AgriNetFusion: Guided hierarchical attention fuses multimodal data to unlock drivers of crop performance

by

Courtney Paige Taylor

(Under the Direction of Abhyuday Mandal and Aditya Mishra)

Abstract

Accurate prediction of crop phenotypes is critical for breeding and agricultural sustainability but challenged by complex interactions between genetics, environment, and management (G×E×M). Existing models often struggle to effectively integrate diverse data modalities or incorporate domain knowledge. AgriNetFusion is a novel deep learning framework that employs Hierarchical Multimodal Attention (HMMA) explicitly guided by phenotype-specific domain knowledge. AgriNetFusion integrates high-dimensional genotype data (using PCA or a novel attentive autoencoder) with dynamic environmental and static field data from large-scale corn trials. Rigorous benchmarking across 11 phenotypes shows AgriNetFusion significantly outperforms standard machine learning and simpler deep learning baselines. Ablation studies confirm the crucial contributions of hierarchy, guided attention, and genotype representation. Interpretability analyses reveal learned modality importances consistent with agronomic priors while highlighting novel interactions. AgriNetFusion provides a powerful and interpretable approach for modeling complex biological systems, advancing predictive capabilities in agriculture and offering a framework adaptable to other multimodal scientific domains.

Index words: Deep learning, Machine Learning, Agriculture, Agronomy

AgriNetFusion: Guided hierarchical attention fuses multimodal data to unlock drivers of crop performance

by

Courtney Paige Taylor

B.S., University of Georgia, 2024

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Master of Science

Athens, Georgia

2025

AgriNetFusion: Guided hierarchical attention fuses multimodal data to unlock drivers of crop performance

by

Courtney Paige Taylor

Major Professor: Abhyuday Mandal
Aditya Mishra

Committee: Shuyang Bai

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
August 2025

Contents

List of Figures

Chapter 1

Introduction

Recent advances in data science and artificial intelligence allow for the analysis of complex patterns and processes across many disciplines, including agriculture. Scientists have leveraged these architectures to predict plant growth and adjust growing processes to optimize yield. However, the use of standard linear models, machine learning, and deep learning techniques come with limitations in capturing all important trends. While linear models allow for relatively easily interpretability, they fail in capturing complex, nonlinear relationships between predictors, missing important components of intricate growing processes (Ansarifar et al., 2021). Deep learning models display optimal performance, but often lack easy interpretability of variable relationships (Nieradzik et al., 2024).

Corn is one of the most widely used and produced crops in the United States, serving as a staple in human diets, livestock feed, and fuel production (U.S. Department of Agriculture, Economic Research Service, 2025). However, this high demand for corn occurs alongside rising unpredictability in environmental factors and major weather events. The genetic and environmental conditions for optimal corn growth and yield must be found to ensure that production is increased to meet this high demand (Hatfield et al., 2018). By identifying robust corn genotypes under multiple environmental conditions, we can ensure that the effects of climate change on corn production are minimized and improve food security.

To analyze the optimal growth of corn, multiple phenotypes such as plant height, flowering time, and grain yield must be considered. Additionally, strength against major weather events provided by genetic

robustness must be considered. These outcomes result from complex interactions between the genetic makeup of a plant, nutrients provided in the soil, treatments added to the field, and weather factors such as heat and water. Thus, identifying the importance of a single environmental factor or gene cannot be the solution to optimal growth nationwide, and sufficient analysis must integrate data across temporal and spatial scales to consider these interactions. The Genomes 2 Fields Initiative publishes annual corn growth and environmental data primarily across multiple US states in order for these factors to be analyzed across varying climate regions. The data provided by the Genomes 2 Field initiative includes annual phenotype measurements across multiple fields and states, weather data for each field location reported multiple times daily, all treatments applied to each field in a year, annual soil nutrient composition, and genetic data for each hybrid or inbred planted. This diverse, high dimensional collection of data at different scales poses challenges for modeling and determining which factors and interactions are most important for corn production. Previous works consider the interactions between these modalities to determine the genetic and environmental conditions for best growth (Sharma et al., 2022). However, researchers have not considered the implicit hierarchy that lies within all of these factors; diverse modalities can have similar influences on corn outcomes and can be grouped together to reveal larger effects.

Our work addresses these limitations by proposing AgriNetFusion, a deep learning framework based on the hypothesis that a hierarchical architecture mirroring natural data relationships, combined with attention mechanisms explicitly guided by domain knowledge, can enhance both predictive accuracy and model interpretability for complex agricultural phenotypes. AgriNetFusion utilizes Hierarchical Multimodal Attention (HMMA), incorporates phenotype-specific guidance based on known agronomic influences, employs Group Lasso for modality selection, and explores advanced genotype representations derived from an attentive autoencoder alongside traditional PCA.

In this study, we (1) present the AgriNetFusion framework architecture, (2) demonstrate its superior predictive performance across 11 key corn phenotypes using large-scale field trial data compared to established baseline methods, (3) validate the contribution of its core components through systematic ablation

studies, and (4) showcase its interpretability by analyzing learned data modality features and attention patterns in the context of existing domain knowledge.

Chapter 2

Data

The Genomes to Fields (G2F) initiative seeks to advance the understanding of how genetic variation in maize interacts with diverse environmental factors to influence plant performance(www.genomes2fields.org). The initiative publishes publicly available datasets of genomic sequencing information, phenotypic, soil, field, agronomic, and weather data which encompasses hundreds of corn hybrids across farms primarily across North America. The collected information spans multiple years, contributing to a robust dataset that allows researchers to track changes over time and explore relationships between genetic markers, environmental factors, and phenotypic traits. The challenge of accurately predicting the factors that influence corn phenotypes lies in the complexity of the involved data. Genetic variation encompasses millions of single nucleotide polymorphisms (SNPs) across thousands of genetic inbred lines, which each potentially influence different traits. Environmental data can be recorded frequently and contain high variability (e.g., weather data) and can depend on farmers' individual practices (e.g., field treatment data). Thus, the number of interacting factors grows exponentially as new variables are considered, and phenotypic traits are often influenced by multiple genes and environmental conditions, leading to complex interactions. Integration of these differing dimensions to uncover existing patterns requires a diverse array of sophisticated approaches, underscoring the importance of broad collaboration. Thus, this public data is a valuable resource which allows researchers to provide their own input into how phenotype outcomes

are influenced by a wide range of genetic and growing factors. This knowledge has the potential to inform the development of maize and maximize production.

2.1 Data Collection

The annual Genomes to Fields data collection includes over 180,000 plots with over 2,500 hybrids and 162 environments in North America since 2014. Experiment locations are shown in Figure 2.1.

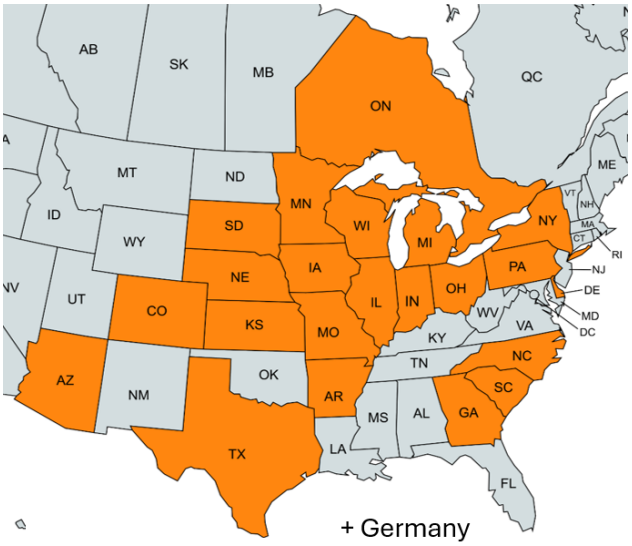


Figure 2.1: Experiment Locations

Phenotype, genotype, weather, soil, field, and agronomic data are recorded each year.

Phenotype Data

The G2F initiative highlights a set of core phenotypic measurements that are crucial for evaluating maize performance. These variables cover a broad range of characteristics, including morphology, phenology, lodging, and yield variables. The phenotypic data includes 11 core traits, each of which is recorded once per year. The 11 core traits can be broken down into four categories:

- Morphology: Plant Height, Ear Height, Stand Count

- Phenology: Silking, Anthesis
- Lodging: Root Lodging, Stalk Lodging
- Yield/Moisture: Grain Yield, Grain Moisture, Test Weight, Plot Weight

The location of these phenotypes on a corn plant are shown in Figure 2.2.

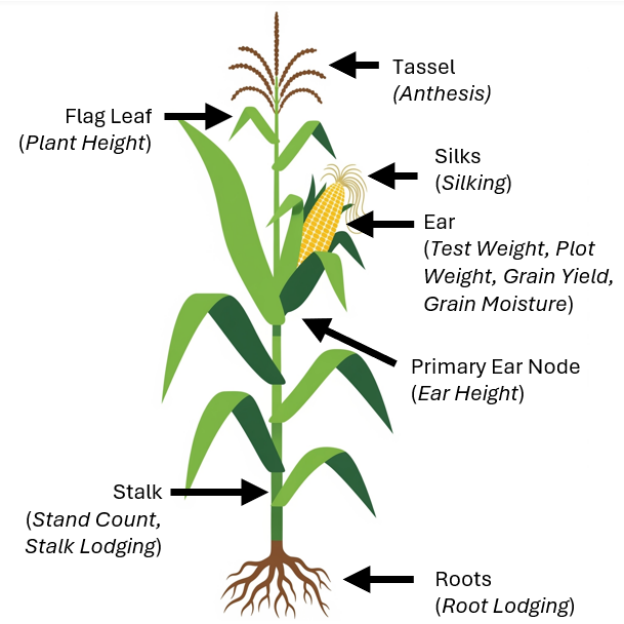


Figure 2.2: Phenotype Locations on Plant

Additional metadata, such as the field location, planting and harvest dates, and comments on specific experimental conditions, are also recorded for each observation. Every plot within a field location is planted with a single type of genetic corn hybrid or inbred (given by its Pedigree name), and these phenotype traits are reported at the plot level. Both hybrids and inbreds are considered in this study. Inbreds contain one inbred line name. Hybrids contain two inbred names, indicating the influence and possible interaction of traits from two inbred lines.

Genotype Data

The genotypic data within the G2F dataset is based on sequencing technologies that allow for the identification of genetic variation in maize inbreds. The dataset covers genetic information collected from

2014 to 2023 and aligns sequencing reads against a comprehensive database of maize genome assemblies. This database, known as the Practical Haplotype Graph (PHG), contains genetic information from 86 genome assemblies. The PHG is used to compute genotypes based on identified haplotypes, which are then aligned with the genotypic data of the 2,193 observed inbreds. The reported data is provided in a Variant Call Format (VCF), where each column corresponds to an inbred line (e.g., B73), each row represents a single nucleotide polymorphism (SNP) identified by its chromosome and position (e.g., s1_120931), and each cell value indicates the allelic state relative to the reference genome. The value 0/0 indicates that both alleles at that SNP are the same as the reference genome (homozygous reference). The values 0/1 and 1/0 indicate that one allele differs from the reference genome at that location (heterozygous), and 1/1 indicates that both alleles differ from the reference (homozygous alternate). These altered genetics can lead to differences in phenotypes depending on the function of the affected gene. The final dataset contains over 437,000 genotypic positions, offering a detailed view of genetic variation across multiple maize lines.

Weather Data

Weather data is a crucial component in understanding how environmental factors affect maize growth. The G2F initiative collects weather data from a Spectrum Watchdog 2700 Weather Station run by the National Weather Service (NWS) near each experimental site. These stations record measurements approximately every 30 minutes during the growing season, but the frequency of data collection varies slightly depending on the location and time of year, with some stations consistently recording data every 30 minutes and others containing gaps or less frequent updates.

The weather variables collected are Temperature, Rainfall, Dew Point, Relative Humidity, Wind Speed, Wind Direction, Wind Gust, Soil Temperature, Soil Moisture, Soil EC, Solar Radiation, UV Light, Photoperiod, and CO_2. The field location, station ID, date, time, and extra comments are also recorded for each observation.

Soil Data

Soil samples are collected annually from each G2F field location and submitted for nutrient and texture analysis. The soil data includes key variables such as pH, organic matter content, and concentrations of essential nutrients such as nitrogen, potassium, and calcium. The field location, collaborator name, dates received and reported at the lab, and extra comments were also recorded for each soil observation.

Field Data

Field information is collected annually from each G2F field location participating in a given year's experiment. The cooperator of each location provides plot layout information, pre and post planting tillage practices, and previous crops planted in order for the organization and practices of the fields to be used in predicting crop phenotype and success.

Agronomic Data

Agronomic information is collected for all treatments and applications applied to each location in a year, in order for researchers to examine the effects of supplemental soil and field treatments on the performance of corn growth. In addition to field location, the important agronomic variables collected in the experiment are: Application or Treatment, Product or Nutrient Applied, Date of Application, Quantity per Acre, and Application Unit.

2.2 Data Limitations

Considering the experiment's evolution across time, collection of data from many sources and collaborators, information within open ended comment fields, and high dimensional datasets, the data cleaning process required detailed examination and editing in order to aggregate data across years and collection types.

2.2.1 Field Location Presence

The Genomes to Fields (G2F) initiative has grown since its start in 2014, adding locations in North America and abroad over time. Nine locations are present every year, while the remaining 30 locations that appear in the data are added at later dates, recorded for several years then removed, or both.

2.2.2 Variable Presence

Due to the annual evolution of the experiment, the organization and presence of measurements differ across years. Additionally, slight formatting changes across years and datasets frequently occurred. For example, Field-Location in the phenotype datasets, Experiment in the 2014-2015 field datasets, and Experiment_Code in the 2016-2022 field datasets represent the same information, which is the representation of a field location given by its state or country abbreviation, experiment, and a single digit representing its replicate number within the location and experiment combination.

Phenotype data was published every year from 2014 through 2022. Hybrid and inbred experiments conducted in 2014 and 2015 were reported as separate datasets, and were combined into one dataset starting in 2016, with a new variable indicating the experiment of each observation. Thirteen measurements of interest were recorded each year, two measurements were recorded for seven years, one measurement was recorded for three years, and the remaining measurements were only recorded in 2014.

Weather data is given every year from 2014 through 2022. Eight measurements of interest are present every year, four measurements are present from 2015 through 2022, and the remaining measurements are present for six or fewer years.

Field data has been reported every year from 2014 through 2022, with a large number of attributes in 2014 and 2015. Many of these variables were moved to the agronomic and soil data once these new datasets were created, indicating identical types of information present across different dataset types depending on the year.

The soil dataset was created in 2015, with plow depth, pH, buffer pH, organic matter level, phosphorus levels, and potassium levels as key measurement variables. Seventeen measurement variables were added in 2016 and reported every year after, and five variables were recorded only in 2016, giving an inconsistent collection of measurements for the first several years of the experiment.

The agronomic dataset was first published in 2016, and each variable has been recorded consistently.

2.2.3 Weather Data Time and Location Differences

The weather data was reported from a Spectrum Watchdog weather station near each field location for the duration of the field's growing season; thus, weather measurements started and ended at different times based on the field location and year. Weather measurements were ideally recorded every thirty minutes during the growing season, but were reported in different intervals and/or had occasional gaps for many locations across all experiment years. Additionally, some field locations collected weather information from the same NWS station, resulting in aggregated field location names.

2.2.4 Open Ended Treatment Sections

Treatment and product variables in the agronomic and early field data allowed for open ended responses for collaborators to describe the treatment applied to their plots, which increased the number of unique non numeric responses to be reviewed before analysis.

2.2.5 Missingness

Levels of missingness were found across all datasets in at least one year for both numeric and non numeric fields.

2.3 Data Pre-Processing

2.3.1 Inconsistent Column Names and Data Values

Equivalent columns in each dataset type were reassigned to common names to ensure matching across years.

Several field locations deviated from the usual reporting style of state abbreviation, experiment indicator, and replicate within location and experiment combination, such as GAH1. Several field locations lacked an experiment indicator, resulting in a field location of the format "IA(?)2". In the 2015 soil data, IA(?)2 and IA(?)3 were matched to IAH2 and IAH3 based on the cooperator name in the soil dataset. The MN(?)1 locations were matched to MNH1 and MNI1 based on the Lab Sample ID, since the cooperator name was identical. In the 2017 soil data, NEH3 (IRRIGATED) and NEH4 (NON IRRIGATED) were matched to NEH3 and NEH4 to map directly across years and other datasets within 2017. The experiment indicators within the 2014 phenotype data were “H” and “inb” for hybrid and inbred, rather than “H” and “I”, so all inbred field locations were edited to match the formatting found in all other datasets.

2.3.2 Weather Data Aggregation and Location Refinement

Due to inconsistent weather reporting dates, times, and intervals across field locations and years, along with the high dimensional data, weather variables were aggregated by week. A common date key was assigned based on the day, month, and year given in the raw data. For each location, the average, standard deviation, minimum, and maximum value for all weather variables except Rainfall. The cumulative sum of rain was found for each week in the year, since the total amount of rain across the growing season is a more useful point for crop performance than the average rain per week.

Since one weather station often provided data for multiple locations, not all field locations in a given year were uniquely reported in the weather data. For example, weather information for all Texas locations in 2022 were represented by observations with the “TXH1_TXH2_TXH3” field location. All weather

observations with these field locations were copied once if the weather station covered two locations and twice if the weather station covered three locations, and unique field location names were assigned to each copy of observations. For example, the single set of "TXH1_TXH2_TXH3" observations was transformed to three sets of identical observations labeled "TXH1", "TXH2", or "TXH3".

2.3.3 Variable and Dataset Formatting

The dates of anthesis and silking for crops were provided in the phenotype dataset in the format of [MM/DD/YYYY]. These date formats were converted to days, indicating the difference between the date the plot was planted at the date of anthesis or silking, since this measurement is more relevant for analysis based on different growing seasons and planting dates across locations.

Before the creation of the agronomic datasets in 2016, several of agronomy-related variables were reported in the field dataset. Thus, Irrigation, Fertilizer, and Herbicide variables in the 2014 and 2015 field datasets were organized into Application or Treatment and Product or Nutrient variables in new 2014 and 2015 agronomic datasets.

Wind speed (S) and wind direction (θ) variables were converted to U and V wind components to represent the eastward and northward wind, respectively, where a positive U wind comes from the west and blows east, and a positive V wind comes from the south and blows north. These new variables were calculated using the formulas $u = -S * \sin(\theta_{\text{rad}})$ and $v = -S * \cos(\theta_{\text{rad}})$.

2.3.4 Master Tables

Key columns were identified to connect all datasets of differing dimensions, measurements, and granularity. Year was used to connect datasets of the same type to create master tables of all phenotype data, all soil data, all field data, all agronomic data, and separate tables for each type of weather data.

2.3.5 Variable Selection and Dataset Aggregation

Variables within the phenotype, weather, soil, field, and agronomic data were dropped if they were recorded for fewer than 5 years. Only variables recorded in over half of the present years were considered since we seek to examine the effects of environmental factors on corn phenotypes across a substantial period of time. Phenotype, weather, soil, field, and agronomic datasets were then each combined across years. Phenotype observations with complete missingness across all measurements were dropped.

2.3.6 Missingness

Weather measurements were limited to weeks with an average missingness of 40% or less. Soil EC, CO_2, and UV light were only recorded for one location each year, so these datasets were dropped entirely. K nearest neighbors imputation was used to resolve the remaining missingness for numerical variables in the weather and soil datasets. This imputation was performed for soil variables with less than 40% missingness and all remaining weather variables. Data across all years was used for imputation.

2.3.7 Genotype Processing

The genotype data was originally recorded in a VCF format, where each column represents an inbred line, each row represents a SNP containing the chromosome number and position, and each value represents the alleles which differ or remain similar to the reference genome (ex: 0/0). These values were encoded to represent 0, 1 or 2 alleles that differ from the reference genome; homozygous references (0/0) were encoded as 0, heterozygous (0/1 or 1/0) were encoded as 1, and homozygous alternates (1/1) were encoded as (2). Additionally, each SNP was compared to a reference genome to determine its exact genomic position, resulting in identifiers of the format V1, V2, etc. that represent the position orders of the SNPs. The processed genotype data contains these encoded genomic positions as the column names, the inbred lines as the row names, and the encoded allele differences as the data values.

2.3.8 Resulting Data

Phenotype data contains 162,724 total observations across 2014-2022. Soil data contains 222 total observations across 2015-2022. Field data contains 255 total observations across 2015-2022. Agronomic data contains 1348 total observations across 2014-2022. Dew Point, Humidity, Solar Radiation, Rain, Wind Speed, Wind Direction, Wind Gust, and Temperature contain 272 observations from 2014-2022. Soil Temperature and Soil Moisture contain 230 observations from 2015-2022. Photoperiod contains 165 observations from 2014-2018. The genotype data contains 2,193 inbred line observations with 16,384 SNP locations. Weather observations are given from the 20th week of the year through the 41st week of the year, except for Rainfall, which contains observations from the 19th week through the 41st week of the year due to lower levels of missingness in the original data.

2.4 Data Matching

2.4.1 Categorization of Open Ended Treatment Variables

The Application or Treatment and Product or Nutrient variables in the Agronomic data allow for open-ended responses regarding the type of treatment and product name applied to the plots, resulting in a large number of unique values. These variables were aggregated into categorical variables derived from the most common treatment and products applied. Application or Treatment values were categorized as Herbicide, Fertilizer, Insecticide, Irrigation, or Other. Product or Nutrient values were categorized as Water, Atrazine, Overhead Sprinkler Irrigation, Acuron, Callisto, or Other. Each observation was assigned to these categories based on key words that appeared in the Application and Product fields.

The Treatment, Previous Crop, Pre-Plant Tillage Methods, and In-Season Tillage Methods variables in the field data allowed for open ended responses as well. Treatment was classified into Standard and Other, since Standard had 173 occurrences, while the other 14 treatments each had fewer than 10 occurrences. Previous Crop was categorized into Soybean (containing soybean and double crop soybeans),

Corn (containing corn), Wheat (containing wheat, winter wheat, and 2019/20 wheat), and Other, since the 7 other crops each had 5 or fewer occurrences. Pre-Season Tillage Methods was categorized into Field Cultivator and Other, since all other types of methods each had fewer than 5 observations each. In season Tillage Methods was categorized into Applied and None, since each type of method each had fewer than 10 observations.

2.4.2 Data Refinement for Model

If multiple soil observations contained the same year and field location, their measurements were averaged to ensure there was one soil observation per location per year.

2.4.3 Matching

The soil, field, agronomic, and weather datasets are linked to the target phenotype dataset through the Year and Field Location variables. Genotype information is linked to the phenotype dataset through the Pedigree, or genetic plant name. For hybrid plants, Pedigree was matched to both respective inbred lines. For inbreds, Pedigree was matched twice to its one respective inbred line, ensuring data dimensions were consistent across both Pedigree types. A final dataset was created with each phenotype observation and its corresponding soil, field, agronomic, genotype, and weekly weather observations.

The complete data processing pipeline is represented in Figure 2.3.

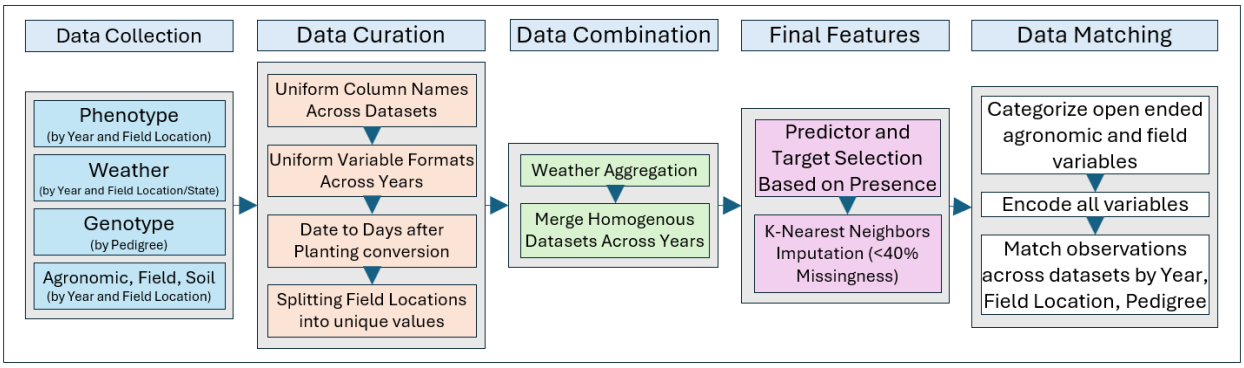


Figure 2.3: Data Pre-Processing Pipeline

This careful consideration of data highlights a large challenge in this phenotype prediction problem. The unique datasets each required thorough consideration and selection of important variables in order for harmonization across all data types, which is required to map corn phenotypes to their respective influential environmental, genetic, and field treatment variables. The need for this extensive preparation underscores the complex nature of agricultural output prediction and reflects the challenges faced in modeling these interactions.

Chapter 3

Modeling

3.1 Genotype Representations

After data preparation, the genotype file contained information for 16,384 SNP locations for each inbred line. Since hybrid plants contain two inbred lines (or "parents"), and inbred plants are treated as such for dimension consistency in this model (with one parent represented twice), over 32,000 genetic attributes must be considered for each observation. This high dimensionality can result in increased computational demands and difficulty in extract meaningful biological patterns from the data. Thus, dimensionality reduction is often applied to SNP data in the aim of using a lower number of important traits in target prediction.

A convolutional autoencoder for genotypic dimensionality reduction was proposed by Ausmees and Nettelblad, 2022, successfully extracting important information into a latent representation of the original data. A similar approach is employed for dimensionality reduction for AgriNetFusion, with the addition of data blocking. Blocking the genotype positions into smaller subsets reduces computational cost but preserves the order of sequential positions and local context (Jo et al., 2022).

The full autoencoder architecture is represented in Figure 3.1.

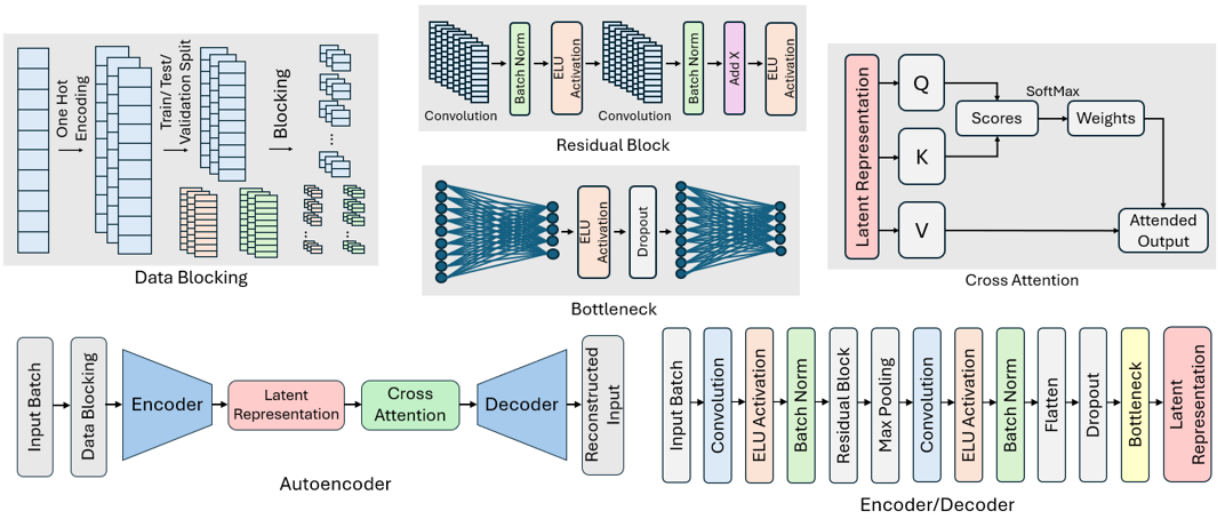


Figure 3.1: Autoencoder Model

The data blocking pipeline was applied to the genotype data in preparation for the autoencoder. Categorical genotype data, with values of 0, 1, or 2 representing the number of allele discrepancies compared to the reference genome, were one-hot encoded into indicator vectors for each of these three unique states, resulting in a 3 dimensional array of the genotype data with the dimensions (number of samples, number of SNPs, number of allele states). Then, 80% of the samples were randomly allocated as training data, 10% of the samples were allocated as testing data, and 10% of the samples were allocated as validation data. For efficient model training, each sample was sectioned into blocks of 50 SNPs. The data was padded with 0s along the second dimension to ensure equal sized blocks. This blocking allows for local patterns to be learned in the model.

An encoder then is applied to each SNP block, consisting of convolutional layers, residual blocks, and non-linear activation functions. The residual block contains two stacked convolutional layers, each followed by batch normalization and ELU activation. This block passes information from one layer to later stages of the network, helping stabilize the training process (He et al., 2015). Batch normalization contributes to gradient and training stablization, and max-pooling layers downsample the data to help

compress it to a latent represention of important features. The bottleneck block performs final feature compression and contains a dropout layer, which randomly zeroes elements with probability 0.10.

Cross attention is applied to the encoder output, or latent representation, to capture dependencies between different input blocks. The latent representation is passed through three separate linear layers to form query, key, and value tensors. Batch matrix multiplication is applied to query and key tensors and processed through the softmax function to obtain attention weights. These weights represent how much each block should attend to all other blocks in the latent representation, enabling cross-block information sharing. The attention weights are then used to compute a weighted sum of value tensors, giving a final attended latent representation of the genotype data. (Vaswani et al., 2023)

A decoder of the genotype data is applied to the attended latent representation, which directly mirrors the genotype encoder and reconstructs the genotype blocks based on learned features. The reconstructed blocks are then concatenated to match original sample formats containing all genotype information for a single inbred line.

Learning rate and weight decay are selected in a 3-stage hyperparameter tuning process, with 2 epochs in the first stage, 15 epochs in the second stage, and 200 epochs in the third stage.

A genotype representation using Principal Component Analysis (PCA) to extract important features from the data is also considered. The top 200 principal components were extracted, which capture approximately 70% of the variance in the original genotype data.

3.2 AgriNetFusion Motivation

Corn phenotypes are a result of many complex interactions between modalities, each capturing information at different scales and frequencies. While each variable may independently influence phenotypes, many share similar roles in plant growth, and this large collection of modalities contain underlying traits and interdependencies amongst each other. For example, rainfall and soil moisture both contain information about the amount of water provided to a plant. The consideration of these similar modalities in a deep learning model can reveal larger patterns about corn growth. AgriNetFusion is designed to explicitly

model this hierarchy, with modalities sectioned into 7 larger modality groups to promote information sharing between similar variables. These modality groups are displayed in Figure 3.2.

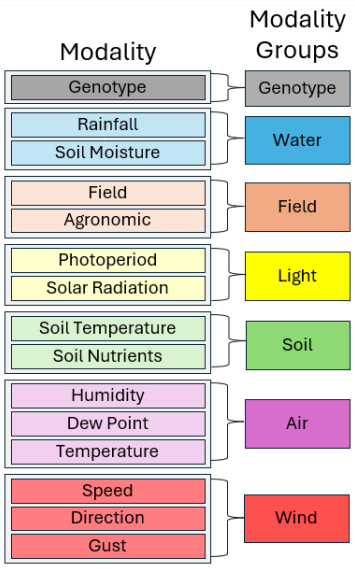


Figure 3.2: Modality Groups

This architecture promotes the consideration of both nuanced, smaller details and broad relationships, leading to important interaction discoveries and accurate phenotype predictions.

3.3 AgriNetFusion Model

The AgriNetFusion model predicts corn phenotypes using individual and shared representations of genotype, environmental, and treatment variables. Figure 3.3 shows the flow of information throughout the model. Intra modality information is captured through individual submodality representations and a joint modality representation. Inter modality information is captured using each joint modality representation to create a single representation that combines information across all modalities. This shared representation is used along with individual submodality representations to predict phenotypes.

Figure 3.3 displays a full representation of AgriNetFusion's architecture. In stage 1 of the model, the genotype autoencoder is used to obtain reconstructed representations of 2,193 corn inbred lines. These representations are then processed with a multilayer perceptron (MLP), and each observations' submodal-

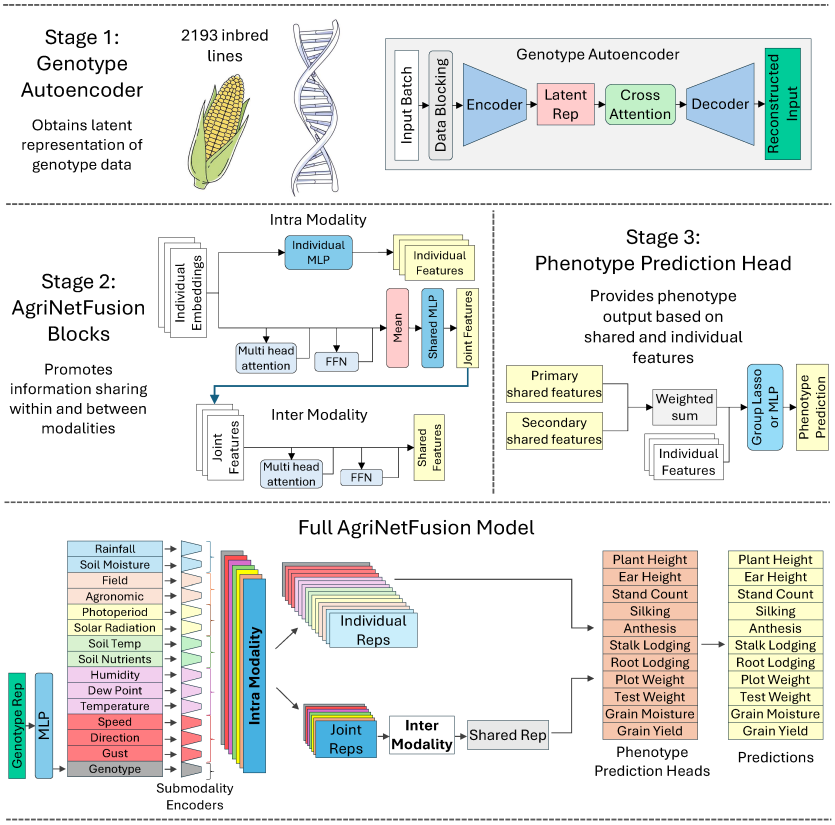


Figure 3.3: AgriNetFusion Architecture

ities are matched through either the Pedigree key or Year and Field Location keys. Then, simple encoders with two linear layers are applied to all submodalities to obtain encoded representations.

In stage 2 of the model, the submodalities are processed in the Intra Modality and Inter Modality blocks to promote information sharing within and between modalities. The Intra Modality block processes the individual submodality embeddings within each group to obtain individual and joint features. An individual MLP is applied to these embeddings to obtain individual features for each submodality. To obtain joint features for each modality group, a shared MLP is applied to the embeddings, along with an optional multi head attention and Feed Forward Network (FFN). The multihead attention module is used to capture important patterns between submodalities in the same group. These joint features are fed into the Inter Modality block, which calculates a shared representation for all modalities. This block

contains optional multihead attention, which promotes information sharing between modality groups, and FFN components to obtain a single shared feature representation for all modalities combined.

In stage 3 of the model, individual submodality representations and the single shared representation are used to predict phenotype output. A weighted sum of shared features are concatenated with individual features and passed to an optional group lasso block and MLP block to obtain a phenotype prediction. The group lasso module is designed to control which modalities contribute most to each phenotype; it consists of a linear layer, batch normalization, GELU activation, and a dropout. An L2 norm of weights is then calculated and summed to calculate regularization loss. Each modality output is stacked, and the final prediction is calculated by summing the outputs across modalities and adding a group lasso penalty term (regularization loss $\times$ lambda).

3.3.1 Model Training

A robust three-stage training pipeline is employed for effective multi-task learning in AgriNetFusion. The process includes uncertainty weighted multi-task loss, adaptive backbone and task specific head freezing, modality guided multi-level early stopping, and a dynamic learning rate schedule.

In stage 1, models are trained for 10 epochs with an early stopping patience of 3. Hyperparameter performance is evaluated using a validation set, and results from stage 1 are saved and used for stage 2.

In stage 2, model weights are initialized from the best performing stage 1 results, and models are trained for 30 epochs with a patience threshold of 10. The learning rate is gradually increased over the initial warmup epochs. In this stage of training, a dynamic freezing system is applied; the shared backbone can be frozen if overall loss plateaus, allowing the model to focus on training the individual phenotype heads. This allows for even training across all aspects of the model. A custom loss function incorporates uncertainty weighting by adjusting the contribution of each phenotype's loss based on its variance. Validation losses are calculated on the model after each epoch to determine the performance of the shared backbone and individual phenotype heads. If the validation loss plateaus, a learning rate scheduler decreases the learning rate by a factor of 0.5.

Stage 3 is the final component of the training process, initialized with results from stage 2. This stage is run for 300 epochs with a patience threshold of 30. The backbone or task-specific phenotype heads can be frozen depending on their validation loss, similar to stage 2. After training, the model is evaluated on a held-out test set. A key feature of this stage is the optional construction of a hybrid model, where the highest performing phenotype heads can be assembled to create a model from multiple training pathways and parameter settings.

3.4 Model Variations

Several model variations were constructed to determine the effectiveness of the attention mechanism, the genotype autoencoder and hierarchal modality structure in effectively capturing relationships in the data, as shown in Figure 3.4.

	AgriNetFusion	AgriNetHMMA	AgriNetSHMMA	AgriNetMLP
Attention	✓	✓	✓	x
Genotype Encoder Rep	✓	X	x	x
Shared Rep	✓	✓	✓	x
Individual Rep	✓	✓	x	✓

Figure 3.4: Model Components

AgriNetFusion and AgriNetHMMA both utilize an attention block with both shared and individual modality representations. AgriNetFusion uses an autoencoder representation for the genotype data, while the AgriNetHMMA model uses a PCA genotype representation, allowing the efficacy of these two genotype representations to become clear. AgriNetSHMMA does not have individual submodality representations, and AgriNetMLP does not have a multi-modality shared representation; comparisons of

these models display the importance of both shared and individual modality information when predicting phenotypes in the AgriNetFusion model.

Additionally, AgriNetFusion variants incorporating or excluding the attention and group lasso blocks were trained, where $(++)$ represents the inclusion of attention and group lasso, $(+-)$ represents the inclusion of attention, $(-+)$ represents the inclusion of group lasso, and $(--)$ represents the exclusion of both.

Chapter 4

Results

Figure 4.1 depicts the training and validation loss of each epoch for the AgriNetFusion model. Figure 4.2 depicts the training loss for each epoch of the genotype autoencoder.

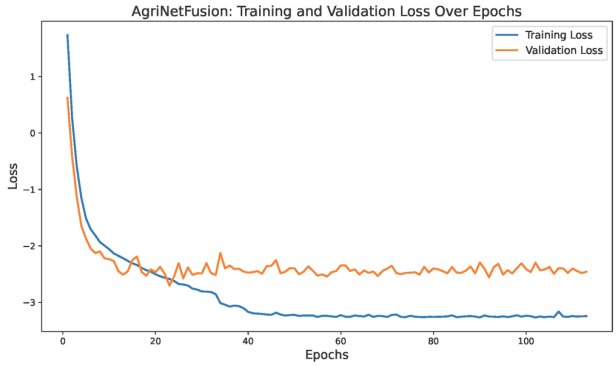


Figure 4.1: AgriNetFusion Training and Validation Loss

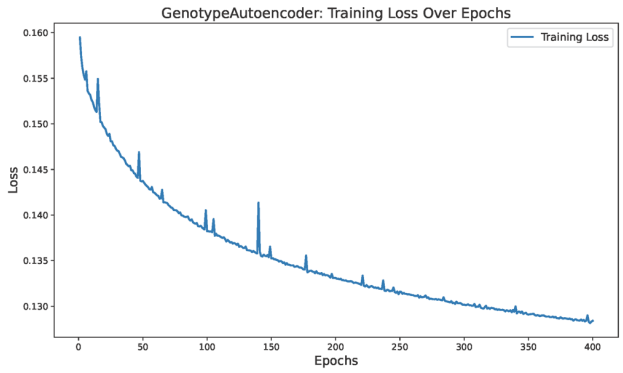


Figure 4.2: Genotype Autoencoder Training Loss

A consistent loss decrease is shown, indicating that the AgriNetFusion model and genotype autoencoder accurately learn features from the data with each epoch.

4.1 Basic Model Comparison

Gradient boosting and decision tree methods are well-performing, widely used methods in machine learning, and have been extended to multi-output prediction (Zhang and Jung, 2019). To determine AgriNetFusion's improvement over these existing techniques, LASSO regression, random forest, CatBoost, and XGBoost models were created as standard machine learning comparisons. One round of hyperparameter tuning was performed on each of these models independently. The alpha parameter was tuned for LASSO regression. The n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, and bootstrap parameters were tuned for the random forest model. The max_depth, learning_rate, subsample, colsample_bytree, and n_estimator parameters are tuned for the XGBoost model. A randomized search of hyperparameter combinations was performed on the XGBoost and random forest models, with 30 random parameter combinations. A grid search was used for the Lasso model tuning, since only one parameter with 20 possible values was tuned. Each tuning algorithm used 5-fold cross validation and mean squared error as the scoring criterion.

Figure 4.3 shows the performance of AgriNetFusion and several basic models against the well-performing XGBoost model. AgriNetFusion(+-) has a lower test error than XGBoost and other basic models for 7 out of the 11 phenotype prediction scenarios. For the remaining phenotypes, AgriNetFusion has a comparable error to the best performing basic models. Thus, since AgriNetFusion provides strong interpretability, it is a favorable option over basic models.

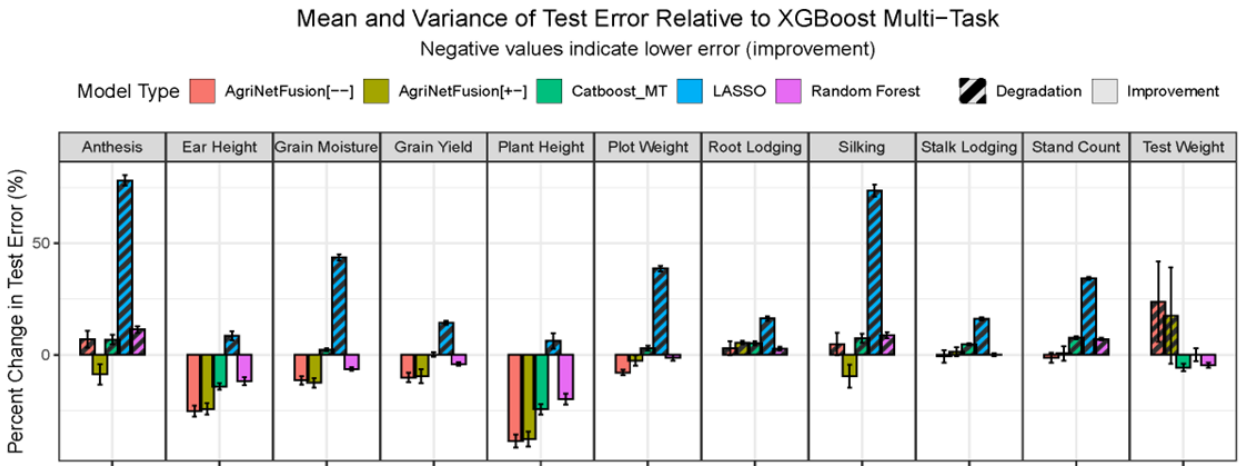


Figure 4.3: Comparison of AgriNetFusion and Basic Models to XGBoost

Figure 4.4 displays a sample of 300 observed and predicted values for each phenotype, grouped into four categories: Phenology, Lodging, Yield and Moisture, and Morphological Traits. Each subplot includes a fitted regression line and the R^2 value. Most phenotypes exhibit strong or moderate relationships between the observed and predicted values, with Phenology traits displaying the strongest relationships. Overall, AgriNetFusion consistently provides estimates close to the observed values, indicating a well-performing model.

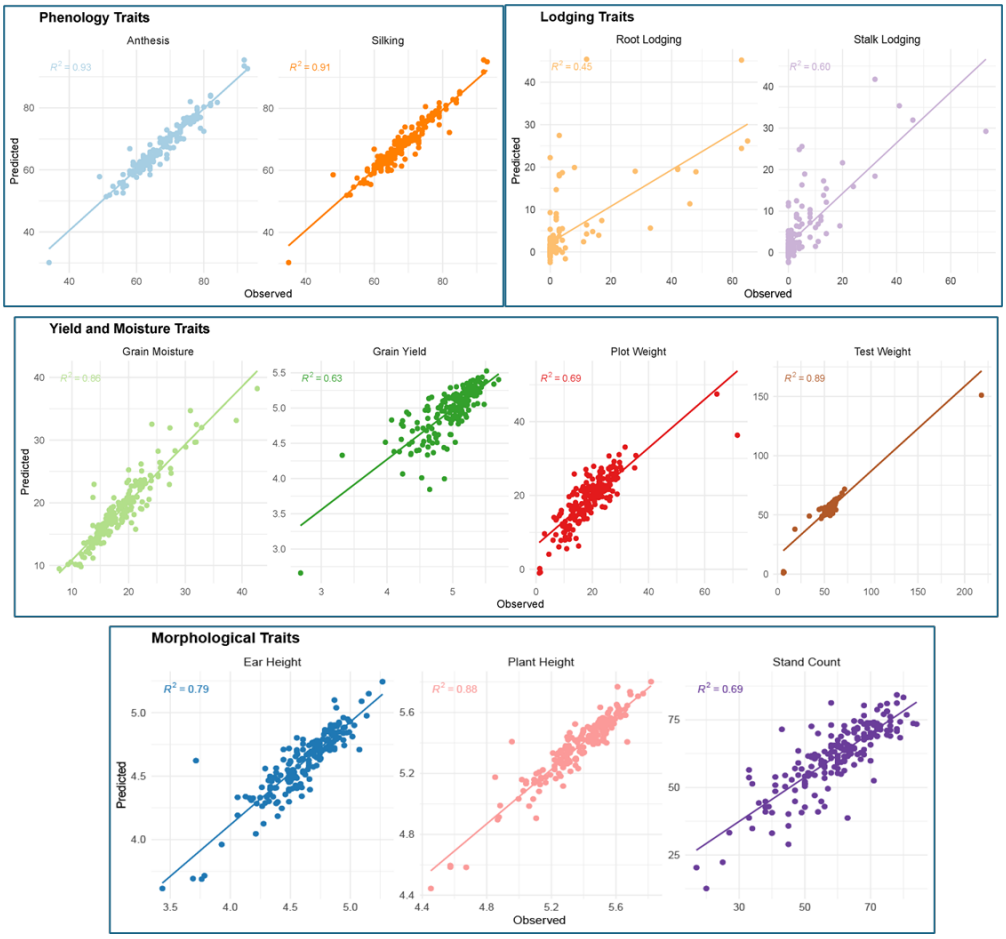


Figure 4.4: Observed versus Predicted Phenotype Observations

4.2 Ablation Study

An ablation study of the AgriNetFusion variants with and without attention, group lasso, the genotype encoder representation, shared modality representations, and individual modality representations reveal the best model for phenotype prediction.

The Comparison of Mean Test Error for Genotype Classification in Figure 4.5 displays a standardized mean test error for the genotype autoencoder and PCA for 11 genotype categories, where lower errors

indicate better classification. The genotype autoencoder consistently has lower mean test error, indicating its superiority in extracting important genotype features.

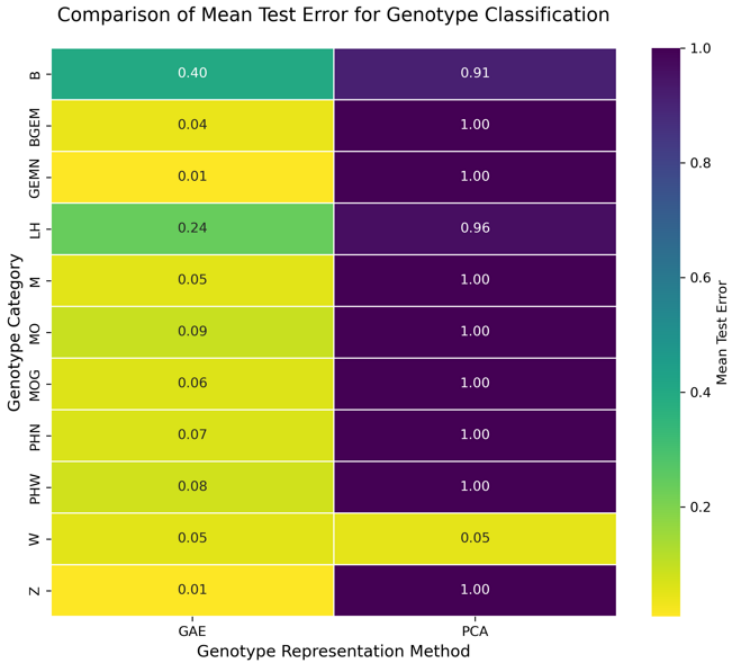


Figure 4.5: Genotype Classification Error Comparison

The AgriNetHMMA model is similar to AgriNetFusion, with the exception of the genotype representation; AgriNetHMMA contains a genotype representation created through PCA, and AgriNetFusion contains a genotype representation from the genotype autoencoder. Thus, comparing the performance of these two models displays the effectiveness of the genotype autoencoder. Figure 4.6 shows that AgriNetHMMA's test error is higher than AgriNetFusion's for 9 out of 11 phenotypes. This indicates the effectiveness of the genotype autoencoder in extracting useful information for phenotype prediction.

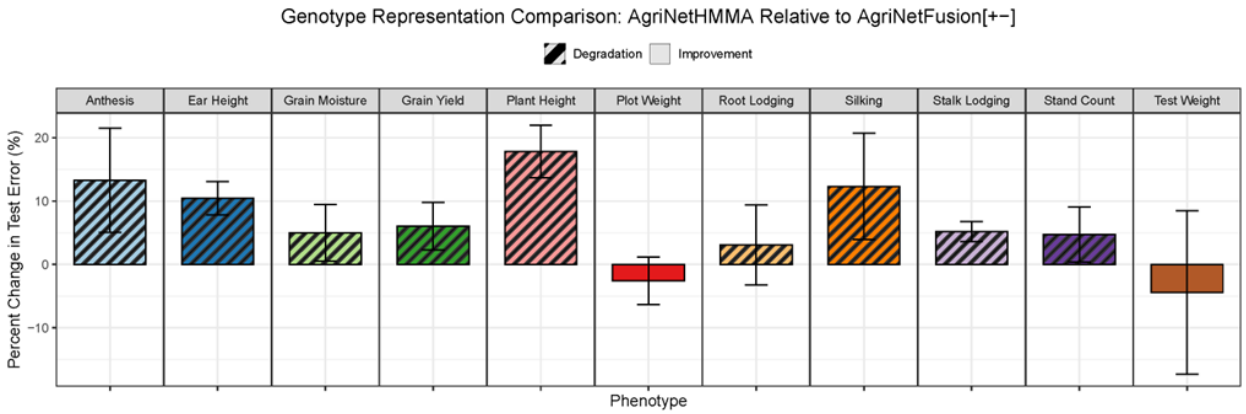


Figure 4.6: Genotype Representation Model Comparison

Figure 4.7 compares test error of the AgriNetFusion model with attention (AgriNetFusion(+−)) as a baseline against three other variants of the model: AgriNetFusion without attention or group lasso (−−), with attention and without group lasso (−+), and with both attention and group lasso (++). AgriNetFusion(++) displays degradation over the baseline for 9 out of 11 phenotypes. AgriNetFusion(−+) displays degradation over the baseline for 6 out of 11 phenotypes. AgriNetFusion(−−) displays degradation over the baseline for 4 out of 11 phenotypes.

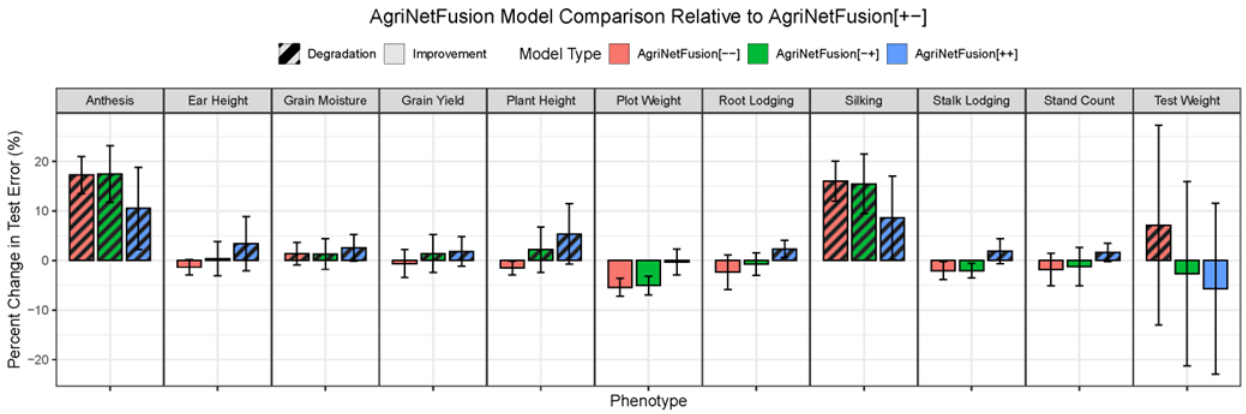


Figure 4.7: AgriNetFusion Model Comparison

The MultiModalMLP model lacks a shared representation of all modalities, so comparing it to the AgriNetFusion model helps reveal the significance of information sharing across modalities. The test error

for the MultiModalMLP model is higher than the test error for AgriNetFusion for 10 out of 11 phenotypes, shown in Figure 4.8, indicating the importance of a shared representation in the model.

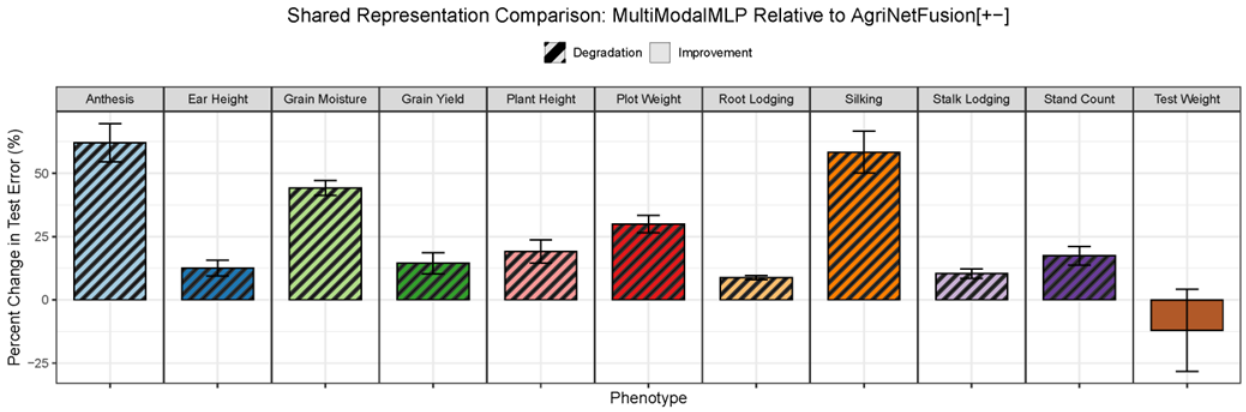


Figure 4.8: Shared Representation Comparison

The SharedHMMA model lacks individual submodality representations when predicting phenotypes, and it has a higher test error than AgriNetFusion for 10 out of 11 phenotypes. This result, shown in Figure 4.9, indicates the importance of individual submodality information when predicting these phenotypes.

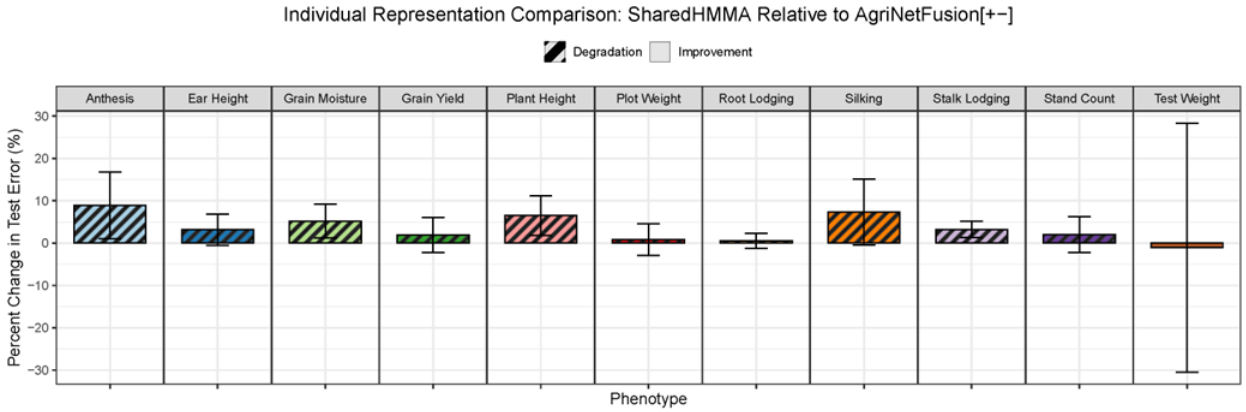
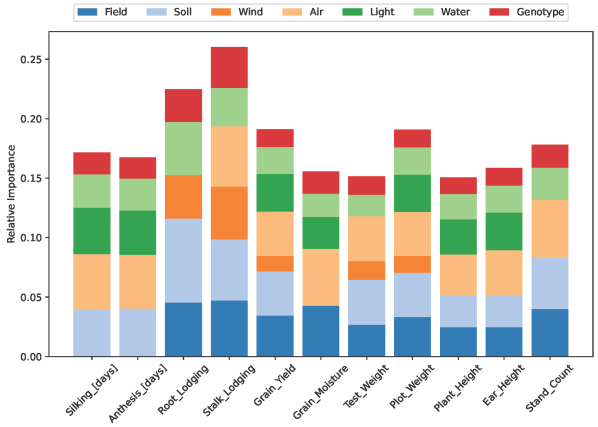


Figure 4.9: Individual Representation Comparison

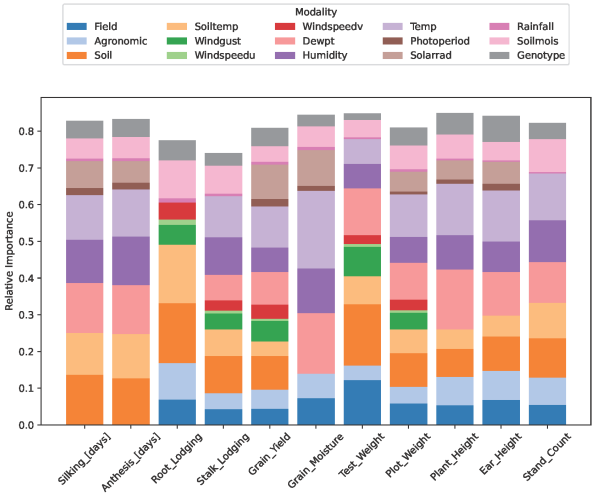
4.3 Interpretation and Modality Contributions

To determine the most influential modalities for each phenotype prediction, SHAP (SHapley Additive exPlanations) values were calculated for each modality and phenotype combination. SHAP values indicate

a feature's contribution to a prediction by summing the difference in the prediction with and without a certain feature for all possible feature subsets (Lundberg and Lee, 2017). SHAP values were calculated for both the modality groups and the individual modalities, and relative importance for all features was calculated by dividing each SHAP value by the sum of all SHAP values. The results are shown in 4.10.



(a) Relative SHAP Contributions for Modality Groups



(b) Relative SHAP Contributions for Individual Modalities

Figure 4.10: Comparison of SHAP Contributions Across Modalities and Submodalities

Individual modalities account for a large portion of relative importance in all phenotype predictions, while modality groups tend to contain relative importance values of 0.15 to 0.20. Water and genotype are the only modality groups with importance for all phenotype predictions, indicating the widespread significance of these higher level representations in predicting a diverse range of phenotypes. Air and soil modality groups fall close behind, with importance for ten phenotypes. Relative importance compositions tend to be noticeably different across phenotypes, with the exception of Anthesis and Silking. These two phenotypes are the only outcomes with high correlations, so similar information can be used to predict both of these target variables. A similar pattern exists in the individual modality importances; Anthesis and Silking have similar important individual modality makeups, with Temperature, Dew Point, and Soil Nutrients as the highest influences of these predictions. These three individual modalities have substantial

importance values across many phenotype predictions, indicating these variables' strong influence in corn outputs.

The importance of interactions between modalities was also considered. Attention scores are learned weights in the attention mechanism of the AgriNetFusion model which quantify the importance of input modalities or modality interactions when making each phenotype prediction. High attention scores indicate that the model is focusing heavily on certain interactions for a specific output. Figure 4.11 displays log-transformed attention scores assigned to interactions between individual modalities that reside in the same modality group, highlighting the important interactions that exist within these modality groups. The bars are categorized into quantile groups, with blue bars indicating the top 10% of phenotype values, and red bars indicating the bottom 10% of phenotype values. Thus, modality interactions with high positive values indicate a strong influence for prediction of the top values of a phenotype, and interactions with more negative values indicate a strong influence for predicting the lowest values of a phenotype. For example, in the ear height panel, the Humidity $\times$ Temperature interaction has a high log attention score in the top 10% quantile, suggesting that this interaction is influential in predicting high ear height values, or the tallest plants. Conversely, Soil Nutrients $\times$ Soil Temperature has a very negative log attention score, so interactions between these soil variables are important for predicting low ear height values. Interactions that appear in both the top and bottom 10% are significant for predicting both high and low performing phenotypes.

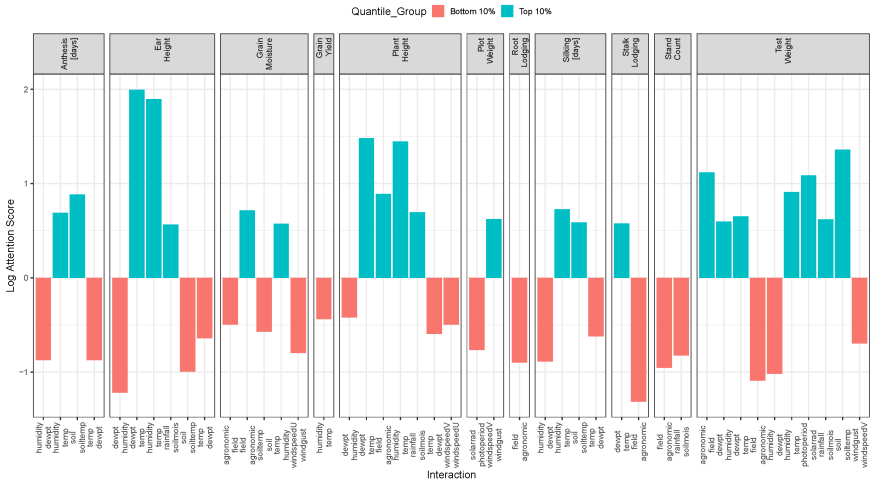


Figure 4.11: Intra Attention

Figure 4.12 displays log attention scores for interactions between modality groups. The Silking and Test Weight phenotypes contain the highest number of interactions, indicating the complex exchange of modalities that determine these corn outputs.

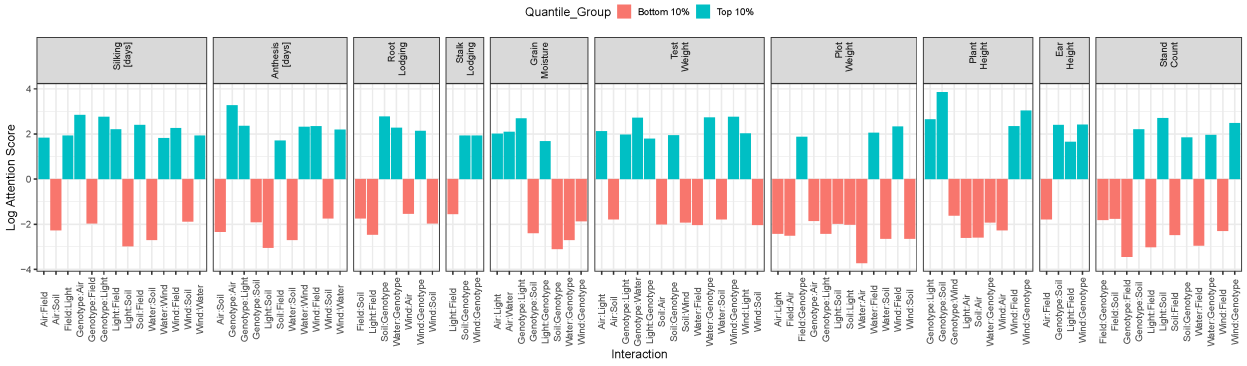


Figure 4.12: Inter Attention

Ultimately, the AgriNetFusion model successfully captures interactions within and between modality groups, uncovering important patterns which affect corn outputs.

Chapter 5

Conclusion

This study presents AgriNetFusion, a novel deep learning framework designed to improve prediction of corn phenotypes and interpret the complex interactions between multimodal environmental, genetic, and treatment data. This framework adds to existing literature by incorporating a hierarchal architecture and guided attention mechanisms which facilitate the use of information from individual submodalities and larger modality groups with shared effects. AgriNetFusion demonstrated strong prediction performance across 11 phenotypes, surpassing traditional machine learning models. An ablation study revealed the success of the hierarchal structure and core components of the model, highlighting the need for both shared modality group representations and individual submodality representations in the model. Additionally, we provide interpretable insights into submodality and group modality contributions in the model, revealing important interactions and relationships. This research contributes to a deeper understanding of corn growth and phenotype prediction which enables deeper insights into the factors and interactions that affect this complex process.

Appendix A

Data Preprocessing (Phenotype, Soil, Agronomic, Field, and Weather)

```
# Importing data from G2F

hybrid_phenotype_2014 <-
↪  read_csv("2014/a._2014_hybrid_phenotypic_data/g2f_2014_hybrid_data_clean.csv")
weather_2014 <- read_csv("2014/b._2014_weather_data/g2f_2014_weather.csv")
inbred_phenotype_2014 <-
↪  read_csv("2014/c._2014_inbred_phenotypic_data/g2f_2014_inbred_data_clean.csv")
field_2014 <-
↪  read_csv("2014/z._2014_supplemental_info/g2f_2014_field_characteristics.csv")

hybrid_phenotype_2015 <-
↪  read_csv("2015/a._2015_hybrid_phenotypic_data/g2f_2015_hybrid_data_clean.csv")
```

```
weather_2015 <- read_csv("2015/b._2015_weather_data/g2f_2015_weather.csv")
inbred_phenotype_2015 <-
 ↪  read_csv("2015/c._2015_inbred_phenotypic_data/g2f_2015_inbred_data_clean.csv")
soil_2015 <-read_csv("2015/d._2015_soil_data/g2f_2015_soil_data.csv")
field_2015 <-
 ↪  read_csv("2015/z._2015_supplemental_info/g2f_2015_field_metadata.csv")


phenotype_2016 <-
 ↪  read_csv("2016/a._2016_hybrid_phenotypic_data/g2f_2016_hybrid_data_clean.csv")
weather_2016 <-
 ↪  read_csv("2016/b._2016_weather_data/g2f_2016_weather_data.csv")
soil_2016 <-read_csv("2016/c._2016_soil_data/g2f_2016_soil_data_clean.csv")
field_2016 <-
 ↪  read_csv("2016/z._2016_supplemental_info/g2f_2016_field_metadata.csv",
                      locale=locale(encoding="latin1"))
field_2016 = field_2016[1:41] #leaving out issue/comment columns
agronomic_2016 <-
 ↪  read_csv("2016/z._2016_supplemental_info/g2f_2016_agronomic_information.csv")


phenotype_2017 <-
 ↪  read_csv("2017/a._2017_hybrid_phenotypic_data/g2f_2017_hybrid_data_clean.csv")
weather_2017 <-
 ↪  read_csv("2017/b._2017_weather_data/g2f_2017_weather_data.csv")
soil_2017 <-read_csv("2017/c._2017_soil_data/g2f_2017_soil_data_clean.csv")
field_2017 <-
 ↪  read_csv("2017/z._2017_supplemental_info/g2f_2017_field_metadata.csv",
```

```
                      locale=locale(encoding="latin1"))
agronomic_2017 <-
 ↪  read_csv("2017/z._2017_supplemental_info/g2f_2017_agronomic
 ↪  information.csv")

phenotype_2018 <-
 ↪  read_csv("2018/a._2018_hybrid_phenotypic_data/g2f_2018_hybrid_data_clean.csv")
weather_2018 <-
 ↪  read_csv("2018/b._2018_weather_data/g2f_2018_weather_clean.csv")
soil_2018 <-read_csv("2018/c._2018_soil_data/g2f_2018_soil_data.csv")
field_2018 <-
 ↪  read_csv("2018/e._2018_supplemental_info/g2f_2018_field_metadata.csv",
                      locale=locale(encoding="latin1"))
agronomic_2018 <-
 ↪  read_csv("2018/e._2018_supplemental_info/g2f_2018_agronomic
 ↪  information.csv")

phenotype_2019 <-
 ↪  read_csv("2019/a._2019_phenotypic_data/g2f_2019_phenotypic_clean_data.csv")
weather_2019 <-
 ↪  read_csv("2019/b._2019_weather_data/2019_weather_cleaned.csv")
soil_2019 <-read_csv("2019/c._2019_soil_data/g2f_2019_soil_data.csv")
field_2019 <-
 ↪  read_csv("2019/z._2019_supplemental_info/g2f_2019_field_metadata.csv")
agronomic_2019 <-
 ↪  read_csv("2019/z._2019_supplemental_info/g2f_2019_agronomic_information.csv")
```

```
phenotype_2020 <-
 ↪  read_csv("2020/a._2020_phenotypic_data/g2f_2020_phenotypic_clean_data.csv")
weather_2020 <-
 ↪  read_csv("2020/b._2020_weather_data/2020_weather_cleaned.csv")
soil_2020 <-read_csv("2020/c._2020_soil_data/g2f_2020_soil_data.csv")
field_2020 <-
 ↪  read_csv("2020/z._2020_supplemental_info/g2f_2020_field_metadata.csv")
agronomic_2020 <-
 ↪  read_csv("2020/z._2020_supplemental_info/g2f_2020_agronomic_information.csv")

phenotype_2021 <-
 ↪  read_csv("2021/a._2021_phenotypic_data/g2f_2021_phenotypic_clean_data.csv")
weather_2021 <-
 ↪  read_csv("2021/b._2021_weather_data/g2f_2021_weather_cleaned.csv")
soil_2021 <-read_csv("2021/c._2021_soil_data/g2f_2021_soil_data.csv")
field_2021 <-
 ↪  read_csv("2021/z._2021_supplemental_info/g2f_2021_field_metadata.csv")
agronomic_2021 <-
 ↪  read_csv("2021/z._2021_supplemental_info/g2f_2021_agronomic_information.csv")

phenotype_2022 <-
 ↪  read_csv("2022/a._2022_phenotypic_data/g2f_2022_phenotypic_clean_data.csv")
weather_2022 <-
 ↪  read_csv("2022/b._2022_weather_data/g2f_2022_weather_cleaned.csv")
```

```
field_2022 <-
↪  read_csv("2022/z._2022_supplemental_info/g2f_2022_field_metadata.csv")
soil_2022 <-read_csv("2022/c._2022_soil_data/g2f_2022_soil_data.csv")
agronomic_2022 <-
↪  read_csv("2022/z._2022_supplemental_info/g2f_2022_agronomic_information.csv")


# Creating uniform column names
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) ==
↪  'Field-Location'] <- 'Field Location'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'Location'] <-
↪  'Field Location'
names(field_2014)[names(field_2014) == 'Experiment'] <- 'Field Location'

names(weather_2014)[names(weather_2014) == 'Month [Local]'] <- 'Month'
names(weather_2014)[names(weather_2014) == 'Time [Local]'] <- 'Time'
names(weather_2014)[names(weather_2014) == 'Year [Local]'] <- 'Year'
names(weather_2014)[names(weather_2014) == 'Day of Year [Local]'] <- 'Day of
↪  Year'
names(weather_2014)[names(weather_2014) == 'Day [Local]'] <- 'Day'

names(field_2014)[names(field_2014) == 'RowSp'] <- 'Day'
names(field_2014)[names(field_2014) == 'Location name'] <- 'State'
names(field_2014)[names(field_2014) == 'Plot length (center to center in
↪  feet)'] <- 'Plot_length (center-alley to center-alley in feet)'
```

```
names(field_2014)[names(field_2014) == 'Alley length (inches)'] <-
↪  'Alley_length (in inches)'
names(field_2014)[names(field_2014) == 'lat'] <- 'Weather_Station_Latitude
↪  (in decimal numbers NOT DMS)'
names(field_2014)[names(field_2014) == 'long'] <- 'Weather_Station_Longitude
↪  (in decimal numbers NOT DMS)'
names(field_2014)[names(field_2014) == 'Row spacing (inches)'] <-
↪  'Row_spacing (in inches)'
names(field_2014)[names(field_2014) == 'Number kernels planted'] <-
↪  'Number_kernels_planted_per_plot (>200 seed/pack for cone planter'
names(field_2014)[names(field_2014) == 'Planter type'] <- 'Type_of_planter
↪  (fluted cone; belt cone; air planter)'
names(field_2014)[names(field_2014) == 'Total nitrogen'] <- 'total N'
names(field_2014)[names(field_2014) == 'Total phosphorus'] <- 'total P'
names(field_2014)[names(field_2014) == 'Total potassium'] <- 'total K'
names(field_2014)[names(field_2014) == 'Previous crop'] <- 'Previous Crop'
names(field_2014)[names(field_2014) == 'Tillage method'] <- 'Tillage'

names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'LOCAL_CHECK
↪  (Yes, No[Blank])'] <- 'Local Check (Yes, No)'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Plot Length
↪  Field'] <- 'Plot length (center-center in feet)'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Alley Length']
↪  <- 'Alley length (in inches)'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Row Spacing']
↪  <- 'Row spacing (in inches)'
```

```
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Plot Area'] <-
↪  'Plot area (ft2)'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Rows/Plot'] <-
↪  'Rows per plot'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == '# Seed'] <- '#
↪  Seed per plot'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Date Planted']
↪  <- 'Date Plot Planted [MM/DD/YY]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Date
↪  Harvested'] <- 'Date Plot Harvested [MM/DD/YY]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Anthesis
↪  [date]'] <- 'Anthesis [MM/DD/YY]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Anthesis
↪  [date]'] <- 'Date Plot Harvested [MM/DD/YY]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Stand Count
↪  [plants]'] <- 'Stand Count [# of plants]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Root Lodging
↪  [plants]'] <- 'Root Lodging [# of plants]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Stalk Lodging
↪  [plants]'] <- 'Stalk Lodging [# of plants]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Grain Yield
↪  [bu/A]'] <- 'Grain Yield (bu/A)'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Plot Discarded
↪  [enter \"yes\" or \"blank\"]'] <- 'Plot Discarded [enter \'yes\' or
↪  blank]'
```

```
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Filler [enter
↪  \"filler\" or \"blank\"]'] <- 'Filler'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Test Weight
↪  [lbs/bu]'] <- 'Test Weight [lbs]'
names(hybrid_phenotype_2014)[names(hybrid_phenotype_2014) == 'Silking
↪  [date]'] <- 'Silking [MM/DD/YY]'

names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'plantingDate']
↪  <- 'Date Plot Planted [MM/DD/YY]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'harvestDate']
↪  <- 'Date Plot Harvested [MM/DD/YY]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'Rep'] <-
↪  'Replicate'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'Genotype'] <-
↪  'Pedigree'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'Discarded'] <-
↪  'Plot Discarded [enter \'yes\' or blank]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'stalkLodging']
↪  <- 'Stalk Lodging [# of plants]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'rootLodging']
↪  <- 'Root Lodging [# of plants]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'standCount']
↪  <- 'Stand Count [# of plants]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'earHeight'] <-
↪  'Ear Height [cm]'
```

```
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'plantHeight']
↪ <- 'Plant Height [cm]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'pollendap'] <-
↪ 'Pollen DAP [days]'
names(inbred_phenotype_2014)[names(inbred_phenotype_2014) == 'silkdap'] <-
↪ 'Silk DAP [days]'

names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) ==
↪ 'Field-Location'] <- 'Field Location'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) ==
↪ 'Field-Location'] <- 'Field Location'
names(weather_2015)[names(weather_2015) == 'Experiment(s)'] <- 'Field
↪ Location'
names(field_2015)[names(field_2015) == 'Experiment'] <- 'Field Location'
names(soil_2015)[names(soil_2015) == 'Experiment'] <- 'Field Location'

names(weather_2015)[names(weather_2015) == 'Time [Local]'] <- 'Time'
names(weather_2015)[names(weather_2015) == 'Soil Moisture [%]'] <- 'Soil
↪ Moisture [%VWC]'

names(field_2015)[names(field_2015) == 'WS_SN'] <-
↪ 'Weather_Station_Serial_Number (Last four digits, e.g. m2700s#####)'
names(field_2015)[names(field_2015) == 'WS Lat'] <-
↪ 'Weather_Station_Latitude (in decimal numbers NOT DMS)'
names(field_2015)[names(field_2015) == 'WS Lon'] <-
↪ 'Weather_Station_Longitude (in decimal numbers NOT DMS)'
```

```
names(field_2015)[names(field_2015) == 'DateIn'] <-
↪  'Date_weather_station_placed'
names(field_2015)[names(field_2015) == 'DateOut'] <-
↪  'Date_weather_station_removed'
names(field_2015)[names(field_2015) == 'corner1 lat'] <-
↪  'Latitude_of_Field_Corner_#1 (lower left)'
names(field_2015)[names(field_2015) == 'corner1 lon'] <-
↪  'Longitude_of_Field_Corner_#1 (lower left)'
names(field_2015)[names(field_2015) == 'corner 2lat'] <-
↪  'Latitude_of_Field_Corner_#2 (lower right)'
names(field_2015)[names(field_2015) == 'corner2 lon'] <-
↪  'Longitude_of_Field_Corner_#2 (lower right)'
names(field_2015)[names(field_2015) == 'corner3 lat'] <-
↪  'Latitude_of_Field_Corner_#3 (upper right)'
names(field_2015)[names(field_2015) == 'corner3 lon'] <-
↪  'Longitude_of_Field_Corner_#3 (upper right)'
names(field_2015)[names(field_2015) == 'corner4 lat'] <-
↪  'Latitude_of_Field_Corner_#4 (upper left)'
names(field_2015)[names(field_2015) == 'corner4 lon'] <-
↪  'Longitude_of_Field_Corner_#4 (upper left)'
names(field_2015)[names(field_2015) == 'PlotLen'] <- 'Plot_length
↪  (center-alley to center-alley in feet)'
names(field_2015)[names(field_2015) == 'AlleyLen'] <- 'Alley_length (in
↪  inches)'
names(field_2015)[names(field_2015) == 'RowSp'] <- 'Row_spacing (in inches)'
```

```
names(field_2015)[names(field_2015) == 'PlanterType'] <- 'Type_of_planter
↪  (fluted cone; belt cone; air planter)'
names(field_2015)[names(field_2015) == 'KernelsPerPlot'] <-
↪  'Number_kernels_planted_per_plot (>200 seed/pack for cone planter'

names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'LOCAL_CHECK
↪  (Yes, No[Blank])'] <- 'Local Check (Yes, No)'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Plot Length
↪  Field'] <- 'Plot length (center-center in feet)'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Alley Length']
↪  <- 'Alley length (in inches)'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Row Spacing']
↪  <- 'Row spacing (in inches)'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Plot Area'] <-
↪  'Plot area (ft2)'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Rows/Plot'] <-
↪  'Rows per plot'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == '# Seed'] <- '#
↪  Seed per plot'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Date Planted']
↪  <- 'Date Plot Planted [MM/DD/YY]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Date
↪  Harvested'] <- 'Date Plot Harvested [MM/DD/YY]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Anthesis
↪  [date]'] <- 'Anthesis [MM/DD/YY]'
```

```
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Anthesis
↪  [date]'] <- 'Date Plot Harvested [MM/DD/YY]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Stand Count
↪  [plants]'] <- 'Stand Count [# of plants]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Root Lodging
↪  [plants]'] <- 'Root Lodging [# of plants]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Stalk Lodging
↪  [plants]'] <- 'Stalk Lodging [# of plants]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Grain Yield
↪  [bu/A]'] <- 'Grain Yield (bu/A)'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Plot Discarded
↪  [enter \"yes\" or \"blank\"]'] <- 'Plot Discarded [enter \'yes\' or
↪  blank]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Filler [enter
↪  \"filler\" or \"blank\"]'] <- 'Filler'

names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Silking
↪  [date]'] <- 'Silking [MM/DD/YY]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Silking
↪  [date]'] <- 'Silking [MM/DD/YY]'

names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Field
↪  Location'] <- 'Drop'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Book Name'] <-
↪  'Field Location'
```

```
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Date Planted']
↪ <- 'Date Plot Planted [MM/DD/YY]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Date
↪ Harvested'] <- 'Date Plot Harvested [MM/DD/YY]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Anthesis
↪ [date]'] <- 'Anthesis [MM/DD/YY]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Anthesis
↪ [date]'] <- 'Anthesis [MM/DD/YY]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Stand Count
↪ [plants]'] <- 'Stand Count [# of plants]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Plant height
↪ [cm]'] <- 'Plant Height [cm]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Ear height
↪ [cm]'] <- 'Ear Height [cm]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Root lodging
↪ [plants]'] <- 'Root Lodging [# of plants]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Stalk lodging
↪ [plants]'] <- 'Stalk Lodging [# of plants]'
names(inbred_phenotype_2015)[names(inbred_phenotype_2015) == 'Plot
↪ Discarded'] <- 'Plot Discarded [enter \'yes\' or blank]'
names(hybrid_phenotype_2015)[names(hybrid_phenotype_2015) == 'Test Weight
↪ [lbs/bu]'] <- 'Test Weight [lbs]'

names(soil_2015)[names(soil_2015) == 'PlowDepth'] <- 'E Depth'
names(soil_2015)[names(soil_2015) == 'PH'] <- '1:1 Soil pH'
names(soil_2015)[names(soil_2015) == 'BpH'] <- 'WDRF Buffer pH'
```

```
names(soil_2015)[names(soil_2015) == 'OM'] <- 'Organic Matter LOI %'
names(soil_2015)[names(soil_2015) == 'P'] <- 'Mehlich P-III ppm P'
names(soil_2015)[names(soil_2015) == 'K'] <- 'Potassium ppm K'

names(phenotype_2016)[names(phenotype_2016) == 'Field-Location'] <- 'Field
↪ Location'
names(field_2016)[names(field_2016) == 'Experiment_Code'] <- 'Field
↪ Location'
names(soil_2016)[names(soil_2016) == 'Location'] <- 'Field Location'
names(agronomic_2016)[names(agronomic_2016) == 'Experiment Code'] <- 'Field
↪ Location'

names(phenotype_2016)[names(phenotype_2016) == 'Plot Length Field'] <- 'Plot
↪ length (center-center in feet)'
names(phenotype_2016)[names(phenotype_2016) == 'Plot Area'] <- 'Plot area
↪ (ft2)'
names(phenotype_2016)[names(phenotype_2016) == 'Tester/Group'] <- 'Tester'
names(phenotype_2016)[names(phenotype_2016) == 'Alley Length'] <- 'Alley
↪ length (in inches)'
names(phenotype_2016)[names(phenotype_2016) == 'Anthesis [date]'] <-
↪ 'Anthesis [MM/DD/YY]'
names(phenotype_2016)[names(phenotype_2016) == 'Date Planted'] <- 'Date Plot
↪ Planted [MM/DD/YY]'
names(phenotype_2016)[names(phenotype_2016) == 'Silking [date]'] <- 'Silking
↪ [MM/DD/YY]'
```

```
names(phenotype_2016)[names(phenotype_2016) == 'Grain Yield [bu/A]'] <-
↪  'Grain Yield (bu/A)'
names(phenotype_2016)[names(phenotype_2016) == '# Seed'] <- '# Seed per
↪  plot'
names(phenotype_2016)[names(phenotype_2016) == 'Root Lodging [plants]'] <-
↪  'Root Lodging [# of plants]'
names(phenotype_2016)[names(phenotype_2016) == 'Row Spacing'] <- 'Row
↪  spacing (in inches)'
names(phenotype_2016)[names(phenotype_2016) == 'Rows/Plot'] <- 'Rows per
↪  plot'
names(phenotype_2016)[names(phenotype_2016) == 'Date Harvested'] <- 'Date
↪  Plot Harvested [MM/DD/YY]'
names(phenotype_2016)[names(phenotype_2016) == 'Plot Discarded [enter
↪  \"Yes\" or \"blank\"]'] <- 'Plot Discarded [enter \'yes\' or blank]'
names(phenotype_2016)[names(phenotype_2016) == 'Filler [enter \"filler\" or
↪  \"blank\"]'] <- 'Filler'
names(phenotype_2016)[names(phenotype_2016) == 'Stand Count [plants]'] <-
↪  'Stand Count [# of plants]'
names(phenotype_2016)[names(phenotype_2016) == 'Stalk Lodging [plants]'] <-
↪  'Stalk Lodging [# of plants]'
names(phenotype_2016)[names(phenotype_2016) == 'Test Weight [lbs/bu]'] <-
↪  'Test Weight [lbs]'
names(phenotype_2016)[names(phenotype_2016) == 'Plot Discarded [enter
↪  \"yes\" or \"blank\"]'] <- 'Plot Discarded [enter \'yes\' or blank]'
names(phenotype_2016)[names(phenotype_2016) == 'LOCAL_CHECK (Yes,
↪  No[Blank])'] <- 'Local Check (Yes, No)'
```

```
names(weather_2016)[names(weather_2016) == 'Rainfall[mm]'] <- 'Rainfall
↪  [mm]'
names(weather_2016)[names(weather_2016) == 'Photoperiod[hours]'] <-
↪  'Photoperiod [hours]'
names(weather_2016)[names(weather_2016) == 'Time[Local]'] <- 'Time'

names(agronomic_2016)[names(agronomic_2016) == 'Application or Treatment']
↪  <- 'Application_or_treatment'
names(agronomic_2016)[names(agronomic_2016) == 'Product/Nutrient Applied']
↪  <- 'Product_or_nutrient_applied'
names(agronomic_2016)[names(agronomic_2016) == 'Date of Application'] <-
↪  'Date_of_application'
names(agronomic_2016)[names(agronomic_2016) == 'Application unit\n(lbs, in,
↪  oz per acre)'] <- 'Application_unit'
names(agronomic_2016)[names(agronomic_2016) == 'Quantity per acre'] <-
↪  'Quantity_per_acre'

names(cooperators_2016)[names(cooperators_2016) == 'Collaborator/PI'] <-
↪  'Collaborator'

names(phenotype_2017)[names(phenotype_2017) == 'Field-Location'] <- 'Field
↪  Location'
names(field_2017)[names(field_2017) == 'Experiment_Code'] <- 'Field
↪  Location'
names(soil_2017)[names(soil_2017) == 'Location'] <- 'Field Location'
```

```
names(agronomic_2017)[names(agronomic_2017) == 'Location'] <- 'Field
↪  Location'


names(phenotype_2017)[names(phenotype_2017) == 'Plot Length Field'] <- 'Plot
↪  length (center-center in feet)'
names(phenotype_2017)[names(phenotype_2017) == 'Plot Area'] <- 'Plot area
↪  (ft2)'
names(phenotype_2017)[names(phenotype_2017) == 'Tester/Group'] <- 'Tester'
names(phenotype_2017)[names(phenotype_2017) == 'Alley Length'] <- 'Alley
↪  length (in inches)'
names(phenotype_2017)[names(phenotype_2017) == 'Anthesis [date]'] <-
↪  'Anthesis [MM/DD/YY]'
names(phenotype_2017)[names(phenotype_2017) == 'Date Planted'] <- 'Date Plot
↪  Planted [MM/DD/YY]'
names(phenotype_2017)[names(phenotype_2017) == 'Silking [date]'] <- 'Silking
↪  [MM/DD/YY]'
names(phenotype_2017)[names(phenotype_2017) == 'Grain Yield [bu/A]'] <-
↪  'Grain Yield (bu/A)'
names(phenotype_2017)[names(phenotype_2017) == '# Seed'] <- '# Seed per
↪  plot'
names(phenotype_2017)[names(phenotype_2017) == 'Root Lodging [plants]'] <-
↪  'Root Lodging [# of plants]'
names(phenotype_2017)[names(phenotype_2017) == 'Row Spacing'] <- 'Row
↪  spacing (in inches)'
names(phenotype_2017)[names(phenotype_2017) == 'Rows/Plot'] <- 'Rows per
↪  plot'
```

```
names(phenotype_2017)[names(phenotype_2017) == 'Date Harvested'] <- 'Date
↪  Plot Harvested [MM/DD/YY]'
names(phenotype_2017)[names(phenotype_2017) == 'Plot Discarded [enter
↪  \"Yes\" or \"blank\"]'] <- 'Plot Discarded [enter \'yes\' or blank]'
names(phenotype_2017)[names(phenotype_2017) == 'Filler [enter \"filler\" or
↪  \"blank\"]'] <- 'Filler'
names(phenotype_2017)[names(phenotype_2017) == 'Stand Count [plants]'] <-
↪  'Stand Count [# of plants]'
names(phenotype_2017)[names(phenotype_2017) == 'Stalk Lodging [plants]'] <-
↪  'Stalk Lodging [# of plants]'
names(phenotype_2017)[names(phenotype_2017) == 'Test Weight [lbs/bu]'] <-
↪  'Test Weight [lbs]'
names(phenotype_2017)[names(phenotype_2017) == 'Plot Discarded [enter
↪  \"yes\" or \"blank\"]'] <- 'Plot Discarded [enter \'yes\' or blank]'
names(phenotype_2017)[names(phenotype_2017) == 'LOCAL_CHECK (Yes,
↪  No[Blank])'] <- 'Local Check (Yes, No)'

names(phenotype_2018)[names(phenotype_2018) == 'Field-Location'] <- 'Field
↪  Location'
names(field_2018)[names(field_2018) == 'Experiment_Code'] <- 'Field
↪  Location'
names(soil_2018)[names(soil_2018) == 'Field ID'] <- 'Field Location'
names(agronomic_2018)[names(agronomic_2018) == 'Location'] <- 'Field
↪  Location'
names(soil_2018)[names(soil_2018) == 'Date Recieved'] <- 'Date Received'
```

```
weather_2018  = weather_2018[1:28] #1000 unnamed columns with no data
names(weather_2018)[names(weather_2018) == 'UVL (uM/m^2s)'] <- 'UV Light
↪  [uM/m2s]'
names(weather_2018)[names(weather_2018) == 'Photoperiod [ hours]'] <-
↪  'Photoperiod [hours]'


names(phenotype_2018)[names(phenotype_2018) == 'Plot Length Field'] <- 'Plot
↪  length (center-center in feet)'
names(phenotype_2018)[names(phenotype_2018) == 'Plot Area'] <- 'Plot area
↪  (ft2)'
names(phenotype_2018)[names(phenotype_2018) == 'Tester/Group'] <- 'Tester'
names(phenotype_2018)[names(phenotype_2018) == 'Alley Length'] <- 'Alley
↪  length (in inches)'
names(phenotype_2018)[names(phenotype_2018) == 'Anthesis [date]'] <-
↪  'Anthesis [MM/DD/YY]'
names(phenotype_2018)[names(phenotype_2018) == 'Date Planted'] <- 'Date Plot
↪  Planted [MM/DD/YY]'
names(phenotype_2018)[names(phenotype_2018) == 'Silking [date]'] <- 'Silking
↪  [MM/DD/YY]'
names(phenotype_2018)[names(phenotype_2018) == 'Grain Yield [bu/A]'] <-
↪  'Grain Yield (bu/A)'
names(phenotype_2018)[names(phenotype_2018) == '# Seed'] <- '# Seed per
↪  plot'
names(phenotype_2018)[names(phenotype_2018) == 'Root Lodging [plants]'] <-
↪  'Root Lodging [# of plants]'
```

```
names(phenotype_2018)[names(phenotype_2018) == 'Row Spacing'] <- 'Row
↪  spacing (in inches)'
names(phenotype_2018)[names(phenotype_2018) == 'Rows/Plot'] <- 'Rows per
↪  plot'
names(phenotype_2018)[names(phenotype_2018) == 'Date Harvested'] <- 'Date
↪  Plot Harvested [MM/DD/YY]'
names(phenotype_2018)[names(phenotype_2018) == 'Plot Discarded [enter
↪  \"Yes\" or \"blank\"]'] <- 'Plot Discarded [enter \'yes\' or blank]'
names(phenotype_2018)[names(phenotype_2018) == 'Filler [enter \"filler\" or
↪  \"blank\"]'] <- 'Filler'
names(phenotype_2018)[names(phenotype_2018) == 'Stand Count [plants]'] <-
↪  'Stand Count [# of plants]'
names(phenotype_2018)[names(phenotype_2018) == 'Stalk Lodging [plants]'] <-
↪  'Stalk Lodging [# of plants]'
names(phenotype_2018)[names(phenotype_2018) == 'Test Weight [lbs/bu]'] <-
↪  'Test Weight [lbs]'

names(phenotype_2019)[names(phenotype_2019) == 'Field-Location'] <- 'Field
↪  Location'
names(field_2019)[names(field_2019) == 'Experiment_Code'] <- 'Field
↪  Location'
names(soil_2019)[names(soil_2019) == 'Location'] <- 'Field Location'
names(agronomic_2019)[names(agronomic_2019) == 'Location'] <- 'Field
↪  Location'
names(phenotype_2019)[names(phenotype_2019) == 'Filler [enter \'filler\' or
↪  blank]'] <- 'Filler'
```

```
names(phenotype_2020)[names(phenotype_2020) == 'Field-Location'] <- 'Field
↪ Location'
names(field_2020)[names(field_2020) == 'Experiment_Code'] <- 'Field
↪ Location'
names(soil_2020)[names(soil_2020) == 'Location'] <- 'Field Location'
names(agronomic_2020)[names(agronomic_2020) == 'Location'] <- 'Field
↪ Location'

names(phenotype_2021)[names(phenotype_2021) == 'Field-Location'] <- 'Field
↪ Location'
names(field_2021)[names(field_2021) == 'Experiment_Code'] <- 'Field
↪ Location'
names(soil_2021)[names(soil_2021) == 'Location'] <- 'Field Location'
names(agronomic_2021)[names(agronomic_2021) == 'Location'] <- 'Field
↪ Location'

names(phenotype_2022)[names(phenotype_2022) == 'Field-Location'] <- 'Field
↪ Location'
names(field_2022)[names(field_2022) == 'Experiment_Code'] <- 'Field
↪ Location'
names(soil_2022)[names(soil_2022) == 'Location'] <- 'Field Location'
names(agronomic_2022)[names(agronomic_2022) == 'Location'] <- 'Field
↪ Location'
names(field_2022)[names(field_2022) == 'Number_kernels_planted_per_plot'] <-
↪ 'Number_kernels_planted_per_plot (>200 seed/pack for cone planters)'
```

```
# Silking and Anthesis conversion to days after planted
phenotype_2020$`Anthesis [MM/DD/YY]` = mdy(phenotype_2020$`Anthesis
↪  [MM/DD/YY]`)
phenotype_2020$`Silking [MM/DD/YY]` = mdy(phenotype_2020$`Silking
↪  [MM/DD/YY]`)
phenotype_2020$`Date Plot Planted [MM/DD/YY]` = mdy(phenotype_2020$`Date
↪  Plot Planted [MM/DD/YY]`)

phenotype_2019$`Anthesis [MM/DD/YY]` = mdy(phenotype_2019$`Anthesis
↪  [MM/DD/YY]`)
phenotype_2019$`Silking [MM/DD/YY]` = mdy(phenotype_2019$`Silking
↪  [MM/DD/YY]`)
phenotype_2019$`Date Plot Planted [MM/DD/YY]` = mdy(phenotype_2019$`Date
↪  Plot Planted [MM/DD/YY]`)

phenotype_2018$`Anthesis [MM/DD/YY]` = mdy(phenotype_2018$`Anthesis
↪  [MM/DD/YY]`)
phenotype_2018$`Silking [MM/DD/YY]` = mdy(phenotype_2018$`Silking
↪  [MM/DD/YY]`)
phenotype_2018$`Date Plot Planted [MM/DD/YY]` = mdy(phenotype_2018$`Date
↪  Plot Planted [MM/DD/YY]`)

phenotype_2017$`Anthesis [MM/DD/YY]` = mdy(phenotype_2017$`Anthesis
↪  [MM/DD/YY]`)
```

```
phenotype_2017$`Silking [MM/DD/YY]` = mdy(phenotype_2017$`Silking
↪  [MM/DD/YY]`)
phenotype_2017$`Date Plot Planted [MM/DD/YY]` = mdy(phenotype_2017$`Date
↪  Plot Planted [MM/DD/YY]`)

phenotype_2016$`Anthesis [MM/DD/YY]` = mdy(phenotype_2016$`Anthesis
↪  [MM/DD/YY]`)
phenotype_2016$`Silking [MM/DD/YY]` = mdy(phenotype_2016$`Silking
↪  [MM/DD/YY]`)
phenotype_2016$`Date Plot Planted [MM/DD/YY]` = mdy(phenotype_2016$`Date
↪  Plot Planted [MM/DD/YY]`)

hybrid_phenotype_2015$`Anthesis [MM/DD/YY]` =
↪  mdy(hybrid_phenotype_2015$`Anthesis [MM/DD/YY]`)
hybrid_phenotype_2015$`Silking [MM/DD/YY]` =
↪  mdy(hybrid_phenotype_2015$`Silking [MM/DD/YY]`)
hybrid_phenotype_2015$`Date Plot Planted [MM/DD/YY]` =
↪  mdy(hybrid_phenotype_2015$`Date Plot Planted [MM/DD/YY]`)

hybrid_phenotype_2014$`Anthesis [MM/DD/YY]` =
↪  mdy(hybrid_phenotype_2014$`Anthesis [MM/DD/YY]`)
hybrid_phenotype_2014$`Silking [MM/DD/YY]` =
↪  mdy(hybrid_phenotype_2014$`Silking [MM/DD/YY]`)
hybrid_phenotype_2014$`Date Plot Planted [MM/DD/YY]` =
↪  mdy(hybrid_phenotype_2014$`Date Plot Planted [MM/DD/YY]`)
```

```
inbred_phenotype_2015$`Anthesis [MM/DD/YY]` =
↪  mdy(inbred_phenotype_2015$`Anthesis [MM/DD/YY]`)
inbred_phenotype_2015$`Silking [MM/DD/YY]` =
↪  mdy(inbred_phenotype_2015$`Silking [MM/DD/YY]`)
inbred_phenotype_2015$`Date Plot Planted [MM/DD/YY]` =
↪  mdy(inbred_phenotype_2015$`Date Plot Planted [MM/DD/YY]`)

inbred_phenotype_2014$`Date Plot Planted [MM/DD/YY]` =
↪  mdy(inbred_phenotype_2014$`Date Plot Planted [MM/DD/YY]`)

hybrid_phenotype_2014$`Silking [days]` = hybrid_phenotype_2014$`Silking
↪  [MM/DD/YY]`-hybrid_phenotype_2014$`Date Plot Planted [MM/DD/YY]`
hybrid_phenotype_2014$`Anthesis [days]` = hybrid_phenotype_2014$`Anthesis
↪  [MM/DD/YY]`-hybrid_phenotype_2014$`Date Plot Planted [MM/DD/YY]`

inbred_phenotype_2015$`Silking [days]` = inbred_phenotype_2015$`Silking
↪  [MM/DD/YY]`-inbred_phenotype_2015$`Date Plot Planted [MM/DD/YY]`
inbred_phenotype_2015$`Anthesis [days]` = inbred_phenotype_2015$`Anthesis
↪  [MM/DD/YY]`-inbred_phenotype_2015$`Date Plot Planted [MM/DD/YY]`

hybrid_phenotype_2015$`Silking [days]` = hybrid_phenotype_2015$`Silking
↪  [MM/DD/YY]`-hybrid_phenotype_2015$`Date Plot Planted [MM/DD/YY]`
hybrid_phenotype_2015$`Anthesis [days]` = hybrid_phenotype_2015$`Anthesis
↪  [MM/DD/YY]`-hybrid_phenotype_2015$`Date Plot Planted [MM/DD/YY]`
```

```
phenotype_2016$`Silking [days]` = phenotype_2016$`Silking
↪  [MM/DD/YY]`-phenotype_2016$`Date Plot Planted [MM/DD/YY]`
phenotype_2016$`Anthesis [days]` = phenotype_2016$`Anthesis
↪  [MM/DD/YY]`-phenotype_2016$`Date Plot Planted [MM/DD/YY]`

phenotype_2017$`Silking [days]` = phenotype_2017$`Silking
↪  [MM/DD/YY]`-phenotype_2017$`Date Plot Planted [MM/DD/YY]`
phenotype_2017$`Anthesis [days]` = phenotype_2017$`Anthesis
↪  [MM/DD/YY]`-phenotype_2017$`Date Plot Planted [MM/DD/YY]`

phenotype_2018$`Silking [days]` = phenotype_2018$`Silking
↪  [MM/DD/YY]`-phenotype_2018$`Date Plot Planted [MM/DD/YY]`
phenotype_2018$`Anthesis [days]` = phenotype_2018$`Anthesis
↪  [MM/DD/YY]`-phenotype_2018$`Date Plot Planted [MM/DD/YY]`

phenotype_2019$`Silking [days]` = phenotype_2019$`Silking
↪  [MM/DD/YY]`-phenotype_2019$`Date Plot Planted [MM/DD/YY]`
phenotype_2019$`Anthesis [days]` = phenotype_2019$`Anthesis
↪  [MM/DD/YY]`-phenotype_2019$`Date Plot Planted [MM/DD/YY]`

phenotype_2020$`Silking [days]` = phenotype_2020$`Silking
↪  [MM/DD/YY]`-phenotype_2020$`Date Plot Planted [MM/DD/YY]`
phenotype_2020$`Anthesis [days]` = phenotype_2020$`Anthesis
↪  [MM/DD/YY]`-phenotype_2020$`Date Plot Planted [MM/DD/YY]`
```

```
phenotype_2021$`Silking [days]` = phenotype_2021$`Silking
↪  [MM/DD/YY]`-phenotype_2021$`Date Plot Planted [MM/DD/YY]`
phenotype_2021$`Anthesis [days]` = phenotype_2021$`Anthesis
↪  [MM/DD/YY]`-phenotype_2021$`Date Plot Planted [MM/DD/YY]`

phenotype_2022$`Silking [days]` = phenotype_2022$`Silking
↪  [MM/DD/YY]`-phenotype_2022$`Date Plot Planted [MM/DD/YY]`
phenotype_2022$`Anthesis [days]` = phenotype_2022$`Anthesis
↪  [MM/DD/YY]`-phenotype_2022$`Date Plot Planted [MM/DD/YY]`


# Weather date formatting
weather_2014$Date_key <- paste(weather_2014$Month, weather_2014$Day,
↪  weather_2014$Year, sep = "-")
weather_2014$Date_key <- mdy(weather_2014$Date_key)

weather_2015$Date_key <- paste(weather_2015$Month, weather_2015$Day,
↪  weather_2015$Year, sep = "-")
weather_2015$Date_key <- mdy(weather_2015$Date_key)

weather_2016$Date_key <- paste(weather_2016$Month, weather_2016$Day,
↪  weather_2016$Year, sep = "-")
weather_2016$Date_key <- mdy(weather_2016$Date_key)

weather_2017$Date_key <- paste(weather_2017$Month, weather_2017$Day,
↪  weather_2017$Year, sep = "-")
```

```r
weather_2017$Date_key <- mdy(weather_2017$Date_key)

weather_2018$Date_key <- paste(weather_2018$Month, weather_2018$Day,
↪  weather_2018$Year, sep = "-")
weather_2018$Date_key <- mdy(weather_2018$Date_key)

weather_2019$Date_key <- paste(weather_2019$Month, weather_2019$Day,
↪  weather_2019$Year, sep = "-")
weather_2019$Date_key <- mdy(weather_2019$Date_key)

weather_2020$Date_key <- paste(weather_2020$Month, weather_2020$Day,
↪  weather_2020$Year, sep = "-")
weather_2020$Date_key <- mdy(weather_2020$Date_key)

weather_2020 = weather_2020[-is.na(weather_2020$Date_key),]

weather_2021$Date_key <- paste(weather_2021$Month, weather_2021$Day,
↪  weather_2021$Year, sep = "-")
weather_2021$Date_key <- mdy(weather_2021$Date_key)

weather_2022$Date_key <- paste(weather_2022$Month, weather_2022$Day,
↪  weather_2022$Year, sep = "-")
weather_2022$Date_key <- mdy(weather_2022$Date_key)


# Field location correction and matching
```

```
phenotype_2017$`Field Location`[phenotype_2017$`Field
↪ Location`=="TXH1-Dry"]<-"TXH1"
phenotype_2017$`Field Location`[phenotype_2017$`Field
↪ Location`=="TXH1-Early"]<-"TXH1"
phenotype_2017$`Field Location`[phenotype_2017$`Field
↪ Location`=="TXH1-Late"]<-"TXH1"
phenotype_2018$`Field Location`[phenotype_2018$`Field
↪ Location`=="TXH1-Dry"]<-"TXH1"
phenotype_2018$`Field Location`[phenotype_2018$`Field
↪ Location`=="TXH1-Early"]<-"TXH1"
phenotype_2018$`Field Location`[phenotype_2018$`Field
↪ Location`=="TXH1-Late"]<-"TXH1"

field_2018$`Field Location`[field_2018$`Field Location`=="TXH1-
↪ Dry"]<-"TXH1"
field_2018$`Field Location`[field_2018$`Field Location`=="TXH1-
↪ Early"]<-"TXH1"
field_2018$`Field Location`[field_2018$`Field Location`=="TXH1-
↪ Late"]<-"TXH1"
field_2018$`Field Location`[field_2018$`Field Location`=="MOH1- rep
↪ 2"]<-"MOH1"
field_2018$`Field Location`[field_2018$`Field Location`=="MOH1- rep
↪ 1"]<-"MOH1"
```

```
soil_2017$`Field Location`[soil_2017$`Field Location`=="NEH3
↪  (IRRIGATED)"]<-"NEH3"
soil_2017$`Field Location`[soil_2017$`Field Location`=="NEH4 (NON
↪  IRRIGATED)"]<-"NEH4"
soil_2017$`Field Location`[soil_2017$`Field Location`=="IA(H4)"]<-"IAH4"

soil_2015$`Field Location`[soil_2015$`Field Location`=="IA(?)2"]<-"IAH2"
soil_2015$`Field Location`[soil_2015$`Field Location`=="IA(?)3"]<-"IAH3"
soil_2015$`Field Location`[soil_2015$`Field Location`=="IA(H4)"]<-"IAH4"
soil_2015[27,5]<-"MNH1"
soil_2015[28,5]<-"MNI1"

inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_DE1",]$`Field Location` = "DEI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_GA2",]$`Field Location` = "GAI2"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_IA1",]$`Field Location` = "IAI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_IA2",]$`Field Location` = "IAI2"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_IA3",]$`Field Location` = "IAI3"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_IL1",]$`Field Location` = "ILI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_IN1",]$`Field Location` = "INI1"
```

```
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_MN2",]$`Field Location` = "MNI2"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_MO1",]$`Field Location` = "MOI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_MO2",]$`Field Location` = "MOI2"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_MO3",]$`Field Location` = "MOI3"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_NC1",]$`Field Location` = "NCI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_NE1",]$`Field Location` = "NEI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_NY1",]$`Field Location` = "NYI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_PA1",]$`Field Location` = "PAI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_TX1",]$`Field Location` = "TXI1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_TX2",]$`Field Location` = "TXI2"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_WI1_MAD",]$`Field Location` = "WII1"
inbred_phenotype_2014[inbred_phenotype_2014$`Field Location` ==
↪  "GXE_inb_WI2_ARL",]$`Field Location` = "WII2"
hybrid_phenotype_2014[hybrid_phenotype_2014$`Field Location` ==
↪  "IAH1a",]$`Field Location` = "IAH1"
```

```
hybrid_phenotype_2014[hybrid_phenotype_2014$`Field Location` ==
↪  "IAH1b",]$`Field Location` = "IAH1"
hybrid_phenotype_2014[hybrid_phenotype_2014$`Field Location` ==
↪  "IAH1c",]$`Field Location` = "IAH1"

field_2014[24, 3] = "GAI2"
field_2014[40, 3] = "TXI3"
field_2014

field_2014[field_2014$`Field Location` == "G2FWI-HYB",]$`Field Location` =
↪  "WIH1"
field_2014[field_2014$`Field Location` == "G2FDE1",]$`Field Location` =
↪  "DEI1"
field_2014[field_2014$`Field Location` == "GXE_inb_IA1",]$`Field Location` =
↪  "IAI1"
field_2014[field_2014$`Field Location` == "GXE_inb_IA2",]$`Field Location` =
↪  "IAI2"
field_2014[field_2014$`Field Location` == "G2FIA3",]$`Field Location` =
↪  "IAI3"
field_2014[field_2014$`Field Location` == "G2FIL1",]$`Field Location` =
↪  "ILI1"
field_2014[field_2014$`Field Location` == "G2FIN1",]$`Field Location` =
↪  "INI1"
field_2014[field_2014$`Field Location` == "G2FMN2",]$`Field Location` =
↪  "MNI2"
```

```
field_2014[field_2014$`Field Location` == "GXE_inb_MO1",]$`Field Location` =
↪  "MOI1"
field_2014[field_2014$`Field Location` == "GXE_inb_BO2",]$`Field Location` =
↪  "BOI2"
field_2014[field_2014$`Field Location` == "GXE_inb_MO3",]$`Field Location` =
↪  "MOI3"
field_2014[field_2014$`Field Location` == "NC1",]$`Field Location` = "NCI1"
field_2014[field_2014$`Field Location` == "G2FNE1",]$`Field Location` =
↪  "NEI1"
field_2014[field_2014$`Field Location` == "G2FNY1",]$`Field Location` =
↪  "NYI1"
field_2014[field_2014$`Field Location` == "GxE_inb_PA1",]$`Field Location` =
↪  "PAI1"
field_2014[field_2014$`Field Location` == "G2F_IN_TX1",]$`Field Location` =
↪  "TXI1"
field_2014[field_2014$`Field Location` == "G2FWI1",]$`Field Location` =
↪  "WII1"
field_2014[field_2014$`Field Location` == "G2FWI2",]$`Field Location` =
↪  "WII2"
unique(field_2014$`Field Location`)


# Creating 2014 and 2015 agronomic datasets from field datasets


agronomic_2015 = rbind(
```

```
  data.frame(`Field Location` = field_2015$`Field Location`,
    ↪ `Application_or_treatment` = "pre-plant herbicide",
    ↪ `Product_or_nutrient_applied` =field_2015$`preplant herb`),
  data.frame(`Field Location` = field_2015$`Field Location`,
    ↪ `Application_or_treatment` = "post-plant herbicide",
    ↪ `Product_or_nutrient_applied` =field_2015$`postplant herb`),
  data.frame(`Field Location` = field_2015$`Field Location`,
    ↪ `Application_or_treatment` = "fertilize",
    ↪ `Product_or_nutrient_applied` =field_2015$`Type of Fert`),
  data.frame(`Field Location` = field_2015$`Field Location`,
    ↪ `Application_or_treatment` = "insecticide",
    ↪ `Product_or_nutrient_applied` =field_2015$`Insecticide`)
)
names(agronomic_2015)[1] = 'Field Location'
agronomic_2015 = agronomic_2015[complete.cases(agronomic_2015),]

agronomic_2014 = rbind(
  data.frame(`Field Location` = field_2014$`Field Location`,
    ↪ `Application_or_treatment` = "insecticide",
    ↪ `Product_or_nutrient_applied` = field_2014$Insecticide),
  data.frame(`Field Location` = field_2014$`Field Location`,
    ↪ `Application_or_treatment` = "pre-plant herbicide",
    ↪ `Product_or_nutrient_applied` =field_2014$`Pre-plant herbicides`),
  data.frame(`Field Location` = field_2014$`Field Location`,
    ↪ `Application_or_treatment` = "post-plant herbicide",
    ↪ `Product_or_nutrient_applied` =field_2014$`Post-plant herbicides`),
```

```
  data.frame(`Field Location` = field_2014$`Field Location`,
  ↪  `Application_or_treatment` = "irrigation",
  ↪  `Product_or_nutrient_applied` =field_2014$`Irrigated?`)
)
names(agronomic_2014)[1] = 'Field Location'
agronomic_2014 = agronomic_2014[c(1:128, 136, 138, 146, 147, 149, 150, 157,
↪  160, 164, 166),]
agronomic_2014[127:138,3] = "water"
agronomic_2014 = agronomic_2014[complete.cases(agronomic_2014),]

# Field location formatting for weather data
split_field_locations <- function(df) {
  expanded_df <- do.call(
    rbind, lapply(
      1:nrow(df), function(i) {
        locations <- unlist(strsplit(as.character(df[i, 1]), "[ _]+")) #
        ↪  breaking field locations into parts based on _
        new_rows <- data.frame(Location = locations, df[i, -1],
        ↪  stringsAsFactors = FALSE) # creating new rows
        return(new_rows)
      }
    )
  )
  colnames(expanded_df) <- colnames(df)
  return(expanded_df)
}
```

```r
# Creating individual datasets per variable for 2022 weather data
measurements <- colnames(weather_2022[10:22])

for (measure in measurements) {
  table_name <- paste("weather_2022_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2022 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>% # making weeks into columns instead of rows
           {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
           ↪  6)))]} %>% # ordering by week
           {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} # adding field location
           ↪  names
         %>% split_field_locations())

  print(paste("Created table:", table_name))
```

```
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2021 weather data
measurements <- colnames(weather_2021[10:22])

for (measure in measurements) {
  table_name <- paste("weather_2021_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2021 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>%
           {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
           ↪  6)))]} %>%
           {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
           ↪  split_field_locations())  # Reorder columns
  print(paste("Created table:", table_name))
```

```
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2020 weather data
weather_2020 = weather_2020[!is.na(weather_2020$`Field Location`),]
measurements <- colnames(weather_2020[10:23])

for (measure in measurements) {
  table_name <- paste("weather_2020_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2020 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>%
           {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
           ↪  6)))]} %>%
           {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
           ↪  split_field_locations())  # Reorder columns
```

```
  print(paste("Created table:", table_name))
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2019 weather data
measurements <- colnames(weather_2019[10:23])

for (measure in measurements) {
  table_name <- paste("weather_2019_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2019 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>%
           {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
           ↪  6)))]} %>%
```

```
            {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
            ↪  split_field_locations())  # Reorder columns
  print(paste("Created table:", table_name))
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2018 weather data
measurements <- colnames(weather_2018[10:24])

for (measure in measurements) {
  table_name <- paste("weather_2018_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2018 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>%
           {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
           ↪  6)))]} %>%
```

```
            {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
            ↪  split_field_locations())  # Reorder columns
  print(paste("Created table:", table_name))
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2017 weather data
measurements <- colnames(weather_2017[9:22])

for (measure in measurements) {
  table_name <- paste("weather_2017_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2017 %>%
            mutate(week = week(Date_key)) %>%
            group_by(`Field Location`, week) %>%
            summarise(
              Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
              SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
              Min = min(as.numeric(get(measure)), na.rm = TRUE),
              Max = max(as.numeric(get(measure)), na.rm = TRUE)
            ) %>%
            ungroup() %>%
            pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
            ↪  Max)) %>%
            {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
            ↪  6)))]} %>%
```

```
            {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
            ↪  split_field_locations())  # Reorder columns
  print(paste("Created table:", table_name))
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2016 weather data
measurements <- colnames(weather_2016[9:22])

for (measure in measurements) {
  table_name <- paste("weather_2016_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2016 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>%
           {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
           ↪  6)))]} %>%
```

```
            {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
            ↪  split_field_locations())  # Reorder columns

  print(paste("Created table:", table_name))
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2015 weather data
measurements <- colnames(weather_2015[9:22])

for (measure in measurements) {
  table_name <- paste("weather_2015_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2015 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>%
```

```
            {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
            ↪  6)))]} %>%
            {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
            ↪  split_field_locations())  # Reorder columns
  print(paste("Created table:", table_name))
  print(head(get(table_name)))
}

# Creating individual datasets per variable for 2014 weather data
measurements <- colnames(weather_2014[12:20])

for (measure in measurements) {
  table_name <- paste("weather_2014_", tolower(gsub(" ", "_", measure)), sep
  ↪  = "")
  assign(table_name, weather_2014 %>%
           mutate(week = week(Date_key)) %>%
           group_by(`Field Location`, week) %>%
           summarise(
             Avg = mean(as.numeric(get(measure)), na.rm = TRUE),
             SDv = sd(as.numeric(get(measure)), na.rm = TRUE),
             Min = min(as.numeric(get(measure)), na.rm = TRUE),
             Max = max(as.numeric(get(measure)), na.rm = TRUE)
           ) %>%
           ungroup() %>%
           pivot_wider(names_from = week, values_from = c(Avg, SDv, Min,
           ↪  Max)) %>%
```

```
            {.[, order(as.numeric(substr(colnames(.), start = 5, stop =
            ↪  6)))]} %>%
            {cbind(.[ncol(.)], .[1:(ncol(.) - 1)])} %>%
            ↪  split_field_locations())  # Reorder columns
  print(paste("Created table:", table_name))
  print(head(get(table_name)))
}

# Cumulative sum for 2022 rainfall
weather_2022_sum_rainfall =
  weather_2022 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%

  group_by(`Field Location`) %>% # grouping so cumsum is based on location
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)

weather_2022_sum_rainfall = pivot_wider(weather_2022_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)

# Ordering 2022 Rainfall
```

```
weather_2022_sum_rainfall = weather_2022_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2022_sum_rainfall)))
weather_2022_sum_rainfall =
  cbind(
    weather_2022_sum_rainfall[ncol(weather_2022_sum_rainfall)],
    weather_2022_sum_rainfall[1:ncol(weather_2022_sum_rainfall)-1]
  )
weather_2022_sum_rainfall = split_field_locations(weather_2022_sum_rainfall)

# Cumulative sum for 2021 rainfall
weather_2021_sum_rainfall =
  weather_2021 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2021_sum_rainfall = pivot_wider(weather_2021_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)

# Ordering 2021 Rainfall
weather_2021_sum_rainfall = weather_2021_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2021_sum_rainfall)))
```

```r
weather_2021_sum_rainfall =
  cbind(
    weather_2021_sum_rainfall[ncol(weather_2021_sum_rainfall)],
    weather_2021_sum_rainfall[1:ncol(weather_2021_sum_rainfall)-1]
  )

weather_2021_sum_rainfall = split_field_locations(weather_2021_sum_rainfall)

# Cumulative sum for 2020 rainfall
weather_2020_sum_rainfall =
  weather_2020 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2020_sum_rainfall = pivot_wider(weather_2020_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)

# Ordering 2020 Rainfall
weather_2020_sum_rainfall = weather_2020_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2020_sum_rainfall)))
weather_2020_sum_rainfall =
```

```
  cbind(
    weather_2020_sum_rainfall[ncol(weather_2020_sum_rainfall)],
    weather_2020_sum_rainfall[1:ncol(weather_2020_sum_rainfall)-1]
  )

weather_2020_sum_rainfall = split_field_locations(weather_2020_sum_rainfall)

# Cumulative sum for 2019 rainfall
weather_2019_sum_rainfall =
  weather_2019 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2019_sum_rainfall = pivot_wider(weather_2019_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)


# Ordering 2019 Rainfall
weather_2019_sum_rainfall = weather_2019_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2019_sum_rainfall)))
weather_2019_sum_rainfall =
```

```
  cbind(
    weather_2019_sum_rainfall[ncol(weather_2019_sum_rainfall)],
    weather_2019_sum_rainfall[1:ncol(weather_2019_sum_rainfall)-1]
  )

weather_2019_sum_rainfall = split_field_locations(weather_2019_sum_rainfall)

# Cumulative sum for 2018 rainfall
weather_2018_sum_rainfall =
  weather_2018 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2018_sum_rainfall = pivot_wider(weather_2018_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)


# Ordering 2018 Rainfall
weather_2018_sum_rainfall = weather_2018_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2018_sum_rainfall)))
weather_2018_sum_rainfall =
```

```
  cbind(
    weather_2018_sum_rainfall[ncol(weather_2018_sum_rainfall)],
    weather_2018_sum_rainfall[1:ncol(weather_2018_sum_rainfall)-1]
  )

weather_2018_sum_rainfall = split_field_locations(weather_2018_sum_rainfall)

# Cumulative sum for 2017 rainfall
weather_2017_sum_rainfall =
  weather_2017 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2017_sum_rainfall = pivot_wider(weather_2017_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)


# Ordering 2017 Rainfall
weather_2017_sum_rainfall = weather_2017_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2017_sum_rainfall)))
weather_2017_sum_rainfall =
```

```
  cbind(
    weather_2017_sum_rainfall[ncol(weather_2017_sum_rainfall)],
    weather_2017_sum_rainfall[1:ncol(weather_2017_sum_rainfall)-1]
  )

weather_2017_sum_rainfall = split_field_locations(weather_2017_sum_rainfall)

# Cumulative sum for 2016 rainfall
weather_2016_sum_rainfall =
  weather_2016 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2016_sum_rainfall = pivot_wider(weather_2016_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)


# Ordering 2016 Rainfall
weather_2016_sum_rainfall = weather_2016_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2016_sum_rainfall)))
weather_2016_sum_rainfall =
```

```
  cbind(
    weather_2016_sum_rainfall[ncol(weather_2016_sum_rainfall)],
    weather_2016_sum_rainfall[1:ncol(weather_2016_sum_rainfall)-1]
  )

weather_2016_sum_rainfall = split_field_locations(weather_2016_sum_rainfall)

# Cumulative sum for 2015 rainfall
weather_2015_sum_rainfall =
  weather_2015 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2015_sum_rainfall = pivot_wider(weather_2015_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)


# Ordering 2015 Rainfall
weather_2015_sum_rainfall = weather_2015_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2015_sum_rainfall)))
weather_2015_sum_rainfall =
```

```
  cbind(
    weather_2015_sum_rainfall[ncol(weather_2015_sum_rainfall)],
    weather_2015_sum_rainfall[1:ncol(weather_2015_sum_rainfall)-1]
  )

weather_2015_sum_rainfall = split_field_locations(weather_2015_sum_rainfall)

# Cumulative sum for 2014 rainfall
weather_2014_sum_rainfall =
  weather_2014 %>%
  mutate(week = week(Date_key)) %>%
  group_by(`Field Location`, week) %>%
  summarise(Weekly_Rainfall = sum(as.numeric(`Rainfall [mm]`), na.rm =
  ↪  TRUE), .groups = "drop") %>%
  arrange(`Field Location`, week) %>%
  group_by(`Field Location`) %>%
  mutate(Cumulative_Rainfall = cumsum(Weekly_Rainfall)) %>%
  select(`Field Location`, week, Cumulative_Rainfall)
weather_2014_sum_rainfall = pivot_wider(weather_2014_sum_rainfall,
↪  names_from = week, values_from = Cumulative_Rainfall)


# Ordering 2014 Rainfall
weather_2014_sum_rainfall = weather_2014_sum_rainfall %>%
↪  dplyr::select(order(names(weather_2014_sum_rainfall)))
weather_2014_sum_rainfall =
```

```
  cbind(
    weather_2014_sum_rainfall[ncol(weather_2014_sum_rainfall)],
    weather_2014_sum_rainfall[1:ncol(weather_2014_sum_rainfall)-1]
  )

weather_2014_sum_rainfall = split_field_locations(weather_2014_sum_rainfall)

# Extracting phenotype columns with substantial presence across years
inbred_phenotype_final_2014 = cbind(`Year` = 2014, inbred_phenotype_2014 %>%
↪  select(`Field Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height
↪  [cm]`, `Stand Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk
↪  Lodging [# of plants]`),`Silking [days]` = NA, `Anthesis [days]` = NA,
↪  `Grain Moisture [%]` = NA, `Plot Weight [lbs]` = NA, `Grain Yield
↪  (bu/A)`= NA, `Test Weight [lbs]` = NA)

hybrid_phenotype_final_2014 = cbind(`Year` = 2014, hybrid_phenotype_2014 %>%
↪  select(`Field Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height
↪  [cm]`, `Stand Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk
↪  Lodging [# of plants]`, `Silking [days]`, `Anthesis [days]`, `Grain
↪  Moisture [%]`, `Plot Weight [lbs]`, `Grain Yield (bu/A)`, `Test Weight
↪  [lbs]`))
```

```
inbred_phenotype_final_2015 = cbind(`Year` = 2015, inbred_phenotype_2015 %>%
↪  select(`Field Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height
↪  [cm]`, `Stand Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk
↪  Lodging [# of plants]`, `Silking [days]`, `Anthesis [days]`), `Grain
↪  Moisture [%]` = NA, `Plot Weight [lbs]` = NA, `Grain Yield (bu/A)`= NA,
↪  `Test Weight [lbs]` = NA)

hybrid_phenotype_final_2015 = cbind(`Year` = 2015, hybrid_phenotype_2015 %>%
↪  select(`Field Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height
↪  [cm]`,  `Stand Count [# of plants]`,`Root Lodging [# of plants]`,
↪  `Stalk Lodging [# of plants]`, `Silking [days]`, `Anthesis [days]`,
↪  `Grain Moisture [%]`, `Plot Weight [lbs]`,`Grain Yield (bu/A)`, `Test
↪  Weight [lbs]`))

phenotype_final_2016 = cbind(`Year` = 2016, phenotype_2016 %>% select(`Field
↪  Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height [cm]`,m`Stand
↪  Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk Lodging [# of
↪  plants]`, `Silking [days]`,
`Anthesis [days]`, `Grain Moisture [%]`, `Plot Weight [lbs]`, `Grain Yield
↪  (bu/A)`. `Test Weight [lbs]`))

phenotype_final_2017 = cbind(`Year` = 2017, phenotype_2017 %>% select(`Field
↪  Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height [cm]`, `Stand
↪  Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk Lodging [# of
↪  plants]`, `Silking [days]`, `Anthesis [days]`, `Grain Moisture [%]`,
↪  `Plot Weight [lbs]`, `Grain Yield (bu/A)`, `Test Weight [lbs]`))
```

```
phenotype_final_2018 = cbind(`Year` = 2018, phenotype_2018 %>% select(`Field
↪  Location`, `Pedigree`, `Plant Height [cm]`,
`Ear Height [cm]`, `Stand Count [# of plants]`, `Root Lodging [# of
↪  plants]`, `Stalk Lodging [# of plants]`, `Silking [days]`, `Anthesis
↪  [days]`, `Grain Moisture [%]`, `Plot Weight [lbs]`, `Grain Yield
↪  (bu/A)`, `Test Weight [lbs]`))

phenotype_final_2019 = cbind(`Year` = 2019, phenotype_2019 %>% select(`Field
↪  Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height [cm]`,  `Stand
↪  Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk Lodging [# of
↪  plants]`, `Silking [days]`,  `Anthesis [days]`, `Grain Moisture [%]`,
↪  `Plot Weight [lbs]`,  `Grain Yield (bu/A)`, `Test Weight [lbs]`))

phenotype_final_2020 = cbind(`Year` = 2020, phenotype_2020 %>% select(`Field
↪  Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height [cm]`, `Stand
↪  Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk Lodging [# of
↪  plants]`, `Silking [days]`, `Anthesis [days]`, `Grain Moisture [%]`,
↪  `Plot Weight [lbs]`, `Grain Yield (bu/A)`, `Test Weight [lbs]`))

phenotype_final_2021 = cbind(`Year` = 2021, phenotype_2021 %>% select(`Field
↪  Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height [cm]`, `Stand
↪  Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk Lodging [# of
↪  plants]`, `Silking [days]`, `Anthesis [days]`, `Grain Moisture [%]`,
↪  `Plot Weight [lbs]`, `Grain Yield (bu/A)`, `Test Weight [lbs]`))
```

```
phenotype_final_2022 = cbind(`Year` = 2022, phenotype_2022 %>% select(`Field
↪  Location`, `Pedigree`, `Plant Height [cm]`, `Ear Height [cm]`, `Stand
↪  Count [# of plants]`, `Root Lodging [# of plants]`, `Stalk Lodging [# of
↪  plants]`, `Silking [days]`, `Anthesis [days]`, `Grain Moisture [%]`,
↪  `Plot Weight [lbs]`, `Grain Yield (bu/A)`, `Test Weight [lbs]`))

# Extracting soil columns with substantial presence across years

soil_final_2015 = cbind(Year = 2015, soil_2015 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`), `1:1 S Salts mmho/cm`
↪  = NA, `Texture No` = NA, `Nitrate-N ppm N` = NA, `lbs N/A` = NA,
↪  `Sulfate-S ppm S` = NA, `Calcium ppm Ca` = NA, `Magnesium ppm Mg` = NA,
↪  `Sodium ppm Na` = NA,`CEC/Sum of Cations me/100g` = NA, `%H Sat` = NA,
↪  `%K Sat` = NA, `%Ca Sat` = NA, `%Mg Sat` = NA, `%Na Sat` = NA, `% Sand`
↪  = NA, `% Silt` = NA, `% Clay` = NA)

soil_final_2016 = cbind(Year = 2016, soil_2016 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`, `1:1 S Salts mmho/cm`,
↪  `Texture No`, `Nitrate-N ppm N`, `lbs N/A`, `Sulfate-S ppm S`, `Calcium
↪  ppm Ca`, `Magnesium ppm Mg`, `Sodium ppm Na`, `CEC/Sum of Cations
↪  me/100g`, `%H Sat`, `%K Sat`, `%Ca Sat`, `%Mg Sat`, `%Na Sat`, `% Sand`,
↪  `% Silt`, `% Clay`))
```

```
soil_final_2017 = cbind(Year = 2017, soil_2017 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`, `1:1 S Salts mmho/cm`,
↪  `Texture No`, `Nitrate-N ppm N`, `lbs N/A`, `Sulfate-S ppm S`, `Calcium
↪  ppm Ca`, `Magnesium ppm Mg`, `Sodium ppm Na`, `CEC/Sum of Cations
↪  me/100g`, `%H Sat`, `%K Sat`, `%Ca Sat`, `%Mg Sat`, `%Na Sat`, `% Sand`,
↪  `% Silt`, `% Clay`))

soil_final_2018 = cbind(Year = 2018, soil_2018 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`, `1:1 S Salts mmho/cm`,
↪  `Texture No`, `Nitrate-N ppm N`, `lbs N/A`, `Sulfate-S ppm S`, `Calcium
↪  ppm Ca`, `Magnesium ppm Mg`, `Sodium ppm Na`, `CEC/Sum of Cations
↪  me/100g`, `%H Sat`, `%K Sat`, `%Ca Sat`, `%Mg Sat`, `%Na Sat`, `% Sand`,
↪  `% Silt`, `% Clay`))

soil_final_2019 = cbind(Year = 2019, soil_2019 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`, `1:1 S Salts mmho/cm`,
↪  `Texture No`, `Nitrate-N ppm N`, `lbs N/A`, `Sulfate-S ppm S`, `Calcium
↪  ppm Ca`, `Magnesium ppm Mg`, `Sodium ppm Na`, `CEC/Sum of Cations
↪  me/100g`, `%H Sat`, `%K Sat`, `%Ca Sat`, `%Mg Sat`, `%Na Sat`, `% Sand`,
↪  `% Silt`, `% Clay`))
```

```
soil_final_2020 = cbind(Year = 2020, soil_2020 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`, `1:1 S Salts mmho/cm`,
↪  `Texture No`, `Nitrate-N ppm N`, `lbs N/A`, `Sulfate-S ppm S`, `Calcium
↪  ppm Ca`, `Magnesium ppm Mg`, `Sodium ppm Na`, `CEC/Sum of Cations
↪  me/100g`, `%H Sat`, `%K Sat`, `%Ca Sat`, `%Mg Sat`, `%Na Sat`, `% Sand`,
↪  `% Silt`, `% Clay`))

soil_final_2021 = cbind(Year = 2021, soil_2021 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`, `1:1 S Salts mmho/cm`,
↪  `Texture No`, `Nitrate-N ppm N`, `lbs N/A`, `Sulfate-S ppm S`, `Calcium
↪  ppm Ca`, `Magnesium ppm Mg`, `Sodium ppm Na`, `CEC/Sum of Cations
↪  me/100g`, `%H Sat`, `%K Sat`, `%Ca Sat`, `%Mg Sat`, `%Na Sat`, `%
↪  Sand`,`% Silt`, `% Clay`))

soil_final_2022 = cbind(Year = 2022, soil_2022 %>% select(`Field
↪  Location`,`E Depth`, `1:1 Soil pH`, `WDRF Buffer pH`, `Organic Matter
↪  LOI %`, `Potassium ppm K`, `Mehlich P-III ppm P`, `1:1 S Salts mmho/cm`,
↪  `Texture No`, `Nitrate-N ppm N`, `lbs N/A`, `Sulfate-S ppm S`, `Calcium
↪  ppm Ca`, `Magnesium ppm Mg`, `Sodium ppm Na`, `CEC/Sum of Cations
↪  me/100g`, `%H Sat`, `%K Sat`, `%Ca Sat`, `%Mg Sat`, `%Na Sat`,`% Sand`,
↪  `% Silt`, `% Clay`))

# Extracting field columns with substantial presence across years
```

```
field_final_2015 <- cbind(Year = 2015, field_2015 %>% select(`Field
↪  Location`, `Treatment`), `Previous_Crop` = NA,
↪  `Pre-plant_tillage_method(s)` = NA, `In-season_tillage_method(s)` = NA)

field_final_2016 <- cbind(Year = 2016, field_2016 %>% select(`Field
↪  Location`, `Treatment`, `Previous_Crop`, `Pre-plant_tillage_method(s)`,
↪  `In-season_tillage_method(s)`))

field_final_2017 <- cbind(Year = 2017, field_2017 %>% select(`Field
↪  Location`, `Treatment`, `Previous_Crop`, `Pre-plant_tillage_method(s)`,
↪  `In-season_tillage_method(s)`))

field_final_2018 <- cbind(Year = 2018, field_2018 %>% select(`Field
↪  Location`, `Treatment`, `Previous_Crop`, `Pre-plant_tillage_method(s)`,
↪  `In-season_tillage_method(s)`))

field_final_2019 <- cbind(Year = 2019, field_2019 %>% select(`Field
↪  Location`, `Treatment`, `Previous_Crop`, `Pre-plant_tillage_method(s)`,
↪  `In-season_tillage_method(s)`))

field_final_2020 <- cbind(Year = 2020, field_2020 %>% select(`Field
↪  Location`, `Treatment`, `Previous_Crop`, `Pre-plant_tillage_method(s)`,
↪  `In-season_tillage_method(s)`))
```

```
field_final_2021 <- cbind(Year = 2021, field_2021 %>% select(`Field
↪  Location`, `Treatment`, `Previous_Crop`, `Pre-plant_tillage_method(s)`,
↪  `In-season_tillage_method(s)`))

field_final_2022 <- cbind(Year = 2022, field_2022 %>% select(`Field
↪  Location`, `Treatment`, `Previous_Crop`, `Pre-plant_tillage_method(s)`,
↪  `In-season_tillage_method(s)`))

# Extracting agronomic columns with substantial presence across years
agronomic_final_2014 <- cbind(Year = 2014, agronomic_2014 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`),
↪  `Quantity_per_acre` = NA, `Application_unit` = NA)

agronomic_final_2015 <- cbind(Year = 2015, agronomic_2015 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`),
↪  `Quantity_per_acre` = NA, `Application_unit` = NA)

agronomic_final_2016 <- cbind(Year = 2016, agronomic_2016 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`,
↪  `Quantity_per_acre`, `Application_unit`))

agronomic_final_2017 <- cbind(Year = 2017, agronomic_2017 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`,
↪  `Quantity_per_acre`, `Application_unit`))
```

```
agronomic_final_2018 <- cbind(Year = 2018, agronomic_2018 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`,
↪  `Quantity_per_acre`, `Application_unit`))

agronomic_final_2019 <- cbind(Year = 2019, agronomic_2019 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`,
↪  `Quantity_per_acre`, `Application_unit`))

agronomic_final_2020 <- cbind(Year = 2020, agronomic_2020 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`,
↪  `Quantity_per_acre`, `Application_unit`))

agronomic_final_2021 <- cbind(Year = 2021, agronomic_2021 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`,
↪  `Quantity_per_acre`, `Application_unit`))

agronomic_final_2022 <- cbind(Year = 2022, agronomic_2022 %>% select(`Field
↪  Location`, `Application_or_treatment`, `Product_or_nutrient_applied`,
↪  `Quantity_per_acre`, `Application_unit`))

# Combining common dataset types across years
phenotype_all = rbind(inbred_phenotype_final_2014,
↪  hybrid_phenotype_final_2014, inbred_phenotype_final_2015,
↪  hybrid_phenotype_final_2015, phenotype_final_2016, phenotype_final_2017,
↪  phenotype_final_2018, phenotype_final_2019, phenotype_final_2020,
↪  phenotype_final_2021, phenotype_final_2022
```

```
)

soil_all = rbind(soil_final_2015, soil_final_2016, soil_final_2017,
↪  soil_final_2018, soil_final_2019,  soil_final_2020, soil_final_2021,
↪  soil_final_2022)

field_all = rbind(field_final_2015, field_final_2016, field_final_2017,
↪  field_final_2018, field_final_2019,  field_final_2020, field_final_2021,
↪  field_final_2022)


agronomic_all = rbind(agronomic_final_2014, agronomic_final_2015,
↪  agronomic_final_2016, agronomic_final_2017, agronomic_final_2018,
↪  agronomic_final_2019, agronomic_final_2020, agronomic_final_2021,
↪  agronomic_final_2022)

temperature_all = list(`weather_2014_temperature_[c]`,
↪  `weather_2015_temperature_[c]`, `weather_2016_temperature_[c]`,
↪  `weather_2017_temperature_[c]`, `weather_2018_temperature_[c]`,
↪  `weather_2019_temperature_[c]`, `weather_2020_temperature_[c]`,
↪  `weather_2021_temperature_[c]`, `weather_2022_temperature_[c]`)
```

```
dewpoint_all = list(`weather_2014_dew_point_[c]`,
 ↪  `weather_2015_dew_point_[c]`, `weather_2016_dew_point_[c]`,
 ↪  `weather_2017_dew_point_[c]`, `weather_2018_dew_point_[c]`,
 ↪  `weather_2019_dew_point_[c]`, `weather_2020_dew_point_[c]`,
 ↪  `weather_2021_dew_point_[c]`, `weather_2022_dew_point_[c]`)

humidity_all = list(`weather_2014_relative_humidity_[%]`,
 ↪  `weather_2015_relative_humidity_[%]`,
 ↪  `weather_2016_relative_humidity_[%]`,
 ↪  `weather_2017_relative_humidity_[%]`,
 ↪  `weather_2018_relative_humidity_[%]`,
 ↪  `weather_2019_relative_humidity_[%]`,
 ↪  `weather_2020_relative_humidity_[%]`,
 ↪  `weather_2021_relative_humidity_[%]`,
 ↪  `weather_2022_relative_humidity_[%]`)

solar_radiation_all = list(`weather_2014_solar_radiation_[w/m2]`,
 ↪  `weather_2015_solar_radiation_[w/m2]`,
 ↪  `weather_2016_solar_radiation_[w/m2]`,
 ↪  `weather_2017_solar_radiation_[w/m2]`,
 ↪  `weather_2018_solar_radiation_[w/m2]`,
 ↪  `weather_2019_solar_radiation_[w/m2]`,
 ↪  `weather_2020_solar_radiation_[w/m2]`,
 ↪  `weather_2021_solar_radiation_[w/m2]`,
 ↪  `weather_2022_solar_radiation_[w/m2]`)
```

```
rainfall_all = list(`weather_2014_sum_rainfall`,
↪  `weather_2015_sum_rainfall`, `weather_2016_sum_rainfall`,
↪  `weather_2017_sum_rainfall`, `weather_2018_sum_rainfall`,
↪  `weather_2019_sum_rainfall`, `weather_2020_sum_rainfall`,
↪  `weather_2021_sum_rainfall`, `weather_2022_sum_rainfall`)

windspeed_all = list(`weather_2014_wind_speed_[m/s]`,
↪  `weather_2015_wind_speed_[m/s]`, `weather_2016_wind_speed_[m/s]`,
↪  `weather_2017_wind_speed_[m/s]`, `weather_2018_wind_speed_[m/s]`,
↪  `weather_2019_wind_speed_[m/s]`, `weather_2020_wind_speed_[m/s]`,
↪  `weather_2021_wind_speed_[m/s]`, `weather_2022_wind_speed_[m/s]`)

winddir_all = list(`weather_2014_wind_direction_[degrees]`,
↪  `weather_2015_wind_direction_[degrees]`,
↪  `weather_2016_wind_direction_[degrees]`,
↪  `weather_2017_wind_direction_[degrees]`,
↪  `weather_2018_wind_direction_[degrees]`,
↪  `weather_2019_wind_direction_[degrees]`,
↪  `weather_2020_wind_direction_[degrees]`,
↪  `weather_2021_wind_direction_[degrees]`,
↪  `weather_2022_wind_direction_[degrees]`)
```

```
windgust_all = list(`weather_2014_wind_gust_[m/s]`,
↪  `weather_2015_wind_gust_[m/s]`, `weather_2016_wind_gust_[m/s]`,
↪  `weather_2017_wind_gust_[m/s]`, `weather_2018_wind_gust_[m/s]`,
↪  `weather_2019_wind_gust_[m/s]`, `weather_2020_wind_gust_[m/s]`,
↪  `weather_2021_wind_gust_[m/s]`, `weather_2022_wind_gust_[m/s]`)

soiltemp_all = list(`weather_2015_soil_temperature_[c]`,
↪  `weather_2016_soil_temperature_[c]`,
↪  `weather_2017_soil_temperature_[c]`,
↪  `weather_2018_soil_temperature_[c]`,
↪  `weather_2019_soil_temperature_[c]`,
↪  `weather_2020_soil_temperature_[c]`,
↪  `weather_2021_soil_temperature_[c]`,
↪  `weather_2022_soil_temperature_[c]`)

soilmois_all = list(`weather_2015_soil_moisture_[%vwc]`,
↪  `weather_2016_soil_moisture_[%vwc]`,
↪  `weather_2017_soil_moisture_[%vwc]`,
↪  `weather_2018_soil_moisture_[%vwc]`,
↪  `weather_2019_soil_moisture_[%vwc]`,
↪  `weather_2020_soil_moisture_[%vwc]`,
↪  `weather_2021_soil_moisture_[%vwc]`,
↪  `weather_2022_soil_moisture_[%vwc]`)
```

```
photoperiod_all = list(`weather_2014_photoperiod_[hours]`,
↪  `weather_2015_photoperiod_[hours]`, `weather_2016_photoperiod_[hours]`,
↪  `weather_2017_photoperiod_[hours]`,`weather_2018_photoperiod_[hours]`)

save(phenotype_all, soil_all, field_all, agronomic_all, temperature_all,
↪  dewpoint_all, humidity_all, solar_radiation_all, rainfall_all,
↪  windspeed_all, winddir_all, windgust_all, soiltemp_all, soilmois_all,
↪  soilec_all, uvlight_all, co2_all, photoperiod_all, file =
↪  'processed_data/all_merged_data.RData')



# K nearest neighbors imputation for soil data
soil_all_imp[49:221,c(8:16, 18:25)] =
↪  impute.knn(as.matrix(soil_all_imp[49:221,c(8:16, 18:25)]))$data
soil_all_imp = split_field_locations(cbind(soil_all_imp[2], soil_all_imp[1],
↪  soil_all_imp[3:25]))
soil_all_imp = cbind(soil_all_imp[2], soil_all_imp[1], soil_all_imp[3:25])

# K nearest neighbors imputation for temperature
all_temp_imp = rbind(
  data.frame(Year = rep(2014, nrow(`weather_2014_temperature_[c]`)),
             `Field Location` = `weather_2014_temperature_[c]`$`Field
              ↪  Location`,
             `weather_2014_temperature_[c]`[30:117]),

  data.frame(Year = rep(2015, nrow(`weather_2015_temperature_[c]`)),
```

```
            `Field Location` = `weather_2015_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2015_temperature_[c]`[38:125]),

data.frame(Year = rep(2016, nrow(`weather_2016_temperature_[c]`)),
            `Field Location` = `weather_2016_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2016_temperature_[c]`[42:129]),

data.frame(Year = rep(2017, nrow(`weather_2017_temperature_[c]`)),
            `Field Location` = `weather_2017_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2017_temperature_[c]`[66:153]),

data.frame(Year = rep(2018, nrow(`weather_2018_temperature_[c]`)),
            `Field Location` = `weather_2018_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2018_temperature_[c]`[42:129]),

data.frame(Year = rep(2019, nrow(`weather_2019_temperature_[c]`)),
            `Field Location` = `weather_2019_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2019_temperature_[c]`[34:121]),

data.frame(Year = rep(2020, nrow(`weather_2020_temperature_[c]`)),
```

```
            `Field Location` = `weather_2020_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2020_temperature_[c]`[38:125]),

  data.frame(Year = rep(2021, nrow(`weather_2021_temperature_[c]`)),
            `Field Location` = `weather_2021_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2021_temperature_[c]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_temperature_[c]`)),
            `Field Location` = `weather_2022_temperature_[c]`$`Field
            ↪ Location`,
            `weather_2022_temperature_[c]`[38:125])
)

all_temp_imp[all_temp_imp == Inf] <- NA
all_temp_imp[all_temp_imp == -Inf] <- NA

all_temp_imp[3:90] = impute.knn(as.matrix(all_temp_imp[3:90]))$data

# K nearest neighbors imputation for dew point
all_dewpt_imp = rbind(
  data.frame(Year = rep(2014, nrow(`weather_2014_dew_point_[c]`)),
            `Field Location` = `weather_2014_dew_point_[c]`$`Field
            ↪ Location`,
            `weather_2014_dew_point_[c]`[30:117]),
```

```
data.frame(Year = rep(2015, nrow(`weather_2015_dew_point_[c]`)),
           `Field Location` = `weather_2015_dew_point_[c]`$`Field
           ↪ Location`,
           `weather_2015_dew_point_[c]`[38:125]),

data.frame(Year = rep(2016, nrow(`weather_2016_dew_point_[c]`)),
           `Field Location` = `weather_2016_dew_point_[c]`$`Field
           ↪ Location`,
           `weather_2016_dew_point_[c]`[42:129]),

data.frame(Year = rep(2017, nrow(`weather_2017_dew_point_[c]`)),
           `Field Location` = `weather_2017_dew_point_[c]`$`Field
           ↪ Location`,
           `weather_2017_dew_point_[c]`[66:153]),

data.frame(Year = rep(2018, nrow(`weather_2018_dew_point_[c]`)),
           `Field Location` = `weather_2018_dew_point_[c]`$`Field
           ↪ Location`,
           `weather_2018_dew_point_[c]`[42:129]),

data.frame(Year = rep(2019, nrow(`weather_2019_dew_point_[c]`)),
           `Field Location` = `weather_2019_dew_point_[c]`$`Field
           ↪ Location`,
           `weather_2019_dew_point_[c]`[34:121]),
```

```
  data.frame(Year = rep(2020, nrow(`weather_2020_dew_point_[c]`)),
             `Field Location` = `weather_2020_dew_point_[c]`$`Field
             ↪  Location`,
             `weather_2020_dew_point_[c]`[38:125]),

  data.frame(Year = rep(2021, nrow(`weather_2021_dew_point_[c]`)),
             `Field Location` = `weather_2021_dew_point_[c]`$`Field
             ↪  Location`,
             `weather_2021_dew_point_[c]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_dew_point_[c]`)),
             `Field Location` = `weather_2022_dew_point_[c]`$`Field
             ↪  Location`,
             `weather_2022_dew_point_[c]`[38:125])
)

all_dewpt_imp[all_dewpt_imp == Inf] <- NA
all_dewpt_imp[all_dewpt_imp == -Inf] <- NA

all_dewpt_imp[3:90] = impute.knn(as.matrix(all_dewpt_imp[3:90]))$data

# K nearest neighbors imputation for humidity
all_humidity_imp = rbind(
  data.frame(Year = rep(2014, nrow(`weather_2014_relative_humidity_[%]`)),
             `Field Location` = `weather_2014_relative_humidity_[%]`$`Field
             ↪  Location`,
```

```
            `weather_2014_relative_humidity_[%]`[30:117]),

data.frame(Year = rep(2015, nrow(`weather_2015_relative_humidity_[%]`)),
            `Field Location` = `weather_2015_relative_humidity_[%]`$`Field
             ↪ Location`,
            `weather_2015_relative_humidity_[%]`[38:125]),

data.frame(Year = rep(2016, nrow(`weather_2016_relative_humidity_[%]`)),
            `Field Location` = `weather_2016_relative_humidity_[%]`$`Field
             ↪ Location`,
            `weather_2016_relative_humidity_[%]`[42:129]),

data.frame(Year = rep(2017, nrow(`weather_2017_relative_humidity_[%]`)),
            `Field Location` = `weather_2017_relative_humidity_[%]`$`Field
             ↪ Location`,
            `weather_2017_relative_humidity_[%]`[66:153]),

data.frame(Year = rep(2018, nrow(`weather_2018_relative_humidity_[%]`)),
            `Field Location` = `weather_2018_relative_humidity_[%]`$`Field
             ↪ Location`,
            `weather_2018_relative_humidity_[%]`[42:129]),

data.frame(Year = rep(2019, nrow(`weather_2019_relative_humidity_[%]`)),
            `Field Location` = `weather_2019_relative_humidity_[%]`$`Field
             ↪ Location`,
            `weather_2019_relative_humidity_[%]`[34:121]),
```

```
  data.frame(Year = rep(2020, nrow(`weather_2020_relative_humidity_[%]`)),
             `Field Location` = `weather_2020_relative_humidity_[%]`$`Field
             ↪ Location`,
             `weather_2020_relative_humidity_[%]`[38:125]),

  data.frame(Year = rep(2021, nrow(`weather_2021_relative_humidity_[%]`)),
             `Field Location` = `weather_2021_relative_humidity_[%]`$`Field
             ↪ Location`,
             `weather_2021_relative_humidity_[%]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_relative_humidity_[%]`)),
             `Field Location` = `weather_2022_relative_humidity_[%]`$`Field
             ↪ Location`,
             `weather_2022_relative_humidity_[%]`[38:125])
)

all_humidity_imp[all_humidity_imp == Inf] <- NA
all_humidity_imp[all_humidity_imp == -Inf] <- NA

all_humidity_imp[3:90] = impute.knn(as.matrix(all_humidity_imp[3:90]))$data

# K nearest neighbors imputation for solar radiation
all_solarrad_imp = rbind(
  data.frame(Year = rep(2014, nrow(`weather_2014_solar_radiation_[w/m2]`)),
```

```
            `Field Location` = `weather_2014_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2014_solar_radiation_[w/m2]`[30:117]),

data.frame(Year = rep(2015, nrow(`weather_2015_solar_radiation_[w/m2]`)),
            `Field Location` = `weather_2015_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2015_solar_radiation_[w/m2]`[38:125]),

data.frame(Year = rep(2016, nrow(`weather_2016_solar_radiation_[w/m2]`)),
            `Field Location` = `weather_2016_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2016_solar_radiation_[w/m2]`[42:129]),

data.frame(Year = rep(2017, nrow(`weather_2017_solar_radiation_[w/m2]`)),
            `Field Location` = `weather_2017_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2017_solar_radiation_[w/m2]`[66:153]),

data.frame(Year = rep(2018, nrow(`weather_2018_solar_radiation_[w/m2]`)),
            `Field Location` = `weather_2018_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2018_solar_radiation_[w/m2]`[42:129]),

data.frame(Year = rep(2019, nrow(`weather_2019_solar_radiation_[w/m2]`)),
```

```
            `Field Location` = `weather_2019_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2019_solar_radiation_[w/m2]`[34:121]),

  data.frame(Year = rep(2020, nrow(`weather_2020_solar_radiation_[w/m2]`)),
            `Field Location` = `weather_2020_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2020_solar_radiation_[w/m2]`[38:125]),

  data.frame(Year = rep(2021, nrow(`weather_2021_solar_radiation_[w/m2]`)),
            `Field Location` = `weather_2021_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2021_solar_radiation_[w/m2]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_solar_radiation_[w/m2]`)),
            `Field Location` = `weather_2022_solar_radiation_[w/m2]`$`Field
            ↪ Location`,
            `weather_2022_solar_radiation_[w/m2]`[38:125])
)

all_solarrad_imp[all_solarrad_imp == Inf] <- NA
all_solarrad_imp[all_solarrad_imp == -Inf] <- NA

all_solarrad_imp[3:90] = impute.knn(as.matrix(all_solarrad_imp[3:90]))$data

# K nearest neighbors imputation for windspeed
```

```
all_windspeed_imp = rbind(
  data.frame(Year = rep(2014, nrow(`weather_2014_wind_speed_[m/s]`)),
             `Field Location` = `weather_2014_wind_speed_[m/s]`$`Field
             ↪ Location`,
             `weather_2014_wind_speed_[m/s]`[30:117]),

  data.frame(Year = rep(2015, nrow(`weather_2015_wind_speed_[m/s]`)),
             `Field Location` = `weather_2015_wind_speed_[m/s]`$`Field
             ↪ Location`,
             `weather_2015_wind_speed_[m/s]`[38:125]),

  data.frame(Year = rep(2016, nrow(`weather_2016_wind_speed_[m/s]`)),
             `Field Location` = `weather_2016_wind_speed_[m/s]`$`Field
             ↪ Location`,
             `weather_2016_wind_speed_[m/s]`[42:129]),

  data.frame(Year = rep(2017, nrow(`weather_2017_wind_speed_[m/s]`)),
             `Field Location` = `weather_2017_wind_speed_[m/s]`$`Field
             ↪ Location`,
             `weather_2017_wind_speed_[m/s]`[66:153]),

  data.frame(Year = rep(2018, nrow(`weather_2018_wind_speed_[m/s]`)),
             `Field Location` = `weather_2018_wind_speed_[m/s]`$`Field
             ↪ Location`,
             `weather_2018_wind_speed_[m/s]`[42:129]),
```

```
  data.frame(Year = rep(2019, nrow(`weather_2019_wind_speed_[m/s]`)),
             `Field Location` = `weather_2019_wind_speed_[m/s]`$`Field
             ↪  Location`,
             `weather_2019_wind_speed_[m/s]`[34:121]),

  data.frame(Year = rep(2020, nrow(`weather_2020_wind_speed_[m/s]`)),
             `Field Location` = `weather_2020_wind_speed_[m/s]`$`Field
             ↪  Location`,
             `weather_2020_wind_speed_[m/s]`[38:125]),

  data.frame(Year = rep(2021, nrow(`weather_2021_wind_speed_[m/s]`)),
             `Field Location` = `weather_2021_wind_speed_[m/s]`$`Field
             ↪  Location`,
             `weather_2021_wind_speed_[m/s]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_wind_speed_[m/s]`)),
             `Field Location` = `weather_2022_wind_speed_[m/s]`$`Field
             ↪  Location`,
             `weather_2022_wind_speed_[m/s]`[38:125])
)

all_windspeed_imp[all_windspeed_imp == Inf] <- NA
all_windspeed_imp[all_windspeed_imp == -Inf] <- NA

all_windspeed_imp[3:90] =
↪  impute.knn(as.matrix(all_windspeed_imp[3:90]))$data
```

```
# K nearest neighbors imputation for wind direction
all_winddir_imp = rbind(
  data.frame(Year = rep(2014,
  ↪  nrow(`weather_2014_wind_direction_[degrees]`)),
            `Field Location` =
             ↪  `weather_2014_wind_direction_[degrees]`$`Field Location`,
            `weather_2014_wind_direction_[degrees]`[30:117]),

  data.frame(Year = rep(2015,
  ↪  nrow(`weather_2015_wind_direction_[degrees]`)),
            `Field Location` =
             ↪  `weather_2015_wind_direction_[degrees]`$`Field Location`,
            `weather_2015_wind_direction_[degrees]`[38:125]),

  data.frame(Year = rep(2016,
  ↪  nrow(`weather_2016_wind_direction_[degrees]`)),
            `Field Location` =
             ↪  `weather_2016_wind_direction_[degrees]`$`Field Location`,
            `weather_2016_wind_direction_[degrees]`[42:129]),

  data.frame(Year = rep(2017,
  ↪  nrow(`weather_2017_wind_direction_[degrees]`)),
            `Field Location` =
             ↪  `weather_2017_wind_direction_[degrees]`$`Field Location`,
            `weather_2017_wind_direction_[degrees]`[66:153]),
```

```
data.frame(Year = rep(2018,
↪  nrow(`weather_2018_wind_direction_[degrees]`)),
           `Field Location` =
            ↪  `weather_2018_wind_direction_[degrees]`$`Field Location`,
           `weather_2018_wind_direction_[degrees]`[42:129]),


data.frame(Year = rep(2019,
↪  nrow(`weather_2019_wind_direction_[degrees]`)),
           `Field Location` =
            ↪  `weather_2019_wind_direction_[degrees]`$`Field Location`,
           `weather_2019_wind_direction_[degrees]`[34:121]),


data.frame(Year = rep(2020,
↪  nrow(`weather_2020_wind_direction_[degrees]`)),
           `Field Location` =
            ↪  `weather_2020_wind_direction_[degrees]`$`Field Location`,
           `weather_2020_wind_direction_[degrees]`[38:125]),


data.frame(Year = rep(2021,
↪  nrow(`weather_2021_wind_direction_[degrees]`)),
           `Field Location` =
            ↪  `weather_2021_wind_direction_[degrees]`$`Field Location`,
           `weather_2021_wind_direction_[degrees]`[42:129]),
```

```
  data.frame(Year = rep(2022,
  ↪  nrow(`weather_2022_wind_direction_[degrees]`)),
             `Field Location` =
              ↪  `weather_2022_wind_direction_[degrees]`$`Field Location`,
             `weather_2022_wind_direction_[degrees]`[38:125])
)

all_winddir_imp[all_winddir_imp == Inf] <- NA
all_winddir_imp[all_winddir_imp == -Inf] <- NA

all_winddir_imp[3:90] = impute.knn(as.matrix(all_winddir_imp[3:90]))$data

# K nearest neighbors imputation for wind gust
all_windgust_imp = rbind(
  data.frame(Year = rep(2014, nrow(`weather_2014_wind_gust_[m/s]`)),
             `Field Location` = `weather_2014_wind_gust_[m/s]`$`Field
              ↪  Location`,
             `weather_2014_wind_gust_[m/s]`[30:117]),

  data.frame(Year = rep(2015, nrow(`weather_2015_wind_gust_[m/s]`)),
             `Field Location` = `weather_2015_wind_gust_[m/s]`$`Field
              ↪  Location`,
             `weather_2015_wind_gust_[m/s]`[38:125]),

  data.frame(Year = rep(2016, nrow(`weather_2016_wind_gust_[m/s]`)),
```

```
            `Field Location` = `weather_2016_wind_gust_[m/s]`$`Field
            ↪ Location`,
            `weather_2016_wind_gust_[m/s]`[42:129]),

data.frame(Year = rep(2017, nrow(`weather_2017_wind_gust_[m/s]`)),
            `Field Location` = `weather_2017_wind_gust_[m/s]`$`Field
            ↪ Location`,
            `weather_2017_wind_gust_[m/s]`[66:153]),

data.frame(Year = rep(2018, nrow(`weather_2018_wind_gust_[m/s]`)),
            `Field Location` = `weather_2018_wind_gust_[m/s]`$`Field
            ↪ Location`,
            `weather_2018_wind_gust_[m/s]`[42:129]),

data.frame(Year = rep(2019, nrow(`weather_2019_wind_gust_[m/s]`)),
            `Field Location` = `weather_2019_wind_gust_[m/s]`$`Field
            ↪ Location`,
            `weather_2019_wind_gust_[m/s]`[34:121]),

data.frame(Year = rep(2020, nrow(`weather_2020_wind_gust_[m/s]`)),
            `Field Location` = `weather_2020_wind_gust_[m/s]`$`Field
            ↪ Location`,
            `weather_2020_wind_gust_[m/s]`[38:125]),

data.frame(Year = rep(2021, nrow(`weather_2021_wind_gust_[m/s]`)),
```

```
            `Field Location` = `weather_2021_wind_gust_[m/s]`$`Field
            ↪ Location`,
            `weather_2021_wind_gust_[m/s]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_wind_gust_[m/s]`)),
            `Field Location` = `weather_2022_wind_gust_[m/s]`$`Field
            ↪ Location`,
            `weather_2022_wind_gust_[m/s]`[38:125])
)

all_windgust_imp[all_windgust_imp == Inf] <- NA
all_windgust_imp[all_windgust_imp == -Inf] <- NA

all_windgust_imp[3:90] = impute.knn(as.matrix(all_windgust_imp[3:90]))$data

# K nearest neighbors imputation for rain
all_rain_imp = rbind(
  data.frame(Year = rep(2014, nrow(weather_2014_sum_rainfall)),
            `Field Location` = weather_2014_sum_rainfall$`Field Location`,
            weather_2014_sum_rainfall[8:30]),

  data.frame(Year = rep(2015, nrow(weather_2015_sum_rainfall)),
            `Field Location` = weather_2015_sum_rainfall$`Field Location`,
            weather_2015_sum_rainfall[10:32]),

  data.frame(Year = rep(2016, nrow(weather_2016_sum_rainfall)),
```

```
            `Field Location` = weather_2016_sum_rainfall$`Field Location`,
            weather_2016_sum_rainfall[11:33]),

data.frame(Year = rep(2017, nrow(weather_2017_sum_rainfall)),
            `Field Location` = weather_2017_sum_rainfall$`Field Location`,
            weather_2017_sum_rainfall[c(11:31, 33:34)]),

data.frame(Year = rep(2018, nrow(weather_2018_sum_rainfall)),
            `Field Location` = weather_2018_sum_rainfall$`Field Location`,
            weather_2018_sum_rainfall[11:33]),

data.frame(Year = rep(2019, nrow(weather_2019_sum_rainfall)),
            `Field Location` = weather_2019_sum_rainfall$`Field Location`,
            weather_2019_sum_rainfall[9:31]),

data.frame(Year = rep(2020, nrow(weather_2020_sum_rainfall)),
            `Field Location` = weather_2020_sum_rainfall$`Field Location`,
            weather_2020_sum_rainfall[10:32]),

data.frame(Year = rep(2021, nrow(weather_2021_sum_rainfall)),
            `Field Location` = weather_2021_sum_rainfall$`Field Location`,
            weather_2021_sum_rainfall[11:33]),

data.frame(Year = rep(2022, nrow(weather_2022_sum_rainfall)),
            `Field Location` = weather_2022_sum_rainfall$`Field Location`,
            weather_2022_sum_rainfall[10:32])
```

```
)

all_rain_imp[3:25] = impute.knn(as.matrix(all_rain_imp[3:25]))$data


# K nearest neighbors imputation for soil temperature
all_soiltemp_imp = rbind(
  data.frame(Year = rep(2015, nrow(`weather_2015_soil_temperature_[c]`)),
             `Field Location` = `weather_2015_soil_temperature_[c]`$`Field
             ↪  Location`,
             `weather_2015_soil_temperature_[c]`[38:125]),

  data.frame(Year = rep(2016, nrow(`weather_2016_soil_temperature_[c]`)),
             `Field Location` = `weather_2016_soil_temperature_[c]`$`Field
             ↪  Location`,
             `weather_2016_soil_temperature_[c]`[42:129]),

  data.frame(Year = rep(2017, nrow(`weather_2017_soil_temperature_[c]`)),
             `Field Location` = `weather_2017_soil_temperature_[c]`$`Field
             ↪  Location`,
             `weather_2017_soil_temperature_[c]`[66:153]),

  data.frame(Year = rep(2018, nrow(`weather_2018_soil_temperature_[c]`)),
             `Field Location` = `weather_2018_soil_temperature_[c]`$`Field
             ↪  Location`,
             `weather_2018_soil_temperature_[c]`[42:129]),
```

```
  data.frame(Year = rep(2019, nrow(`weather_2019_soil_temperature_[c]`)),
             `Field Location` = `weather_2019_soil_temperature_[c]`$`Field
             ↪ Location`,
             `weather_2019_soil_temperature_[c]`[34:121]),

  data.frame(Year = rep(2020, nrow(`weather_2020_soil_temperature_[c]`)),
             `Field Location` = `weather_2020_soil_temperature_[c]`$`Field
             ↪ Location`,
             `weather_2020_soil_temperature_[c]`[38:125]),

  data.frame(Year = rep(2021, nrow(`weather_2021_soil_temperature_[c]`)),
             `Field Location` = `weather_2021_soil_temperature_[c]`$`Field
             ↪ Location`,
             `weather_2021_soil_temperature_[c]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_soil_temperature_[c]`)),
             `Field Location` = `weather_2022_soil_temperature_[c]`$`Field
             ↪ Location`,
             `weather_2022_soil_temperature_[c]`[38:125])
)

all_soiltemp_imp[all_soiltemp_imp == Inf] <- NA
all_soiltemp_imp[all_soiltemp_imp == -Inf] <- NA

all_soiltemp_imp[3:90] = impute.knn(as.matrix(all_soiltemp_imp[3:90]))$data
```

```
# K nearest neighbors imputation for soil moisture
all_soilmois_imp = rbind(
  data.frame(Year = rep(2015, nrow(`weather_2015_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2015_soil_moisture_[%vwc]`$`Field
             ↪  Location`,
             `weather_2015_soil_moisture_[%vwc]`[38:125]),

  data.frame(Year = rep(2016, nrow(`weather_2016_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2016_soil_moisture_[%vwc]`$`Field
             ↪  Location`,
             `weather_2016_soil_moisture_[%vwc]`[42:129]),

  data.frame(Year = rep(2017, nrow(`weather_2017_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2017_soil_moisture_[%vwc]`$`Field
             ↪  Location`,
             `weather_2017_soil_moisture_[%vwc]`[66:153]),

  data.frame(Year = rep(2018, nrow(`weather_2018_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2018_soil_moisture_[%vwc]`$`Field
             ↪  Location`,
             `weather_2018_soil_moisture_[%vwc]`[42:129]),

  data.frame(Year = rep(2019, nrow(`weather_2019_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2019_soil_moisture_[%vwc]`$`Field
             ↪  Location`,
```

```
            `weather_2019_soil_moisture_[%vwc]`[34:121]),

  data.frame(Year = rep(2020, nrow(`weather_2020_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2020_soil_moisture_[%vwc]`$`Field
             ↪ Location`,
             `weather_2020_soil_moisture_[%vwc]`[38:125]),

  data.frame(Year = rep(2021, nrow(`weather_2021_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2021_soil_moisture_[%vwc]`$`Field
             ↪ Location`,
             `weather_2021_soil_moisture_[%vwc]`[42:129]),

  data.frame(Year = rep(2022, nrow(`weather_2022_soil_moisture_[%vwc]`)),
             `Field Location` = `weather_2022_soil_moisture_[%vwc]`$`Field
             ↪ Location`,
             `weather_2022_soil_moisture_[%vwc]`[38:125])
)

all_soilmois_imp[all_soilmois_imp == Inf] <- NA
all_soilmois_imp[all_soilmois_imp == -Inf] <- NA

all_soilmois_imp[3:90] = impute.knn(as.matrix(all_soilmois_imp[3:90]))$data

# K nearest neighbors imputation for photoperiod
all_photoperiod_imp = rbind(
  data.frame(Year = rep(2014, nrow(`weather_2014_photoperiod_[hours]`)),
```

```
            `Field Location` = `weather_2014_photoperiod_[hours]`$`Field
            ↪ Location`,
            `weather_2014_photoperiod_[hours]`[30:113]),

  data.frame(Year = rep(2015, nrow(`weather_2015_photoperiod_[hours]`)),
            `Field Location` = `weather_2015_photoperiod_[hours]`$`Field
            ↪ Location`,
            `weather_2015_photoperiod_[hours]`[38:121]),

  data.frame(Year = rep(2016, nrow(`weather_2016_photoperiod_[hours]`)),
            `Field Location` = `weather_2016_photoperiod_[hours]`$`Field
            ↪ Location`,
            `weather_2016_photoperiod_[hours]`[42:125]),

  data.frame(Year = rep(2017, nrow(`weather_2017_photoperiod_[hours]`)),
            `Field Location` = `weather_2017_photoperiod_[hours]`$`Field
            ↪ Location`,
            `weather_2017_photoperiod_[hours]`[66:149]),

  data.frame(Year = rep(2018, nrow(`weather_2018_photoperiod_[hours]`)),
            `Field Location` = `weather_2018_photoperiod_[hours]`$`Field
            ↪ Location`,
            `weather_2018_photoperiod_[hours]`[42:125])

)
```

```
all_photoperiod_imp[all_photoperiod_imp == Inf] <- NA
all_photoperiod_imp[all_photoperiod_imp == -Inf] <- NA

all_photoperiod_imp[3:86] =
↪  impute.knn(as.matrix(all_photoperiod_imp[3:86]))$data

names(phenotype_all) <- gsub(" ", "_", names(phenotype_all))
names(soil_all_imp) <- gsub(" ", "_", names(soil_all_imp))
names(field_all) <- gsub(" ", "_", names(field_all))
names(agronomic_all) <- gsub(" ", "_", names(agronomic_all))
names(all_dewpt_imp)[2] <- "Field_Location"
names(all_humidity_imp)[2] <- "Field_Location"
names(all_solarrad_imp)[2] <- "Field_Location"
names(all_rain_imp)[2] <- "Field_Location"
names(all_windspeed_imp)[2] <- "Field_Location"
names(all_winddir_imp)[2] <- "Field_Location"
names(all_windgust_imp)[2] <- "Field_Location"
names(all_soiltemp_imp)[2] <- "Field_Location"
names(all_soilmois_imp)[2] <- "Field_Location"
names(all_temp_imp)[2] <- "Field_Location"
names(all_photoperiod_imp)[2] <- "Field_Location"

# Preparing agronomic data for categorization and encoding
agronomic_encoded = agronomic_all
agronomic_encoded$Application_or_treatment =
↪  tolower(agronomic_encoded$Application_or_treatment)
```

```r
agronomic_encoded$Product_or_nutrient_applied <- iconv(
  agronomic_encoded$Product_or_nutrient_applied,
  from = "",
  to = "UTF-8",
  sub = "byte"
)
agronomic_encoded$Product_or_nutrient_applied =
↪  tolower(agronomic_encoded$Product_or_nutrient_applied)

# Preparing field data for categorization and encoding
field_encoded = field_all
field_encoded$Previous_Crop = tolower(field_encoded$Previous_Crop)
field_encoded$`Pre.plant_tillage_method.s.` =
↪  tolower(field_encoded$`Pre.plant_tillage_method.s.`)
field_encoded$`In.season_tillage_method.s.` =
↪  tolower(field_encoded$`In.season_tillage_method.s.`)
field_encoded$Treatment = tolower(field_encoded$Treatment)

# Categorization for Application or Treatment
categorizeApp <- function(df) {
  df <- df %>%
    mutate(Application_Category = case_when(
      is.na(Application_or_treatment) ~ "NA",
      str_detect(tolower(Application_or_treatment), "herbicide") ~
      ↪  "Herbicide",
```

```
      str_detect(tolower(Application_or_treatment), "fertilizer") ~
      ↪  "Fertilizer",
      str_detect(tolower(Application_or_treatment), "fertilize") ~
      ↪  "Fertilizer",
      str_detect(tolower(Application_or_treatment), "irrigation") ~
      ↪  "Irrigation",
      TRUE ~ "Other"
    ))
  return(df)
}

# Categorization for Product or nutrient applied
categorizeProd <- function(df) {
  df <- df %>%
    mutate(Product_Category = case_when(
      is.na(Product_or_nutrient_applied) ~ "NA",
      str_detect(tolower(Product_or_nutrient_applied), "water") ~ "water",
      str_detect(tolower(Product_or_nutrient_applied), "atrazine") ~
      ↪  "atrazine",
      TRUE ~ "Other"
    ))
  return(df)
}

# Categorization for Treatment
categorizeFieldTreat <- function(df) {
```

```
  df <- df %>%
    mutate(Treatment_Category = case_when(
      is.na(Treatment) ~ "NA",
      str_detect(tolower(Treatment), "standard") ~ "standard",
      TRUE ~ "Other"
    ))
  return(df)
}

# Categorization for Previous Crop
categorizePrevCrop <- function(df) {
  df <- df %>%
    mutate(PrevCrop_Category = case_when(
      is.na(Previous_Crop) ~ "NA",
      str_detect(tolower(Previous_Crop), "soybean|soybeans") ~ "soybean",
      str_detect(tolower(Previous_Crop), "corn") ~ "corn",
      str_detect(tolower(Previous_Crop), "wheat") ~ "wheat",
      TRUE ~ "Other"
    ))
  return(df)
}

# Categorization for Pre plant tillage methods
categorizePreTill <- function(df) {
  df <- df %>%
    mutate(PreTill_Category = case_when(
```

```
      is.na(`Pre.plant_tillage_method.s.`) ~ "NA",
      str_detect(tolower(`Pre.plant_tillage_method.s.`), "cultivator|field
      ↪  cultivat") ~ "field cultivator",
      str_detect(tolower(`Pre.plant_tillage_method.s.`), "none") ~ "NA",
      TRUE ~ "Other"
    ))
  return(df)
}

# Categorization for In season tillage methods
categorizeSeasonTill <- function(df) {
  df <- df %>%
    mutate(SeasonTill_Category = case_when(
      is.na(`In.season_tillage_method.s.`) ~ "NA",
      str_detect(tolower(`In.season_tillage_method.s.`), "none") ~ "None",
      TRUE ~ "Applied"
    ))
  return(df)
}

# Encoding agronomic data
agronomic_encoded = categorizeApp(agronomic_encoded)
agronomic_encoded = categorizeProd(agronomic_encoded)
agronomic_encoded = agronomic_encoded[c(1,2,7,8)]
agronomic_encoded <- dummy_cols(
  agronomic_encoded,
```

```
  select_columns = c("Application_Category", "Product_Category"),
  remove_selected_columns = TRUE,
  ignore_na = TRUE,
  remove_first_dummy = FALSE
)
agronomic_encoded <- agronomic_encoded %>%
  group_by(Year, Field_Location) %>%
  summarise(across(everything(), max, na.rm = TRUE), .groups = "drop")
agronomic_encoded <- agronomic_encoded %>%
  select(-contains("Product_Category_NA"),
         -contains("Application_Category_NA"))
agronomic_encoded <- agronomic_encoded %>%
  mutate(across(-c(Year, Field_Location), ~ as.integer(.)))

# Encoding field data
field_encoded = categorizeFieldTreat(field_encoded)
field_encoded = categorizePrevCrop(field_encoded)
field_encoded = categorizePreTill(field_encoded)
field_encoded = categorizeSeasonTill(field_encoded)
field_encoded = field_encoded[c(1,2,7:10)]
encode_with_na_first <- function(x) {
  x_char <- as.character(x)
  x_char[is.na(x_char)] <- "NA"

  lvls <- unique(c("NA", sort(unique(x_char[x_char != "NA"]))))
```

```
  as.integer(factor(x_char, levels = lvls)) - 1
}
field_encoded$Treatment_Category     <-
↪  encode_with_na_first(field_encoded$Treatment_Category)
field_encoded$PrevCrop_Category      <-
↪  encode_with_na_first(field_encoded$PrevCrop_Category)
field_encoded$PreTill_Category       <-
↪  encode_with_na_first(field_encoded$PreTill_Category)
field_encoded$SeasonTill_Category    <-
↪  encode_with_na_first(field_encoded$SeasonTill_Category)

field_encoded <- field_encoded %>%
  group_by(Year, Field_Location) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE), .groups = "drop")



# Soil encoding
soil_encoded$`X1.1_Soil_pH` = as.numeric(soil_encoded$`X1.1_Soil_pH`)
soil_encoded$`CEC.Sum_of_Cations_me.100g` =
↪  as.numeric(soil_encoded$`CEC.Sum_of_Cations_me.100g`)
soil_encoded <- soil_encoded %>%
  group_by(Year, `Field_Location`) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE), .groups = "drop")

# Phenotype encoding
phenotype_encoded = phenotype_all
```

```
phenotype_encoded <- phenotype_encoded %>%
  mutate(
    Pedigree_1 = str_split_fixed(Pedigree, "/", 2)[, 1],
    Pedigree_2 = str_split_fixed(Pedigree, "/", 2)[, 2]
  )

master_keys = phenotype_encoded[1:2]


# Joining all data
soil_joined = left_join(x = master_keys, y = soil_encoded, by = master_keys)
field_joined = left_join(x = master_keys, y = field_encoded, by =
↪  master_keys)
agronomic_joined = left_join(x = master_keys, y = agronomic_encoded, by =
↪  master_keys)

dewpoint_joined = left_join(x = master_keys, y = dewpoint_all_imp, by =
↪  master_keys)
humidity_joined = left_join(x = master_keys, y = humidity_all_imp, by =
↪  master_keys)
solarrad_joined = left_join(x = master_keys, y = solarrad_all_imp, by =
↪  master_keys)
rainfall_joined = left_join(x = master_keys, y = rainfall_all_imp, by =
↪  master_keys)
windspeed_joined = left_join(x = master_keys, y = windspeed_all_imp, by =
↪  master_keys)
```

```
winddir_joined = left_join(x = master_keys, y = winddir_all_imp, by =
↪  master_keys)
windgust_joined = left_join(x = master_keys, y = windgust_all_imp, by =
↪  master_keys)
soiltemp_joined = left_join(x = master_keys, y = soiltemp_all_imp, by =
↪  master_keys)
soilmois_joined = left_join(x = master_keys, y = soilmois_all_imp, by =
↪  master_keys)
temperature_joined = left_join(x = master_keys, y = temperature_all_imp, by
↪  = master_keys)
photoperiod_joined = left_join(x = master_keys, y = photoperiod_all_imp, by
↪  = master_keys)
```

Bibliography

Ansarifar, J., Wang, L., & Archontoulis, S. V. (2021). An interaction regression model for crop yield prediction. *Scientific Reports*.

Ausmees, K., & Nettelblad, C. (2022). A deep learning framework for characterization of genotype data. *G3*.

Hatfield, J., Dold, C., & Fahad, S. (2018). Corn: Production and human health in changing climate. *London: Books on Demand*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. https://arxiv.org/abs/1512.03385

Jo, T., Nho, K., Bice, P., & Saykin, A. J. (2022). Deep learning-based identification of genetic variants: Application to alzheimer's disease classification. *Briefings in Bioinformatics*.

Lundberg, S., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. https://arxiv.org/abs/1705.07874

Nieradzik, L., Stephani, H., Sieburg-Rockel, J., Helmling, S., Olbrich, A., & Keuper, J. (2024). Challenging the black box: A comprehensive evaluation of attribution maps of cnn applications in agriculture and forestry. https://arxiv.org/abs/2402.11670

Sharma, S., Partap, A., de Luis Balaguer, M. A., Malvar, S., & Chandra, R. (2022). Deepg2p: Fusing multi-modal data to improve crop production. https://arxiv.org/abs/2211.05986

U.S. Department of Agriculture, Economic Research Service. (2025). Feed grains sector at a glance [Accessed: 2025-06-30]. https://www.ers.usda.gov/topics/crops/corn-and-other-feed-grains/feed-grains-sector-at-a-glance

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need. https://arxiv.org/abs/1706.03762

Zhang, Z., & Jung, C. (2019). Gbdt-mo: Gradient boosted decision trees for multiple outputs. https://arxiv.org/abs/1909.04373