3.csv', index=False)
frames = [c_, c_1, c_2]
result = pd.concat(frames)
display(result)
```

```
result.to_csv('/home/s417377/lastversion_gwas_analysis/overlapping_result/
,→gene1_2_trans_result_genes.csv', index=False)
```

AC_1 Count and AC_0 Count

```
%%R
library(tidyverse)
SNPs<-read.csv('~/lastversion_gwas_analysis/SNPs_11_10.csv')
a<-read.csv("/home/s417377/lastversion_gwas_analysis/overlapping_result/
,→ger_1_1_32.csv")
#AT2G21830
AT2G21830<- SNPs[which(SNPs$gene == "AT2G21830"),]
setwd("/home/s417377/trans_genes/AT2G21830")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
}
for (i in 1:length(Y)){
Z<-merge(AT2G21830[,c(4,5)],Y[[i]][,c(13,14)],by='SNP')
b[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
cat(i,'\n')
}
load("/home/s417377/lastversion_gwas_analysis/F_95/X_f95_2.rda")
b1<-X[,"2- 9304447"]
b2<-X[,"2- 9304693"]
b1<-as.data.frame(b1)
b2<-as.data.frame(b2)
write.csv( b1,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→filtered_2_9304447.csv", row.names = TRUE)
write.csv( b2,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→filtered_2_9304693.csv", row.names = TRUE)
setwd("~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/")
SNP <- list()
csvfiles<-list.files(pattern = "*.csv")
SNP <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
e<-merge(Y[[1]], SNP[[1]], by.="X",)
e<-merge(Y[[1]], SNP[[1]], by="X")
e1<-merge(Y[[2]], SNP[[1]], by="X")
e2<-merge(Y[[3]], SNP[[1]], by="X")
e3<-merge(Y[[4]], SNP[[1]], by="X")
e4<-merge(Y[[5]], SNP[[1]], by="X")
e5<-merge(Y[[6]], SNP[[2]], by="X")
e6<-merge(Y[[7]], SNP[[1]], by="X")
e7<-merge(Y[[8]], SNP[[2]], by="X")
write.csv( e,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT1G42680_AT4G00970_2_9304447.csv", row.names = TRUE)
write.csv( e1,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT2G24600_AT2G29710_2_9304447.csv", row.names = TRUE)
write.csv( e2,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT2G24600_AT4G00970_2_9304447.csv", row.names = TRUE)
write.csv( e3,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT2G29710_AT2G24600_2_9304447.csv", row.names = TRUE)
write.csv( e4,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT3G47010_AT4G00970_2_9304447.csv", row.names = TRUE)
write.csv( e5,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT4G00970_AT1G42680_2_9304693.csv", row.names = TRUE)
write.csv( e6,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT4G00970_AT2G24600_2_9304447.csv", row.names = TRUE)
write.csv( e7,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→AT4G00970_AT3G47010_2_9304693.csv", row.names = TRUE)
#
AT2G18735<- SNPs[which(SNPs$gene == "AT2G18735"),]
setwd("/home/s417377/trans_genes/AT2G18735")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
}
for (i in 1:length(Y)){
Z<-merge(AT2G18735[,c(4,5)],Y[[i]][,c(13,14)],by='SNP')
b[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
cat(i,'\n')
}
load("/home/s417377/lastversion_gwas_analysis/F_95/X_f95_2.rda")
b1<-X[,"2- 8122148"]
b2<-X[,"2- 8122723"]
b1<-as.data.frame(b1)
b2<-as.data.frame(b2)
write.csv( b1,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→filtered_2_8122148.csv", row.names = TRUE)
write.csv( b2,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→filtered_2_8122723.csv", row.names = TRUE)
write.csv( b,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→SNPsAT2G18735_.csv", row.names = TRUE)
#
AT1G08840<- SNPs[which(SNPs$gene == "AT1G08840"),]
setwd("/home/s417377/trans_genes/AT1G08840")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
}
for (i in 1:length(Y)){
Z<-merge(AT1G08840[,c(4,5)],Y[[i]][,c(13,14)],by='SNP')
b[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
cat(i,'\n')
}
load("/home/s417377/lastversion_gwas_analysis/F_95/X_f95_1.rda")
b1<-X[,"1- 2830188"]
b2<-X[,"1- 2833798"]
b3<-X[,"1- 2832154"]
b1<-as.data.frame(b1)
b2<-as.data.frame(b2)
b3<-as.data.frame(b3)
write.csv( b1,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→filtered_1_2830188.csv", row.names = TRUE)
write.csv( b2,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→filtered_1_2833798.csv", row.names = TRUE)
write.csv( b3,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→filtered_1_2832154.csv", row.names = TRUE)
write.csv( b,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→________________SNPsAT1G08840_.csv", row.names = TRUE)
#AT1G08840
#"AT1G13430_AT2G16365.csv"
setwd("/home/s417377/trans_genes/AT1G08840")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
}
d<-a[which("AT1G13430_AT2G16365.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[1]][,c(5,14)],by='SNP')
b[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b<-as.data.frame(b)
}
Z<-merge(mylist[[1]][,c(4,5)],Y[[1]][,c(5,14)],by='SNP')
b<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
write.csv( b,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→SNPsAT1G08840_.csv", row.names = TRUE)
#"AT2G16365_AT1G13430.csv"
d<-a[which("AT2G16365_AT1G13430.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b1<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[2]][,c(5,14)],by='SNP')
b1[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b1<-as.data.frame(b1)
}
#"AT2G16365_AT3G47010.csv"
d<-a[which("AT2G16365_AT3G47010.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b2<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[3]][,c(5,14)],by='SNP')
b2[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b2<-as.data.frame(b2)
}
#"AT3G47010_AT2G16365.csv"
d<-a[which("AT3G47010_AT2G16365.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
```

```
}
b3<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[4]][,c(5,14)],by='SNP')
b3[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b3<-as.data.frame(b3)
}
#"AT3G47010_AT4G00970.csv"
d<-a[which("AT3G47010_AT4G00970.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
20
}
b4<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[5]][,c(5,14)],by='SNP')
b4[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b4<-as.data.frame(b4)
}
#"AT4G00970_AT3G47010.csv"
d<-a[which("AT4G00970_AT3G47010.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b5<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[6]][,c(5,14)],by='SNP')
b5[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b5<-as.data.frame(b5)
}
#AT2G18735
setwd("/home/s417377/trans_genes/AT2G18735")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
}
#"AT2G29710_AT2G24600.csv"
d<-a[which("AT2G29710_AT2G24600.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[1]][,c(5,14)],by='SNP')
f[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
```

```
f<-as.data.frame(f)
}
#"AT2G24600_AT2G29710.csv"
d<-a[which("AT2G24600_AT2G29710.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f1<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[2]][,c(5,14)],by='SNP')
f1[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f1<-as.data.frame(f1)
}
#"AT2G24600_AT4G00970.csv"
d<-a[which("AT2G24600_AT4G00970.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f2<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[3]][,c(5,14)],by='SNP')
f2[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f2<-as.data.frame(f2)
}
#"AT3G12020_AT3G19040.csv"
d<-a[which("AT3G12020_AT3G19040.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f3<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[4]][,c(5,14)],by='SNP')
f3[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f3<-as.data.frame(f3)
}
#"AT3G19040_AT3G12020.csv"
d<-a[which("AT3G19040_AT3G12020.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f4<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[5]][,c(5,14)],by='SNP')
f4[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
```

```
f4<-as.data.frame(f4)
}
#"AT4G00970_AT2G24600.csv"
d<-a[which("AT4G00970_AT2G24600.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f5<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[6]][,c(5,14)],by='SNP')
f5[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f5<-as.data.frame(f5)
}
```

Draw Venn Diagrams

```
#Import libraries
from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
from matplotlib_venn import venn3, venn3_circles
from matplotlib import pyplot as plt
%matplotlib inline
from collections import Counter
from matplotlib_venn import venn2, venn3
import matplotlib.pyplot as plt
sets = Counter()
sets['100'] = 7
sets['010'] = 62
sets['001'] = 16
sets['110'] = 639
sets['101'] = 0
sets['111'] = 0
labels = ('Cis', 'Trans', 'Non Hits')
plt.figure(figsize=(7,7))
ax = plt.gca()
colors = ['darkviolet','deepskyblue','blue']
v = venn3(subsets=sets, set_labels=(",",","), ax=ax,set_colors=
colors,alpha=0.7)
h = []
for i in sets:
h.append(v.get_patch_by_id(i))
l = ['Trans Hits only','Cis Hits only','Non Hits','Cis and Trans Hits shared␣
,→',",
","]
ax.legend(handles=h, labels=l, title="Legend",loc=3)
#plt.title('Number of SNPs Hits ')
output_file = "/home/s417377/lastversion_gwas_analysis/overlapping_result/
,→Perm_GWAS_Hits.png"
plt.savefig(output_file, dpi=700, facecolor='w')
plt.show()
sets = Counter()
sets['100'] = 11
sets['010'] = 201
sets['001'] = 34
sets['110'] = 478
sets['101'] = 0
sets['111'] = 0
labels = ('Cis', 'Trans', 'Non Hits')
plt.figure(figsize=(7,7))
ax = plt.gca()
colors = ['darkviolet','deepskyblue','blue']
v = venn3(subsets=sets, set_labels=(",",","), ax=ax,set_colors=
colors,alpha=0.7)
25
h = []
for i in sets:
h.append(v.get_patch_by_id(i))
l = ['Trans Hits only','Cis Hits only','Non Hits','Cis and Trans Hits shared␣
,→',",
","]
ax.legend(handles=h, labels=l, title="Legend",loc=3)
#plt.title('Number of SNPs Hits ')
output_file = "/home/s417377/lastversion_gwas_analysis/overlapping_result/
,→GWAS_Hits.png"
plt.savefig(output_file, dpi=700, facecolor='w')
plt.show()
```

Common Trans Genes

```
%% R
#AT1G08840
#"AT1G13430_AT2G16365.csv"
setwd("/home/s417377/trans_genes/AT1G08840")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
} d<-a[which("AT1G13430_AT2G16365.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[1]][,c(5,14)],by='SNP')
b[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b<-as.data.frame(b)
}
Z<-merge(mylist[[1]][,c(4,5)],Y[[1]][,c(5,14)],by='SNP')
b<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
write.csv( b,"~/lastversion_gwas_analysis/PermGWAS_Scripts/AC_1_0_files/
,→SNPsAT1G08840_.csv", row.names = TRUE)
#"AT2G16365_AT1G13430.csv"
d<-a[which("AT2G16365_AT1G13430.csv"==a$file_1),]
d<-unique(d$gens_trans)
27
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b1<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[2]][,c(5,14)],by='SNP')
b1[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b1<-as.data.frame(b1)
}
#"AT2G16365_AT3G47010.csv"
```

```
d<-a[which("AT2G16365_AT3G47010.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b2<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[3]][,c(5,14)],by='SNP')
b2[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b2<-as.data.frame(b2)
}
#"AT3G47010_AT2G16365.csv"
d<-a[which("AT3G47010_AT2G16365.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b3<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[4]][,c(5,14)],by='SNP')
b3[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
28
b3<-as.data.frame(b3)
}
#"AT3G47010_AT4G00970.csv"
d<-a[which("AT3G47010_AT4G00970.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b4<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[5]][,c(5,14)],by='SNP')
b4[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b4<-as.data.frame(b4)
}
#"AT4G00970_AT3G47010.csv"
d<-a[which("AT4G00970_AT3G47010.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
b5<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[6]][,c(5,14)],by='SNP')
b5[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
b5<-as.data.frame(b5)
}
```

```
#AT2G18735
setwd("/home/s417377/trans_genes/AT2G18735")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
29
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
}
#"AT2G29710_AT2G24600.csv"
d<-a[which("AT2G29710_AT2G24600.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
} f<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[1]][,c(5,14)],by='SNP')
f[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f<-as.data.frame(f)
}
#"AT2G24600_AT2G29710.csv"
d<-a[which("AT2G24600_AT2G29710.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f1<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[2]][,c(5,14)],by='SNP')
f1[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f1<-as.data.frame(f1)
}
#"AT2G24600_AT4G00970.csv"
d<-a[which("AT2G24600_AT4G00970.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f2<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[3]][,c(5,14)],by='SNP')
f2[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f2<-as.data.frame(f2)
}
#"AT3G12020_AT3G19040.csv"
d<-a[which("AT3G12020_AT3G19040.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
```

```
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f3<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[4]][,c(5,14)],by='SNP')
f3[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f3<-as.data.frame(f3)
}
#"AT3G19040_AT3G12020.csv"
d<-a[which("AT3G19040_AT3G12020.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f4<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[5]][,c(5,14)],by='SNP')
f4[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f4<-as.data.frame(f4)
}
#"AT4G00970_AT2G24600.csv"
d<-a[which("AT4G00970_AT2G24600.csv"==a$file_1),]
d<-unique(d$gens_trans)
d<-as.data.frame(d)
colnames(d)<-"gene"
mylist<-c()
31
for (i in 1:nrow(d)){
e<-SNPs[which(d$gene[i]==SNPs$gene),]
mylist[[length(mylist)+1]]<-e
cat(i,'\n')
}
f5<-c()
for (i in 1:length((mylist))){
Z<-merge(mylist[[i]][,c(4,5)],Y[[6]][,c(5,14)],by='SNP')
f5[i]<-as.data.frame(paste(as.character(na.omit((Z$SNP))),collapse=','))
f5<-as.data.frame(f5)
}
```

Manhattan Plot

```
%% R
library(lattice)
manhattan.plot<-function(chr, pos, pvalue,
sig.level=NA, annotate=NULL, ann.default=list(),
should.thin=T, thin.pos.places=2, thin.logp.places=2,
xlab="Chromosome", ylab=expression(-log[10](p-value)),
col=c("black","gray"), panel.extra=NULL, pch=20, cex=0.
,→8,...) {
if (length(chr)==0) stop("chromosome vector is empty")
if (length(pos)==0) stop("position vector is empty")
if (length(pvalue)==0) stop("pvalue vector is empty")
#make sure we have an ordered factor
if(!is.ordered(chr)) {
chr <- ordered(chr)
} else {
chr <- chr[,drop=T]
}
#make sure positions are in kbp
```

```
if (any(pos>1e6)) pos<-pos/1e6;
#calculate absolute genomic position
#from relative chromosomal positions
posmin <- tapply(pos,chr, min);
posmax <- tapply(pos,chr, max);
posshift <- head(c(0,cumsum(posmax)),-1);
names(posshift) <- levels(chr)
genpos <- pos + posshift[chr];
getGenPos<-function(cchr, cpos) {
p<-posshift[as.character(cchr)]+cpos
return(p)
}
#parse annotations
grp <- NULL
ann.settings <- list()
label.default<-list(x="peak",y="peak",adj=NULL, pos=3, offset=0.5,
col=NULL, fontface=NULL, fontsize=NULL, show=F)
parse.label<-function(rawval, groupname) {
r<-list(text=groupname)
if(is.logical(rawval)) {
if(!rawval) {r$show <- F}
} else if (is.character(rawval) || is.expression(rawval)) {
if(nchar(rawval)>=1) {
r$text <- rawval
}
} else if (is.list(rawval)) {
r <- modifyList(r, rawval)
}
return(r)
}
if(!is.null(annotate)) {
if (is.list(annotate)) {
grp <- annotate[[1]]
} else {
grp <- annotate
}
if (!is.factor(grp)) {
grp <- factor(grp)
}
} else {
grp <- factor(rep(1, times=length(pvalue)))
}
ann.settings<-vector("list", length(levels(grp)))
ann.settings[[1]]<-list(pch=pch, col=col, cex=cex, fill=col, label=label.
,→default)
if (length(ann.settings)>1) {
lcols<-trellis.par.get("superpose.symbol")$col
lfills<-trellis.par.get("superpose.symbol")$fill
for(i in 2:length(levels(grp))) {
ann.settings[[i]]<-list(pch=pch,
col=lcols[(i-2) %% length(lcols) +1 ],
fill=lfills[(i-2) %% length(lfills) +1 ],
cex=cex, label=label.default);
ann.settings[[i]]$label$show <- T
}
names(ann.settings)<-levels(grp)
}
for(i in 1:length(ann.settings)) {
if (i>1) {ann.settings[[i]] <- modifyList(ann.settings[[i]], ann.default)
ann.settings[[i]]$label <- modifyList(ann.settings[[i]]$label,
parse.label(ann.settings[[i]]$label, ␣
,→levels(grp)[i]))
}
if(is.list(annotate) && length(annotate)>1) {
user.cols <- 2:length(annotate)
ann.cols <- c()
if(!is.null(names(annotate[-1])) && all(names(annotate[-1])!="")) {
ann.cols<-match(names(annotate)[-1], names(ann.settings))
```

```
} else {
ann.cols<-user.cols-1
}
for(i in seq_along(user.cols)) {
if(!is.null(annotate[[user.cols[i]]]$label)) {
annotate[[user.cols[i]]]$label<-parse.label(annotate[[user.
,→cols[i]]]$label,
levels(grp)[ann.cols[i]])
}
ann.settings[[ann.cols[i]]]<-modifyList(ann.settings[[ann.cols[i]]],
annotate[[user.cols[i]]])
}
}
rm(annotate)
#reduce number of points plotted
if(should.thin) {
thinned <- unique(data.frame(
logp=round(-log10(pvalue),thin.logp.places),
pos=round(genpos,thin.pos.places),
chr=chr,
grp=grp)
)
logp <- thinned$logp
genpos <- thinned$pos
chr <- thinned$chr
grp <- thinned$grp
rm(thinned)

} else {
logp <- -log10(pvalue)
}
rm(pos, pvalue)
gc()
#custom axis to print chromosome names
axis.chr <- function(side,...) {
if(side=="bottom") {
panel.axis(side=side, outside=T,
at=((posmax+posmin)/2+posshift),
labels=levels(chr),
ticks=F, rot=0,
check.overlap=F
)
} else if (side=="top" || side=="right") {
panel.axis(side=side, draw.labels=F, ticks=F);
}
else {
axis.default(side=side,...);
}
}
#make sure the y-lim covers the range (plus a bit more to look nice)
prepanel.chr<-function(x,y,...) {
A<-list();
maxy<-ceiling(max(y, ifelse(!is.na(sig.level), -log10(sig.level), 0)))+.5;
A$ylim=c(0,maxy);
A;
}
xyplot(logp~genpos, chr=chr, groups=grp,
axis=axis.chr, ann.settings=ann.settings,
prepanel=prepanel.chr, scales=list(axs="i"),
panel=function(x, y, ..., getgenpos) {
if(!is.na(sig.level)) {
#add significance line (if requested)
panel.abline(h=-log10(sig.level), lty=2);
}
panel.superpose(x, y, ..., getgenpos=getgenpos);
if(!is.null(panel.extra)) {
panel.extra(x,y, getgenpos, ...)
}
```

```
},
panel.groups = function(x,y,..., subscripts, group.number) {
A<-list(...)
#allow for different annotation settings
gs <- ann.settings[[group.number]]
A$col.symbol <- gs$col[(as.numeric(chr[subscripts])-1) %%␣
,→length(gs$col) + 1]
A$cex <- gs$cex[(as.numeric(chr[subscripts])-1) %% length(gs$cex) +␣
,→1]
A$pch <- gs$pch[(as.numeric(chr[subscripts])-1) %% length(gs$pch) +␣
,→1]
A$fill <- gs$fill[(as.numeric(chr[subscripts])-1) %% length(gs$fill)␣
,→+ 1]
A$x <- x
A$y <- y
do.call("panel.xyplot", A)
#draw labels (if requested)
if(gs$label$show) {
gt<-gs$label
names(gt)[which(names(gt)=="text")]<-"labels"
gt$show<-NULL
if(is.character(gt$x) | is.character(gt$y)) {
peak = which.max(y)
center = mean(range(x))
if (is.character(gt$x)) {
if(gt$x=="peak") {gt$x<-x[peak]}
if(gt$x=="center") {gt$x<-center}

}
if (is.character(gt$y)) {
if(gt$y=="peak") {gt$y<-y[peak]}
}
}
if(is.list(gt$x)) {
gt$x<-A$getgenpos(gt$x[[1]],gt$x[[2]])
}
do.call("panel.text", gt)
}
},
xlab=xlab, ylab=ylab,
panel.extra=panel.extra, getgenpos=getGenPos, ...
);
}
```

```
%%R
library(tidyverse)
library(dplyr)
vignette('qqman')
ara11<-read.csv("~/lastversion_gwas_analysis/ara11_50.csv")
setwd("/home/s417377/trans_genes/AT2G18735/maf_filtered/")
Y <- list()
csvfiles<-list.files(pattern = "*.csv")
Y <- lapply(csvfiles, function(x) {
file <- read.csv(x)
return(file)
})
for (i in 1:length((csvfiles))){
Y[[i]]<-mutate(Y[[i]], SNP=paste(CHR, sep="- ", POS))
}
for (i in 1:length((csvfiles))){
colnames(Y[[i]])[4]<-"chr"
colnames(Y[[i]])[5]<-"pos"
colnames(Y[[i]])[6]<-"pvalue"
}
#"AT2G29710_AT2G24600"#make annotation factor
ann<-rep(1, length(Y[[1]]$pvalue))
ann[with(Y[[1]], chr==2 & pos>=12648573 & pos<= 12750343)]<-2
ann[with(Y[[1]], chr==2 & pos==9137974)]<-3
```

```
ann[with(Y[[1]], chr==2 & pos==9303977)]<-4
ann[with(Y[[1]], chr==2 & pos==17833766)]<-5
ann[with(Y[[1]], chr==4 & pos==840051)]<-6
ann<-factor(ann, levels=1:6, labels=c("","-",␣
,→"9137974","9303977","17833766","840051"))
#draw plot with annotation
manhattan.plot(Y[[1]]$chr, Y[[1]]$pos, Y[[1]]$pvalue, annotate=ann,␣
,→ylim=c(0,50))
#"AT2G24600_AT2G29710"#make annotation factor
ann<-rep(1, length(Y[[3]]$pvalue))
ann[with(Y[[3]], chr==2 & pos>=10402034 & pos<= 10504593)]<-2
ann[with(Y[[3]], chr==2 & pos==9137974)]<-3
ann[with(Y[[3]], chr==2 & pos==9303977)]<-4
ann[with(Y[[3]], chr==2 & pos==17833769)]<-5
ann[with(Y[[3]], chr==4 & pos==840051)]<-6
ann<-factor(ann, levels=1:6, labels=c("","-",␣
,→"9137974","9303977","17833769","840051"))
#draw plot with annotation
manhattan.plot(Y[[3]]$chr, Y[[3]]$pos, Y[[3]]$pvalue, annotate=ann,␣
,→ylim=c(0,50))
#"AT2G24600_AT4G00970.csv"#make annotation factor
ann<-rep(1, length(Y[[2]]$pvalue))
ann[with(Y[[2]], chr==4 & pos>=368327 & pos<= 471885)]<-2
ann[with(Y[[2]], chr==2 & pos==8122148)]<-3
ann[with(Y[[2]], chr==2 & pos==9345189)]<-4
ann[with(Y[[2]], chr==2 & pos==17833784)]<-5
ann[with(Y[[2]], chr==4 & pos==839000)]<-6
ann<-factor(ann, levels=1:6, labels=c("","-",␣
,→"8122148","9345189","17833784","839000"))
#draw plot with annotation
manhattan.plot(Y[[2]]$chr, Y[[2]]$pos, Y[[2]]$pvalue, annotate=ann,␣
,→ylim=c(0,50))
#"AT4G00970_AT2G24600.csv"#make annotation factor
ann<-rep(1, length(Y[[6]]$pvalue))
ann[with(Y[[6]], chr==2 & pos>=10402034 & pos<= 10504593)]<-2
ann[with(Y[[6]], chr==2 & pos==8122148)]<-3
ann[with(Y[[6]], chr==2 & pos==9345189)]<-4
ann[with(Y[[6]], chr==2 & pos==17833784)]<-5
ann[with(Y[[6]], chr==4 & pos==839000)]<-6
ann<-factor(ann, levels=1:6, labels=c("","-",␣
,→"8122148","9345189","17833784","839000"))
#draw plot with annotation
manhattan.plot(Y[[6]]$chr, Y[[6]]$pos, Y[[6]]$pvalue, annotate=ann,␣
,→ylim=c(0,50))
#"AT3G12020_AT3G19040.csv"
ann<-rep(1, length(Y[[4]]$pvalue))
ann[with(Y[[4]], chr==3 & pos>=6517021 & pos<= 6517021)]<-2
ann[with(Y[[4]], chr==2 & pos==8122723)]<-3
ann<-factor(ann, levels=1:3, labels=c("","-", "8122723"))
#draw plot with annotation
manhattan.plot(Y[[4]]$chr, Y[[4]]$pos, Y[[4]]$pvalue, annotate=ann,␣
,→ylim=c(0,50))
#"AT3G19040_AT3G12020.csv" #make annotation factor
ann<-rep(1, length(Y[[5]]$pvalue))
ann[with(Y[[5]], chr==3 & pos>=3775980 & pos<= 3885146)]<-2
ann[with(Y[[5]], chr==2 & pos==8122723)]<-3
ann<-factor(ann, levels=1:3, labels=c("","-", "8122723"))
#draw plot with annotation
manhattan.plot(Y[[5]]$chr, Y[[5]]$pos, Y[[5]]$pvalue, annotate=ann,␣
,→ylim=c(0,50))
```

List of Figures

List of Tables

Affirmation/ Eidesstattliche Erklärung

I hereby confirm that my master thesis entitled Genome-Wide Association Studies to Infer Signs of Selection in *Arabidopsis thaliana*
is the result of my own work. /
I did not receive any support from commercial consultants. /
I have given due reference to all sources and materials used in the thesis and have listed and specified
them. /
I confirm that this thesis has not yet been submitted as part of another examination process neither in
identical nor in similar form. /
I agree that the thesis can be checked for plagiarism also by using a software. /

Hiermit erkläre ich an Eides statt, dass ich die Masterarbeit mit dem Titel Selektion der Genome-Wide Association Studies to Infer Signs of Selection in *Arabidopsis thaliana*
eigenständig und eigenhändig angefertigt habe.
Ich habe keine Unterstützung kommerzieller Berater erhalten.
Ich habe alle in der Arbeit verwendeten Quellen und Materialien ordnungsgemäß zitiert, aufgelistet
und spezifiziert
Ich erkläre, dass die vorliegende Arbeit weder in gleicher noch in ähnlicher Form bereits in einem
anderen Prüfungsverfahren vorgelegen hat.
Ich bestätige, dass die Thesis auch mit Hilfe einer Software auf Plagiat untersucht werden kann

Würzburg, June 21, 2022 Leyla Sırkıntı

References

1. Uffelmann, E., et al., *Genome-wide association studies.* Nature Reviews Methods Primers, 2021. **1**(1): p. 59.
2. Torella, A., et al., *The position of nonsense mutations can predict the phenotype severity: A survey on the DMD gene.* PloS one, 2020. **15**(8): p. e0237803-e0237803.
3. Lee, T. and I. Lee, *Genome-Wide Association Studies in Arabidopsis thaliana: Statistical Analysis and Network-Based Augmentation of Signals.* Methods in molecular biology (Clifton, N.J.), 2021. **2200**: p. 187-210.
4. Alqudah, A.M., et al., *Genome-wide and SNP network analyses reveal genetic control of spikelet sterility and yield-related traits in wheat.* Scientific reports, 2020. **10**(1): p. 2098-2098.
5. Nogales, A. and M. L DeDiego, *Host Single Nucleotide Polymorphisms Modulating Influenza A Virus Disease in Humans.* Pathogens (Basel, Switzerland), 2019. **8**(4): p. 168.
6. Tak, Y.G. and P.J. Farnham, *Making sense of GWAS: using epigenomics and genome engineering to understand the functional relevance of SNPs in non-coding regions of the human genome.* Epigenetics & Chromatin, 2015. **8**(1): p. 57.
7. Bush, W.S. and J.H. Moore, *Chapter 11: Genome-wide association studies.* PLoS computational biology, 2012. **8**(12): p. e1002822-e1002822.
8. Chandra, A., et al., *Genome-wide association studies: applications and insights gained in Ophthalmology.* Eye (London, England), 2014. **28**(9): p. 1066-1079.
9. Manolio, T.A. and F.S. Collins, *The HapMap and genome-wide association studies in diagnosis and therapy.* Annual review of medicine, 2009. **60**: p. 443-456.
10. Tam, V., et al., *Benefits and limitations of genome-wide association studies.* Nat Rev Genet, 2019. **20**(8): p. 467-484.
11. Visscher, P.M., et al., *Five years of GWAS discovery.* Am J Hum Genet, 2012. **90**(1): p. 7-24.
12. Tsai, H.-Y., et al., *Genomic prediction and GWAS of yield, quality and disease-related traits in spring barley and winter wheat.* Scientific Reports, 2020. **10**(1): p. 3347.
13. Challa, S. and N.R.R. Neelapu, *Chapter 9 - Genome-Wide Association Studies (GWAS) for Abiotic Stress Tolerance in Plants*, in *Biochemical, Physiological and*

Molecular Avenues for Combating Abiotic Stress Tolerance in Plants, S.H. Wani, Editor. 2018, Academic Press. p. 135-150.

14. Manik, S.M.N., et al., *Genome-Wide Association Study Reveals Marker Trait Associations (MTA) for Waterlogging-Triggered Adventitious Roots and Aerenchyma Formation in Barley.* International journal of molecular sciences, 2022. **23**(6): p. 3341.
15. Marees, A.T., et al., *A tutorial on conducting genome-wide association studies: Quality control and statistical analysis.* International journal of methods in psychiatric research, 2018. **27**(2): p. e1608-e1608.
16. Park, J.-H., et al., *Distribution of allele frequencies and effect sizes and their interrelationships for common genetic susceptibility variants.* Proceedings of the National Academy of Sciences, 2011. **108**(44): p. 18026-18031.
17. Chan, Y., et al., *An excess of risk-increasing low-frequency variants can be a signal of polygenic inheritance in complex diseases.* Am J Hum Genet, 2014. **94**(3): p. 437-52.
18. Park, J.-H., et al., *Distribution of allele frequencies and effect sizes and their interrelationships for common genetic susceptibility variants.* Proceedings of the National Academy of Sciences of the United States of America, 2011. **108**(44): p. 18026-18031.
19. Kido, T., et al., *Are minor alleles more likely to be risk alleles?* BMC Medical Genomics, 2018. **11**(1): p. 3.
20. Zhu, Z., et al., *Enrichment of Minor Alleles of Common SNPs and Improved Risk Prediction for Parkinson's Disease.* PloS one, 2015. **10**(7): p. e0133421-e0133421.
21. Coleman, J.R., et al., *Quality control, imputation and analysis of genome-wide genotyping data from the Illumina HumanCoreExome microarray.* Brief Funct Genomics, 2016. **15**(4): p. 298-304.
22. Ramzan, F., et al., *Combining Random Forests and a Signal Detection Method Leads to the Robust Detection of Genotype-Phenotype Associations.* Genes, 2020. **11**(8): p. 892.
23. Hopkins, Z.H., C. Moreno, and A.M. Secrest, *Confidence intervals in dermatology.* British Journal of Dermatology, 2019 **180**(4): p. e115-e115.
24. Pearson, T.A. and T.A. Manolio, *How to interpret a genome-wide association study.* Jama, 2008. **299**(11): p. 1335-44.
25. Fadista, J., et al., *The (in)famous GWAS P-value threshold revisited and updated for*

low-frequency variants. Eur J Hum Genet, 2016. **24**(8): p. 1202-5.

26. John, M., et al., *Efficient Permutation-based Genome-wide Association Studies for Normal and Skewed Phenotypic Distributions.* bioRxiv, 2022: p. 2022.04.05.487185.
27. Hopkins, Z.H., C. Moreno, and A.M. Secrest, *Confidence intervals in dermatology.* British Journal of Dermatology, 2019 **180**(4): p. e115-e115.
28. *Analysis of the genome sequence of the flowering plant Arabidopsis thaliana.* Nature, 2000. **408**(6814): p. 796-815.
29. Weigel, D. and R. Mott, *The 1001 genomes project for Arabidopsis thaliana.* Genome biology, 2009. **10**(5): p. 107-107.
30. *1,135 Genomes Reveal the Global Pattern of Polymorphism in Arabidopsis thaliana.* Cell, 2016. **166**(2): p. 481-491.
31. A. Korte and, J.A.F., *GWAS.* 2020.
32. Geller, F., et al., *Genome-wide association study identifies four loci associated with eruption of permanent teeth.* PLoS genetics, 2011. **7**(9): p. e1002275-e1002275.
33. Hwang, I.S., et al., *The pepper cysteine/histidine-rich DC1 domain protein CaDC1 binds both RNA and DNA and is required for plant cell death and defense response.* New Phytol, 2014. **201**(2): p. 518-530.
34. van den Berg, S., et al., *Significance testing and genomic inflation factor using high-density genotypes or whole-genome sequence data.* J Anim Breed Genet, 2019. **136**(6): p. 418-429.
35. Monroe, J.G., et al., *Mutation bias reflects natural selection in Arabidopsis thaliana.* Nature, 2022. **602**(7895): p. 101-105.