Australian Government
Wine Australia

MACQUARIE University
SYDNEY · AUSTRALIA

Accurate and early yield predictions through advanced statistical modelling

FINAL REPORT

Project number: **MQ 1401**

Principal Investigator: **Dr Steven Van Sluyter**

Research organisation: **Macquarie University**

Date: **3 April 2020**

Accurate and early yield predictions through advanced statistical modelling

Wine Australia project MQ 1401

Final report

3 April 2020

Macquarie University

Summary

This project created a modelling approach that predicts block-level yields based on weather. Models are trained with historical vineyard data, provided by growers, matched with historical weather data. To make predictions, current season vineyard and weather data are used with the trained models to produce yield predictions. The strongest predictors for yields were the average yield of the block and weather at flowering.

The accuracy of the models depended heavily on the sources of data. The most accurate predictions for a single grower were <10% error; the least accurate were >20% error.

Although prediction errors were higher than the project goal of 5%, the approach of using machine learning to predict vineyard yields based on weather data is promising. The main obstacle to higher prediction accuracy is the quality of historical grower data for training models. Also, reformatting grower data to be machine readable and consistent is prohibitively time consuming in many cases. In the future, data formatting problems could be resolved by growers using standard data formats, or at least consistent, well designed formats within each organisation.

Average yields are the best predictors of future yields. For some blocks there was so little variation year-to-year that their historical average could simply be used as the yield prediction with little error on average. Our models predicted yield variations from historical averages—as opposed to predicting yields outright with no previous knowledge of block-level historical yields. We only used data from blocks with 3 or more years of historical yield data.

A key component of our approach was standardizing historical data based on the assumption that all cultivar x region x grower combinations co-vary. For example, the yields of all Chardonnay in McLaren Vale from Grower A deviate from the average historical yield in the same way year-to-year. Extending that approach to regions using high quality regional historical yield data could be used to predict variation from the average yield for each cultivar x region combination. Regional predictions would be less laborious and could be extended to growers without requiring extensive historical data from them.

Contents

1 OVERVIEW 3

1.1 Background 3

1.2 Modelling summary 4

2 MODELLING APPROACH 7

2.1 Data cleaning 7

2.2 Data scaling 7

2.3 Training and predicting 8

3 RESULTS SUMMARY 9

3.1 Historical average yields as predictors of current season yields 9

3.2 Model results 10

3.2.1 Linear regression 13

3.2.2 Mixed linear model 14

3.2.3 Generalised additive model 15

3.2.4 Artificial neural network 16

4 CONCLUSION 17

5 METHODS 19

5.1 Generalized Additive Model 20

5.2 Linear Regression Model 37

5.3 ANN 54

5.4 Mixed Linear Model 69

5.5 Automated variable selection 86

1 Overview

1.1 Background

Grapevine yields vary substantially and unpredictably year-to-year because they are influenced in complex and poorly understood ways by vineyard histories and by several weather variables. This project directly addressed the GWRDC 2014/15 gaps analysis R&D priorities of accurate vineyard crop forecasting and yield estimation.

Managing yields through informed decision-making is of central importance for producers because yield-quality interactions and yield targets specified by buyers affect income. On the expense side, accurate and early yield predictions allow more efficient vineyard practices and planning. Coordination between growers and wineries is disrupted when yields differ from predictions, and so poor predictions have flow-on effects through wine production and distribution chains.

GWRDC-funded analysis (CSH96/1) estimated that 10% of industry expenditure is dedicated to coping with seasonal variation and that half of that could be saved through exact yield forecasts. The estimated cost of inaccurate forecasts is $100m to $200m annually.

The challenge of yield prediction is that the independent factors that affect yields are well known, but the ways in which they interact overall to determine yields are not well known. The reason for the knowledge gap is that modelling interactions among multiple variables requires advanced statistical methods not previously utilized in grape yield prediction tools.

The main hypothesis of this project was that yield predictions can be made using advanced statistical models that do not require a mechanistic understanding of the underlying drivers of variation. In other words, we can make predictions about *how* weather affects grapevine yields without understanding *why* each weather variable affects yields.

Yields are influenced by weather up to 18 months prior to harvest and by crop load and vegetative growth of the previous season. As a result, yields vary substantially from year-to-year because of compounding weather effects, with costly unpredictability. The general ways in which weather affects individual components of yield, such as the initiation of inflorescences and flowering, are known. However, a model incorporating all the contributors to yield has not been created. The reason for the lack of a comprehensive model is that computational complexity grows exponentially with each additional variable. Standard statistical methods used to identify individual components of yield are not suitable for assessing all the components as a whole—it is simply impossible. The only methods capable of modelling multiple non-linear relationships are machine learning methods, such as artificial neural networks. The basis of this project is the use of machine learning to create an accurate and easy to use yield prediction tool.

The general approach was based on Mike Trought's work (e.g. 2005 ASVO Proceedings). Trought developed a simple model to forecast yields for

Marlborough Sauvignon blanc using temperatures at two key developmental stages. Bunches per shoot has a linear relationship with temperature at initiation, and bunch weight a sigmoidal relationship with temperature at flowering, so the combination of the two in a linear model is a compromise. Even so, just those two variables do a reasonable job of forecasting yields at regional levels. Trought's work demonstrates that a weather-based approach is feasible, and our plan was to add more variables (necessitating machine learning) in order to make forecasts accurate at the block level for multiple cultivars and regions.

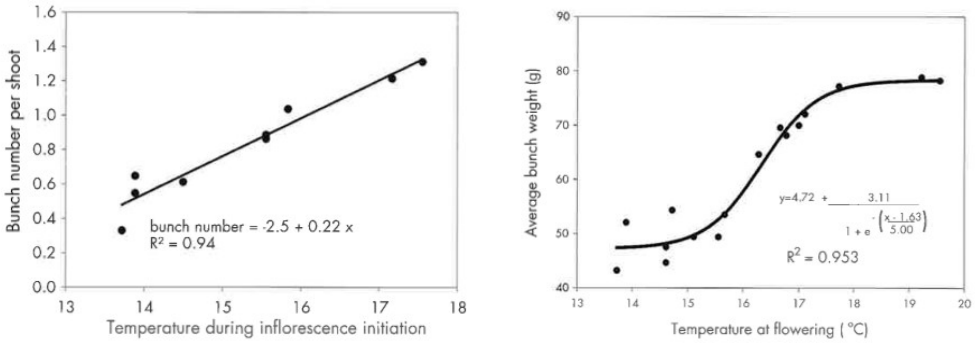


Figure 1. A simple model using temperature shortly after budburst of the previous season and temperature at flowering produced estimates of regional yields for Marlborough Sauvignon Blanc (From Trought 2005, ASVO seminar proceedings, Mildura).

1.2 Modelling summary

We developed four different types of models that differ in their statistical approaches. All make yield predictions based on weather variables and all must be trained using historical vineyard data. Two of the models are traditional statistical models; two are forms of machine learning (Figure 2).

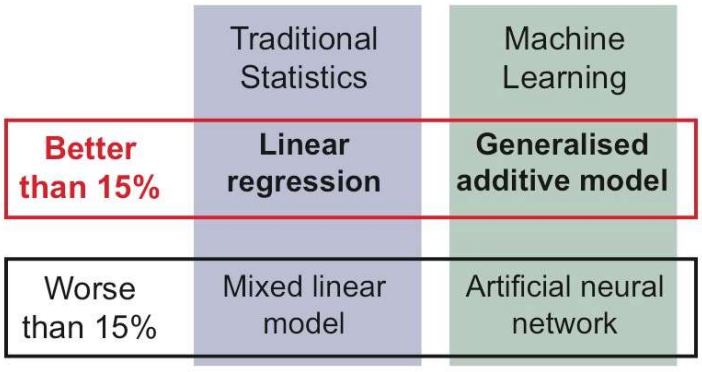


Figure 2. Two of the four modelling approaches produced average prediction errors better than 15% when used with high quality data sets.

The models make predictions for unknown yields based on the historical data for a block and the weather for that growing season and, in some cases, weather from the previous growing season. The mixed linear model and artificial neural network models also considered the cultivar and region as categorical variables.

We developed automated methods to dramatically reduce the computational time required to screen weather variable combinations for their predictive power. For example, there are over 65,000 possible combinations of 16 weather variables. We used automated methods to rapidly reduce the number of variable combinations, and then manually refined those selected combinations using a two-step model training and testing procedure (Figure 3).

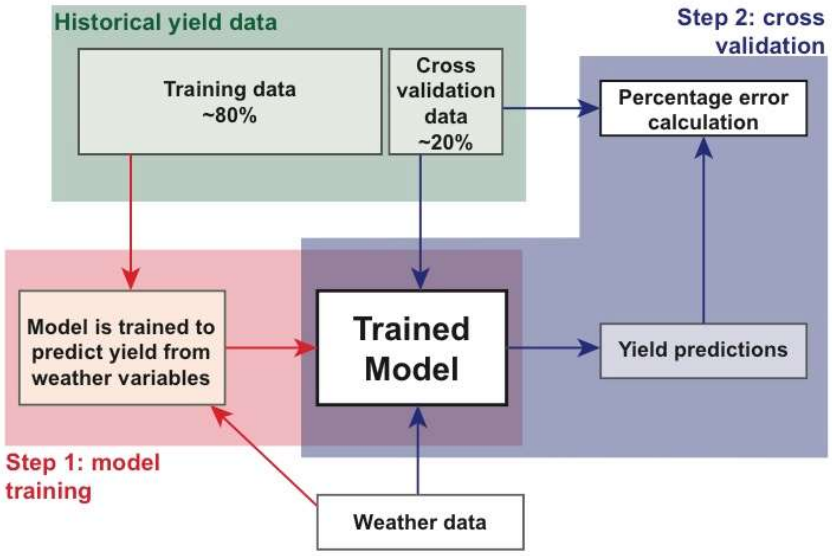


Figure 3. The four model types were all trained and tested using the same two-step process: Step 1, model training; Step 2, cross validation. Model training used randomly selected subsets of historical yield data to establish the relationships between weather variables and yield. In the cross validation step the trained models were used to make predictions for the block × year combinations withheld from the training data. Prediction error was calculated by comparing those predicted yields versus the actual yields in the withheld data set. The two-step process was repeated with different combinations of weather variables until the prediction error was minimised for each model type. The % errors we report are based on 5 iterations of the two-step process, with different training data selected randomly each time.

Model accuracy was measured by making yield predictions for block × year combinations withheld from the training data set, and then comparing those predictions to the actual yields.

In addition to trialling thousands of combinations of weather variables, we tested previous vintage yield as a predictor. Average block yields over time were also used in models. In this report we present data from the combination of

variables that produced the most accurate predictions for each of the four types of models.

Vineyard data

During model development we found that better predictions could be achieved by creating single models that combined data from multiple regions and cultivars. In mixed linear models and artificial neural networks, 'Region' and 'cultivar' were included as variables. In that way predictions were based on more robust training data. For example, predictions for Barossa Shiraz yields were weighted toward training data from all Shiraz across Australia and all cultivars within the Barossa instead of only previous Barossa Shiraz data. However, the most accurate predictions were made by Generalized Additive Models and Linear Regression models that did not use cultivar and region as categorical predictors.

Weather data

Weather data were downloaded from the Bureau of Meteorology web server. The weather data collection was performed using a custom software package in the statistical environment R. Vineyard location coordinates and time frames were entered into the software, then the data from the nearest weather station downloaded automatically.

The main challenge for compiling weather data is adjusting the timeframes of the weather data based on vine development stages. For example, the most significant weather variables we tested were around the time of flowering. However, flowering occurs at different times in different years. Therefore, we used times of flowering and budburst as reference points to normalize weather data across years (Table 1).

Table 1. Examples of categories defining weather variables for use in models

Vine growth stage	Timing relative to growth stage	Weather measurement (fortnightly averages)
• Previous season bud burst	• 2 weeks before	• Minimum temperature
• Current season bud burst	• 1 week before, 1 week after	• Maximum temperature
• Current season flowering	• 2 weeks after	• Precipitation
		• Solar exposure

2 Modelling approach

Our models are designed to be continuously improved as the training dataset grows. Therefore, the strategy is a modelling approach, not a single model. It uses historical vineyard data at the block level, from growers, and historical weather, to train models. Then it uses current season weather to make predictions of current season yields. The approach consists of three parts: cleaning, scaling, training and predicting.

2.1 Data cleaning

Grower datasets often contain yield observations from years of catastrophic, unpredictable losses (e.g. from hail), erroneous yields from, for example, two blocks being harvested together, or simple data entry errors. Our approach removes outlier yield observations by the following:

1) Divide each yield for a block by the historical block average (mean or median).
2) Determine the 5th and 95th percentile for the (block yield)/(average block yield) across the entire dataset.
3) Remove observations that fall outside the 5th-95th percentile range.

2.2 Data scaling

1) Re-calculate block means based on the cleaned dataset.
2) Divide each block-level yield by its corresponding mean block yield. This value is the normalized yield.
3) Calculate the standard deviation for each cultivar x grower x region combination. For example, the standard deviation across all Barossa Chardonnay for Grower A.
4) Subtract 1 from each normalized yield, then divide that by the corresponding cultivar x grower x region standard deviation. This is the scaled yield.

Scaled yield approximates a z-score (z-score = (observation – mean)/standard deviation). With enough historical data per block it would be ideally calculated based on non-normalized values and the standard deviation of each block, which would make it a true z-score. We have found in grower yield data that the magnitudes of variation across years differs consistently by cultivar. That is why, for datasets of too few block-level observations over time to calculate accurate standard deviations, we can approximate z-scores using the standard deviation of each cultivar x grower x region combination.

Scaled values also account for differences across cultivars in the magnitudes of their yield responses to weather. We have used types of models that can accommodate categorical predictors such as cultivar (artificial neural networks and mixed linear models), which could effectively create sub-models for each region x cultivar combination, if necessary, and if the training data support them.

2.3 Training and predicting

We have used freely available R packages for four types of models: linear regression, mixed linear model, generalised additive model, artificial neural network. The successful implementation of the different types of models illustrates that the overall approach is model-independent.

The physiological theory is based on ASVO conference proceedings, Mildura, from Dr Mike Trought in 2005. Trought created a simple predictive model for the regional yield of Marlborough Sauvignon Blanc using two weather variables: temperature at budburst of the previous season; temperature at flowering of the current season. The approach of the current project extends that by including multiple possible weather variables across variable time windows centred around budburst and flowering. The reasoning is that, even though the Trought paper demonstrated an effect of temperature on yield, there could be additional unknown relationships among weather variables and yield. At the least, the temperatures that affect yield at budburst and flowering are those of the plant tissues, which are affected by many more factors than simple air temperature. In addition to weather around previous season budburst, we trial weather at current season budburst.

Even as few as 16 weather variables, and variable time windows, results in 65,000 possible models. Therefore, the modelling approach is based around selecting the best model with each new training dataset instead of incrementally improving a single model.

Artificial neural networks and mixed linear models are allowed the possibility to use categorical predictors, such as region and cultivar, in addition to weather variables. We have also trialled yield of the previous season as a predictor for current season yield.

1. Weather data have been scraped from the BOM website based on the BOM station nearest each vineyard block using an R script gifted to the project by a researcher at Macquarie University. Weather data are typically averaged over a time, such as a fortnight, before using to train models.
2. Weather variables are selected in a semi-automated way in R comparing yield-weather relationships based on correlation coefficients. The top weather variable combinations are then compared using Akaike Information Criterion (AIC) and cross-validation errors of trained models.
3. Prediction: once trained on scaled historical yield and weather data, the models predict scaled yield values. The actual yield predictions in tonnes per hectare are made by back-transforming scaled yields.
4. Cross validation is done by withholding some of the historical data from the training dataset, training a model, making predictions based on the withheld data, and measuring error of predictions vs actual yields in the cross-validation set.
5. For artificial neural networks, the numbers of potential weather variables are first reduced by selecting with generalised additive models to reduce overfitting by the ANNs.

We calculate error as:

$$\% \text{ error} = 100 \times (\text{predicted yield} - \text{actual yield})/\text{actual yield}$$

For model comparisons we then average the absolute values of the % errors for each potential model.

3 Results summary

3.1 Historical average yields as predictors of current season yields

We found that the magnitude of year-to-year yield differences varies by cultivar (Table 2). The low variation of some cultivars, such as Muscat Gordo, demonstrates that, in many cases, historical averages are excellent predictors of yields. We found that weather conditions at budburst of the previous season did significantly affect yields ($p < 0.05$) but explained <10% of overall yield variation and did not substantially improve model performance measure by cross-validation. Therefore, we focused on weather time windows based around current season flowering dates. Similarly, in some cases the yield of the previous season accounted for a small but statistically significant amount of variation, but was not used as a predictor because the benefits of including it in models was offset by how its inclusion limited the number of useful observation years per block (i.e., the first year of observations for a block could not be used; the year following a catastrophic loss could not be used).

Table 2. Year-to-year yield variation based on 153 observations of 38 blocks, at least 3 years per block from one grower.

Cultivar	% variation
Muscat Gordo	7
Pinot Gris	9
Merlot	13
Shiraz	14
Cabernet Sauvignon	14
Tempranillo	16
Malbec	18
Durif	20
Chardonnay	21
Orange Muscat	24

For a single vintage, we compared the prediction accuracy of our best performing model, a Generalized Additive Model (GAM), with historical averages as predictions, and predictions based on grower berry and bunch counts (Figure 4). On average, the GAM was the most accurate approach, and the average errors

of counting and using historical averages as predictions were not substantially different.

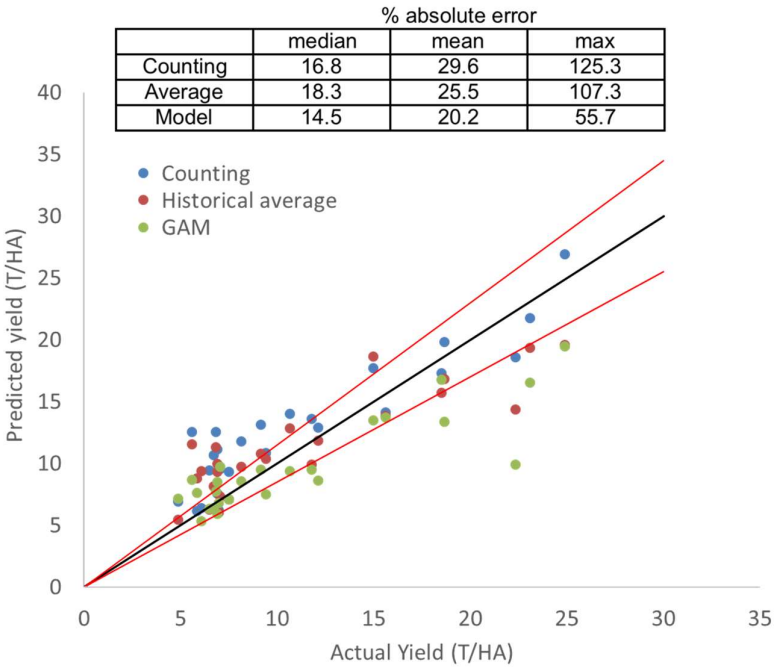


Figure 4. Comparison of three prediction methods: our Generalized Additive Model, using historical averages as current season predictions, berry and bunch counting. The black line indicates perfect predictions; red lines indicate ±15% error. Data were from 25 blocks of multiple cultivars from a single grower. % error = 100 x (predicted yield – actual yield)/actual yield. Median, mean, and maximum errors were of absolute values of % errors.

The different magnitudes of variation across cultivars motivated our data scaling method for training and using models. We assumed that weather drove effects on yields in the same ways, but at different magnitudes depending on cultivars. To capture that our scaling method divided block-level normalized yields by standard deviations across all blocks of that cultivar per region x grower.

3.2 Model results

The accuracies of the four types of models are summarized in Figure 5. Generalized Additive Models (GAM) were consistently the most accurate. Linear regression models were similarly accurate. Surprisingly, mixed linear models were outperformed by simple linear models; potentially because our scaling approach accounted for cultivar and region-level variation. Also, in some cases during cross validation the test data set contained categories that were poorly represented in randomly generated training data sets. Artificial Neural Networks were the least accurate models, most likely because we did not use large enough sets of training data to overcome the tendency of Artificial Neural Networks to

overfit training sets. It is possible that Random Forest models could capture the benefits of ANNs, but with smaller training sets.

Average year-to-year yield variation by block ranged from <10% to approximately 30%. The best GAMs explained approximately 50% of the variation in year-to-year yields judged by correlation coefficients, which corresponds with the best median errors in the 10% to 15% range. Finding relationships that explain the remaining variation becomes increasingly difficult. We found statistically significant, but small, relationships between yield and weather variables at budburst of the previous season. However, including weather at or shortly after budburst of the previous season did not improve prediction accuracy measured by cross validation. Similarly, we could not find weather variables during or after veraison that improved model accuracy, although weak correlations with yields suggested some effects.

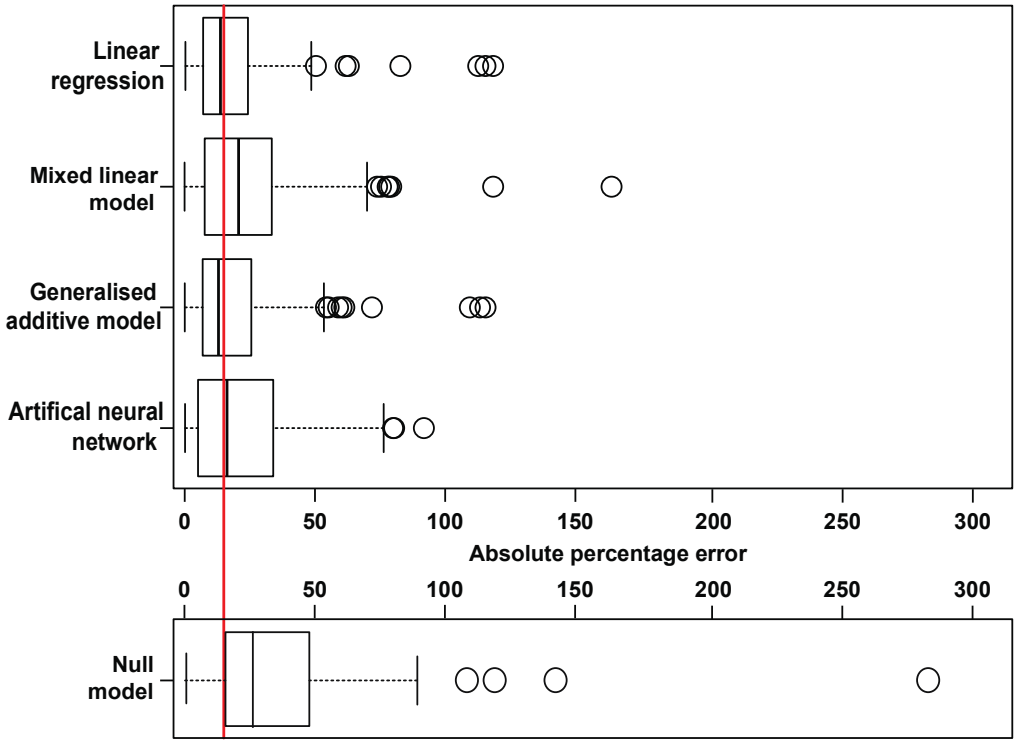


Figure 5. Box plots of absolute percentage error calculated from actual yield harvest observations and yield harvests predicted using each of the four modelling techniques. Boxes represent the upper and lower quartiles with the internal line signifying the median error. Whiskers show minimum and maximum errors (excluding likely outliers shown as circles). The red line is set at 15% error; median lines to the left of the red line signify models that exceed 15% error. The Null model assumes that weather does not affect yield; each predicted yield is simply the historical block average.

The following four sections contain statistical summaries of the four model types. The summaries are based on cross validation results. For each model, five rounds of training and cross validation were performed, and the results

combined. The resulting predictions were compared against actual observed yields and prediction error was calculated as:

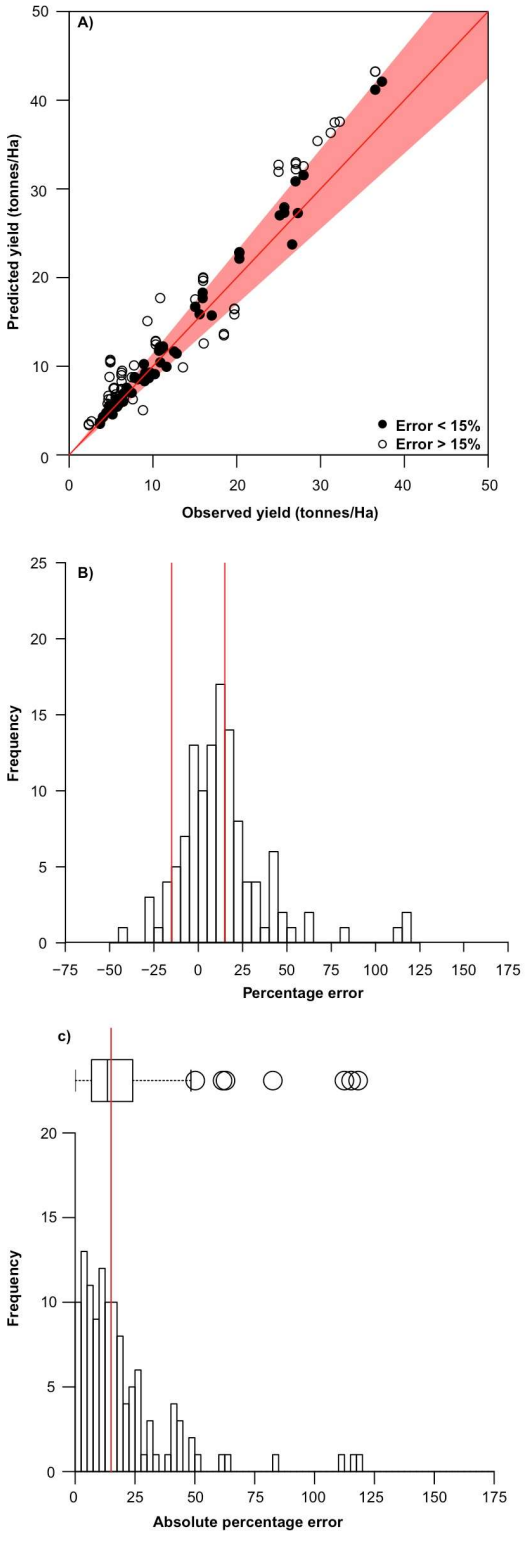
$$\% \text{ error} = 100 \times (\text{predicted yield} - \text{actual yield})/\text{actual yield}$$

The 'B' figure panels in the statistical summaries are histograms of % error. Across all models the greatest prediction errors are over predictions. We believe this is caused by losses that occur post fruit set, which our models cannot currently predict.

The model error distributions all pass a normality test, meaning that the errors follow a normal distribution pattern. The whiskers on the box plots are designed to capture ~99% of a normal distribution, with the outliers indicated as circles outside the whiskers. In all cases the outliers are from over predictions, rather than under predictions, which further suggests that the worst predictions are from losses the models cannot currently predict—those that occur post fruit set. Some post fruit set losses are totally unpredictable; for that reason, we do not consider maximum model error a reliable measure of model performance.

Assuming that the outliers are caused by post fruit set losses, the box plot of, for example, the linear regression model, can be interpreted as follows: a quarter of predictions are better than 6.8% inaccurate, half are better than the median of 13.5% inaccurate, three quarters are better than 24.1% inaccurate, and 99% of predictions are better than 48.4% inaccurate.

Absolute error, used in the 'C' figure panels, refers to the absolute value of the % errors (i.e. -23% would be binned in the histograms and box plots as +23%).



3.2.1 Linear regression

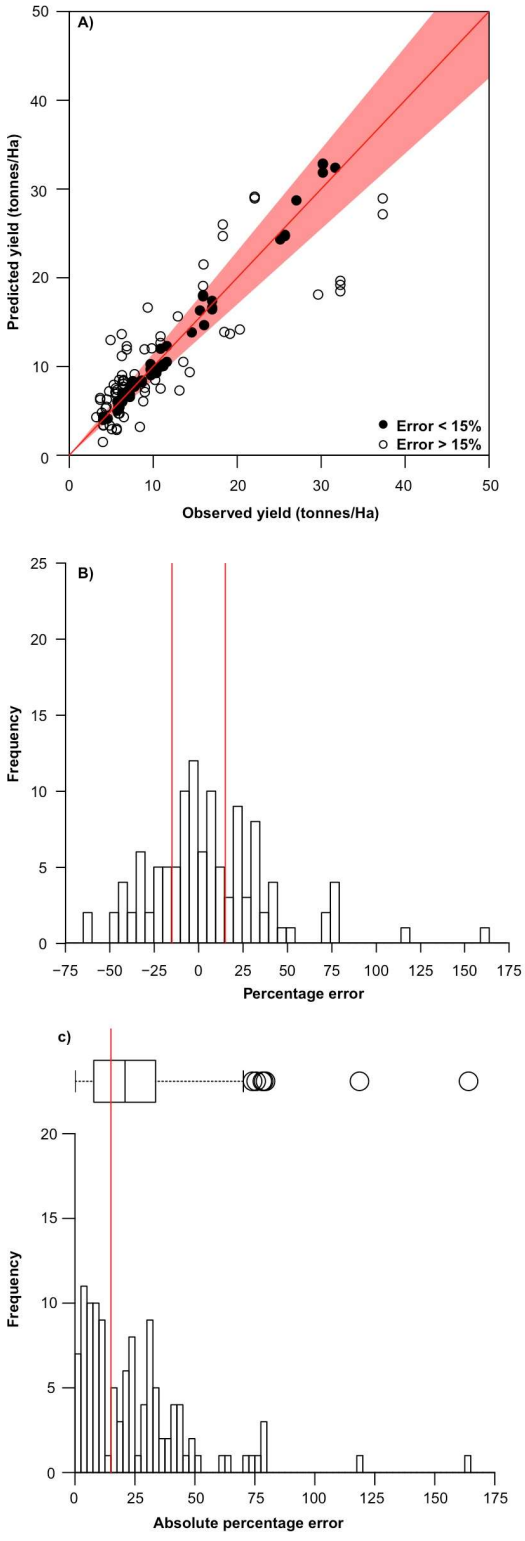
- Median error = 13.5%
- Technique: traditional statistics
- Weather variables: precipitation, max temp, min temp
- Strengths: simple
- Weaknesses: Real-world systems are more complex

Figure 6. Cross validation results using the best combination of weather predictors for a linear regression model.

A) Predicted yield versus actual observed yield showing ± 15% (red shaded area) around a perfect prediction (red line). Filled circles represent predictions that fall within the 15% range.

B) Frequency distribution of the percentage error between predicted and observed yields (bin size = 5) showing the ± 15% error margin between two red lines.

C) Box plot and frequency distribution (bin size = 2.5) of absolute percentage error between observed and predicted yields. The red line indicates 15% error and open circles represent potential outliers in the data.



3.2.2 Mixed linear model

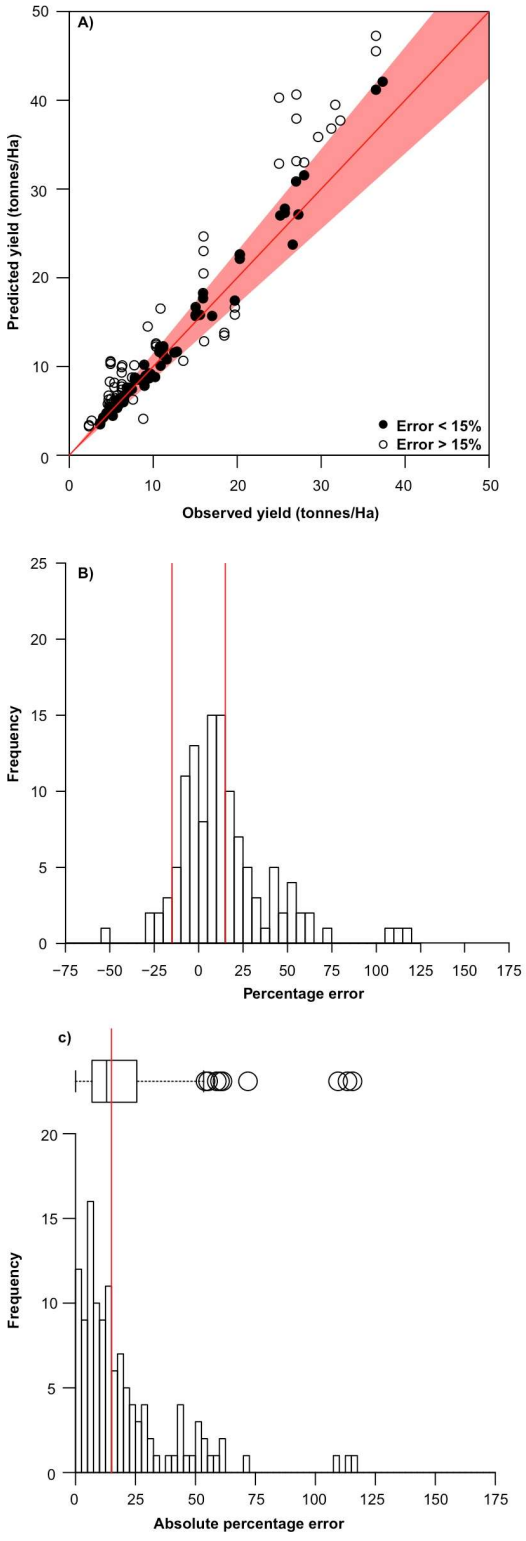
- Median error = 20.9%
- Technique: traditional statistics
- Weather variables: min temp at two time points, max temp, precipitation, solar radiation
- Strengths: can incorporate categorical variables such as cultivar and region; more transparent than machine learning methods.
- Weaknesses: requires large amounts of training data (1000s of observations); cannot model as much complexity as machine learning methods.

Figure A2. Cross validation results using the best combination of weather predictors for a mixed linear model.

A) Predicted yield versus actual observed yield showing ± 15% (red shaded area) around a perfect prediction (red line). Filled circles represent predictions that fall within the 15% range.

B) Frequency distribution of the percentage error between predicted and observed yields (bin size = 5) showing the ± 15% error margin between two red lines.

C) Box plot and frequency distribution (bin size = 2.5) of absolute percentage error between observed and predicted yields. The red line indicates 15% error and open circles represent potential outliers in the data.



3.2.3 Generalised additive model

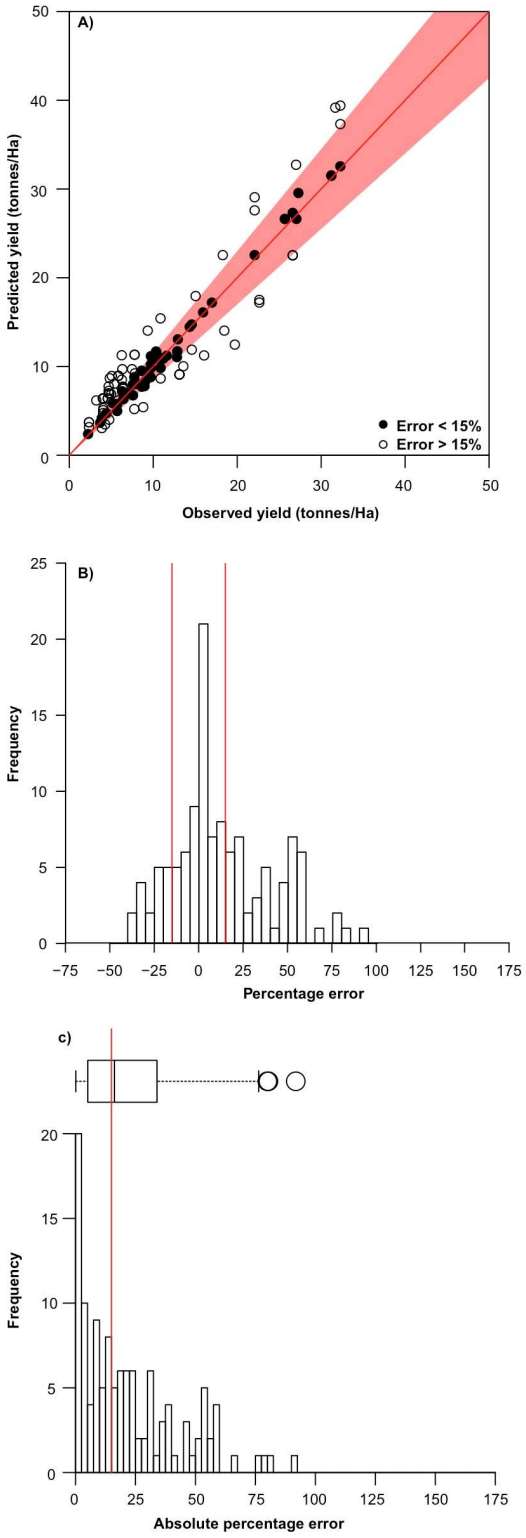
- Median error = 13.0%
- Technique: machine learning
- Weather variables: precipitation, max temp, min temp
- Strengths: Simple; can accommodate more complexity than a linear model; can potentially use categorical predictors such as region and cultivar
- Weaknesses: not transparent; extrapolations to circumstances outside the training data produce high prediction errors.

Figure A3. Cross validation results using the best combination of weather predictors for a generalised additive model.

A) Predicted yield versus actual observed yield showing ± 15% (red shaded area) around a perfect prediction (red line). Filled circles represent predictions that fall within the 15% range.

B) Frequency distribution of the percentage error between predicted and observed yields (bin size = 5) showing the ± 15% error margin between two red lines.

C) Box plot and frequency distribution (bin size = 2.5) of absolute percentage error between observed and predicted yields. The red line indicates 15% error and open circles represent potential outliers in the data.



3.2.4 Artificial neural network

- Median error = 16.2%
- Technique: machine learning
- Weather variables: 12 variables based around current year flowering dates.
- Strengths: can model highly complex relationships, including categorical variables such as region and cultivar
- Weaknesses: not transparent; requires large amounts of training data (1,000s of observations), extrapolations to circumstances outside the training data produce high prediction errors.

Figure A4. Cross validation results using the best combination of weather predictors for an artificial neural network.

A) Predicted yield versus actual observed yield showing ± 15% (red shaded area) around a perfect prediction (red line). Filled circles represent predictions that fall within the 15% range.

B) Frequency distribution of the percentage error between predicted and observed yields (bin size = 5) showing the ± 15% error margin between two red lines.

C) Box plot and frequency distribution (bin size = 2.5) of absolute percentage error between observed and predicted yields. The red line indicates 15% error and open circles represent potential outliers in the data.

4 Conclusion

The project demonstrated the feasibility of modelling block-level yields using historical vineyard data and weather data. It produced four types of models, two that produce predictions with errors less than 15% on average, and methods that automate the screening process for selecting the best weather variable combinations to use in models.

The two best performing models used only three weather variables as predictors, which illustrates the effectiveness of using weather to predict yield. We expected prediction accuracy would increase with the inclusion of additional weather variables, made possible by the inclusion of more data than our preliminary models. We tried training models with >2000 observations, which allowed us to include more weather variables without overfitting the data, and did not observe an increase in accuracy. Accuracy varied by grower, which suggests that model accuracy is at least partly driven by the quality of the historical grower data. Smaller amount of higher quality data with simple models outperformed larger amounts of data of variable quality used with more complex models.

The quality of grower data was likely affected by several factors that interfered with model accuracy. For example, blocks that were harvested differently in different years by combining or splitting blocks. Similarly, inconsistent block names year-to-year were manually, but potentially incorrectly, cleaned. Unaccounted losses from fruit damage such as frost and disease also affected data quality. We excluded years when growers noted substantial damage, but some growers did not note damage. Lastly, there were inconsistencies across growers in how dates of flowering and budburst were defined and recorded.

An additional potential source of grower-level model accuracy variation is the accuracy of weather data. We used weather data from the BOM station nearest each block. The best predictions were from flat regions and the worst were from hilly regions, which might have been coincidental, but could have reflected larger differences between actual weather data and BOM data in hilly regions.

Recommendations

1. *Improved data formatting*. Most of the project time was spent cleaning grower data so that it was machine readable and consistent across growers. Real-world adoption of the modelling approach will benefit from growers organising their data into consistent, machine-readable formats. The formats do not all have to be the same across growers, but they must follow simple guidelines. Requirements include consistent names and abbreviations for blocks and cultivars and not using merged cells in spreadsheets. Variables, such as block names, cultivars, yields, and growth stage dates should be contained in spreadsheet columns; observations for each variable should be contained in rows. A controlled

vocabulary for cultivars, growth stages, and causes of crop losses would be beneficial.

2. *Higher resolution weather data.* We used BOM data because it is free. However, model accuracy could be improved by using higher spatial resolution weather data; for example, from a commercial gridded weather product.
3. *Regional predictions.* Commercial implementation of the modelling approach would be much simpler if yearly predictions were made at the region level instead of the block level. Our block level predictions simply predict yearly variation from the average yield. Our normalising and scaling methods are done at the block/cultivar x grower level, but it is likely that variation mainly occurs at the cultivar x region level. For example, all Chardonnay in the Adelaide Hills potentially varies in response to weather in the same way. If that is the case, then modelling region level yields could be done with subsets of grower data, then extrapolated to cultivar x region blocks that were not included in training data sets.
4. *Growth stage modelling.* Growers define and collect grow stage information differently; some growers do not collect growth stage data at all. Like yield variation possibly being consistent across cultivar x region combinations, it is likely that growth stage timing is similar across cultivar x region combinations. For example, all Chardonnay in the Adelaide Hills potentially flowers at roughly the same time. If that is the case, then growth stage data from a subset of growers could be used to make predictions from grower data sets that do not include growth stage data. Also, year-to-year variation for the dates of early growth stages such as budburst and flowering could potentially be explained by simple growing degree day models. If that is the case, then collecting growth stage dates from growers would not be required for predicting yields.
5. *Inclusion of remote sensing data.* We struggled to find weather variables at or post veraison that explained yield variation and our best performing models only included weather variables around flowering. It could be possible to use overall canopy observations, such as NDVI, to explain some of the year-to-year yield variation driven by conditions post-veraison. NDVI could effectively integrate several variables related to water availability and photosynthesis that would be difficult to capture independently.

5 Methods

The following subsections contain indicative code in the R statistical environment for training the four model types and testing them by cross validation. The fifth subsection, 5.5, describes how weather variables were ranked by their explanatory power prior to manually trialling combinations of them iteratively, prioritizing higher ranking variables.

Data cleaning and scaling was performed prior to running the models below. Data were inputted from a csv file with the following headers for vineyard data:

Header	Definition
Grower	The name of the grower who provided the data
Vineyard	The group of blocks from a grower within a region
Variety	Grape cultivar
Block	The smallest consistent definition of yield measurements provided by a grower
Block Area (Ha)	Block areas
YieldperHAactual	Block level tonnes of grapes per hectare
Prev.yr.YieldperHAactual	The yield per hectare of the previous season
F commencement	Flowering commencement date
STNDYieldperHAactual	Scaled yield
Prev.yr.STNDYieldperHAactual	Scaled yield of the previous season
bbdate	Budburst date
F completion (90% capfall)	Flowering completion date: 80% or 90% capfall
Yield Year	The year of the yield observation

Weather data were from the BOM station nearest the vineyard. In the examples below, weather variables are labelled in the format [growth stage]_[time relative to growth stage]_[weather variable]. For example, "f_pre14_tmax" refers to the average maximum temperature over the 14 days prior to the completion of flowering. The abbreviation "bb_minus6post7_solar" refers to the average solar radiation of the fortnight in which the date of budburst is the 7th day.

The models in the following subsections include cross validation results from 5 rounds indicating % errors for each round. For example, the following indicates a mean error of 18.6%, median error of 7.8%, and a maximum error of 102%:

```
> mean(per.error)
[1] 0.1860467
> median(per.error)
[1] 0.07764013
> range(per.error)
[1] 0.0008615156 1.0249417332
```

5.1 Generalized Additive Model

Generic scripts to cross validate

```
setwd("~/Grapevine Project/Combined grower data for analysis/standar
dised yields")
InputYieldData <- read.csv("~/Grapevine Project/Combined grower data
 for analysis/standardised yields/InputYieldData.csv")
View(InputYieldData)
##organise this data and omit 2011 yields
```

this will sort data by year

```
sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),]
View(sorted.by.year)
```

below scripts will help to read off row labels;

```
x <- c(1:136)
write.table(x, file = "x.txt")
x <- read.csv("~/Grapevine Project/Combined grower data for analysis
/standardised yields/x.txt", sep="")
sorted.by.year.row.label <- cbind(x, sorted.by.year)
View(sorted.by.year.row.label)
```

##set up TRAINING DATA SET this will set-up the training data set

```
years.to.2011 <- sorted.by.year.row.label[1:15,]
View(years.to.2011)
years.from.2011 <- sorted.by.year.row.label[32:136,]
View(years.from.2011)
train.data <- rbind(years.to.2011,years.from.2011)
View(train.data)
##Randomly divide training data set into training and test sets.See
above scripts for reading in original data set.
set.seed(92)
rand <- sample(1:500, 120)
train <- cbind(train.data, rand)
train.sort <- train[order(train$rand),]
traindata <- train.sort[1:96,]
testdata <- train.sort[97:120,]
model <- gam(STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus6p
ost7_tmax) + s(f_minus6post7_tmin), data = traindata)
summary(model)
res <- residuals(model)
fit <- fitted(model)
plot(fit,res)
qqnorm(res)
shapiro.test(res)
est <- predict(model, testdata)
write.table(est, file = "fitted.txt")
```

```
fitted <- read.csv("~/Grapevine Project/Combined grower data for ana
lysis/standardised yields/fitted.txt", sep="")
View(fitted)
per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testdat
a$STNDYieldperHAactual
mean(per.error)
median(per.error)
range(per.error)
hist(per.error, breaks = seq(0, 8, by = 0.05))
```

A five-fold cross validation of this GAM

CV 1

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(123)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- gam(STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus
6post7_tmax) + s(f_minus6post7_tmin), data = traindata)
```

```
> summary(model)

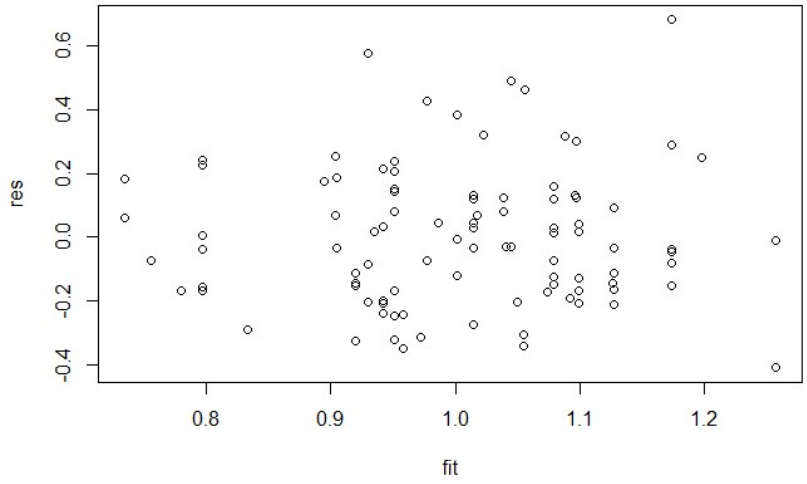
Family: gaussian
Link function: identity

Formula:
STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus6post7_tmax) +
    s(f_minus6post7_tmin)

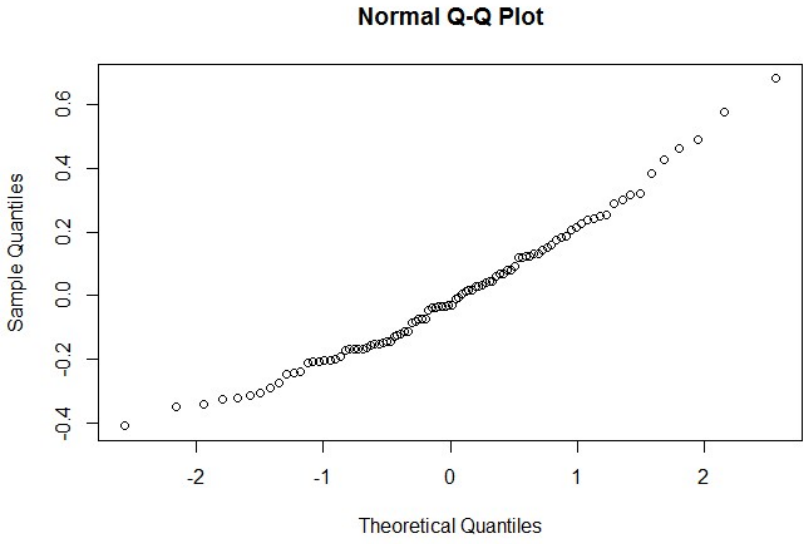
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.00351    0.02269   44.22   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                        edf Ref.df      F  p-value
s(f_post14_precip)    1.000  1.000  4.509 0.036362 *
s(f_minus6post7_tmax) 1.697  2.158 10.333 6.29e-05 ***
s(f_minus6post7_tmin) 1.000  1.000 12.281 0.000703 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.198   Deviance explained =   23%
GCV = 0.051977  Scale est. = 0.049434  n = 96
> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```



```
> qqnorm(res)
```

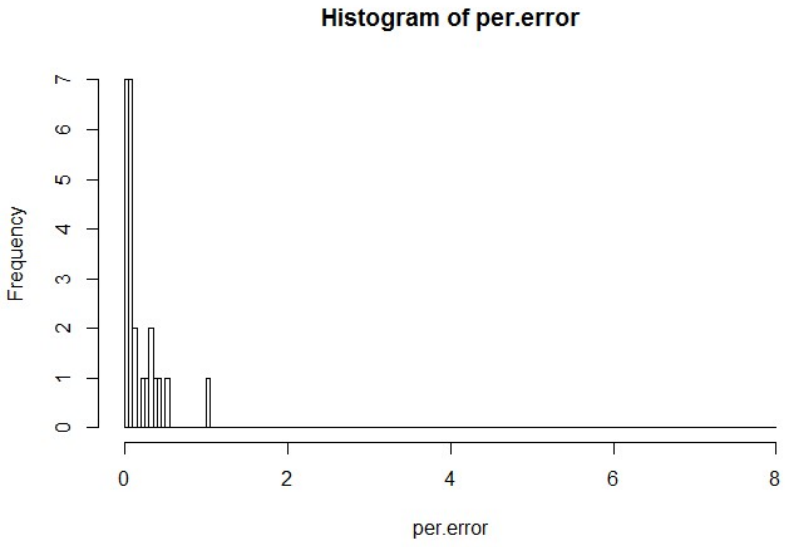



```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.97195, p-value = 0.03722


> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.1860467
> median(per.error)
[1] 0.07764013
> range(per.error)
[1] 0.0008615156 1.0249417332
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 2

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(25)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
```

```
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- gam(STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus
6post7_tmax) + s(f_minus6post7_tmin), data = traindata)
> summary(model)

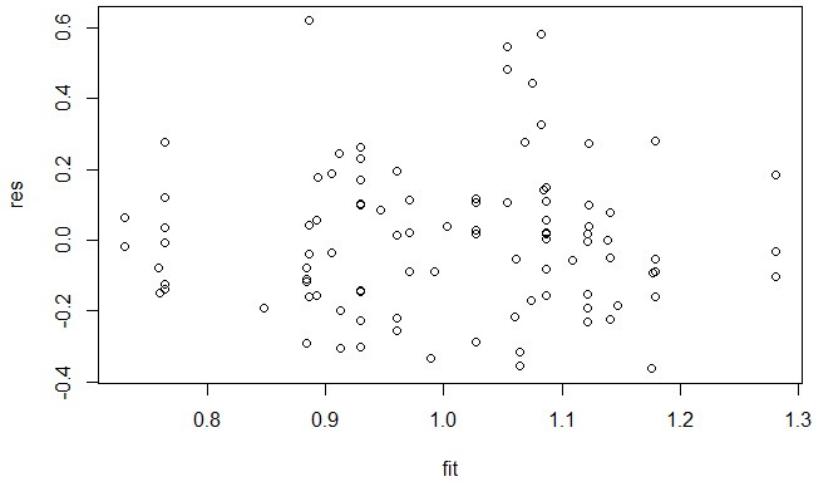
Family: gaussian
Link function: identity

Formula:
STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus6post7_tmax) +
    s(f_minus6post7_tmin)

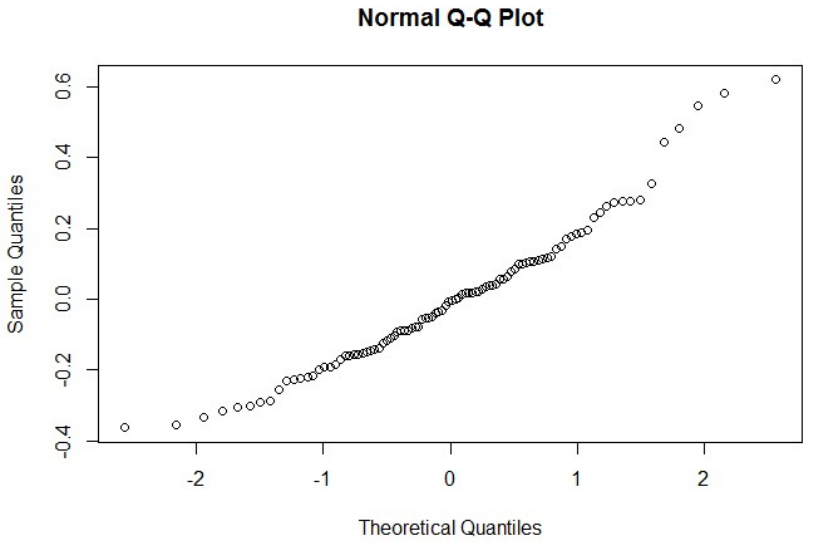
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.0020     0.0215   46.61   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                        edf Ref.df      F  p-value
s(f_post14_precip)    1.000  1.000  6.362   0.0134 *
s(f_minus6post7_tmax) 2.091  2.649  9.958 2.61e-05 ***
s(f_minus6post7_tmin) 1.000  1.000 20.734 1.55e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.268   Deviance explained =   30%
GCV = 0.046843  Scale est. = 0.044359  n = 96
> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



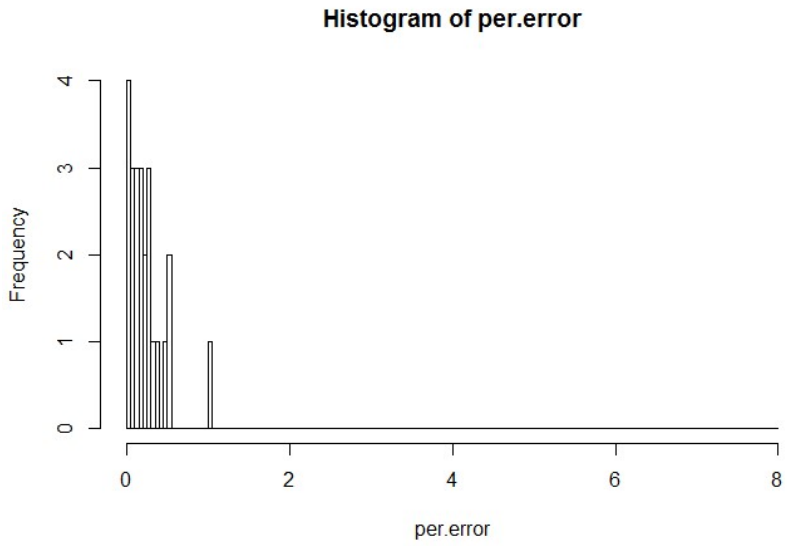

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.96261, p-value = 0.00777

> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
```

```
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.2395437
> median(per.error)
[1] 0.1617583
> range(per.error)
[1] 0.0070364 1.0449316
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 3

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
```

```
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(46)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- gam(STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus
6post7_tmax) + s(f_minus6post7_tmin), data = traindata)
> summary(model)

Family: gaussian
Link function: identity

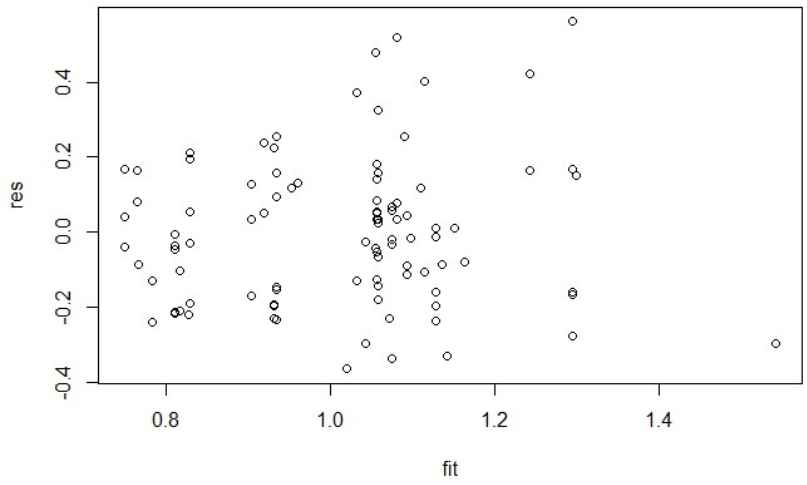
Formula:
STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus6post7_tmax) +
    s(f_minus6post7_tmin)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.01245    0.02096    48.3   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

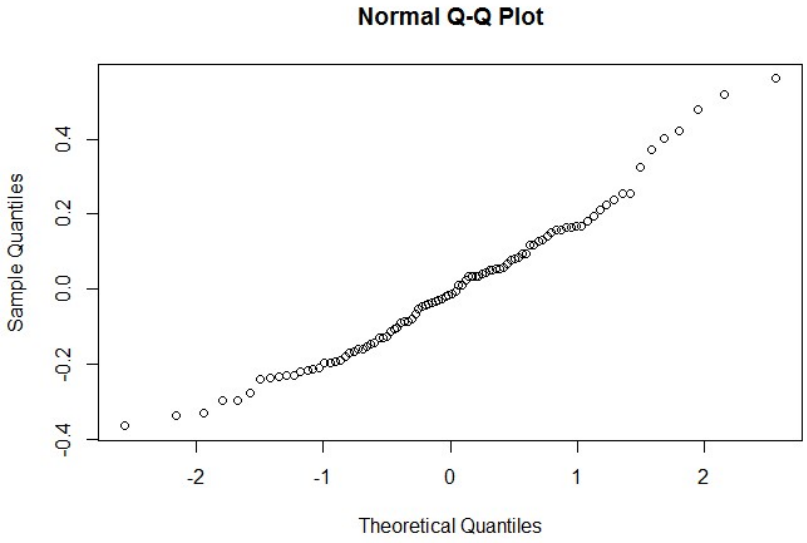
Approximate significance of smooth terms:
                        edf Ref.df     F  p-value
s(f_post14_precip)    1.000  1.000 6.981 0.009751 **
s(f_minus6post7_tmax) 5.286  6.311 6.995 2.44e-06 ***
s(f_minus6post7_tmin) 2.732  3.381 7.220 0.000189 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.355   Deviance explained = 41.6%
GCV = 0.04709  Scale est. = 0.042176  n = 96
```

```
> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```

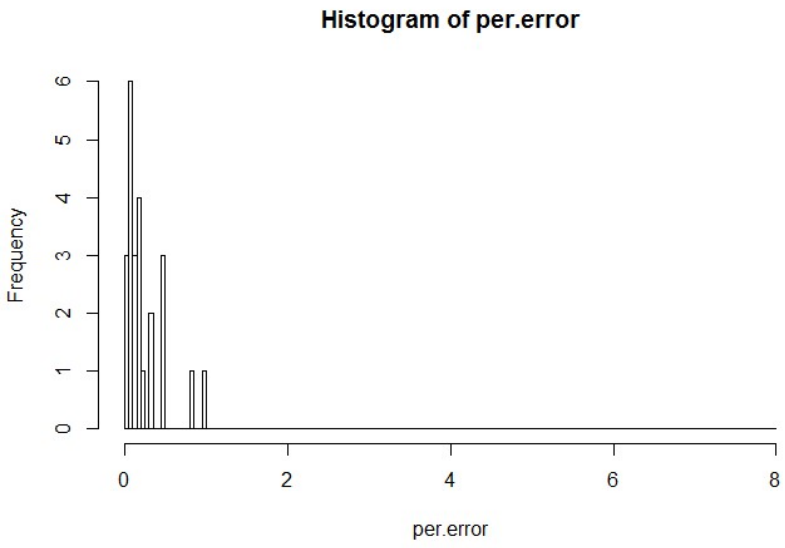



```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.97184, p-value = 0.03655
```

```
> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for analysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testdata$STNDYieldperHAactual
> mean(per.error)
[1] 0.2396266
> median(per.error)
[1] 0.1371192
> range(per.error)
[1] 0.04410851 0.98762447
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 4

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower data for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
```

```
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(80)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- gam(STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus
6post7_tmax) + s(f_minus6post7_tmin), data = traindata)
> summary(model)

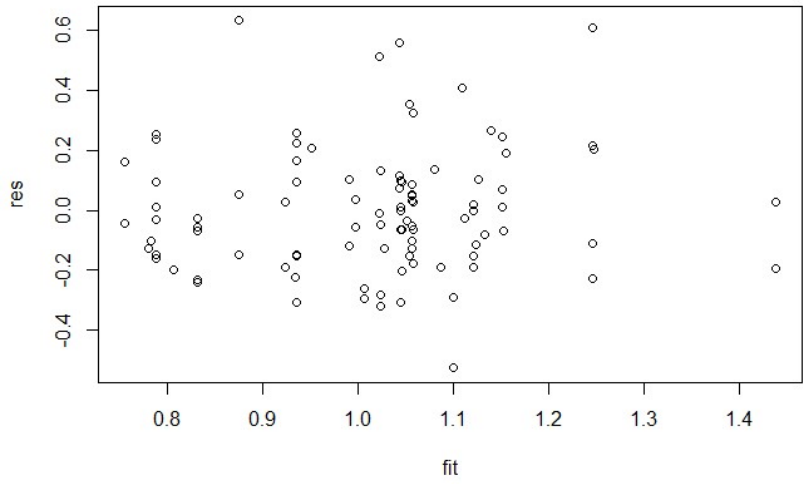
Family: gaussian
Link function: identity

Formula:
STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus6post7_tmax) +
    s(f_minus6post7_tmin)

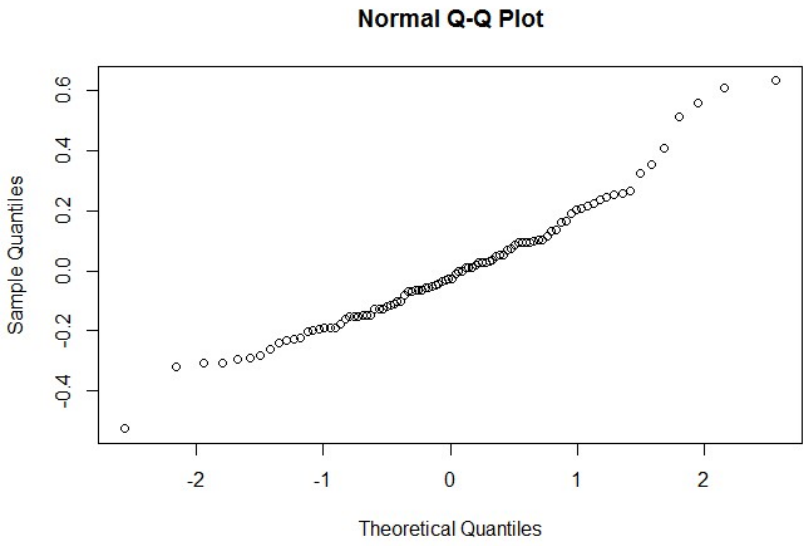
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.01370    0.02232   45.41   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                        edf Ref.df     F  p-value
s(f_post14_precip)    1.694  2.066 4.562 0.013261 *
s(f_minus6post7_tmax) 4.271  5.177 4.817 0.000464 ***
s(f_minus6post7_tmin) 2.548  3.147 4.217 0.006455 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
R-sq.(adj) =  0.281   Deviance explained = 34.5%
GCV = 0.053091  Scale est. = 0.04783   n = 96
> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



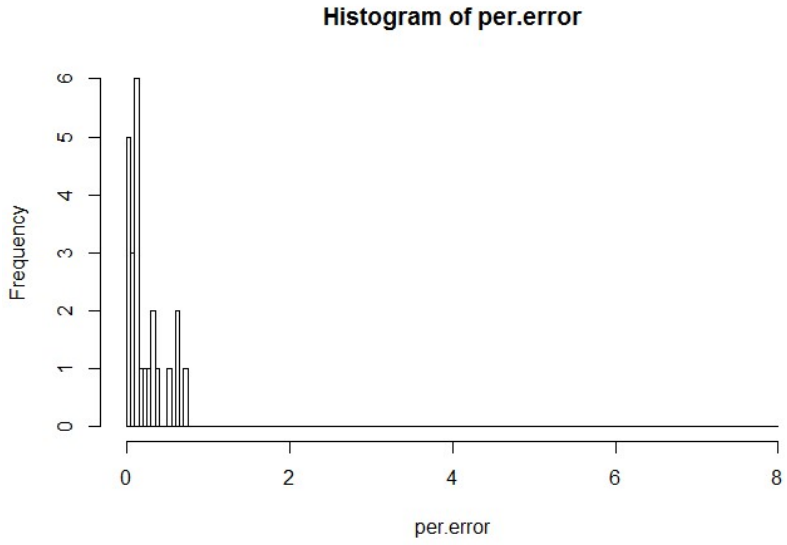

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
```

```
W = 0.96249, p-value = 0.007614


> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.2240343
> median(per.error)
[1] 0.1443068
> range(per.error)
[1] 0.01017955 0.70112286
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 5

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
```

```
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(92)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- gam(STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus
6post7_tmax) + s(f_minus6post7_tmin), data = traindata)
> summary(model)

Family: gaussian
Link function: identity

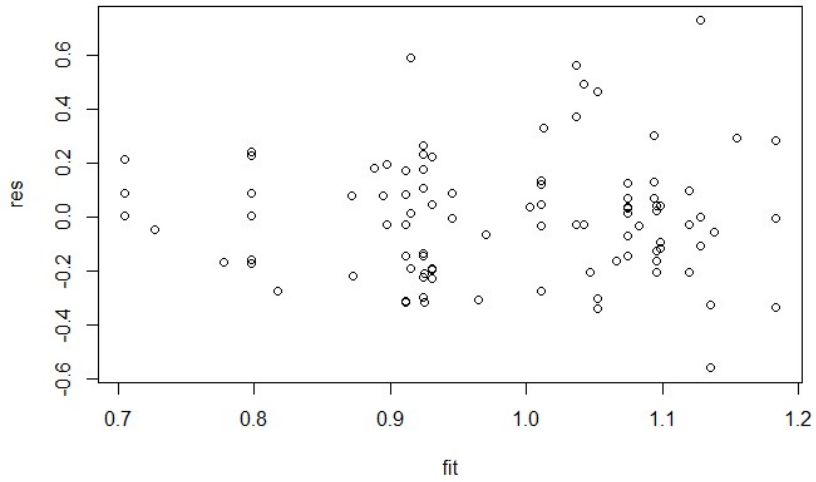
Formula:
STNDYieldperHAactual ~ s(f_post14_precip) + s(f_minus6post7_tmax) +
    s(f_minus6post7_tmin)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9839     0.0235   41.87   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

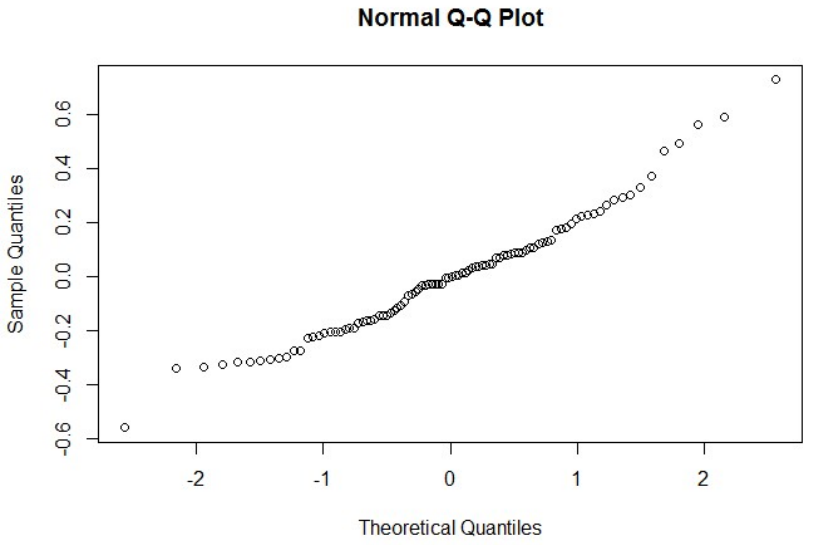
Approximate significance of smooth terms:
                      edf Ref.df      F  p-value
s(f_post14_precip)      1      1  4.141 0.044680 *
s(f_minus6post7_tmax)   1      1 18.604 3.89e-05 ***
s(f_minus6post7_tmin)   1      1 12.007 0.000798 ***
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.188   Deviance explained = 21.3%
GCV = 0.055311  Scale est. = 0.053007  n = 96
> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```

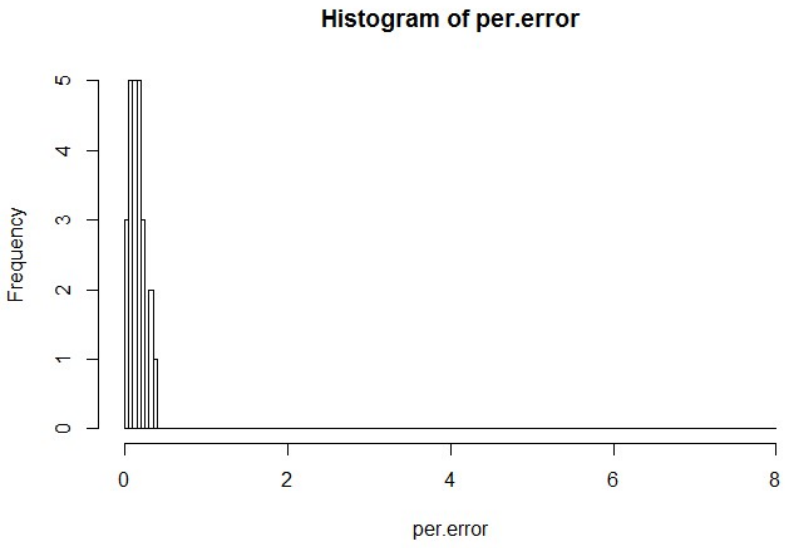



```
> shapiro.test(res)

        Shapiro-Wilk normality test
```

```
data:  res
W = 0.97273, p-value = 0.04261


> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.1498264
> median(per.error)
[1] 0.1324928
> range(per.error)
[1] 0.00708687 0.37721819
> hist(per.error, breaks = seq(0, 8, by = 0.05))
>
```



5.2 Linear Regression Model

Generic scripts to CV this final Linear Model

```
setwd("~/Grapevine Project/Combined grower data for analysis/standar
dised yields")
InputYieldData <- read.csv("~/Grapevine Project/Combined grower data
 for analysis/standardised yields/InputYieldData.csv")
View(InputYieldData)
##organise this data and omit 2011 yields
```

this will sort data by year

```
sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),]
View(sorted.by.year)
```

below scripts will help to read off row labels;

```
x <- c(1:136)
write.table(x, file = "x.txt")
x <- read.csv("~/Grapevine Project/Combined grower data for analysis
/standardised yields/x.txt", sep="")
sorted.by.year.row.label <- cbind(x, sorted.by.year)
View(sorted.by.year.row.label)
```

##set up TRAINING DATA SET this will set-up the training data set

```
years.to.2011 <- sorted.by.year.row.label[1:15,]
View(years.to.2011)
years.from.2011 <- sorted.by.year.row.label[32:136,]
View(years.from.2011)
train.data <- rbind(years.to.2011,years.from.2011)
View(train.data)
##Randomly divide training data set into training and test sets.See
above scripts for reading in original data set.
set.seed(92)
rand <- sample(1:500, 120)
train <- cbind(train.data, rand)
train.sort <- train[order(train$rand),]
traindata <- train.sort[1:96,]
testdata <- train.sort[97:120,]
model <- lm(STNDYieldperHAactual ~  f_post14_precip + f_minus6post7_
tmax + f_minus6post7_tmin, data = traindata)
summary(model)
res <- residuals(model)
fit <- fitted(model)
plot(fit,res)
qqnorm(res)
shapiro.test(res)
est <- predict(model, testdata)
write.table(est, file = "fitted.txt")
```

```
fitted <- read.csv("~/Grapevine Project/Combined grower data for ana
lysis/standardised yields/fitted.txt", sep="")
View(fitted)
per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testdat
a$STNDYieldperHAactual
mean(per.error)
median(per.error)
range(per.error)
hist(per.error, breaks = seq(0, 8, by = 0.05))
```

CV 1

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(123)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- lm(STNDYieldperHAactual ~  f_post14_precip + f_minus6post
7_tmax + f_minus6post7_tmin, data = traindata)
> summary(model)
```

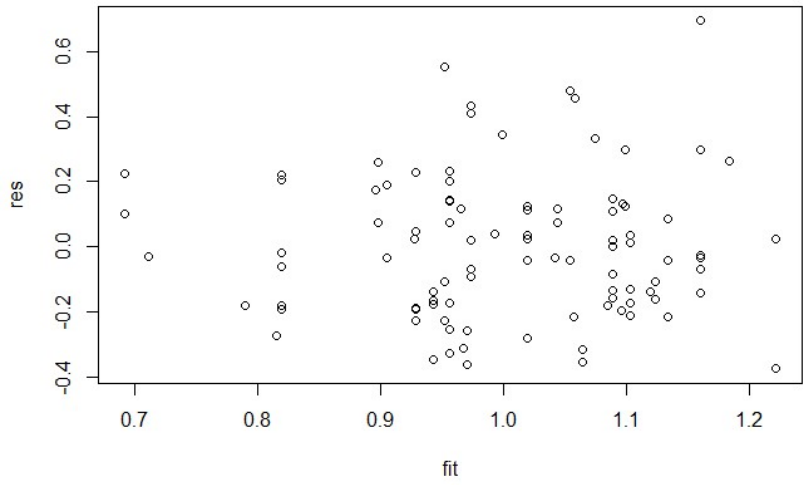
```
Call:
lm(formula = STNDYieldperHAactual ~ f_post14_precip + f_minus6post7_
tmax +
    f_minus6post7_tmin, data = traindata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.37620 -0.17487 -0.02693  0.12624  0.69496

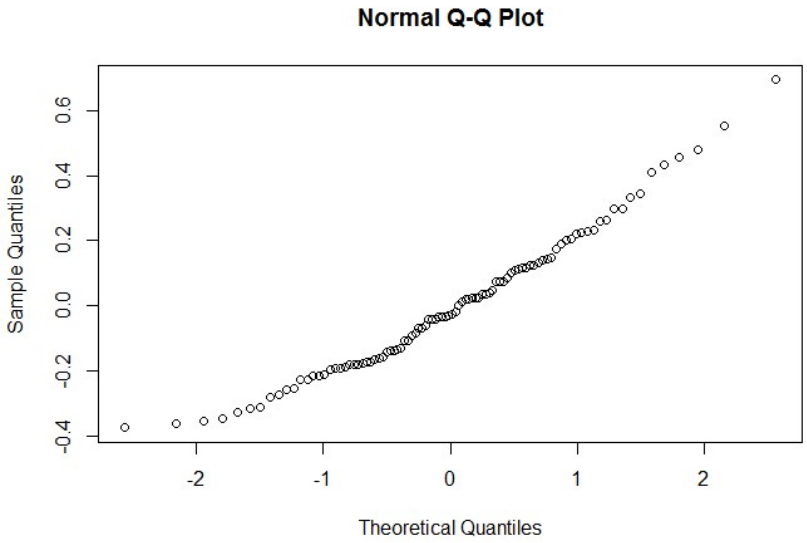
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.24679    0.20711   1.192  0.23647
f_post14_precip    -0.04879    0.02391  -2.041  0.04412 *
f_minus6post7_tmax  0.04613    0.01009   4.572  1.5e-05 ***
f_minus6post7_tmin -0.03701    0.01116  -3.316  0.00131 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2239 on 92 degrees of freedom
Multiple R-squared:  0.2125,   Adjusted R-squared:  0.1868
F-statistic: 8.276 on 3 and 92 DF,  p-value: 6.228e-05

> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```

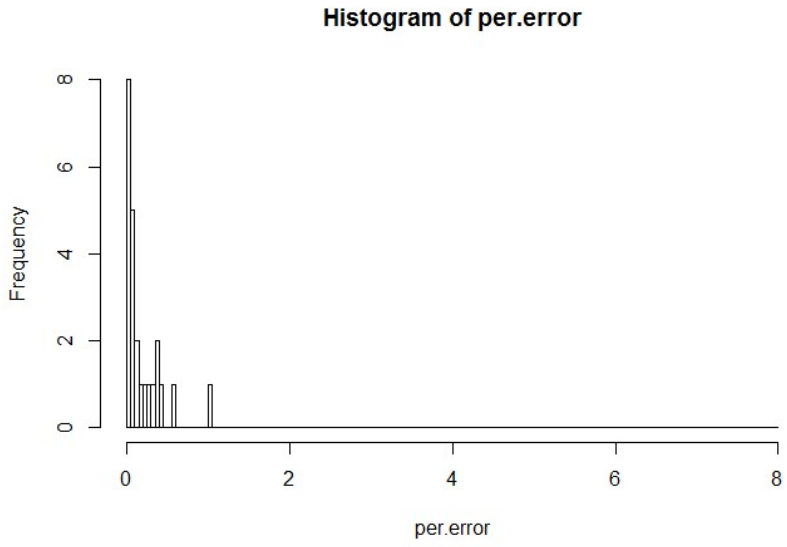



```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.97016, p-value = 0.02741


> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.1875739
> median(per.error)
[1] 0.07419729
> range(per.error)
[1] 0.01569236 1.01615138
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 2

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(25)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
```

```
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- lm(STNDYieldperHAactual ~  f_post14_precip + f_minus6post
7_tmax + f_minus6post7_tmin, data = traindata)
> summary(model)

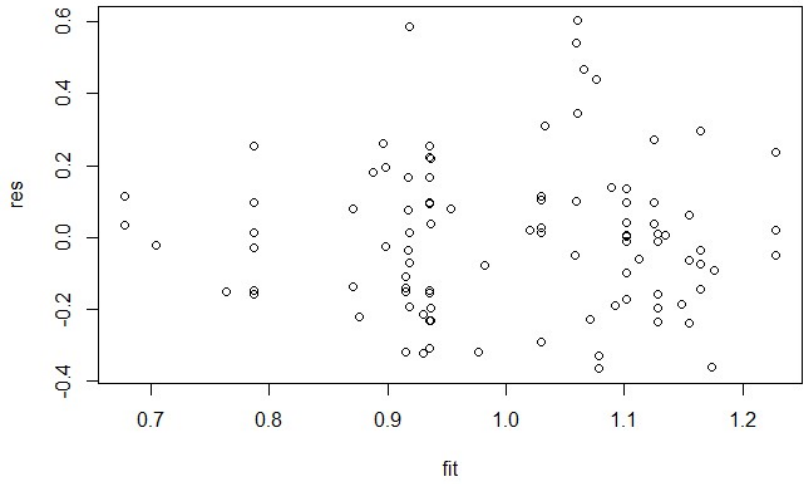
Call:
lm(formula = STNDYieldperHAactual ~ f_post14_precip + f_minus6post7_
tmax +
    f_minus6post7_tmin, data = traindata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.36694 -0.15342 -0.00349  0.10152  0.60316

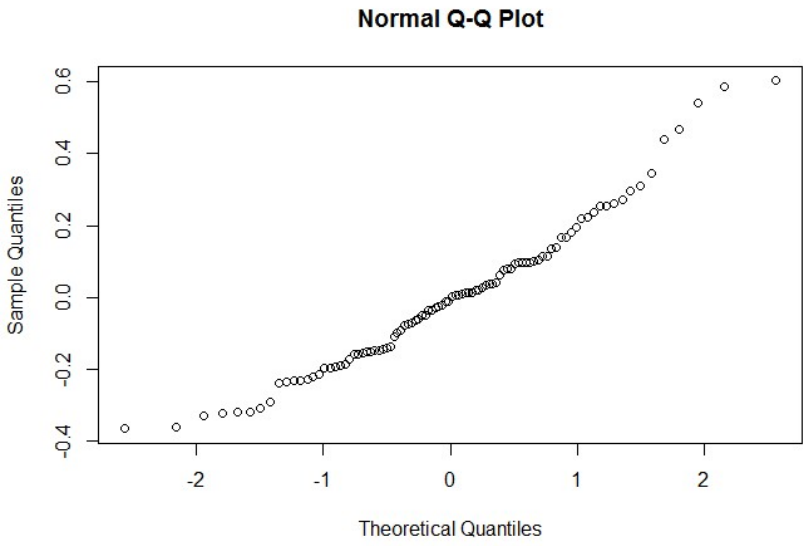
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.268443   0.208397   1.288   0.2009
f_post14_precip    -0.053877   0.021932  -2.457   0.0159 *
f_minus6post7_tmax  0.049267   0.009971   4.941 3.47e-06 ***
f_minus6post7_tmin -0.046774   0.010768  -4.344 3.60e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2129 on 92 degrees of freedom
Multiple R-squared:  0.276,    Adjusted R-squared:  0.2524
F-statistic: 11.69 on 3 and 92 DF,  p-value: 1.474e-06

> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



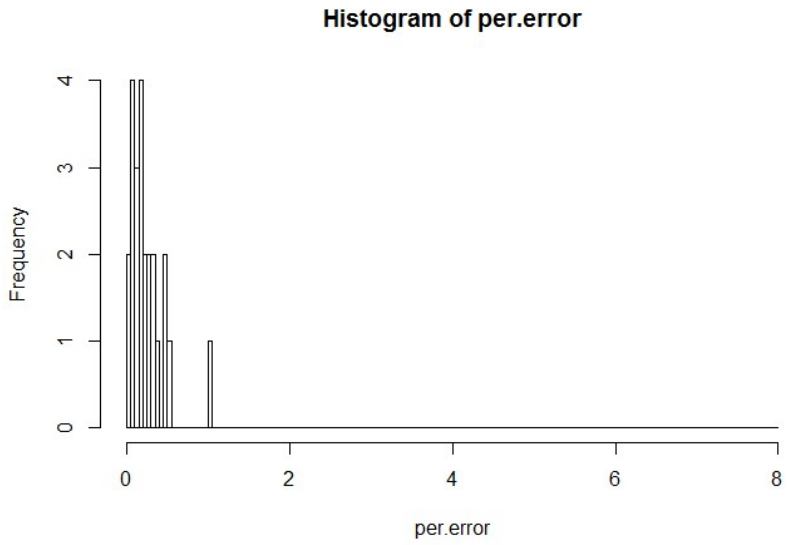

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.96564, p-value = 0.01278

> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
```

```
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.2445156
> median(per.error)
[1] 0.1755402
> range(per.error)
[1] 0.01450896 1.04278692
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 3

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
```

```
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(46)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- lm(STNDYieldperHAactual ~  f_post14_precip + f_minus6post
7_tmax + f_minus6post7_tmin, data = traindata)
> summary(model)

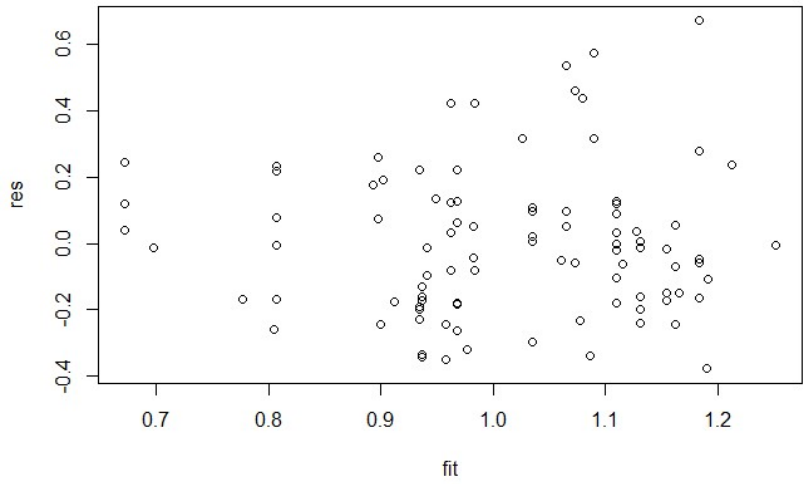

Call:
lm(formula = STNDYieldperHAactual ~ f_post14_precip + f_minus6post7_
tmax +
    f_minus6post7_tmin, data = traindata)


Residuals:
     Min       1Q   Median       3Q      Max
-0.37907 -0.16852 -0.01286  0.11907  0.67206

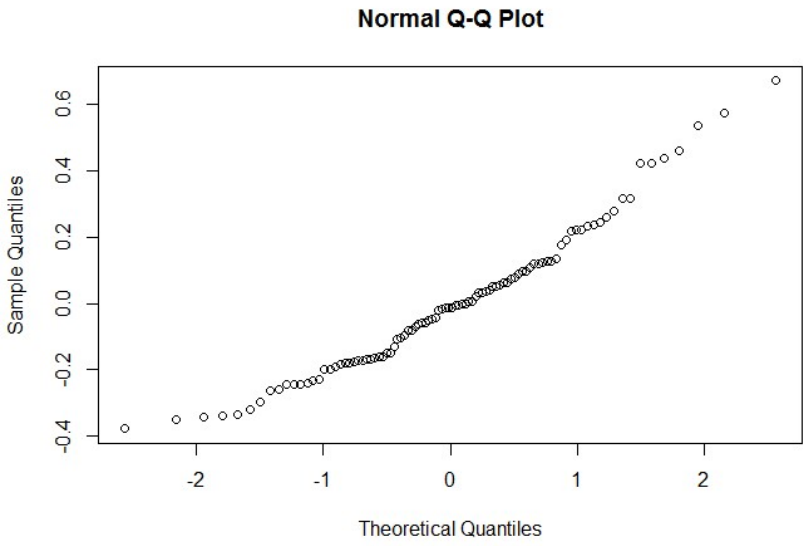

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.199021   0.200032   0.995   0.3224
f_post14_precip    -0.050098   0.024197  -2.070   0.0412 *
f_minus6post7_tmax  0.051493   0.009808   5.250 9.74e-07 ***
f_minus6post7_tmin -0.044546   0.010921  -4.079 9.62e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.2232 on 92 degrees of freedom
Multiple R-squared:  0.2615,   Adjusted R-squared:  0.2374
F-statistic: 10.86 on 3 and 92 DF,  p-value: 3.57e-06


> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



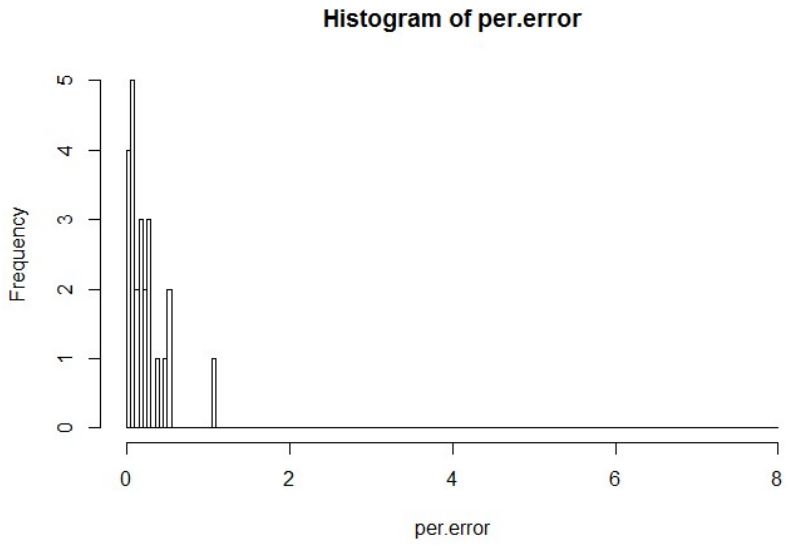

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.96273, p-value = 0.007925

> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
```

```
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.2302673
> median(per.error)
[1] 0.1628447
> range(per.error)
[1] 0.007838943 1.070537617
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 4

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:15,]
```

```
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(80)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- lm(STNDYieldperHAactual ~  f_post14_precip + f_minus6post
7_tmax + f_minus6post7_tmin, data = traindata)
> summary(model)

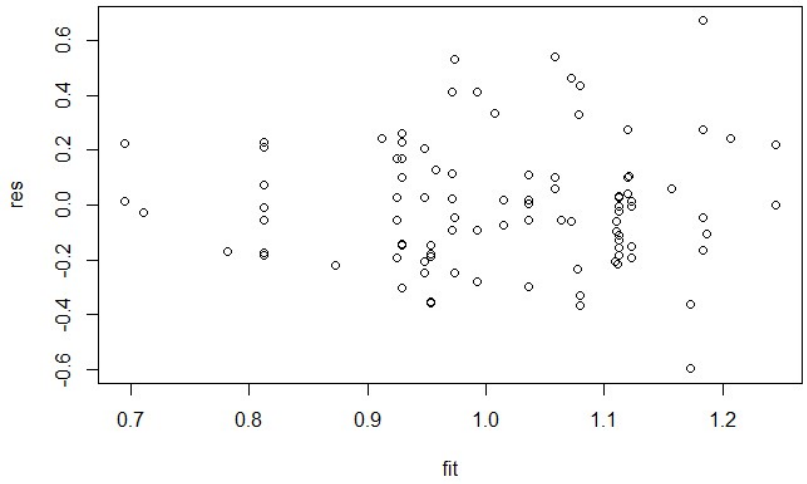

Call:
lm(formula = STNDYieldperHAactual ~ f_post14_precip + f_minus6post7_
tmax +
    f_minus6post7_tmin, data = traindata)


Residuals:
     Min       1Q   Median       3Q      Max
-0.59801 -0.16672 -0.00863  0.10930  0.67233

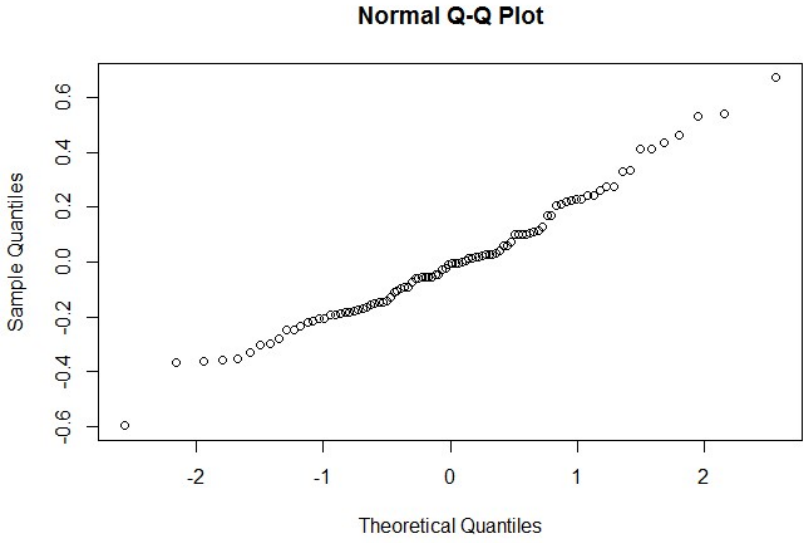

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.26016    0.22256   1.169  0.24543
f_post14_precip    -0.06392    0.02448  -2.612  0.01052 *
f_minus6post7_tmax  0.04666    0.01079   4.325 3.86e-05 ***
f_minus6post7_tmin -0.03737    0.01123  -3.329  0.00125 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2307 on 92 degrees of freedom
Multiple R-squared:  0.2249,   Adjusted R-squared:  0.1996
F-statistic: 8.896 on 3 and 92 DF,  p-value: 3.09e-05


> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



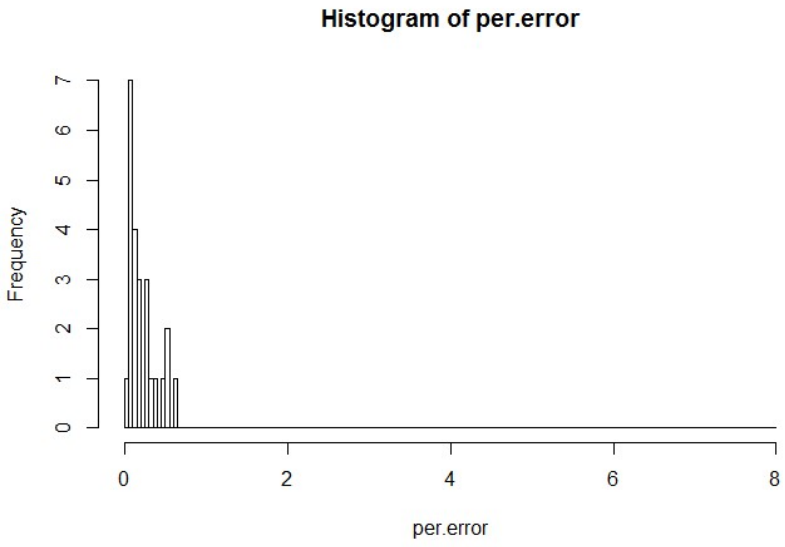

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.9794, p-value = 0.1357

> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
```

```
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.2102585
> median(per.error)
[1] 0.1501554
> range(per.error)
[1] 0.00944328 0.63406917
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



CV 5

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> View(sorted.by.year)
> ## below scripts will help to read off row labels;
> x <- c(1:136)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> View(sorted.by.year.row.label)
> ##set up TRAINING DATA SET this will set-up the training data set
```

```
> years.to.2011 <- sorted.by.year.row.label[1:15,]
> View(years.to.2011)
> years.from.2011 <- sorted.by.year.row.label[32:136,]
> View(years.from.2011)
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(92)
> rand <- sample(1:500, 120)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:120,]
> model <- lm(STNDYieldperHAactual ~  f_post14_precip + f_minus6post
7_tmax + f_minus6post7_tmin, data = traindata)
> summary(model)

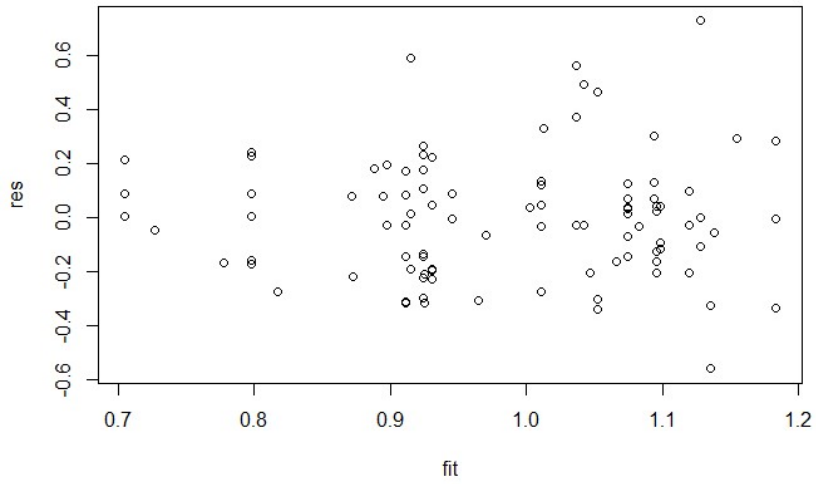

Call:
lm(formula = STNDYieldperHAactual ~ f_post14_precip + f_minus6post7_
tmax +
    f_minus6post7_tmin, data = traindata)


Residuals:
     Min       1Q   Median       3Q      Max
-0.56074 -0.16580 -0.00379  0.11061  0.72803

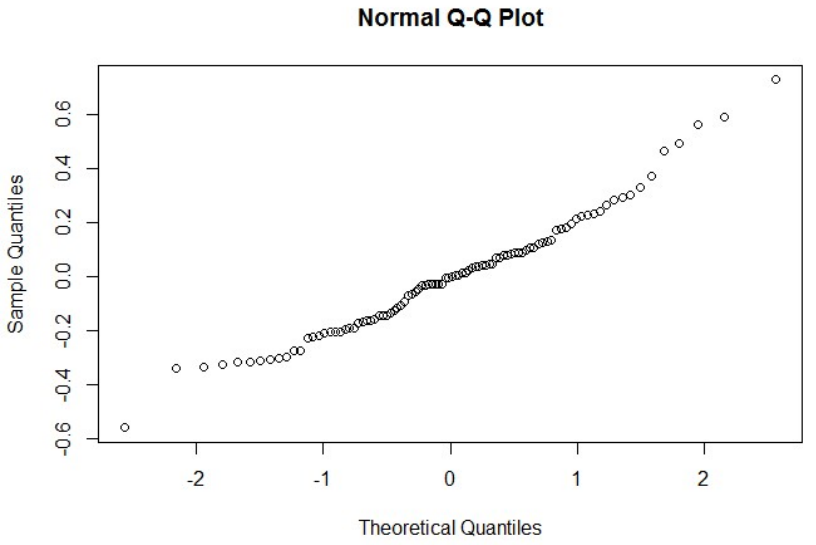

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.351822   0.208167   1.690 0.094396 .
f_post14_precip    -0.048483   0.023825  -2.035 0.044732 *
f_minus6post7_tmax  0.042676   0.009894   4.313 4.04e-05 ***
f_minus6post7_tmin -0.040467   0.011678  -3.465 0.000807 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.2302 on 92 degrees of freedom
Multiple R-squared:  0.2135,   Adjusted R-squared:  0.1878
F-statistic: 8.323 on 3 and 92 DF,  p-value: 5.9e-05


> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



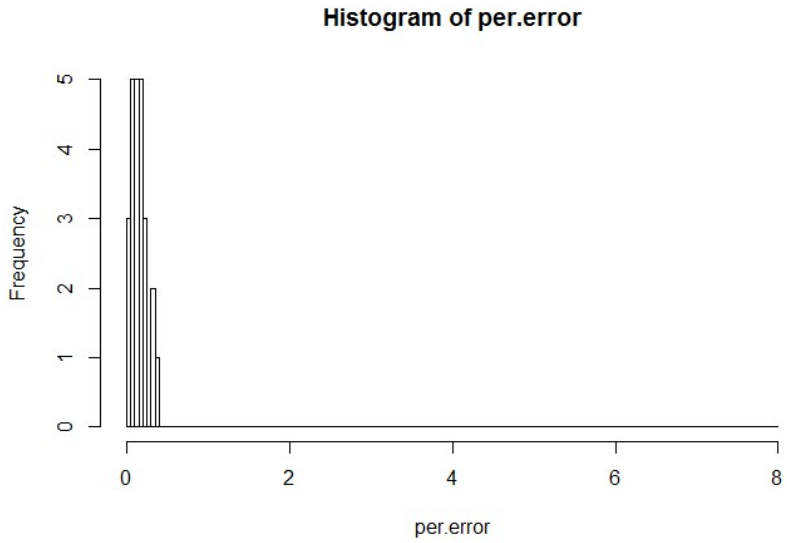

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.97273, p-value = 0.04261

> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
```

```
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.1498264
> median(per.error)
[1] 0.1324928
> range(per.error)
[1] 0.00708687 0.37721819
> hist(per.error, breaks = seq(0, 8, by = 0.05))
```



5.3 ANN

ANN via CARET- F dates only- with categorical variables

Generic scrip to CV ANN fitted via CARET;

```
setwd("~/Grapevine Project/Combined grower data for analysis/standar
dised yields")

InputYieldData <- read.csv("~/Grapevine Project/Combined grower data
 for analysis/standardised yields/InputYieldData.csv")

##organise this data and omit 2011 yields
```

this will sort data by year

```
sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),]
```

below scripts will help to read off row labels;

```
x <- c(1:246)

write.table(x, file = "x.txt")

x <- read.csv("~/Grapevine Project/Combined grower data for analysis
/standardised yields/x.txt", sep="")

sorted.by.year.row.label <- cbind(x, sorted.by.year)
```

##set up TRAINING DATA SET this will set-up the training data set

```
years.to.2011 <- sorted.by.year.row.label[1:102,]

years.from.2011 <- sorted.by.year.row.label[129:246,]

train.data <- rbind(years.to.2011,years.from.2011)

##Randomly divide training data set into training and test sets.See
above scripts for reading in original data set.

set.seed(92)

rand <- sample(1:1000, 220)

train <- cbind(train.data, rand)

train.sort <- train[order(train$rand),]

traindata <- train.sort[1:176,]

testdata <- train.sort[177:220,]

View(traindata)

View(testdata)

model <- caret::train(STNDYieldperHAactual ~ Preiv.yr.STNDYieldperHA
actual + Grower + Vineyard + f_pre14_tmax + f_pre14_tmin + f_pre14_p
recip + f_pre14_solar + f_post14_tmax + f_post14_tmin + f_post14_pre
cip + f_post14_solar + f_minus6post7_tmax + f_minus6post7_tmin + f_m
inus6post7_precip + f_minus6post7_solar, data = traindata, method='n
net', linout=TRUE, trace=FALSE, tuneGrid=expand.grid(.size=c(1,5,10)
,.decay=c(0,0.001,0.1)))

est <- predict(model, testdata)

write.table(est, file = "fitted.txt")

fitted <- read.csv("~/Grapevine Project/Combined grower data for ana
lysis/standardised yields/fitted.txt", sep="")

View(fitted)
per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testdat
a$STNDYieldperHAactual

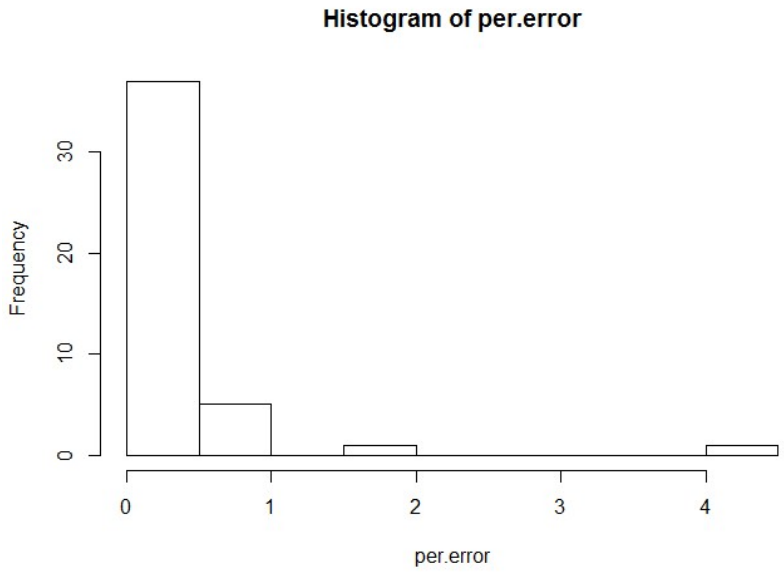
mean(per.error)
```

```
median(per.error)
range(per.error)
hist(per.error)
hist(per.error, breaks = seq(0, 10, by = 0.05))
model <- lm(testdata$STNDYieldperHAactual ~ fitted$x)
summary(model)
plot(testdata$STNDYieldperHAactual,fitted$x)
abline(a=0, b=1)
```

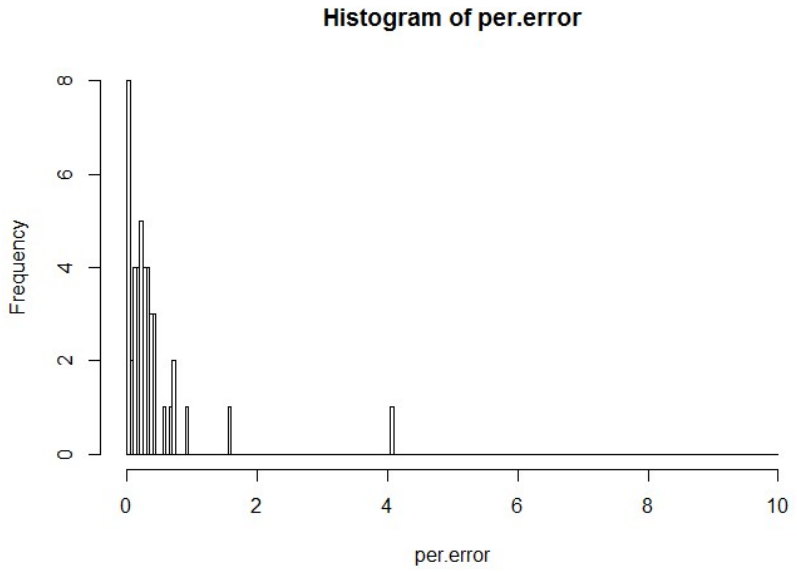
CV 1

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ## below scripts will help to read off row labels;
> x <- c(1:246)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:102,]
> years.from.2011 <- sorted.by.year.row.label[129:246,]
> train.data <- rbind(years.to.2011,years.from.2011)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(123)
> rand <- sample(1:1000, 220)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:176,]
> testdata <- train.sort[177:220,]
> View(traindata)
> View(testdata)
> model <- caret::train(STNDYieldperHAactual ~ Preiv.yr.STNDYieldper
HAactual + Grower + Vineyard + f_pre14_tmax + f_pre14_tmin + f_pre14
_precip + f_pre14_solar + f_post14_tmax + f_post14_tmin + f_post14_p
recip + f_post14_solar + f_minus6post7_tmax + f_minus6post7_tmin + f
_minus6post7_precip + f_minus6post7_solar, data = traindata, method=
'nnet', linout=TRUE, trace=FALSE, tuneGrid=expand.grid(.size=c(1,5,1
0),.decay=c(0,0.001,0.1)))
Loading required package: nnet
```

```
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInf
o,  :
  There were missing values in resampled performance measures.
> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.3802757
> median(per.error)
[1] 0.2375601
> range(per.error)
[1] 0.004897497 4.064171931
> hist(per.error)
```




```
> hist(per.error, breaks = seq(0, 10, by = 0.05))
```




```
> model <- lm(testdata$STNDYieldperHAactual ~ fitted$x)
> summary(model)

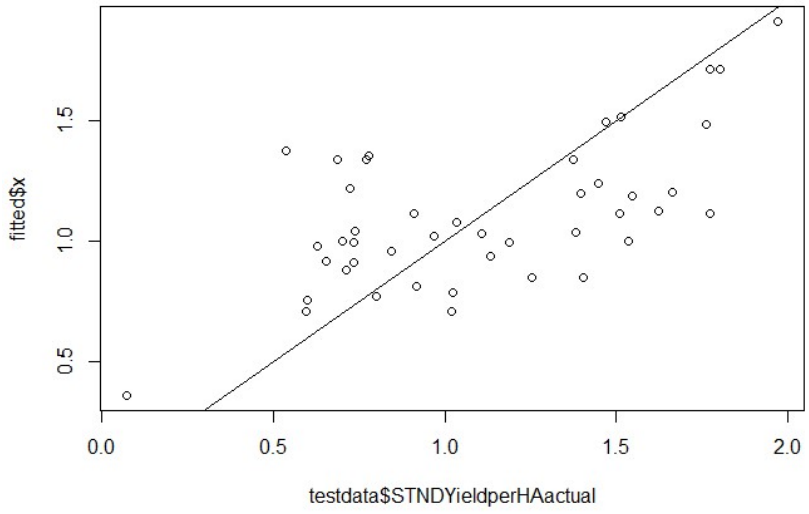
Call:
lm(formula = testdata$STNDYieldperHAactual ~ fitted$x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.81867 -0.22778  0.03838  0.22746  0.65319

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.1106     0.2029   0.545    0.589
fitted$x      0.9050     0.1775   5.100  7.7e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.35 on 42 degrees of freedom
Multiple R-squared:  0.3824,   Adjusted R-squared:  0.3677
F-statistic: 26.01 on 1 and 42 DF,  p-value: 7.698e-06

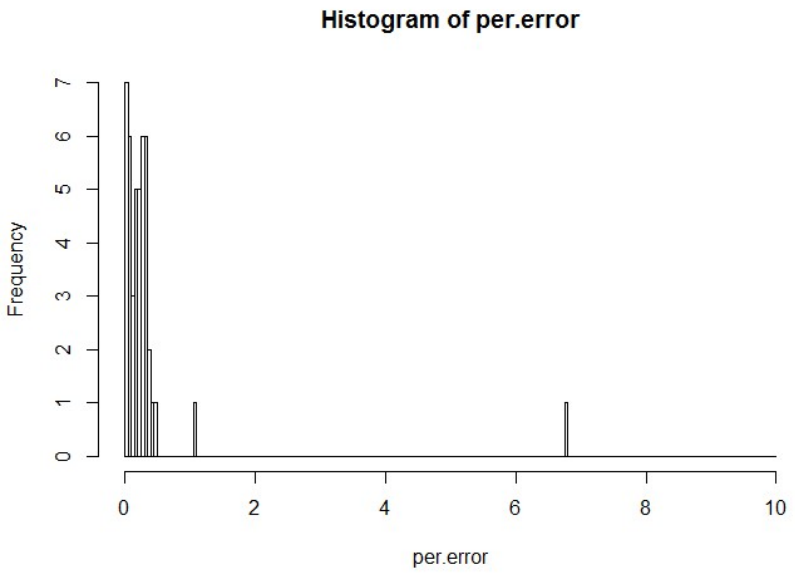
> plot(testdata$STNDYieldperHAactual,fitted$x)
> abline(a=0, b=1)
```



CV 2

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ## below scripts will help to read off row labels;
> x <- c(1:246)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:102,]
> years.from.2011 <- sorted.by.year.row.label[129:246,]
> train.data <- rbind(years.to.2011,years.from.2011)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(25)
> rand <- sample(1:1000, 220)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:176,]
```

```
> testdata <- train.sort[177:220,]
> View(traindata)
> View(testdata)
> model <- caret::train(STNDYieldperHAactual ~ Preiv.yr.STNDYieldper
HAactual + Grower + Vineyard + f_pre14_tmax + f_pre14_tmin + f_pre14
_precip + f_pre14_solar + f_post14_tmax + f_post14_tmin + f_post14_p
recip + f_post14_solar + f_minus6post7_tmax + f_minus6post7_tmin + f
_minus6post7_precip + f_minus6post7_solar, data = traindata, method=
'nnet', linout=TRUE, trace=FALSE, tuneGrid=expand.grid(.size=c(1,5,1
0),.decay=c(0,0.001,0.1)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInf
o,  :
  There were missing values in resampled performance measures.
> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.3657269
> median(per.error)
[1] 0.2069333
> range(per.error)
[1] 0.00238968 6.76005531
> hist(per.error)
> hist(per.error, breaks = seq(0, 10, by = 0.05))
```




```
> model <- lm(testdata$STNDYieldperHAactual ~ fitted$x)
> summary(model)
```

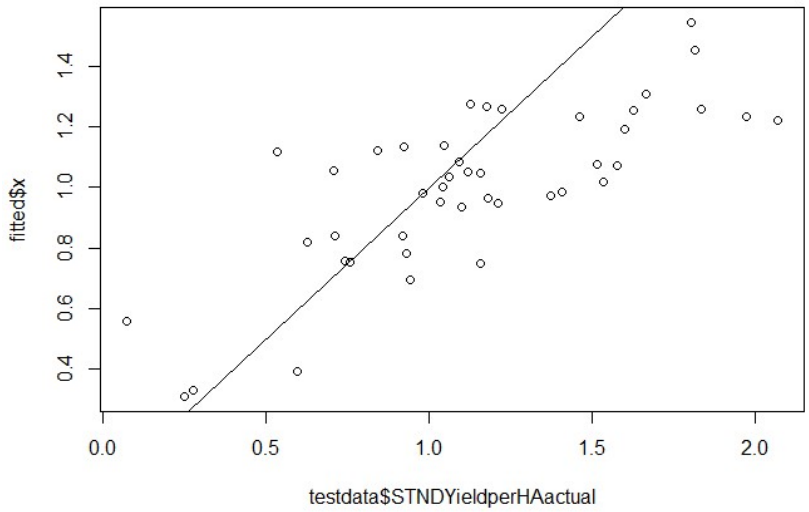
```
Call:
lm(formula = testdata$STNDYieldperHAactual ~ fitted$x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.74996 -0.16526  0.00655  0.20838  0.65187

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1737     0.1728  -1.005    0.321
fitted$x      1.3051     0.1670   7.815 1.02e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2957 on 42 degrees of freedom
Multiple R-squared:  0.5925,   Adjusted R-squared:  0.5828
F-statistic: 61.07 on 1 and 42 DF,  p-value: 1.016e-09
```

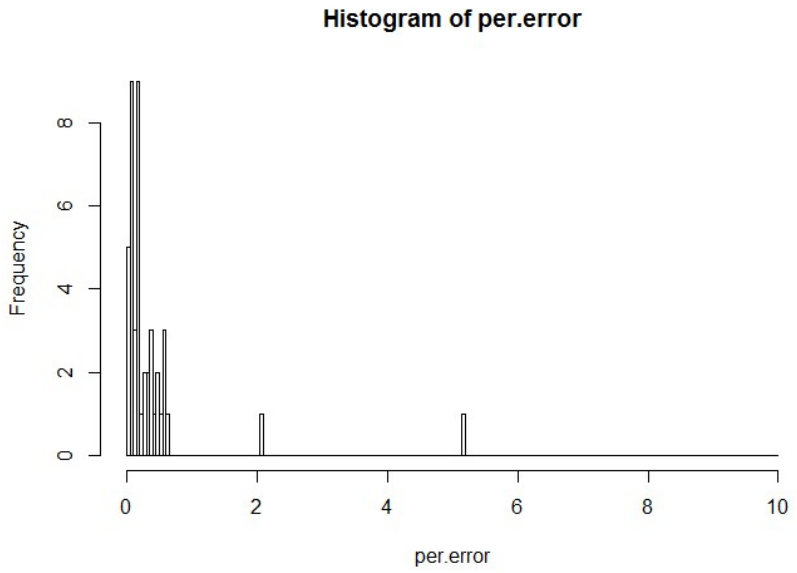
```
> plot(testdata$STNDYieldperHAactual,fitted$x)
> abline(a=0, b=1)
```



CV 3

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ## below scripts will help to read off row labels;
> x <- c(1:246)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:102,]
> years.from.2011 <- sorted.by.year.row.label[129:246,]
> train.data <- rbind(years.to.2011,years.from.2011)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(46)
> rand <- sample(1:1000, 220)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:176,]
> testdata <- train.sort[177:220,]
> View(traindata)
> View(testdata)
> model <- caret::train(STNDYieldperHAactual ~ Preiv.yr.STNDYieldper
HAactual + Grower + Vineyard + f_pre14_tmax + f_pre14_tmin + f_pre14
_precip + f_pre14_solar + f_post14_tmax + f_post14_tmin + f_post14_p
recip + f_post14_solar + f_minus6post7_tmax + f_minus6post7_tmin + f
_minus6post7_precip + f_minus6post7_solar, data = traindata, method=
'nnet', linout=TRUE, trace=FALSE, tuneGrid=expand.grid(.size=c(1,5,1
0),.decay=c(0,0.001,0.1)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInf
o,  :
  There were missing values in resampled performance measures.
> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.3809489
> median(per.error)
```

```
[1] 0.1797587
> range(per.error)
[1] 0.0001392552 5.1740580573
> hist(per.error)
> hist(per.error, breaks = seq(0, 10, by = 0.05))
```




```
> model <- lm(testdata$STNDYieldperHAactual ~ fitted$x)
> summary(model)

Call:
lm(formula = testdata$STNDYieldperHAactual ~ fitted$x)

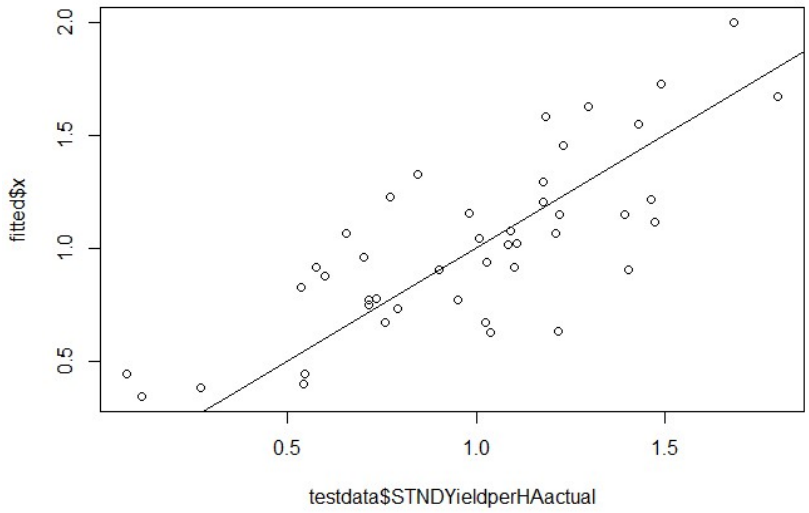
Residuals:
     Min       1Q   Median       3Q      Max
-0.46758 -0.17830  0.00559  0.13822  0.53188

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.1938     0.1078   1.797   0.0795 .
fitted$x      0.7796     0.1000   7.793 1.09e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2515 on 42 degrees of freedom
Multiple R-squared:  0.5912,   Adjusted R-squared:  0.5814
F-statistic: 60.73 on 1 and 42 DF,  p-value: 1.091e-09

> plot(testdata$STNDYieldperHAactual,fitted$x)
```

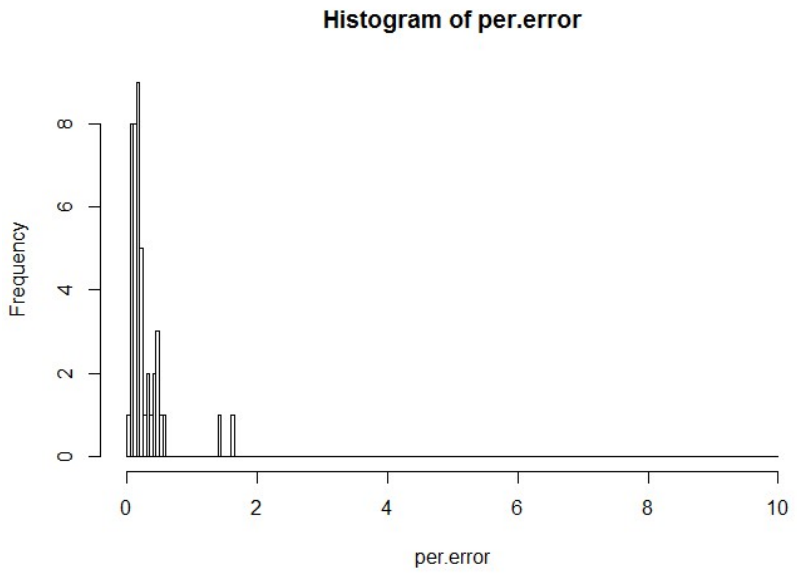
```
> abline(a=0, b=1)
```



CV 4

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ## below scripts will help to read off row labels;
> x <- c(1:246)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:102,]
> years.from.2011 <- sorted.by.year.row.label[129:246,]
> train.data <- rbind(years.to.2011,years.from.2011)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(80)
> rand <- sample(1:1000, 220)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:176,]
```

```
> testdata <- train.sort[177:220,]
> View(traindata)
> View(testdata)
> model <- caret::train(STNDYieldperHAactual ~ Preiv.yr.STNDYieldper
HAactual + Grower + Vineyard + f_pre14_tmax + f_pre14_tmin + f_pre14
_precip + f_pre14_solar + f_post14_tmax + f_post14_tmin + f_post14_p
recip + f_post14_solar + f_minus6post7_tmax + f_minus6post7_tmin + f
_minus6post7_precip + f_minus6post7_solar, data = traindata, method=
'nnet', linout=TRUE, trace=FALSE, tuneGrid=expand.grid(.size=c(1,5,1
0),.decay=c(0,0.001,0.1)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInf
o,  :
  There were missing values in resampled performance measures.
> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.2743206
> median(per.error)
[1] 0.1728725
> range(per.error)
[1] 0.04155466 1.62990238
> hist(per.error)
> hist(per.error, breaks = seq(0, 10, by = 0.05))
```




```
> model <- lm(testdata$STNDYieldperHAactual ~ fitted$x)
> summary(model)
```

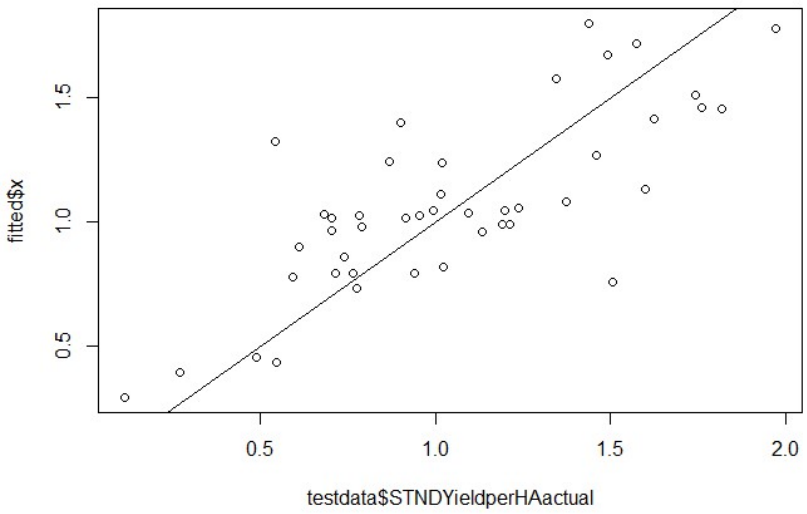
```
Call:
lm(formula = testdata$STNDYieldperHAactual ~ fitted$x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.74074 -0.18472 -0.04401  0.20980  0.74734

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.05767    0.13588   0.424    0.673
fitted$x     0.92514    0.12024   7.694  1.5e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2822 on 42 degrees of freedom
Multiple R-squared:  0.585,    Adjusted R-squared:  0.5751
F-statistic:  59.2 on 1 and 42 DF,  p-value: 1.503e-09
```

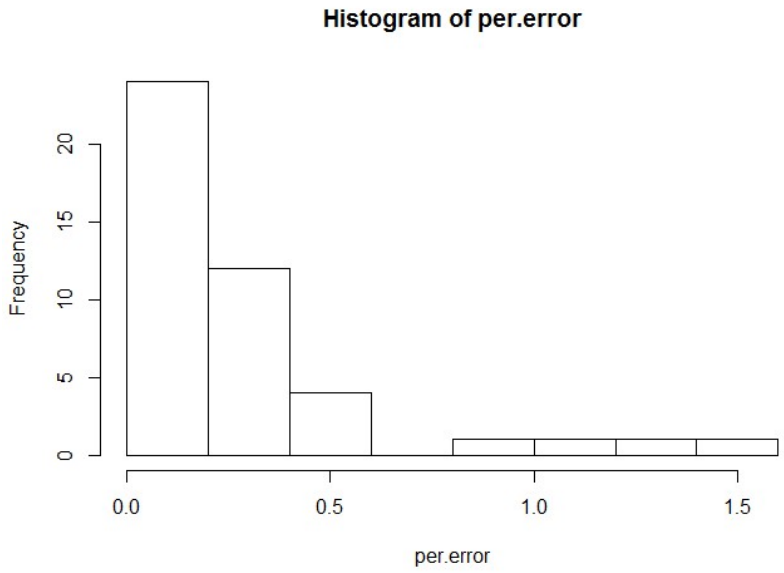
```
> plot(testdata$STNDYieldperHAactual,fitted$x)
> abline(a=0, b=1)
```



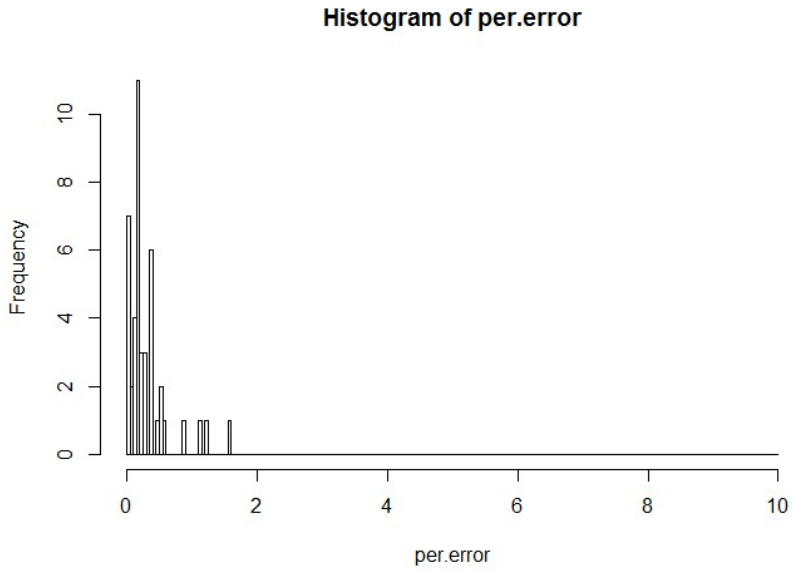
CV 5

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> ##organise this data and omit 2011 yields
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ## below scripts will help to read off row labels;
> x <- c(1:246)
> write.table(x, file = "x.txt")
> x <- read.csv("~/Grapevine Project/Combined grower data for analys
is/standardised yields/x.txt", sep="")
> sorted.by.year.row.label <- cbind(x, sorted.by.year)
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year.row.label[1:102,]
> years.from.2011 <- sorted.by.year.row.label[129:246,]
> train.data <- rbind(years.to.2011,years.from.2011)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(92)
> rand <- sample(1:1000, 220)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:176,]
> testdata <- train.sort[177:220,]
> View(traindata)
> View(testdata)
> model <- caret::train(STNDYieldperHAactual ~ Preiv.yr.STNDYieldper
HAactual + Grower + Vineyard + f_pre14_tmax + f_pre14_tmin + f_pre14
_precip + f_pre14_solar + f_post14_tmax + f_post14_tmin + f_post14_p
recip + f_post14_solar + f_minus6post7_tmax + f_minus6post7_tmin + f
_minus6post7_precip + f_minus6post7_solar, data = traindata, method=
'nnet', linout=TRUE, trace=FALSE, tuneGrid=expand.grid(.size=c(1,5,1
0),.decay=c(0,0.001,0.1)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInf
o,  :
  There were missing values in resampled performance measures.
> est <- predict(model, testdata)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$STNDYieldperHAactual) / testd
ata$STNDYieldperHAactual
> mean(per.error)
[1] 0.3038485
> median(per.error)
[1] 0.1738754
```

```
> range(per.error)
[1] 0.015532 1.591700
> hist(per.error)
```




```
> hist(per.error, breaks = seq(0, 10, by = 0.05))
```




```
> model <- lm(testdata$STNDYieldperHAactual ~ fitted$x)
> summary(model)

Call:
lm(formula = testdata$STNDYieldperHAactual ~ fitted$x)

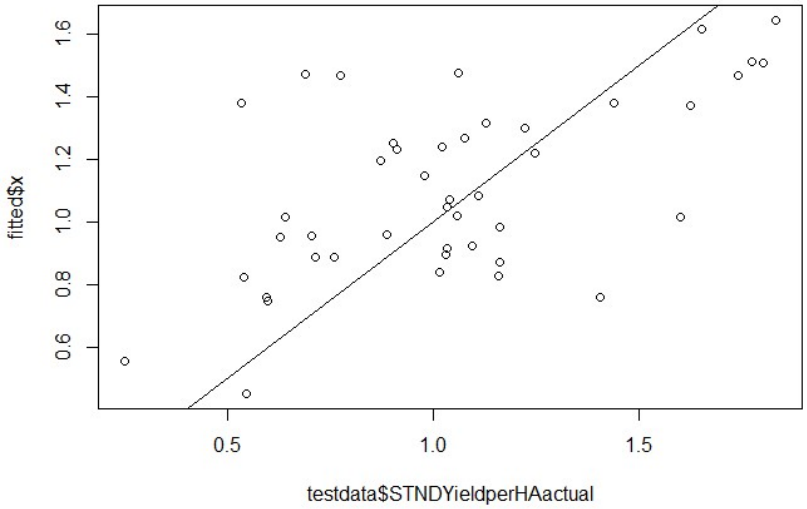
Residuals:
     Min       1Q   Median       3Q      Max
```

```
-0.72916 -0.22460  0.02111  0.19717  0.63581

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.1658     0.1914   0.866    0.391
fitted$x      0.7944     0.1675   4.744 2.43e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3149 on 42 degrees of freedom
Multiple R-squared:  0.3489,   Adjusted R-squared:  0.3334
F-statistic:  22.5 on 1 and 42 DF,  p-value: 2.435e-05

> plot(testdata$STNDYieldperHAactual,fitted$x)
> abline(a=0, b=1)
```



5.4 Mixed Linear Model

Generic scripts to CV this mixed linear model

```
InputYieldData <- read.csv("~/Grapevine Project/Combined grower data
 for analysis/standardised yields/InputYieldData.csv")
View(InputYieldData)
```

this will sort data by year

```
sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),]
```

##set up TRAINING DATA SET this will set-up the training data set

```
years.to.2011 <- sorted.by.year[1:14,]
years.from.2011 <- sorted.by.year[30:134,]
train.data <- rbind(years.to.2011,years.from.2011)
View(train.data)
##Randomly divide training data set into training and test sets.See
above scripts for reading in original data set.
set.seed(92)
rand <- sample(1:500, 119)
train <- cbind(train.data, rand)
train.sort <- train[order(train$rand),]
traindata <- train.sort[1:96,]
testdata <- train.sort[97:119,]
model <- lmer(formula = ActualYieldperHA ~ f_pre14_tmin + f_minus6po
st7_tmax + f_minus6post7_tmin + f_minus6post7_precip + f_minus6post7
_solar + (1|Variety), data = traindata)
summary(model)
res <- residuals(model)
fit <- fitted(model)
plot(fit,res)
qqnorm(res)
shapiro.test(res)
est <- predict(model, testdata, allow.new.levels = T)
write.table(est, file = "fitted.txt")
fitted <- read.csv("~/Grapevine Project/Combined grower data for ana
lysis/standardised yields/fitted.txt", sep="")
View(fitted)
per.error <- abs(fitted$x - testdata$ActualYieldperHA) / testdata
$ActualYieldperHA
mean(per.error)
median(per.error)
range(per.error)
hist(per.error, breaks = seq(0, 3.55, by = 0.05))
```

CV 1 – is extrapolating

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year[1:14,]
> years.from.2011 <- sorted.by.year[30:134,]
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(123)
> rand <- sample(1:500, 119)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:119,]
> model <- lmer(formula = ActualYieldperHA ~ f_pre14_tmin + f_minus6
post7_tmax + f_minus6post7_tmin + f_minus6post7_precip + f_minus6pos
t7_solar + (1|Variety), data = traindata)
> summary(model)
Linear mixed model fit by REML ['lmerMod']
Formula: ActualYieldperHA ~ f_pre14_tmin + f_minus6post7_tmax + f_mi
nus6post7_tmin +
    f_minus6post7_precip + f_minus6post7_solar + (1 | Variety)
   Data: traindata


REML criterion at convergence: 513.3


Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.85912 -0.47149 -0.04648  0.45982  3.09036


Random effects:
 Groups   Name        Variance Std.Dev.
 Variety  (Intercept) 38.567   6.210
 Residual              7.039   2.653
Number of obs: 96, groups:  Variety, 19


Fixed effects:
                    Estimate Std. Error t value
(Intercept)           3.6481     4.5680   0.799
f_pre14_tmin          0.8249     0.4210   1.959
```

```
f_minus6post7_tmax     1.7287     0.2778   6.223
f_minus6post7_tmin    -1.6958     0.3519  -4.820
f_minus6post7_precip  -1.4495     0.6549  -2.213
f_minus6post7_solar   -0.9913     0.2905  -3.413

Correlation of Fixed Effects:
               (Intr) f_p14_ f_mns6pst7_tmx f_mns6pst7_tmn f_mns6pst
7_p
f_pre14_tmn    -0.047

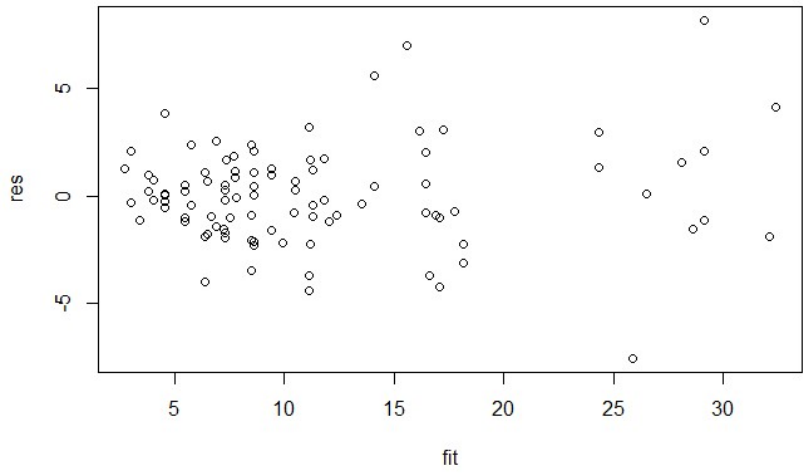
f_mns6pst7_tmx -0.275  0.651

f_mns6pst7_tmn  0.129 -0.843 -0.856

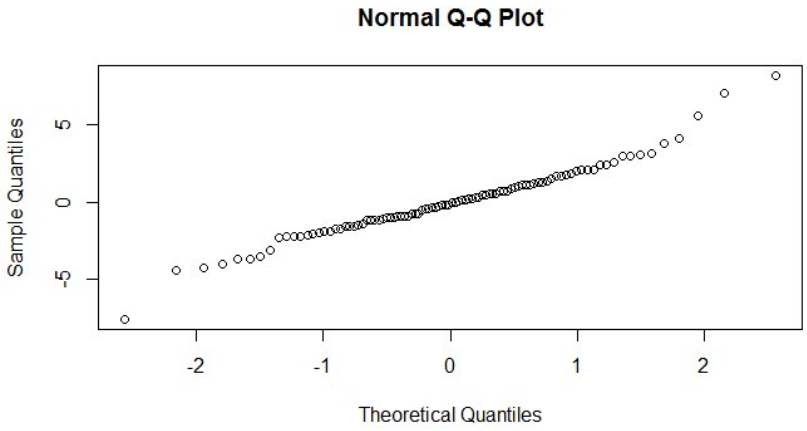
f_mns6pst7_p   -0.413 -0.470 -0.490          0.435

f_mns6pst7_s   -0.318 -0.759 -0.757          0.745          0.763
```

```
> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```

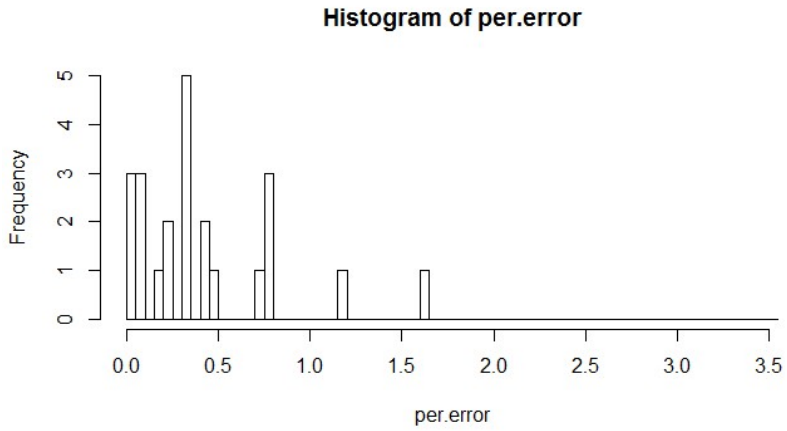



```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.965, p-value = 0.01149


> est <- predict(model, testdata, allow.new.levels = T)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$ActualYieldperHA) / testdata$
ActualYieldperHA
> mean(per.error)
[1] 0.4222743
> median(per.error)
[1] 0.3195706
> range(per.error)
[1] 0.02260246 1.64074617
> hist(per.error, breaks = seq(0, 3.55, by = 0.05))
```



CV 2 – is extrapolating

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year[1:14,]
> years.from.2011 <- sorted.by.year[30:134,]
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(25)
> rand <- sample(1:500, 119)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:119,]
> model <- lmer(formula = ActualYieldperHA ~ f_pre14_tmin + f_minus6
post7_tmax + f_minus6post7_tmin + f_minus6post7_precip + f_minus6pos
t7_solar + (1|Variety), data = traindata)
> summary(model)
Linear mixed model fit by REML ['lmerMod']
Formula: ActualYieldperHA ~ f_pre14_tmin + f_minus6post7_tmax + f_mi
nus6post7_tmin +
    f_minus6post7_precip + f_minus6post7_solar + (1 | Variety)
   Data: traindata


REML criterion at convergence: 520.3
```

```
Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.30591 -0.55537  0.03429  0.53082  2.79072

Random effects:
 Groups   Name        Variance Std.Dev.
 Variety  (Intercept) 51.949   7.208
 Residual              7.409   2.722
Number of obs: 96, groups:  Variety, 18

Fixed effects:
                    Estimate Std. Error t value
(Intercept)           5.0811     4.8470   1.048
f_pre14_tmin          0.6599     0.4326   1.525
f_minus6post7_tmax    1.5406     0.3023   5.096
f_minus6post7_tmin   -1.5133     0.3848  -3.932
f_minus6post7_precip -1.5053     0.6338  -2.375
f_minus6post7_solar  -0.8429     0.3006  -2.805

Correlation of Fixed Effects:
               (Intr) f_p14_ f_mns6pst7_tmx f_mns6pst7_tmn f_mns6pst
7_p
f_pre14_tmn    -0.083

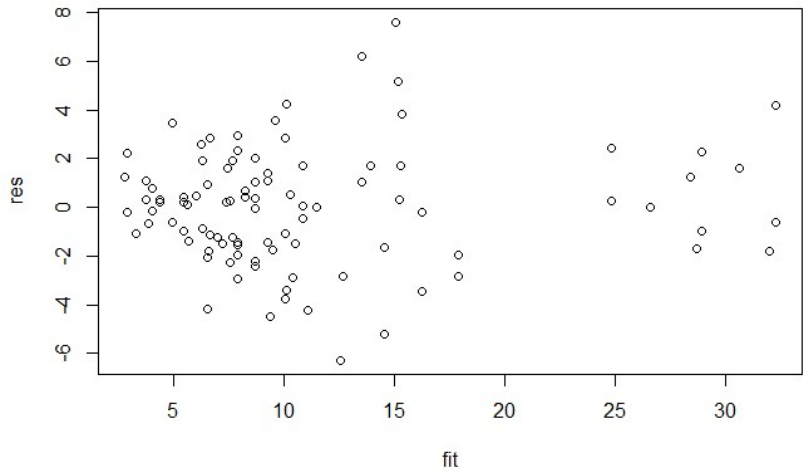
f_mns6pst7_tmx -0.319  0.688

f_mns6pst7_tmn  0.193 -0.865 -0.872

f_mns6pst7_p   -0.323 -0.503 -0.550          0.475

f_mns6pst7_s   -0.271 -0.764 -0.764          0.742          0.787
```

```
> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```

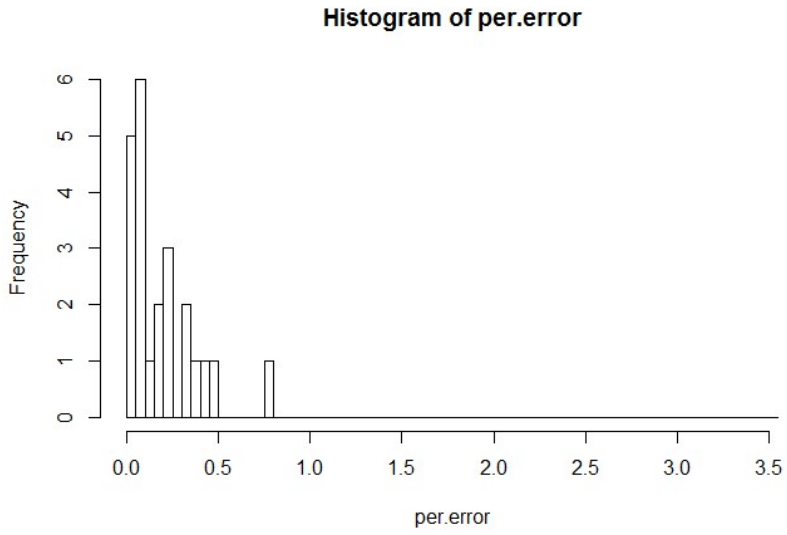



```
> qqnorm(res)
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.99036, p-value = 0.7189


> est <- predict(model, testdata, allow.new.levels = T)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$ActualYieldperHA) / testdata$
ActualYieldperHA
> mean(per.error)
[1] 0.1955149
> median(per.error)
[1] 0.1105425
> range(per.error)
[1] 0.0009776546 0.7553685188
> hist(per.error, breaks = seq(0, 3.55, by = 0.05))
```



CV 3 – is extrapolating

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year[1:14,]
> years.from.2011 <- sorted.by.year[30:134,]
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(46)
> rand <- sample(1:500, 119)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:119,]
> model <- lmer(formula = ActualYieldperHA ~ f_pre14_tmin + f_minus6
post7_tmax + f_minus6post7_tmin + f_minus6post7_precip + f_minus6pos
t7_solar + (1|Variety), data = traindata)
> summary(model)
Linear mixed model fit by REML ['lmerMod']
Formula: ActualYieldperHA ~ f_pre14_tmin + f_minus6post7_tmax + f_mi
nus6post7_tmin +
    f_minus6post7_precip + f_minus6post7_solar + (1 | Variety)
```

```
   Data: traindata

REML criterion at convergence: 529.4

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.10685 -0.52754 -0.00346  0.45844  3.12771

Random effects:
 Groups   Name        Variance Std.Dev.
 Variety  (Intercept) 33.196   5.762
 Residual              9.019   3.003
Number of obs: 96, groups:  Variety, 19

Fixed effects:
                    Estimate Std. Error t value
(Intercept)           8.4635     4.9329   1.716
f_pre14_tmin          0.8759     0.4456   1.966
f_minus6post7_tmax    1.8627     0.3024   6.159
f_minus6post7_tmin   -1.7436     0.3851  -4.528
f_minus6post7_precip -2.4071     0.7090  -3.395
f_minus6post7_solar  -1.3405     0.3010  -4.454

Correlation of Fixed Effects:
               (Intr) f_p14_ f_mns6pst7_tmx f_mns6pst7_tmn f_mns6pst
7_p
f_pre14_tmn    -0.147

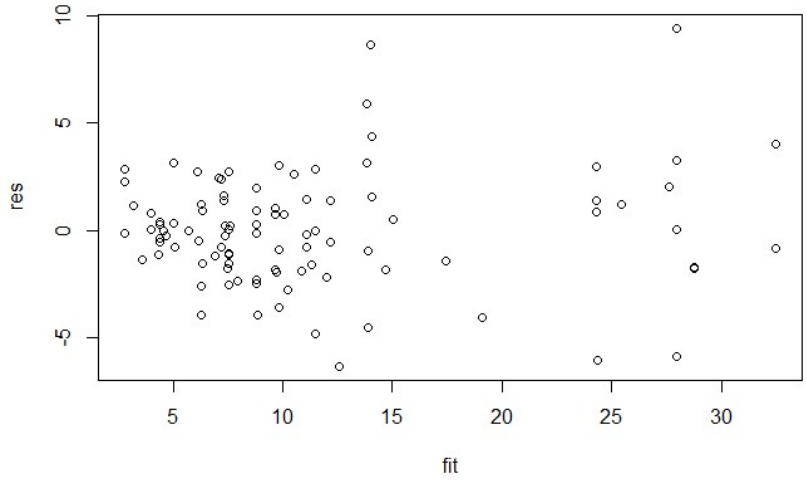
f_mns6pst7_tmx -0.319  0.578

f_mns6pst7_tmn  0.235 -0.834 -0.804

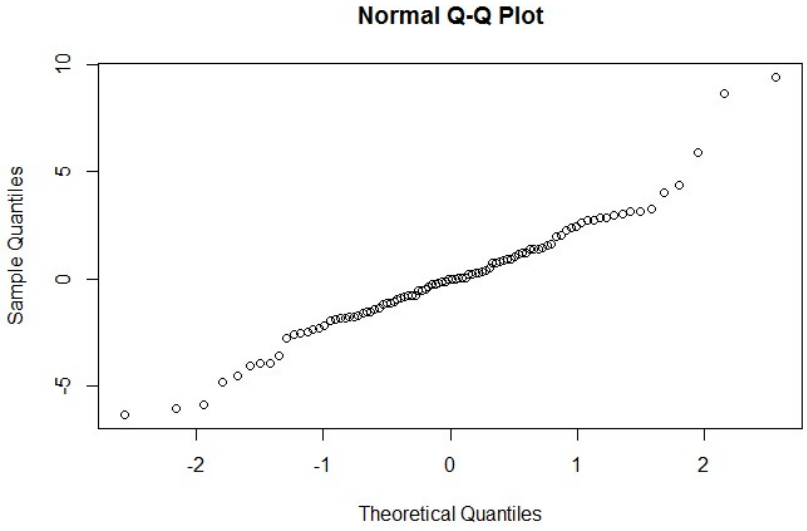
f_mns6pst7_p   -0.389 -0.355 -0.474          0.326

f_mns6pst7_s   -0.305 -0.639 -0.740          0.635          0.756

> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



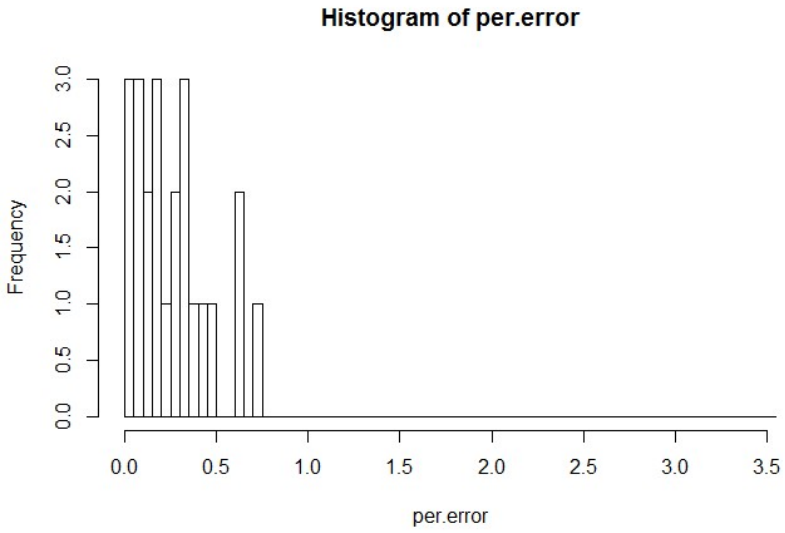

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.96394, p-value = 0.009649

> est <- predict(model, testdata, allow.new.levels = T)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
```

```
> per.error <- abs(fitted$x - testdata$ActualYieldperHA) / testdata$
ActualYieldperHA
> mean(per.error)
[1] 0.263471
> median(per.error)
[1] 0.2141933
> range(per.error)
[1] 0.01564173 0.74049008
> hist(per.error, breaks = seq(0, 3.55, by = 0.05))
```



CV 4 – is extrapolating

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year[1:14,]
> years.from.2011 <- sorted.by.year[30:134,]
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(80)
> rand <- sample(1:500, 119)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:119,]
```

```
> model <- lmer(formula = ActualYieldperHA ~ f_pre14_tmin + f_minus6
post7_tmax + f_minus6post7_tmin + f_minus6post7_precip + f_minus6pos
t7_solar + (1|Variety), data = traindata)
> summary(model)
Linear mixed model fit by REML ['lmerMod']
Formula: ActualYieldperHA ~ f_pre14_tmin + f_minus6post7_tmax + f_mi
nus6post7_tmin +
    f_minus6post7_precip + f_minus6post7_solar + (1 | Variety)
   Data: traindata

REML criterion at convergence: 533

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.54113 -0.49563 -0.04306  0.57318  2.87206

Random effects:
 Groups   Name        Variance Std.Dev.
 Variety  (Intercept) 46.178   6.795
 Residual              8.871   2.978
Number of obs: 96, groups:  Variety, 18

Fixed effects:
                     Estimate Std. Error t value
(Intercept)            3.8817     4.7386   0.819
f_pre14_tmin           0.9988     0.4725   2.114
f_minus6post7_tmax     1.6906     0.3112   5.433
f_minus6post7_tmin    -1.7789     0.4138  -4.299
f_minus6post7_precip  -1.6020     0.7145  -2.242
f_minus6post7_solar   -1.0129     0.3206  -3.159

Correlation of Fixed Effects:
               (Intr) f_p14_ f_mns6pst7_tmx f_mns6pst7_tmn f_mns6pst
7_p
f_pre14_tmn    -0.093

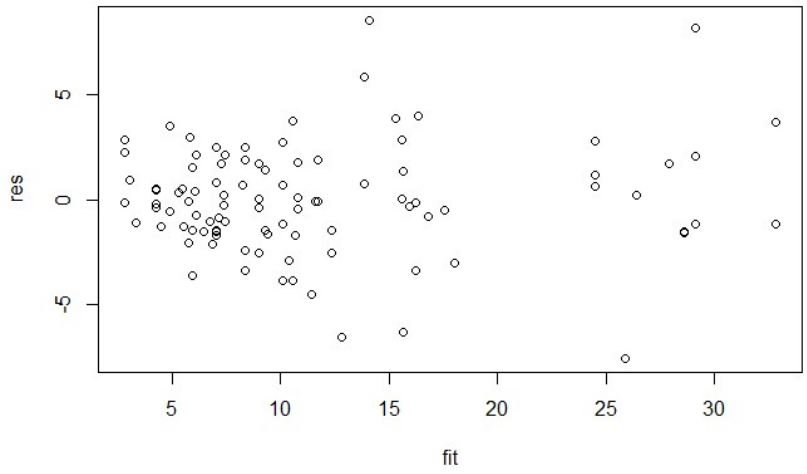
f_mns6pst7_tmx -0.233  0.697

f_mns6pst7_tmn  0.133 -0.867 -0.875

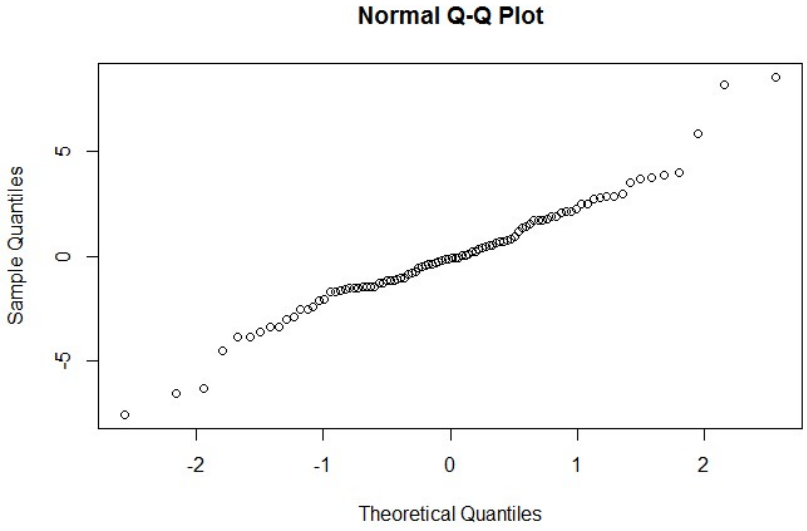
f_mns6pst7_p   -0.402 -0.474 -0.539          0.472

f_mns6pst7_s   -0.288 -0.758 -0.805          0.768          0.773

> res <- residuals(model)
> fit <- fitted(model)
> plot(fit,res)
```




```
> qqnorm(res)
```



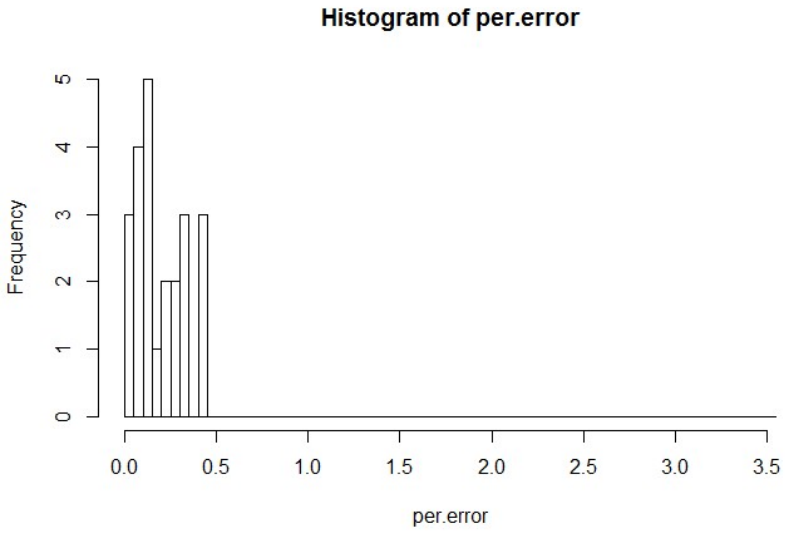

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.9686, p-value = 0.02101

> est <- predict(model, testdata, allow.new.levels = T)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
```

```
> per.error <- abs(fitted$x - testdata$ActualYieldperHA) / testdata$
ActualYieldperHA
> mean(per.error)
[1] 0.1903668
> median(per.error)
[1] 0.1320684
> range(per.error)
[1] 0.006823966 0.443931721
> hist(per.error, breaks = seq(0, 3.55, by = 0.05))
```



CV 5 – is extrapolating

```
> InputYieldData <- read.csv("~/Grapevine Project/Combined grower da
ta for analysis/standardised yields/InputYieldData.csv")
> View(InputYieldData)
> ### this will sort data by year
> sorted.by.year <- InputYieldData[order(InputYieldData$Yield.Year),
]
> ##set up TRAINING DATA SET this will set-up the training data set
> years.to.2011 <- sorted.by.year[1:14,]
> years.from.2011 <- sorted.by.year[30:134,]
> train.data <- rbind(years.to.2011,years.from.2011)
> View(train.data)
> ##Randomly divide training data set into training and test sets.Se
e above scripts for reading in original data set.
> set.seed(92)
> rand <- sample(1:500, 119)
> train <- cbind(train.data, rand)
> train.sort <- train[order(train$rand),]
```

```
> traindata <- train.sort[1:96,]
> testdata <- train.sort[97:119,]
> model <- lmer(formula = ActualYieldperHA ~ f_pre14_tmin + f_minus6
post7_tmax + f_minus6post7_tmin + f_minus6post7_precip + f_minus6pos
t7_solar + (1|Variety), data = traindata)
> summary(model)
Linear mixed model fit by REML ['lmerMod']
Formula: ActualYieldperHA ~ f_pre14_tmin + f_minus6post7_tmax + f_mi
nus6post7_tmin +
    f_minus6post7_precip + f_minus6post7_solar + (1 | Variety)
   Data: traindata

REML criterion at convergence: 524.7

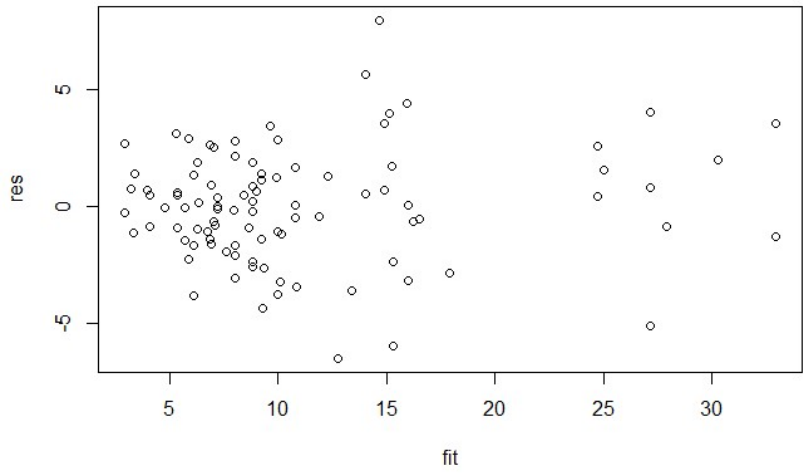
Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.33551 -0.49419 -0.00154  0.51114  2.87233

Random effects:
 Groups   Name        Variance Std.Dev.
 Variety  (Intercept) 48.320   6.951
 Residual              7.723   2.779
Number of obs: 96, groups:  Variety, 19

Fixed effects:
                    Estimate Std. Error t value
(Intercept)           6.1274     4.7274   1.296
f_pre14_tmin          0.6765     0.4198   1.612
f_minus6post7_tmax    1.3356     0.3000   4.451
f_minus6post7_tmin   -1.4105     0.3790  -3.722
f_minus6post7_precip -1.2961     0.6353  -2.040
f_minus6post7_solar  -0.7520     0.2881  -2.610

Correlation of Fixed Effects:
               (Intr) f_p14_ f_mns6pst7_tmx f_mns6pst7_tmn f_mns6pst
7_p
f_pre14_tmn    -0.168
f_mns6pst7_tmx -0.322  0.667
f_mns6pst7_tmn  0.193 -0.837 -0.858
f_mns6pst7_p   -0.297 -0.436 -0.512          0.409
f_mns6pst7_s   -0.230 -0.722 -0.781          0.734          0.745
> res <- residuals(model)
```

```
> fit <- fitted(model)
> plot(fit,res)
```

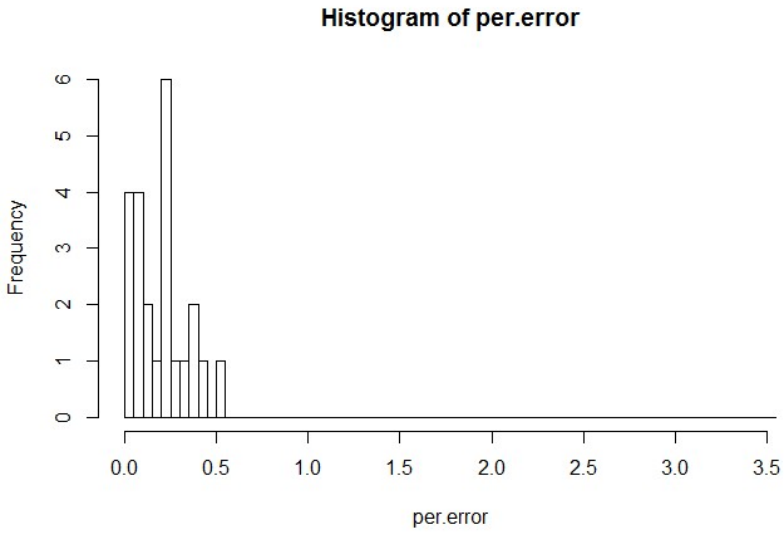



```
> qqnorm(res)
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.99116, p-value = 0.7797


> est <- predict(model, testdata, allow.new.levels = T)
> write.table(est, file = "fitted.txt")
> fitted <- read.csv("~/Grapevine Project/Combined grower data for a
nalysis/standardised yields/fitted.txt", sep="")
> View(fitted)
> per.error <- abs(fitted$x - testdata$ActualYieldperHA) / testdata$
ActualYieldperHA
> mean(per.error)
[1] 0.1951497
> median(per.error)
[1] 0.2087013
> range(per.error)
[1] 0.01489394 0.51662680
> hist(per.error, breaks = seq(0, 3.55, by = 0.05))
```


Histogram of per.error
Frequency
0
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
per.error

5.5 Automated variable selection

To do automated Variable selection in R with RFE LVQ and correlations

example data and scripts

Recursive Feature Elimination (RFE).

```
InputYieldData <- read.csv("~/Grapevine Project/Combined grower data
 for analysis/standardised yields/InputYieldData.csv")
YeildDIVBYmean.prev.year. <- InputYieldData$YeildDIVBYmean.prev.year
.
testweatherdata <- InputYieldData[,10:33]
Yield.per.Ha.divided.by.mean.yield.ha.per.block <- InputYieldData$Yi
eld.per.Ha.divided.by.mean.yield.ha.per.block
testANN <- cbind(Yield.per.Ha.divided.by.mean.yield.ha.per.block , Y
eildDIVBYmean.prev.year., testweatherdata)
View(testANN)
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
results <- rfe(testANN[,2:26], testANN [,1], sizes=c(1:8), rfeContro
l=control)
print(results)
```

```
Recursive feature selection

Outer resampling method: Cross-Validated (10 fold)

Resampling performance over subset size:

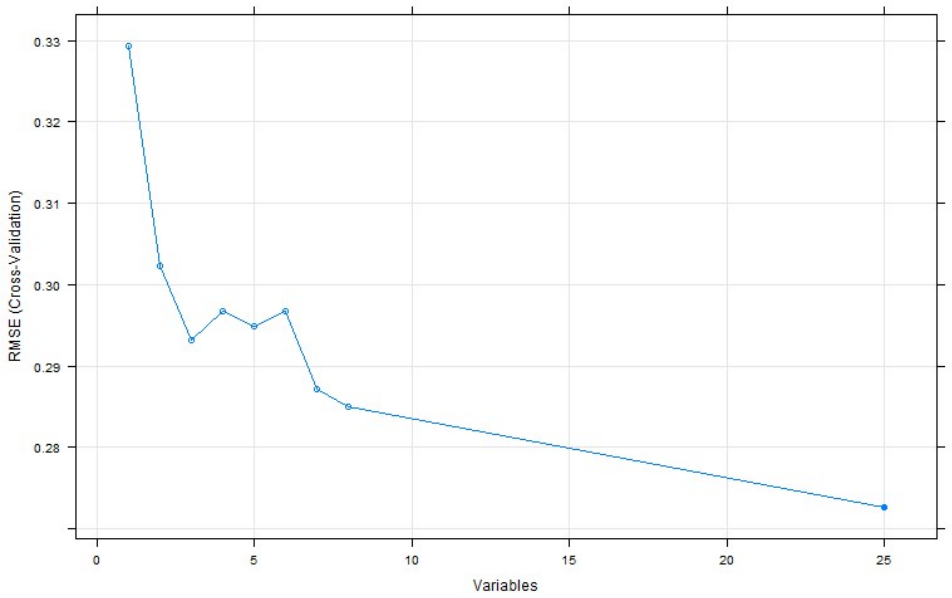
 Variables   RMSE Rsquared  RMSESD RsquaredSD Selected
         1 0.3293   0.4053 0.07162     0.1336
         2 0.3022   0.4716 0.06706     0.1422
         3 0.2931   0.5129 0.06998     0.1598
         4 0.2967   0.5075 0.07258     0.1851
         5 0.2950   0.5144 0.07010     0.1714
         6 0.2967   0.5051 0.07677     0.1922
         7 0.2872   0.5332 0.07381     0.1777
         8 0.2850   0.5416 0.07513     0.1795
        25 0.2727   0.5752 0.07280     0.1665        *

The top 5 variables (out of 25):
   bb_post14_solar, f_post14_solar, bb_minus6post7_solar, f_minus6po
st7_solar, f_post14_precip
```

```
predictors(results)
```

```
 [1] "bb_post14_solar"           "f_post14_solar"           "bb_min
us6post7_solar"
 [4] "f_minus6post7_solar"       "f_post14_precip"          "f_post
14_tmax"
 [7] "bb_pre14_tmax"             "f_pre14_solar"            "YeildD
IVBYmean.prev.year."
[10] "bb_post14_precip"          "f_pre14_tmax"             "f_minu
s6post7_precip"
[13] "bb_pre14_solar"            "f_minus6post7_tmin"       "f_pre1
4_precip"
[16] "bb_pre14_tmin"             "f_post14_tmin"            "bb_pre
14_precip"
[19] "bb_minus6post7_precip"     "f_minus6post7_tmax"       "bb_pos
t14_tmax"
[22] "bb_minus6post7_tmax"       "f_pre14_tmin"             "bb_min
us6post7_tmin"
[25] "bb_post14_tmin"
```

```
plot(results, type=c("g", "o"))
```



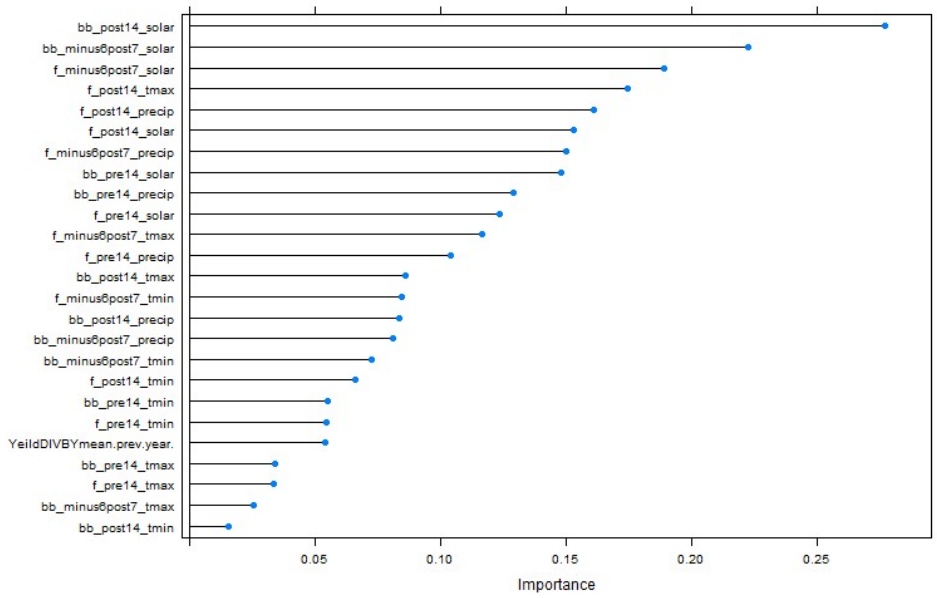
Learning Vector Quantization **(LVQ) model**

```
InputYieldData <- read.csv("~/Grapevine Project/Combined grower data
 for analysis/standardised yields/InputYieldData.csv")
YeildDIVBYmean.prev.year. <- InputYieldData$YeildDIVBYmean.prev.year
.
testweatherdata <- InputYieldData[,10:33]
Yield.per.Ha.divided.by.mean.yield.ha.per.block <- InputYieldData$Yi
eld.per.Ha.divided.by.mean.yield.ha.per.block
```

```
testANN <- cbind(Yield.per.Ha.divided.by.mean.yield.ha.per.block , Y
eildDIVBYmean.prev.year., testweatherdata)
View(testANN)
##got to make sure I can repeat this analysis
set.seed(7)
## prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
## train the model
model <- train(Yield.per.Ha.divided.by.mean.yield.ha.per.block ~., d
ata=testANN, method="neuralnet", preProcess="scale", trControl=contr
ol)
There were 50 or more warnings (use warnings() to see the first 50)


## estimate variable importance
importance <- varImp(model, scale=FALSE)


## plot importance
plot(importance)
```



ID autocorrelated variables using CARET;

```
InputYieldData <- read.csv("~/Grapevine Project/Combined grower data
 for analysis/standardised yields/InputYieldData.csv")
YeildDIVBYmean.prev.year. <- InputYieldData$YeildDIVBYmean.prev.year
.
testweatherdata <- InputYieldData[,10:33]
Yield.per.Ha.divided.by.mean.yield.ha.per.block <- InputYieldData$Yi
eld.per.Ha.divided.by.mean.yield.ha.per.block
```

```
testANN <- cbind(Yield.per.Ha.divided.by.mean.yield.ha.per.block , Y
eildDIVBYmean.prev.year., testweatherdata)
View(testANN)
## to make sure these results are repeatable
set.seed(7)
## calculate correlation matrix
correlationMatrix <- cor(testANN[,2:26])
## summarize the correlation matrix
print(correlationMatrix)

## find attributes that are highly corrected (ideally >0.75)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.5)
## print indexes of highly correlated attributes
print(highlyCorrelated)
 [1] 10  6  2 21 25 15 23  8 12 14 11 22  7  9
```