Winter 2023
SurvMeth 686
Project #2

Cheng, Chia Wen

Abstract

This paper builds and trains two neural network models to predict the varieties of wheat with provided datasets. The ultimate goal is to reach the highest accuracy of prediction with the seven continuous independent variables. Since the data is complete without missing values, the pre-processing work focuses on assigning each class of "variety" with a unique indicator, and standardizing the independent variables. High covariances among the input variables and nonlinearities between the input and the output variables are neglected because they do not strongly skew our goal of accessing the highest prediction accuracy with neural network models. With the assistance of "NeuralNetTools," "nnet," and "caret" packages in R programming, I build two neural network models with adequately high overall accuracy scores of over 0.95. The agreement scores between classifiers are also high at 0.975 and 0.95, respectively. For the reason of comparison, an ordinal logistic model is also built, and the prediction accuracy is close to but poorer than the two neural network models.

Exploratory Analysis

The dependent variable we are interested in is a categorical variable composed of levels 1, 2, and 3, representing three different varieties of wheat: Kama, Rosa, and Canadian.[1] There are 150 observations in the provided training dataset and 60 in the test dataset. All observations are complete without missing values. In addition, all 7 independent variables are continuous. According to the histograms, "area," "asymmetry," "groove," "length," and "perimeter" are positively skewed with more observations on the left side and means greater than medians; "compactness" is negatively skewed; and "width" is more of normally distributed, but still with a mean greater than its median.

There are several high covariances, with values over 0.8, between the independent variables. However, in building neural network, I am not too worried about the high correlations between X's because reducing model complexity is not a primary issue, which is contrary to regression models.

The principle component analysis (PCA) shows multiple features. 65.3% of the variance is explained by the first component (PC1), and it depends the most and almost evenly on "area," "width," and "perimeter." 20.6% of the variance is explained by PC2, which "variety"

[1] seeds Data Set. UCI Center for Machine Learning and Intelligent Systems. Retrieved on April 7 , 2023, from https://archive.ics.uci.edu/ml/datasets/seeds#.

and “asymmetry” contribute the most to it. 9.7% of the variance is explained by PC3, which is substantially affected by “compactness” and “asymmetry.”

To carefully examine bivariate relationships between each independent variable and “variety,” I develop Q-Q plots, boxplots, strip plots, mean/SEM plots, bar plots, and get correlation scores for each pair. The only pair that is close to linear relationship is “variety” as a function of “asymmetry.” It has the greatest adjusted R-squared value; however, 0.3548 still indicates a weak to moderate linear association between the predictor and the outcome. The second greatest adjusted R-squared happens in “variety” as a function of “compactness,” which is 0.2593, and is considered weak in this linear model being accounting for variance in the outcome.

Feature Engineering

For the purpose of building neural network to solve a classification problem, which is to predict the variety of wheat, I take two steps of transformations.

1. Encoding the factor variable, “variety,” such that each class has an indicator variable that I can feed to the neural network for training. I then have three additional binary variables for both training and test datasets, which are named “variety1,” “variety2,” and “variety3,” with values of 0 as ineffective and 1 as effective.
2. Standardizing the independent, continuous variables so that the means are equal to zero and the standard deviations are equal to one. Since neural networks use activation functions between -1 and +1, it is important to scale variables down to a common scale so that the model can accurately compare predicted and actual values.[2] Failure to standardize the data will typically result in the prediction value remaining the same across all observations, regardless of the input values.[3]

After the two transformations, I combine the unique indicators of “variety” and the standardized input variable to new training and test datasets. I will then use these datasets to select hyperparameters and build neural network models for prediction.

Since I am not applying the logistic regression methodology, I am not concerned about being over-parameterized of extracting each class in “variety” to an individual indicator. The high covariances between several independent variables are not of concerned either because existence of uninformative, or less important, variables that will only increase model complexity does not severely impact our goal to predict with a neural network. Furthermore, I choose not to transform the input variables in addition to standardizing them, although non-linear bivariate relationships are frequently seen, because they affect little in prediction accuracy of the neural network.

---

[2] Neural Networks in R Tutorial. Learn by Marketing. Retrieved on April 9, 2023, from https://www.learnbymarketing.com/tutorials/neural-networks-in-r-tutorial/.

[3] neuralnet: Train and Test Neural Networks using R. datascience+. Retrieved on April 10, 2023, from https://datascienceplus.com/neuralnet-train-and-test-neural-networks-using-r/.

Hyperparameter Selection

I utilize two methods with two different programming packages to select the optimize hyperparameters for the most accurate predictions of our neural network models.

1. With the package "NeuralNetTools," I first build a model with formula $variety1 + variety2 + variety3 \sim area + perimeter + compactness + length + width + asymmetry + groove$. I specify the number of nodes on a single hidden layer at 10 and the number of repetitions at 10, trying to see which converges to a best solution. However, the smallest error I receive, which is 0.024814, is not sufficiently small. Since there are only seven input variables, I guess that the complexity of my model is not quite high. Thus, I reduce the number of nodes on a single hidden layer to 5, and get the smallest error at 0.013631 with 426 steps and 1 repetition.
2. With the packages "caret" and "nnet," I have the algorithm experiment 5 sizes, which are 1, 5, 9, 13, and 17, and 11 weight decays from 0 to 1 with 0.1 as intervals. I also use the leave-a-group-out k-fold cross-validation and set parameter k to 10, so that the function outputs the results every 10 iterations, to pick the best size and weight decay. 55 options, each with a maximum iteration number of 500, are tested. Some iterations reach 90 times, but many stop before or at 60 times. In addition, the changes after 60 iterations are slight. So I modify the "maxit" value to 60, informing the algorithm to stop iterating when reaching 60 iterations. I gain the highest accuracy above 0.95 at hidden units of 13 and weight decay of approximately 0.1.
    a. "Length" and "groove" are on top of the variable importance in our second model. I can conclude that features "length" and "groove" are decisive of wheat variety, but I cannot summarize their effects and magnitudes with current information. If I am to build a regression model, these two variables should stay in the model, while other variables need further examination of their necessities.

I also build an ordinal logistic regression model that predicts possibilities for the ordinal, categorical dependent variable, "variety." Nonetheless, the assumptions I hold and the adjustments I make do not fully qualify those of a logistic regression model. Hence, this model is developed merely for entry-level comparison of the prediction accuracy.

Evaluate Predictions

Prediction accuracy is assessed with the function confusionMatrix in R programming for both neural network models.

1. The first neural network built by "NeuralNetTools" has a satisfied overall accuracy at 0.9833. There is only one mislabeled outcome. The lowest sensitivity (true positive rate) takes place in the third class of "variety," while the lowest specificity

(true negative rate) happens in the first class. Class two has the highest balanced accuracy, because no error is detected. The Kappa value is 0.975, revealing a high agreement between the classifiers. But I have to be aware of that, since our target class distribution is balanced with 20 in each variety in the test dataset, it is easier to reach higher values of Cohen's kappa.[4]

2. The second neural network built by using "caret" and "nnet" also has a satisfied overall accuracy at 0.9667. There are two mispredicted outcomes, one in the first class and another in the third. Thus, the highest sensitivity (true positive rate), specificity (true negative rate), and balanced accuracy all take place in the second class of "variety," while the first and the third classes share similar values in those performance two has the highest balanced accuracy, because they both have one error detected. The Kappa value is 0.95, revealing a high agreement between the classifiers.

To conclude, our model performed well in prediction accuracy. In addition, the ordinal logistic regression model has minorly more errors in prediction, but the prediction accuracy is still close to both of the neural network models.

[4] Maarit Widmann. (April 2020.) Cohen's Kappa: What it is, when to use it, and how to avoid its pitfalls. THENEWSTAND. Retrieved on April 15, 2023, from https://thenewstack.io/cohens-kappa-what-it-is-when-to-use-it-and-how-to-avoid-its-pitfalls/.

SurvMeth 686 Project 2-Appendix

Cheng, Chia Wen

2023-04-16

```
setwd("G:/My Drive/0. study abroad/academic/9. 2023 Winter/8. SurvMeth 686 Model Machine Learning II/2.
seeds_test <- read.csv('seeds_test.csv') # 60 obs, 8 vars
seeds_train <- read.csv('seeds_train.csv') # 150 obs, 8 vars

## packages
library(ggplot2)
library(MASS)
library(maxLik)
library(lmtest)
library(pROC)
library(caret)
library(dplyr)
library(tidyr)
library(DataExplorer)
library(tidyverse)
library(NeuralNetTools)
library(neuralnet)
library(nnet)

# examine data
names(seeds_test) # area; perimeter; compactness; length; width; asymmetry; groove; variety
```

```
## [1] "area"        "perimeter"   "compactness" "length"      "width"
## [6] "asymmetry"   "groove"      "variety"
```

```
summary(seeds_test)
```

```
##       area         perimeter      compactness         length
##  Min.   :10.80   Min.   :12.57   Min.   :0.8198   Min.   :4.981
##  1st Qu.:12.35   1st Qu.:13.52   1st Qu.:0.8563   1st Qu.:5.301
##  Median :14.40   Median :14.37   Median :0.8719   Median :5.501
##  Mean   :14.91   Mean   :14.60   Mean   :0.8706   Mean   :5.648
##  3rd Qu.:17.72   3rd Qu.:15.89   3rd Qu.:0.8866   3rd Qu.:6.034
##  Max.   :21.18   Max.   :17.23   Max.   :0.9066   Max.   :6.666
##      width         asymmetry         groove         variety
##  Min.   :2.683   Min.   :1.018   Min.   :4.782   Min.   :1
##  1st Qu.:2.933   1st Qu.:2.642   1st Qu.:5.036   1st Qu.:1
##  Median :3.259   Median :3.599   Median :5.246   Median :2
##  Mean   :3.260   Mean   :3.569   Mean   :5.417   Mean   :2
##  3rd Qu.:3.526   3rd Qu.:4.576   3rd Qu.:5.890   3rd Qu.:3
##  Max.   :4.033   Max.   :6.715   Max.   :6.451   Max.   :3
```

```
summary(seeds_train)
```

```
##       area         perimeter      compactness         length
##  Min.   :10.59   Min.   :12.41   Min.   :0.8081   Min.   :4.899
##  1st Qu.:12.24   1st Qu.:13.41   1st Qu.:0.8576   1st Qu.:5.245
##  Median :14.34   Median :14.29   Median :0.8750   Median :5.541
##  Mean   :14.82   Mean   :14.54   Mean   :0.8712   Mean   :5.621
##  3rd Qu.:17.11   3rd Qu.:15.64   3rd Qu.:0.8878   3rd Qu.:5.979
##  Max.   :20.97   Max.   :17.25   Max.   :0.9183   Max.   :6.675
##      width         asymmetry          groove         variety
##  Min.   :2.630   Min.   :0.7651   Min.   :4.519   Min.   :1
##  1st Qu.:2.955   1st Qu.:2.5615   1st Qu.:5.056   1st Qu.:1
##  Median :3.226   Median :3.6080   Median :5.223   Median :2
##  Mean   :3.258   Mean   :3.7525   Mean   :5.405   Mean   :2
##  3rd Qu.:3.563   3rd Qu.:4.8560   3rd Qu.:5.870   3rd Qu.:3
##  Max.   :4.032   Max.   :8.4560   Max.   :6.550   Max.   :3
```

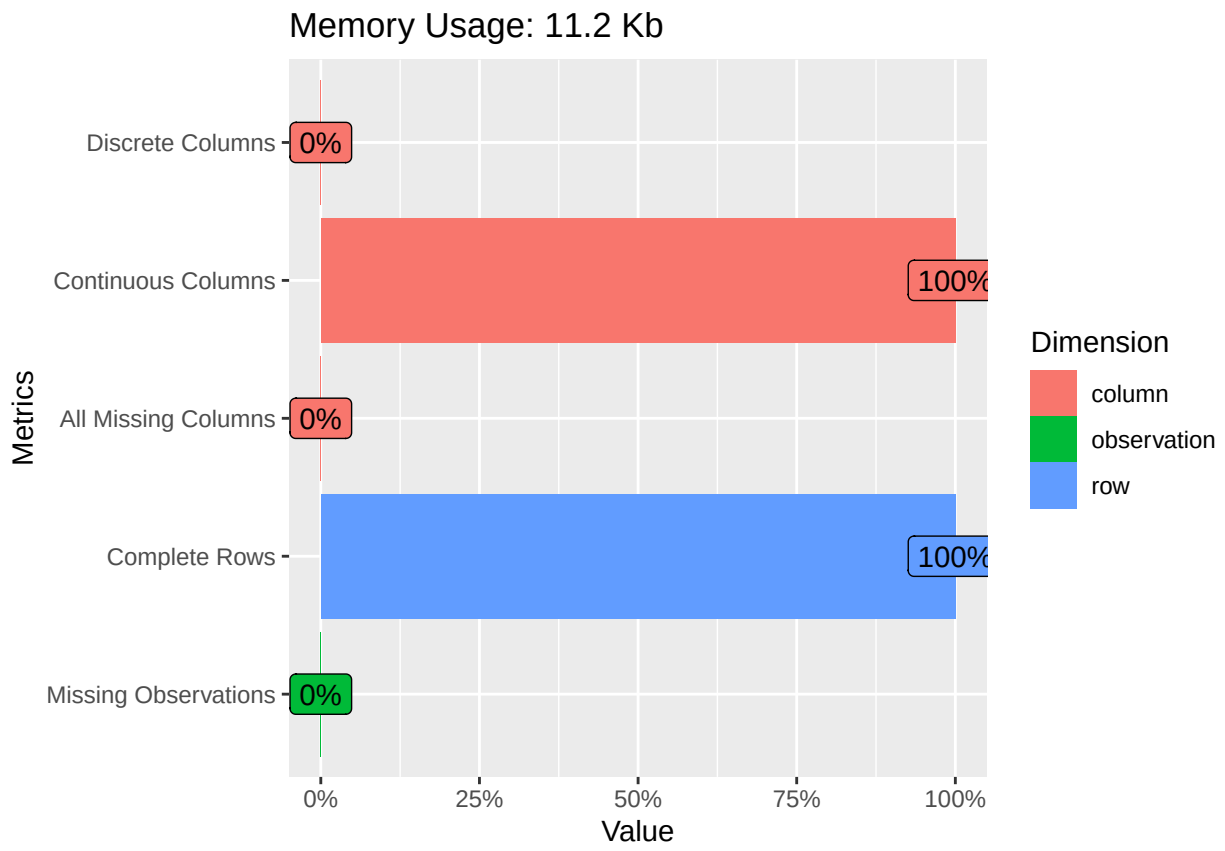
```
str(seeds_train)
```

```
## 'data.frame':    150 obs. of  8 variables:
##  $ area       : num  15.3 13.8 16.1 14.4 14.7 ...
##  $ perimeter  : num  14.8 13.9 15 14.2 14.5 ...
##  $ compactness: num  0.871 0.895 0.903 0.895 0.88 ...
##  $ length     : num  5.76 5.32 5.66 5.39 5.56 ...
##  $ width      : num  3.31 3.38 3.56 3.31 3.26 ...
##  $ asymmetry  : num  2.22 2.26 1.35 2.46 3.59 ...
##  $ groove     : num  5.22 4.8 5.17 4.96 5.22 ...
##  $ variety    : int  1 1 1 1 1 1 1 1 1 1 ...
```

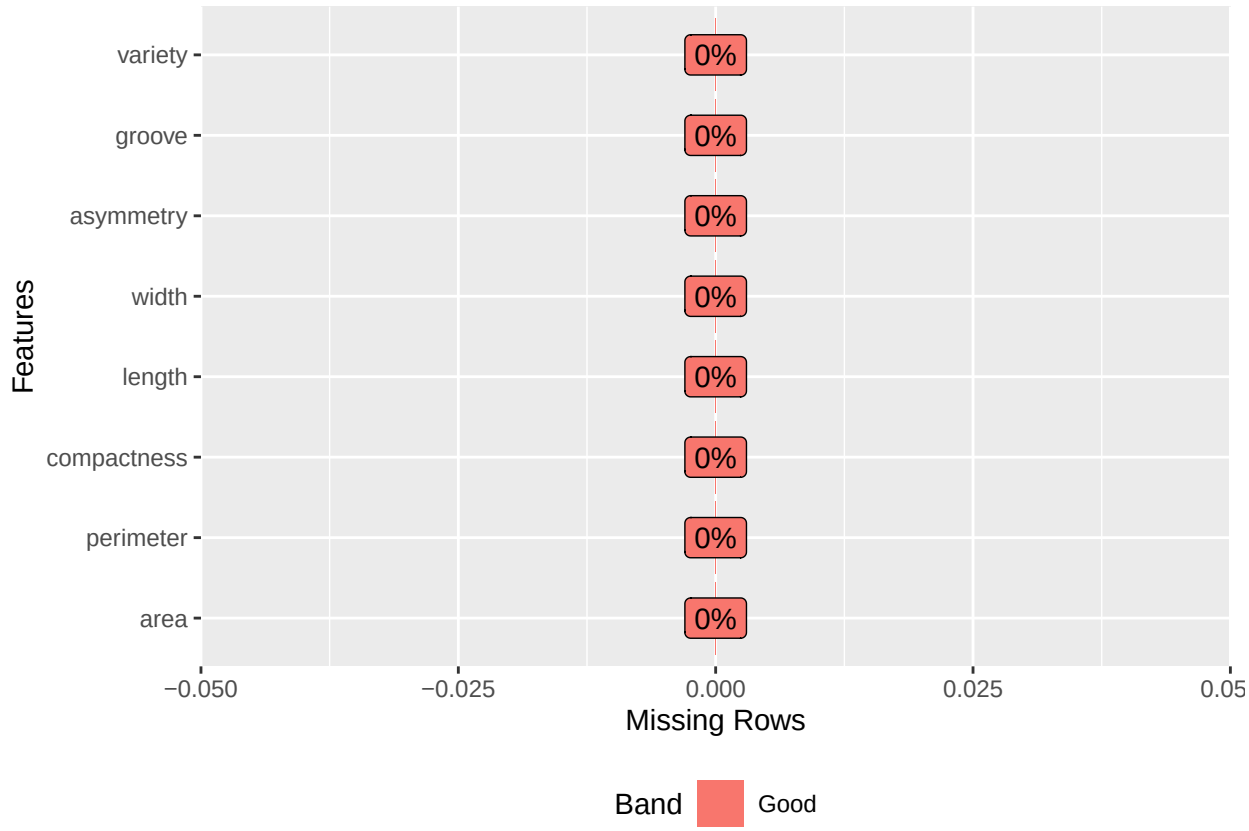
```
# Exploratory Analysis
plot_str(seeds_train)
introduce(seeds_train) # no missing values
```

```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1  150       8               0                   8                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0           150               1200        11488
```

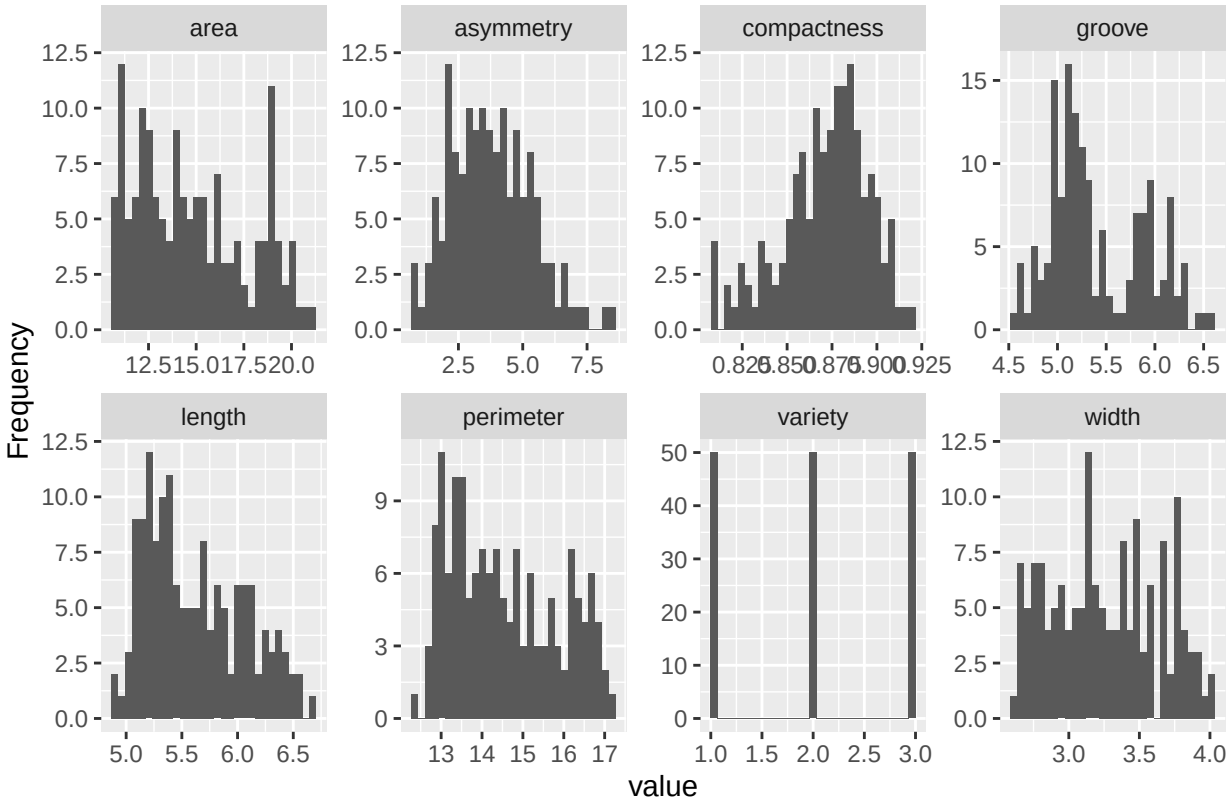
```
plot_intro(seeds_train)
```



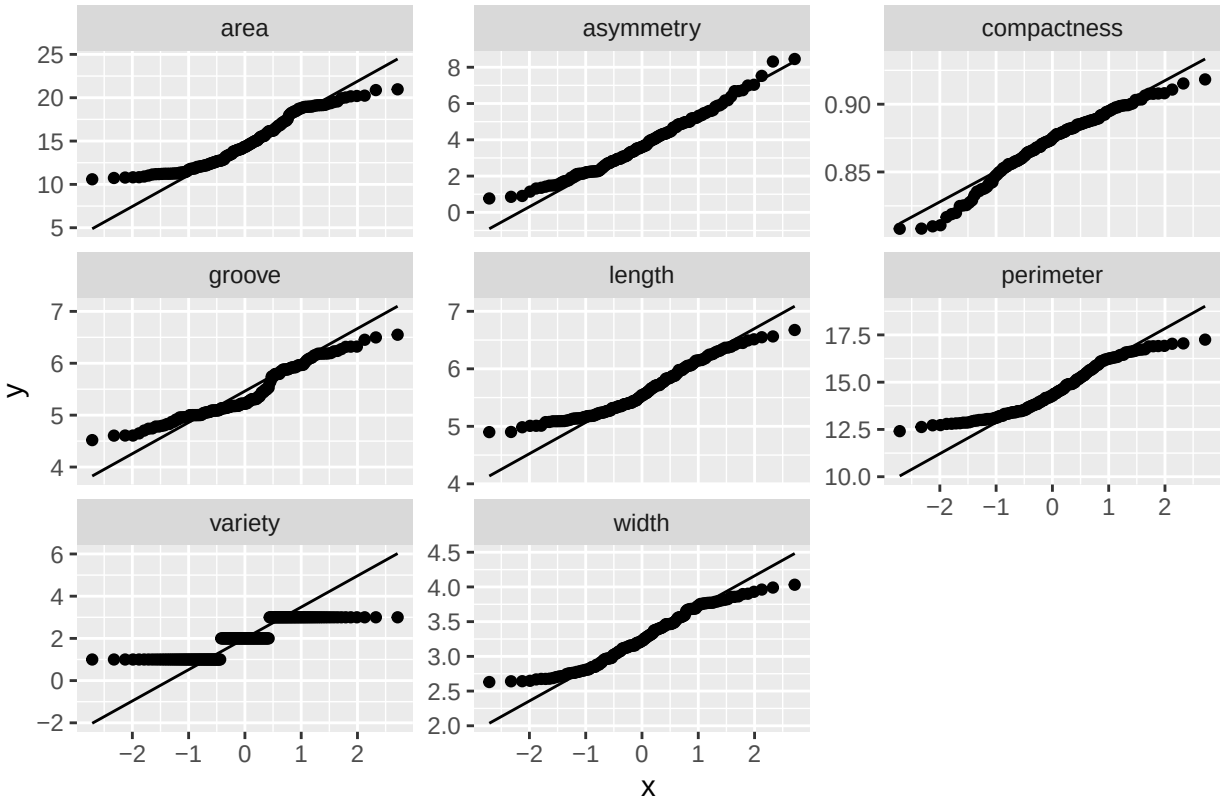

```
plot_missing(seeds_train)
```



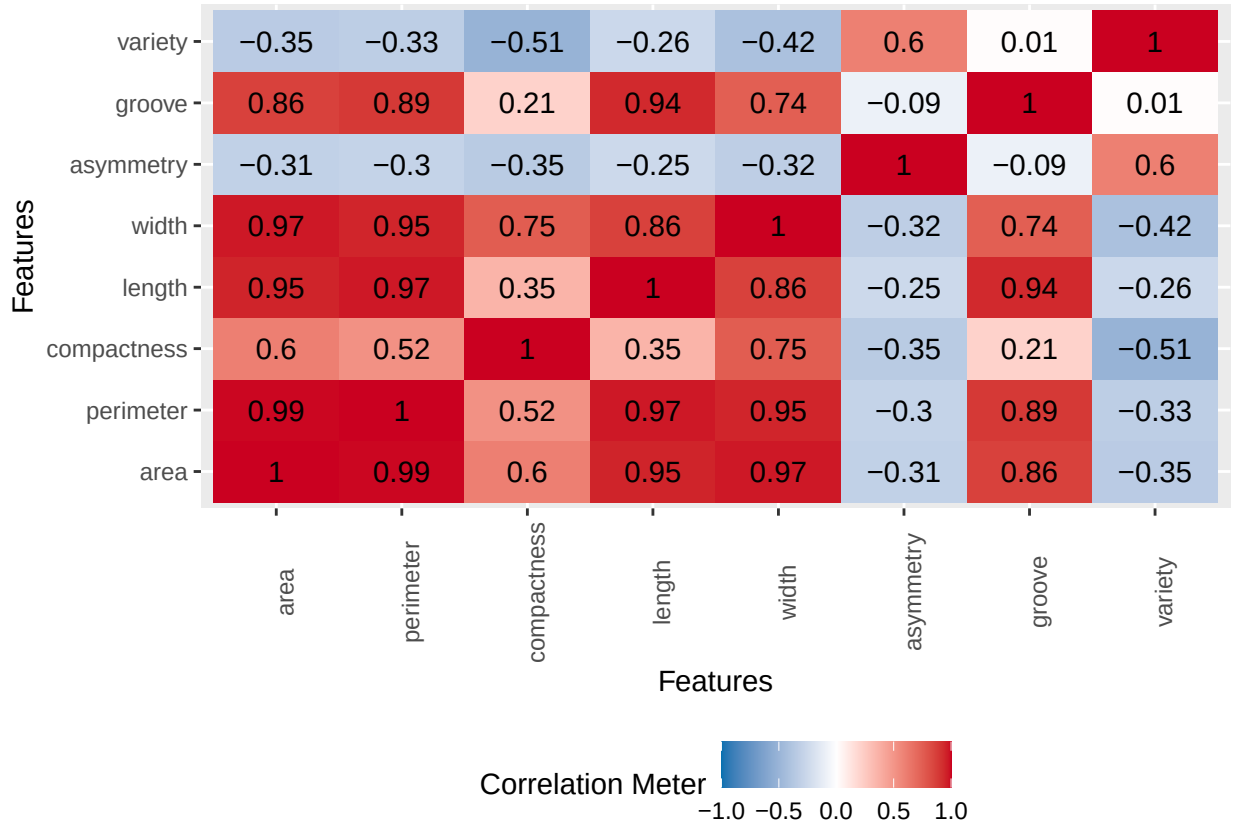

```
plot_histogram(seeds_train)
```



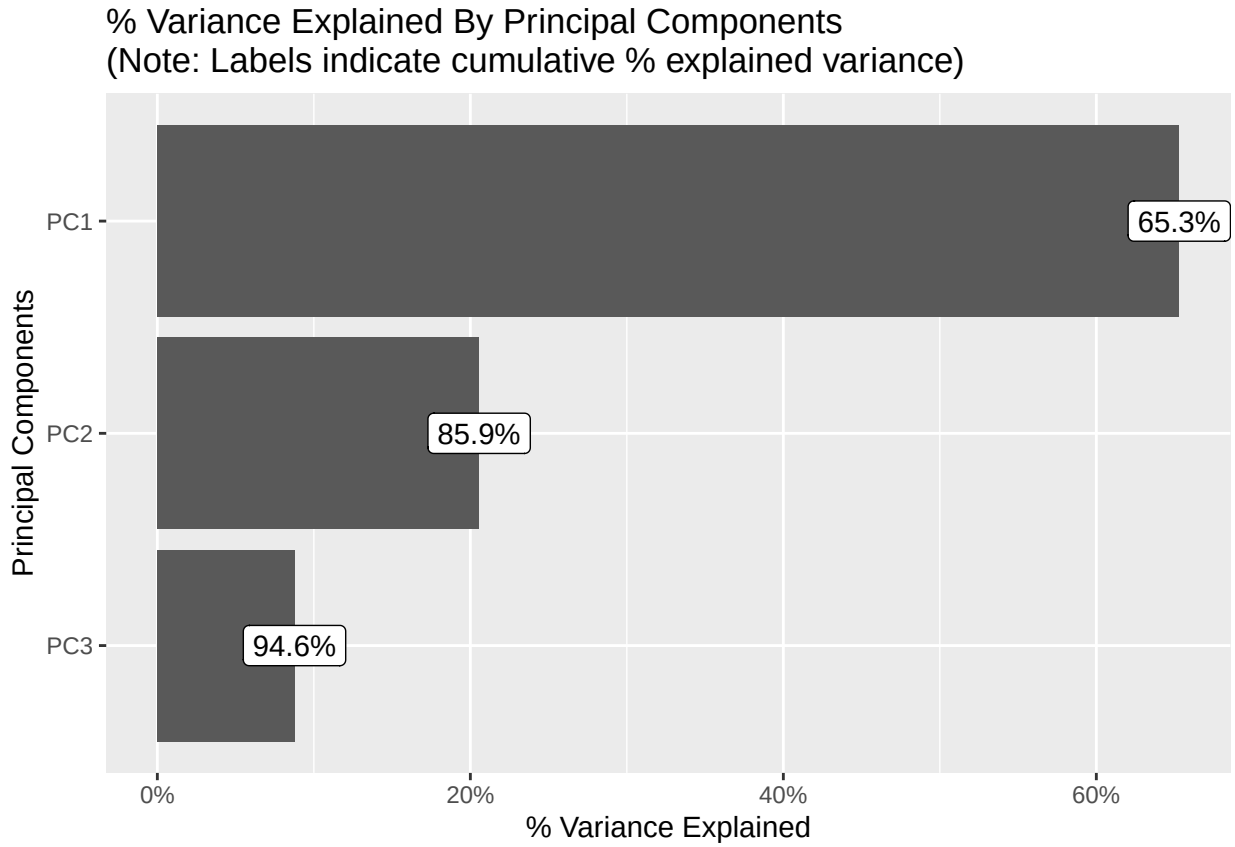

```
plot_qq(seeds_train)
```

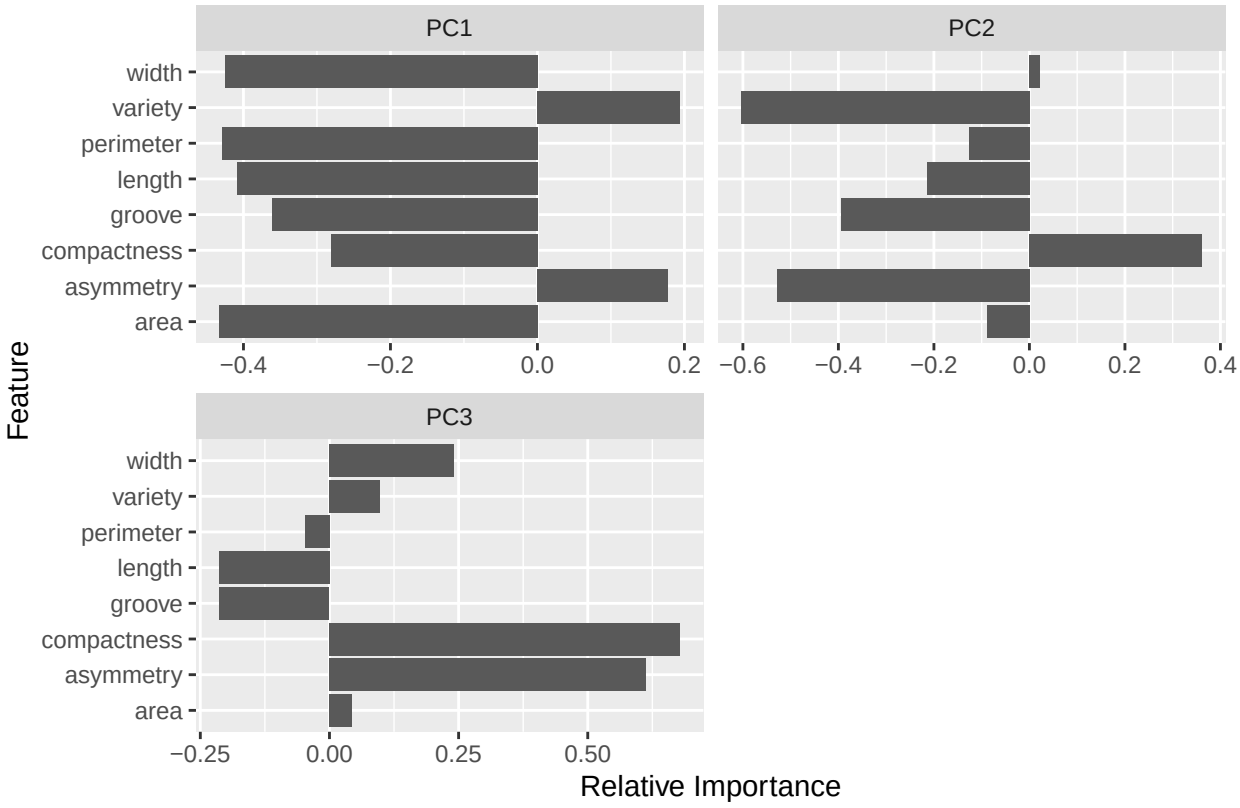



```
plot_correlation(na.omit(seeds_train))
```

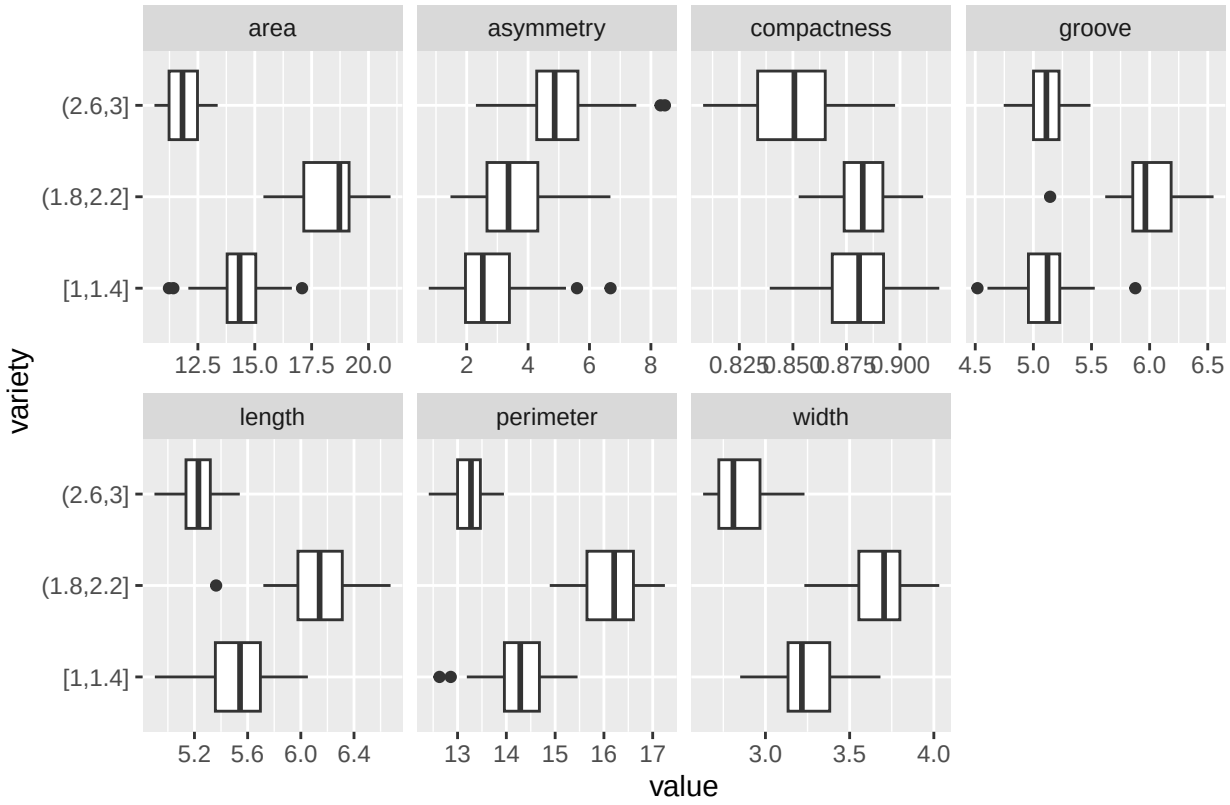



```
## variance cap of 0.95 let to 76 p components, nrow, ncol limits the presentation of components to the
plot_prcomp(seeds_train,variance_cap=0.95,nrow=2,ncol=2)
```

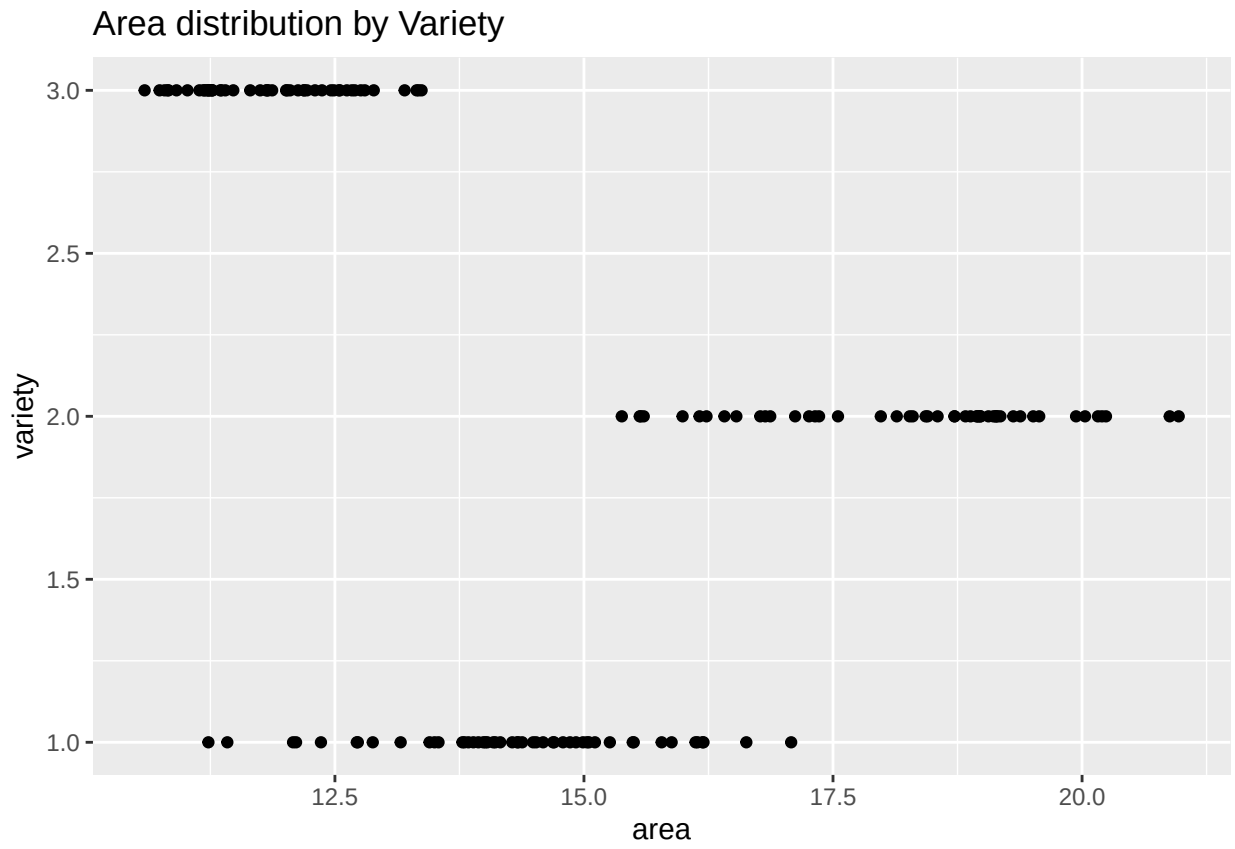

% Variance Explained By Principal Components
(Note: Labels indicate cumulative % explained variance)
Principal Components
PC1
PC2
PC3
65.3%
85.9%
94.6%
0%
20%
40%
60%
% Variance Explained



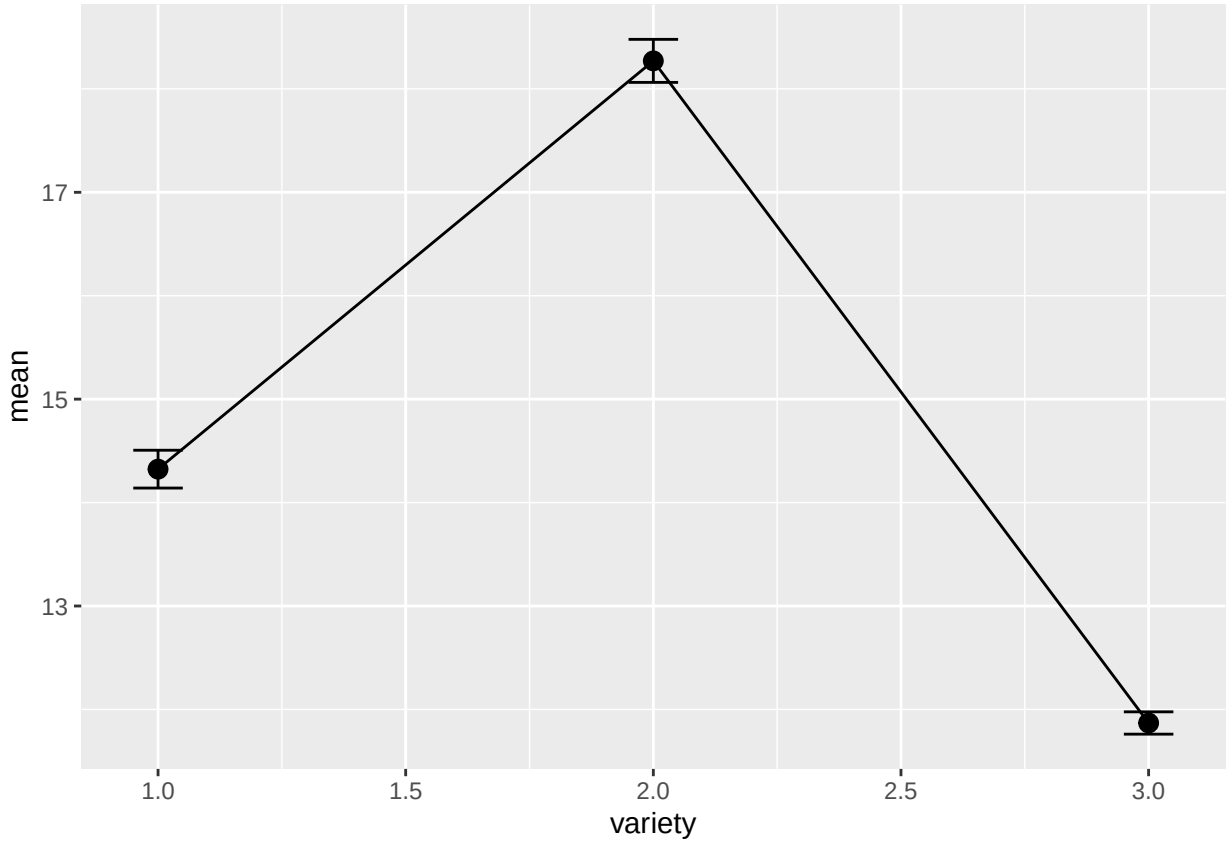

```
plot_boxplot(seeds_train,by="variety")
```



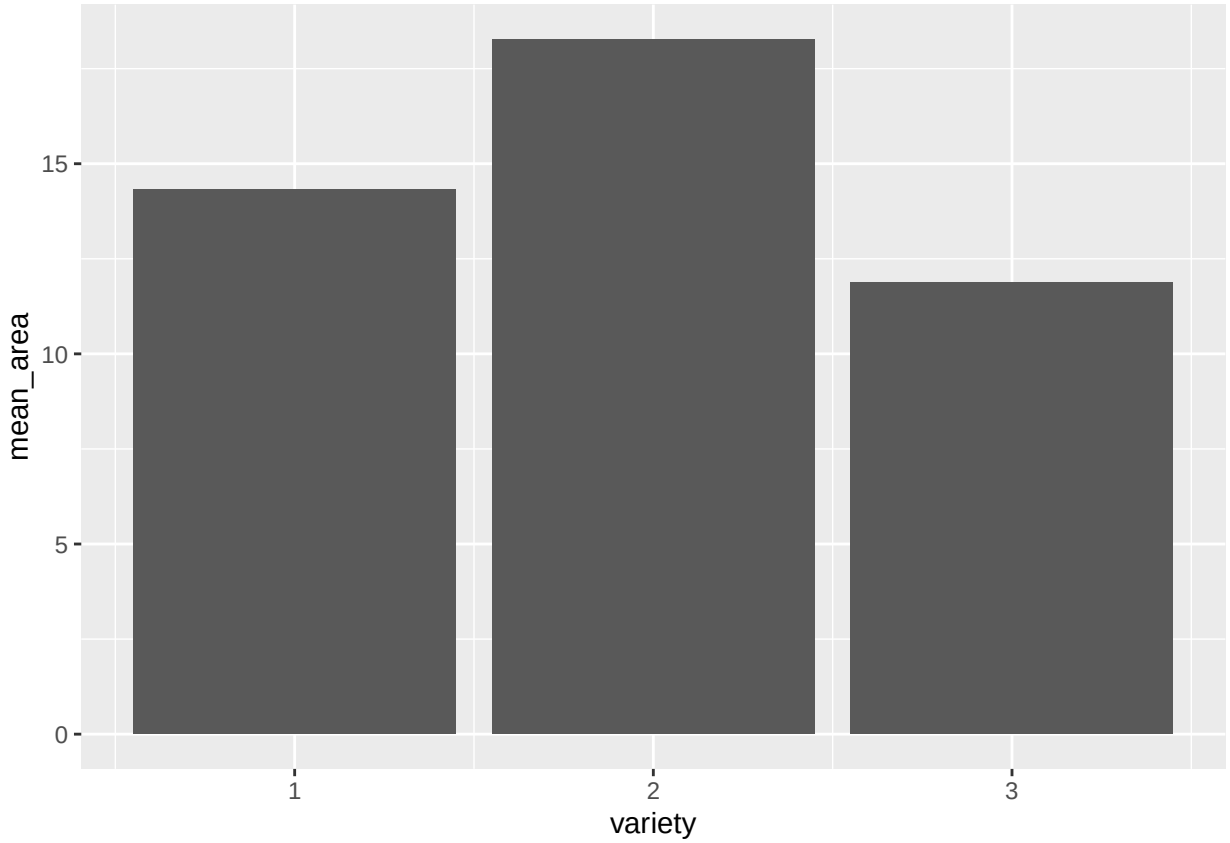

```
## bivariate relationships between X's (continuous) and Y, variety (categorical)
### area
ggplot(seeds_train,  # strip plot
       aes(y = variety,
           x = area)) +
  geom_point() +
  labs(title = "Area distribution by Variety")
```




```
plotdata_area <- seeds_train %>% # mean/sem plot
  group_by(variety) %>%
  summarize(n = n(),
            mean = mean(area),
            sd = sd(area),
            se = sd / sqrt(n),
            ci = qt(0.975, df = n - 1) * sd / sqrt(n))
ggplot(plotdata_area,
       aes(x = variety,
           y = mean,
           group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  geom_errorbar(aes(ymin = mean - se,
                    ymax = mean + se),
                width = .1)
```




```
plotdata_area_1 <- seeds_train %>% # bar plot
  group_by(variety) %>%
  summarize(mean_area = mean(area))
ggplot(plotdata_area_1,
       aes(x = variety,
           y = mean_area)) +
  geom_bar(stat = "identity")
```



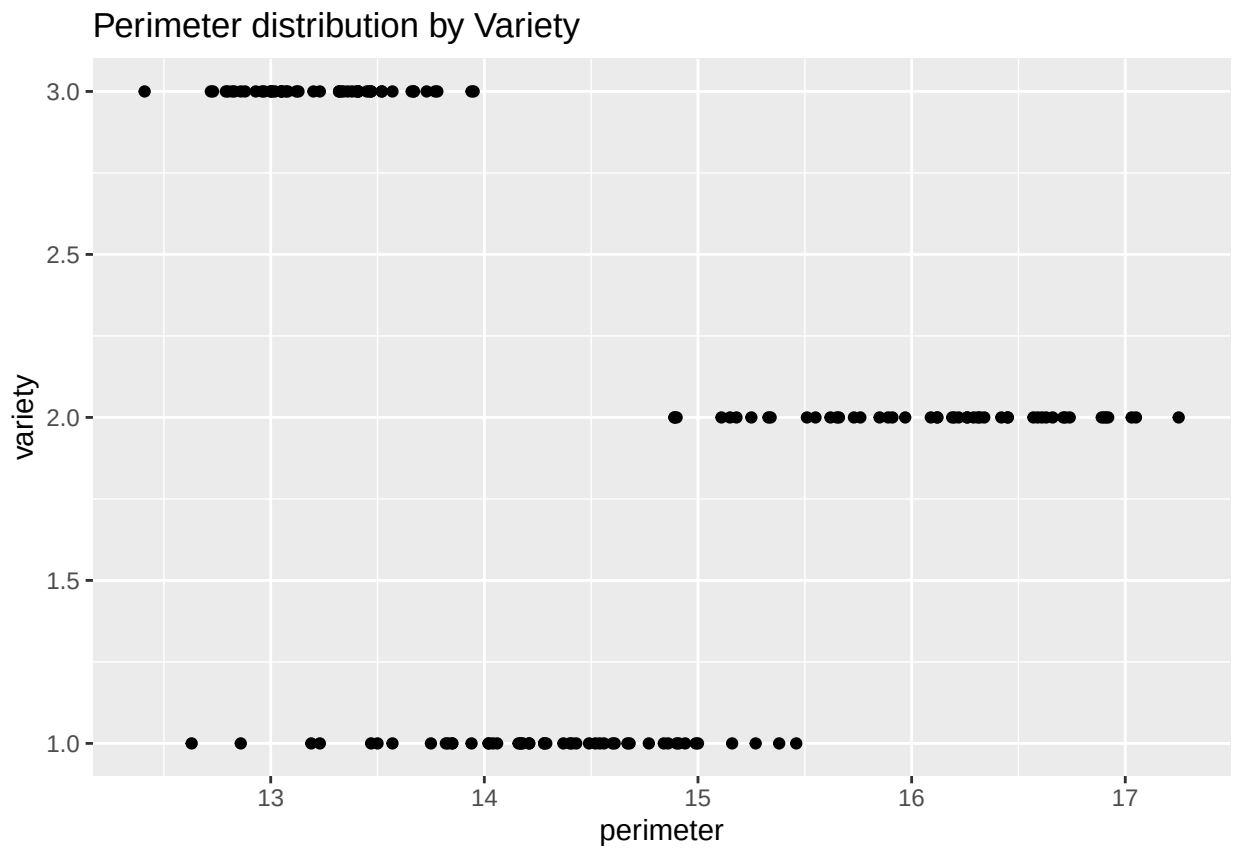

```
cor(seeds_train$area, seeds_train$variety)
```

```
## [1] -0.3457642
```

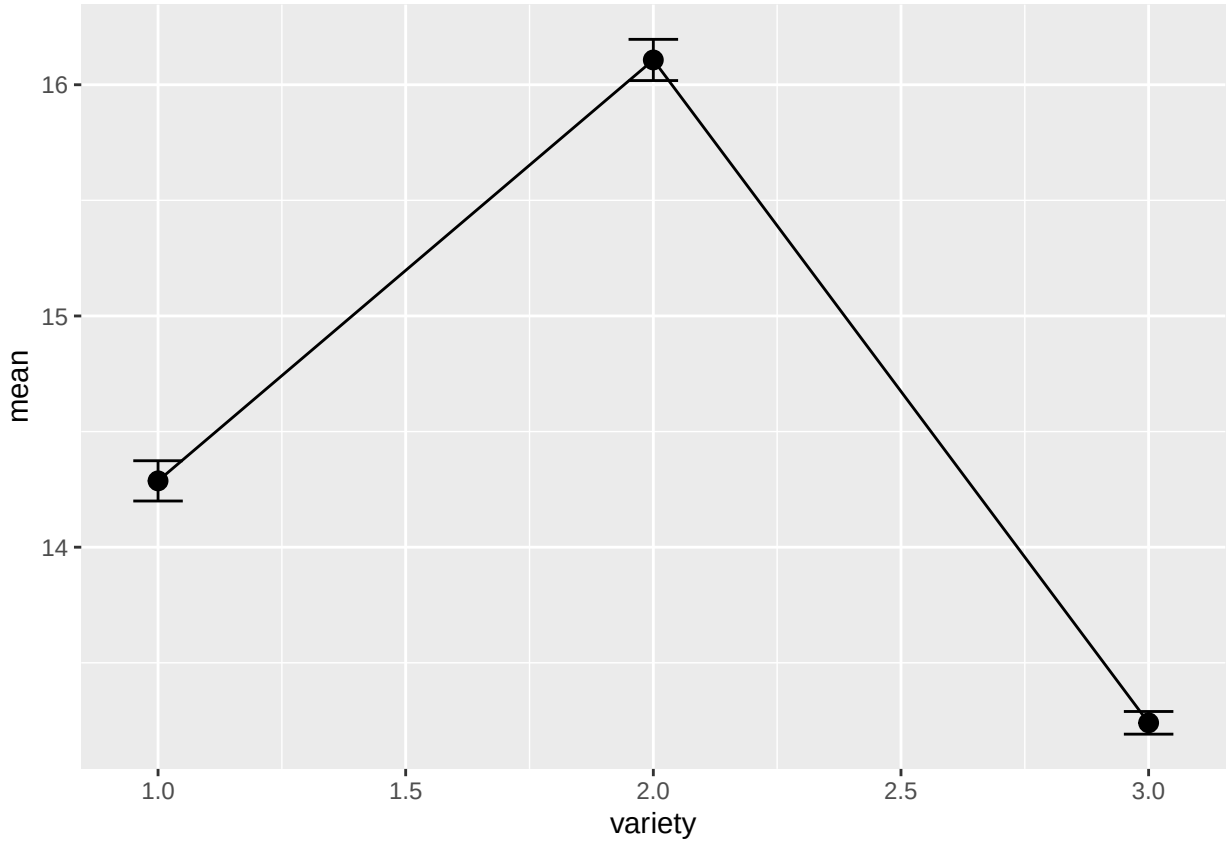
```
fit_area <- lm(variety ~ area, data=seeds_train)
summary(fit_area)
```

```
##
## Call:
## lm(formula = variety ~ area, data = seeds_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3498 -0.9762  0.3798  0.6502  0.8587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.44372    0.32815  10.494  < 2e-16 ***
## area        -0.09741    0.02173  -4.483 1.47e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7713 on 148 degrees of freedom
## Multiple R-squared:  0.1196, Adjusted R-squared:  0.1136
## F-statistic:  20.1 on 1 and 148 DF,  p-value: 1.466e-05
```

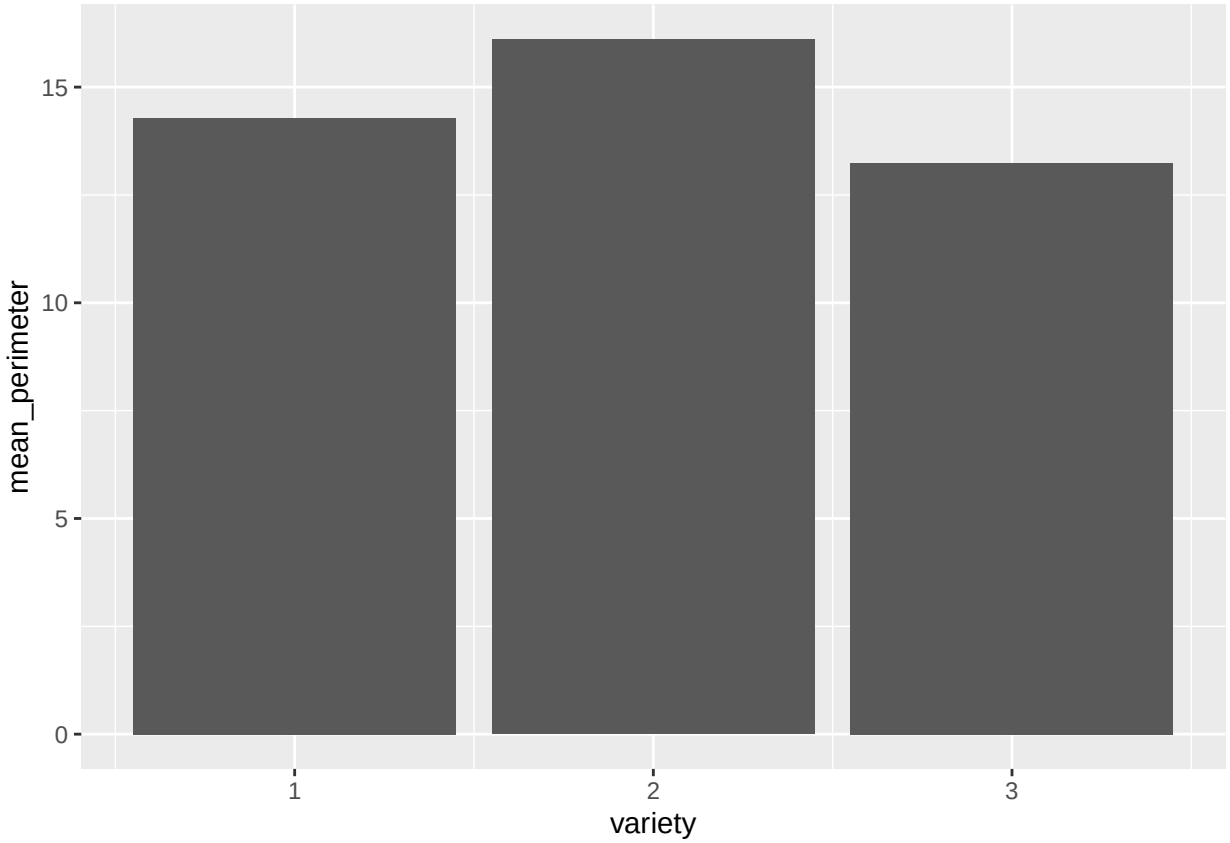
```
### perimeter
ggplot(seeds_train,  # strip plot
       aes(y = variety,
           x = perimeter)) +
  geom_point() +
  labs(title = "Perimeter distribution by Variety")
```




```
plotdata_perimeter <- seeds_train %>% # mean/sem plot
  group_by(variety) %>%
  summarize(n = n(),
            mean = mean(perimeter),
            sd = sd(perimeter),
            se = sd / sqrt(n),
            ci = qt(0.975, df = n - 1) * sd / sqrt(n))
ggplot(plotdata_perimeter,
       aes(x = variety,
           y = mean,
           group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  geom_errorbar(aes(ymin = mean - se,
                    ymax = mean + se),
                width = .1)
```




```
plotdata_perimeter_1 <- seeds_train %>% # bar plot
  group_by(variety) %>%
  summarize(mean_perimeter = mean(perimeter))
ggplot(plotdata_perimeter_1,
       aes(x = variety,
           y = mean_perimeter)) +
  geom_bar(stat = "identity")
```



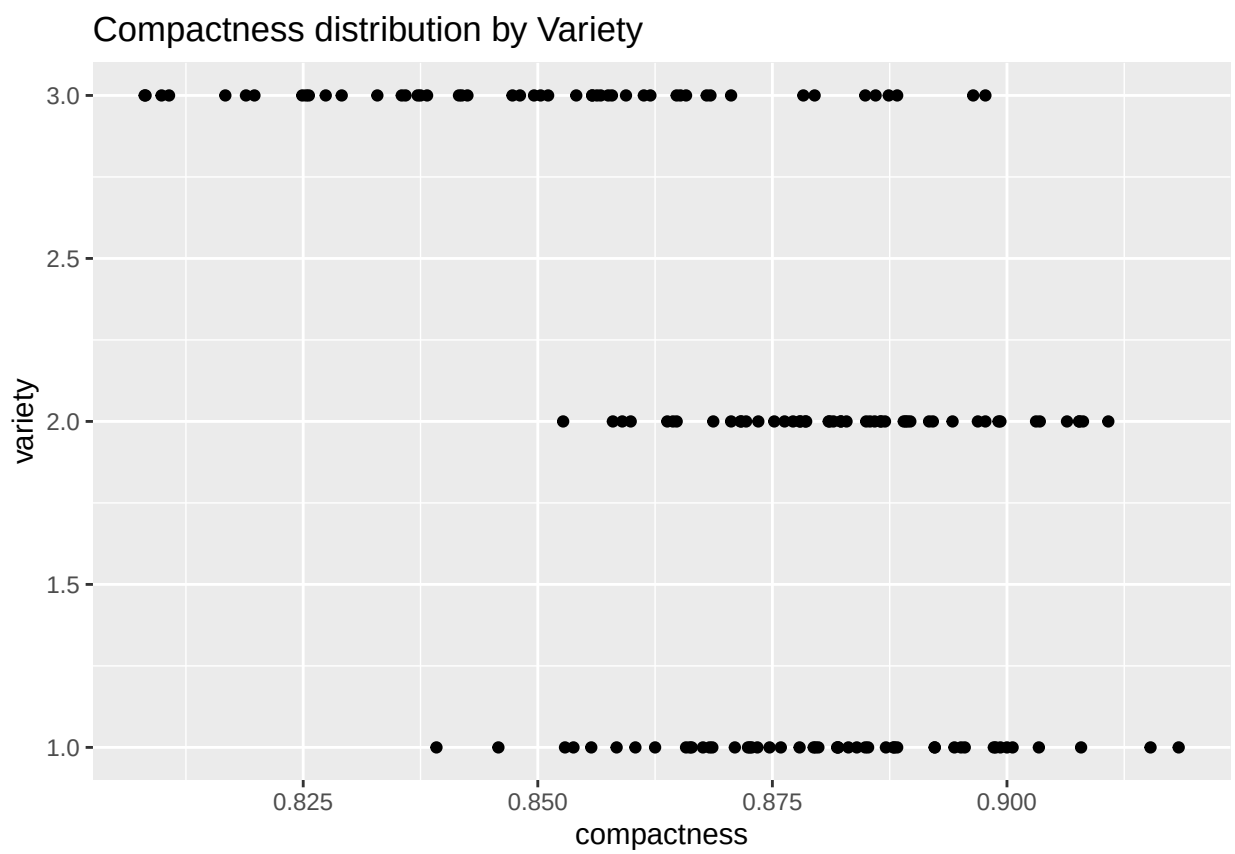

```
cor(seeds_train$perimeter, seeds_train$variety)
```

```
## [1] -0.328031
```

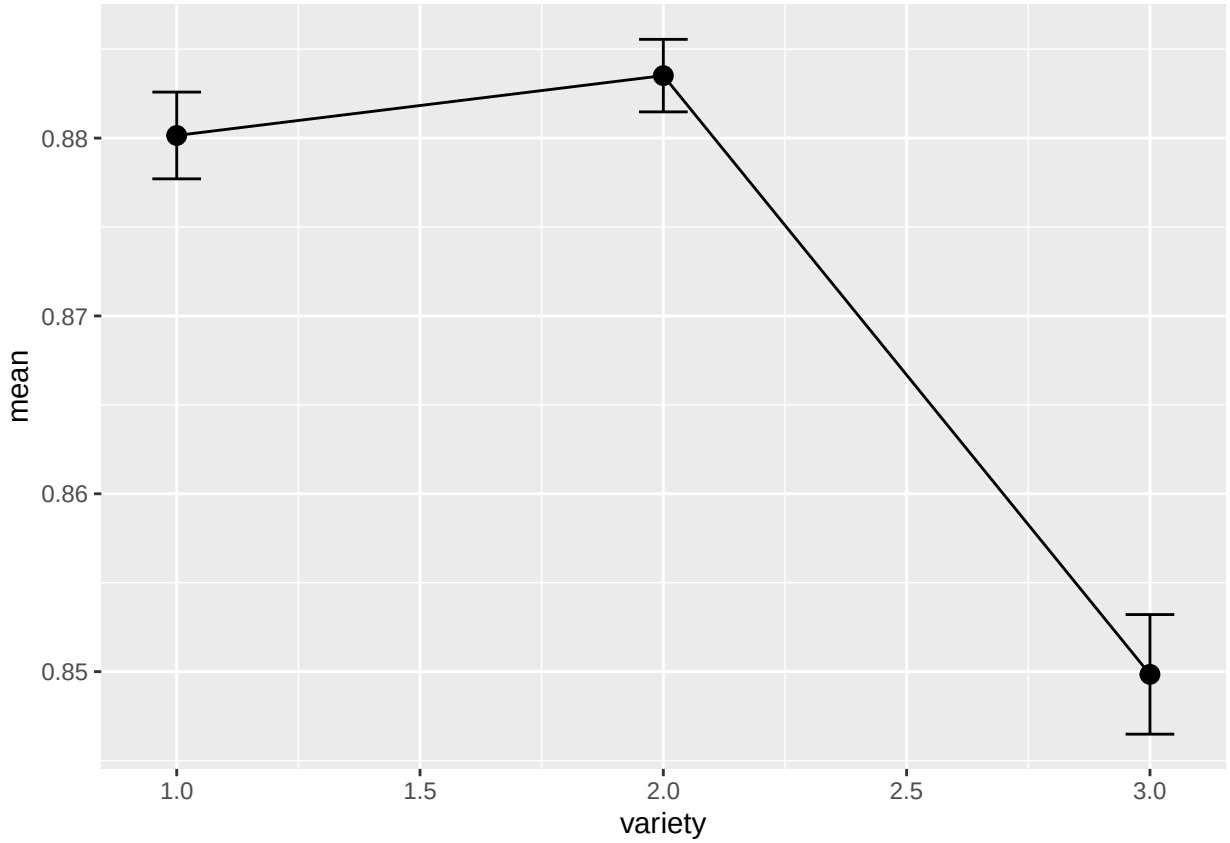
```
fit_perimeter <- lm(variety ~ perimeter, data=seeds_train)
summary(fit_perimeter)
```

```
##
## Call:
## lm(formula = variety ~ perimeter, data = seeds_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3938 -0.9676  0.3425  0.6807  0.8777
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.99137    0.71095   7.021 7.42e-11 ***
## perimeter   -0.20567    0.04868  -4.224 4.17e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7765 on 148 degrees of freedom
## Multiple R-squared:  0.1076, Adjusted R-squared:  0.1016
## F-statistic: 17.85 on 1 and 148 DF,  p-value: 4.169e-05
```

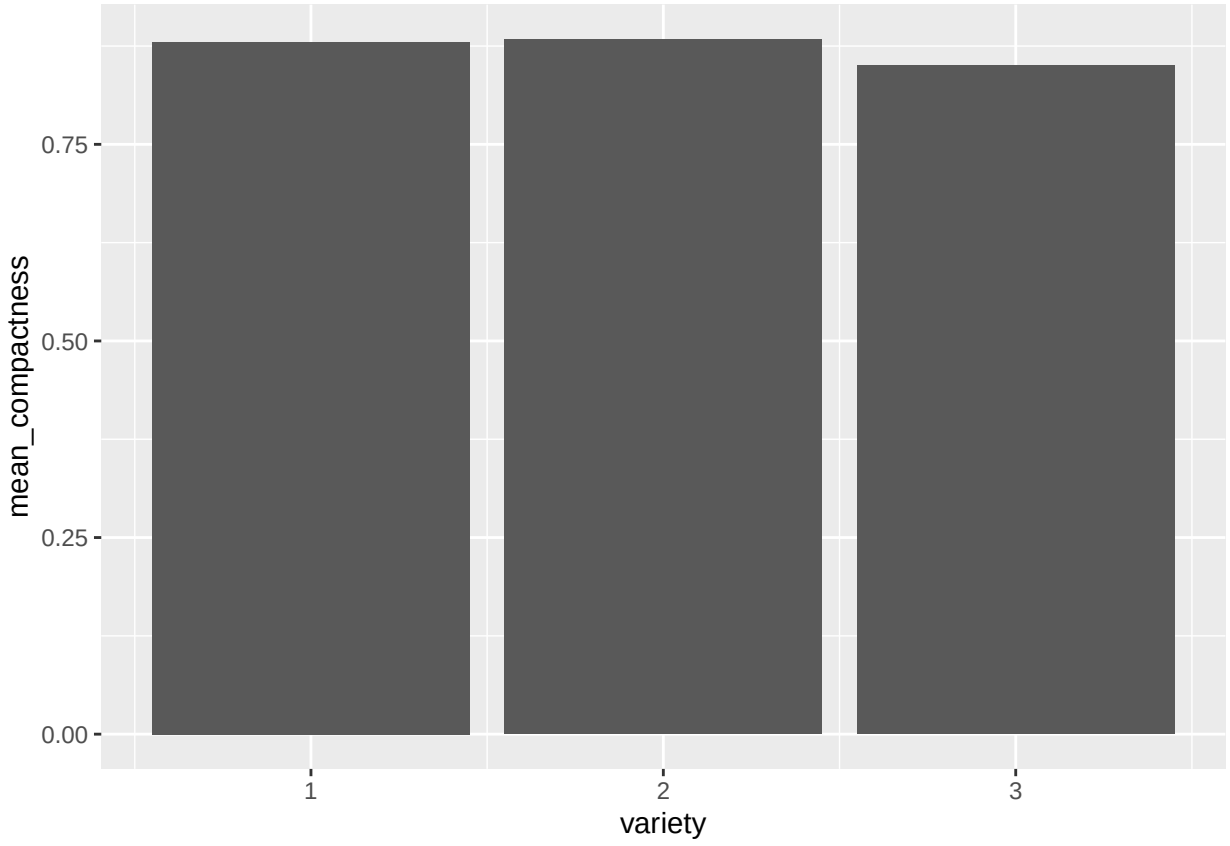
```
### compactness
ggplot(seeds_train,  # strip plot
       aes(y = variety,
           x = compactness)) +
  geom_point() +
  labs(title = "Compactness distribution by Variety")
```




```
plotdata_compactness <- seeds_train %>% # mean/sem plot
  group_by(variety) %>%
  summarize(n = n(),
            mean = mean(compactness),
            sd = sd(compactness),
            se = sd / sqrt(n),
            ci = qt(0.975, df = n - 1) * sd / sqrt(n))
ggplot(plotdata_compactness,
       aes(x = variety,
           y = mean,
           group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  geom_errorbar(aes(ymin = mean - se,
                    ymax = mean + se),
                width = .1)
```




```
plotdata_compactness_1 <- seeds_train %>% # bar plot
  group_by(variety) %>%
  summarize(mean_compactness = mean(compactness))
ggplot(plotdata_compactness_1,
       aes(x = variety,
           y = mean_compactness)) +
  geom_bar(stat = "identity")
```



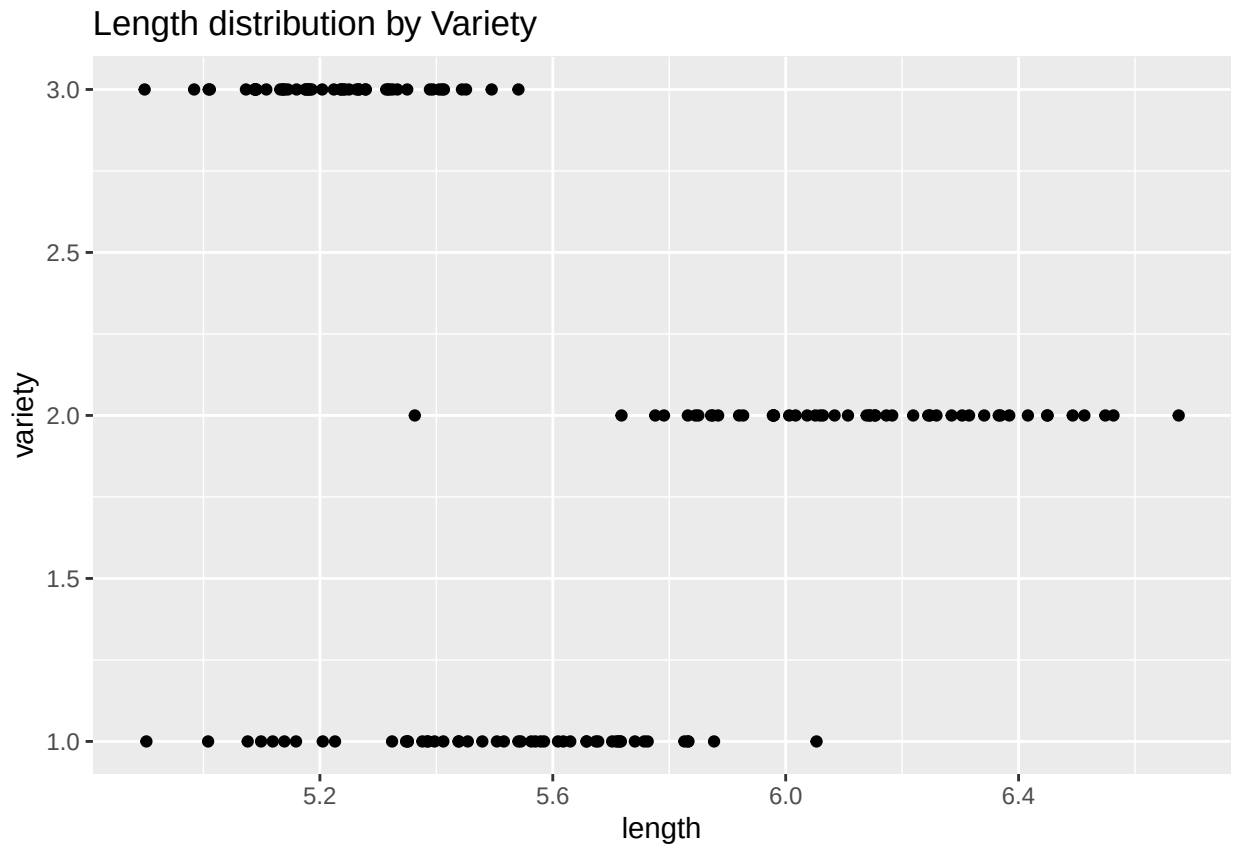

```
cor(seeds_train$compactness, seeds_train$variety)
```

```
## [1] -0.5140942
```

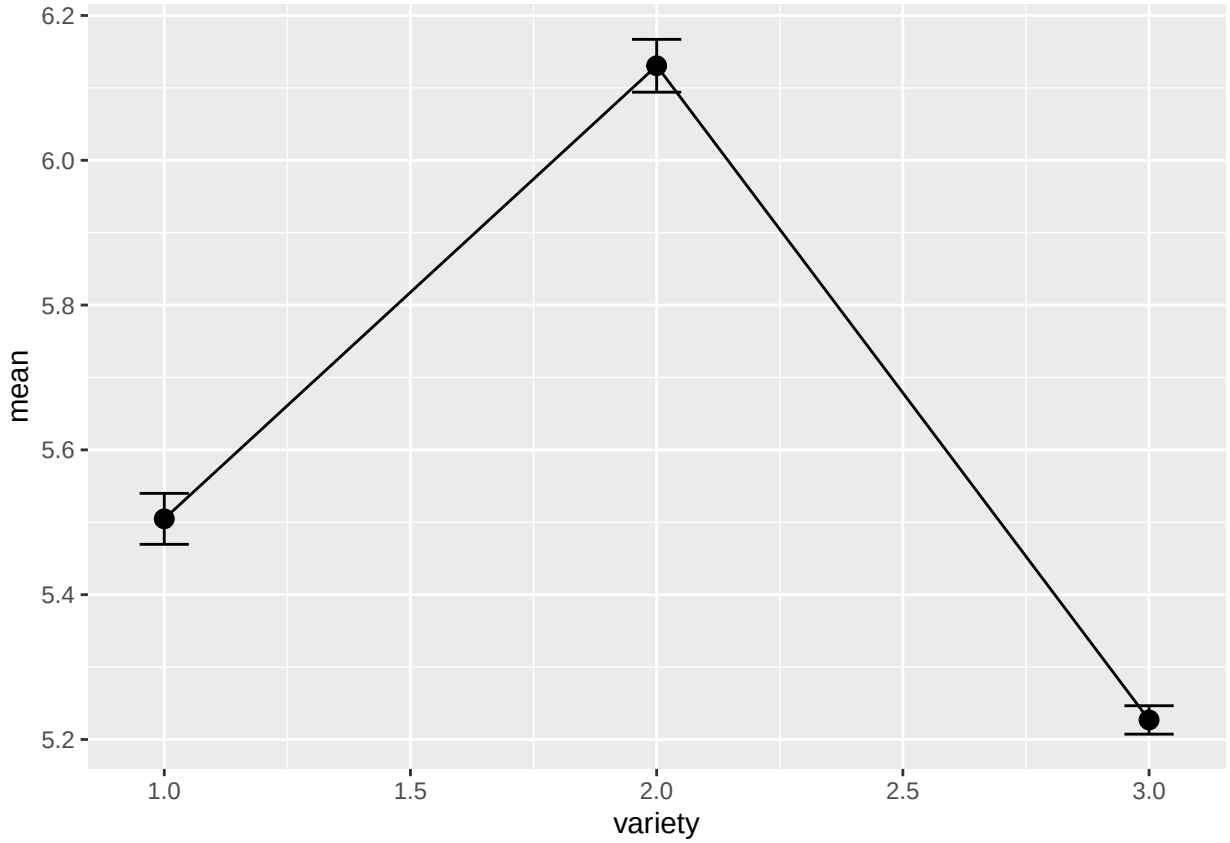
```
fit_compactness <- lm(variety ~ compactness, data=seeds_train)
summary(fit_compactness)
```

```
##
## Call:
## lm(formula = variety ~ compactness, data = seeds_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5577 -0.6222  0.1235  0.4904  1.4628
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   17.197      2.085   8.248 8.06e-14 ***
## compactness  -17.444      2.392  -7.292 1.71e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7051 on 148 degrees of freedom
## Multiple R-squared:  0.2643, Adjusted R-squared:  0.2593
## F-statistic: 53.17 on 1 and 148 DF,  p-value: 1.712e-11
```

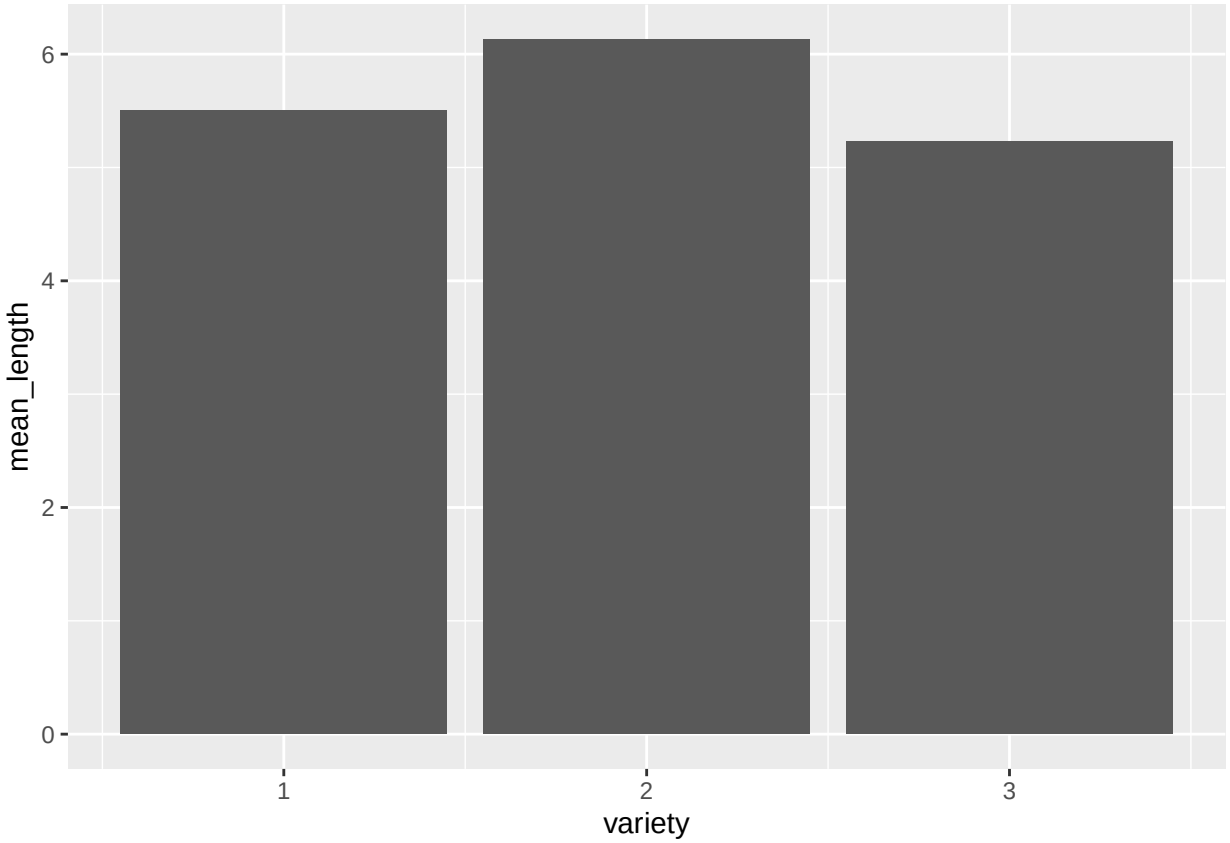
```
### length
ggplot(seeds_train,  # strip plot
       aes(y = variety,
           x = length)) +
  geom_point() +
  labs(title = "Length distribution by Variety")
```




```
plotdata_length <- seeds_train %>% # mean/sem plot
  group_by(variety) %>%
  summarize(n = n(),
          mean = mean(length),
          sd = sd(length),
          se = sd / sqrt(n),
          ci = qt(0.975, df = n - 1) * sd / sqrt(n))
ggplot(plotdata_length,
       aes(x = variety,
           y = mean,
           group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  geom_errorbar(aes(ymin = mean - se,
                    ymax = mean + se),
                width = .1)
```




```
plotdata_length_1 <- seeds_train %>% # bar plot
  group_by(variety) %>%
  summarize(mean_length = mean(length))
ggplot(plotdata_length_1,
       aes(x = variety,
           y = mean_length)) +
  geom_bar(stat = "identity")
```



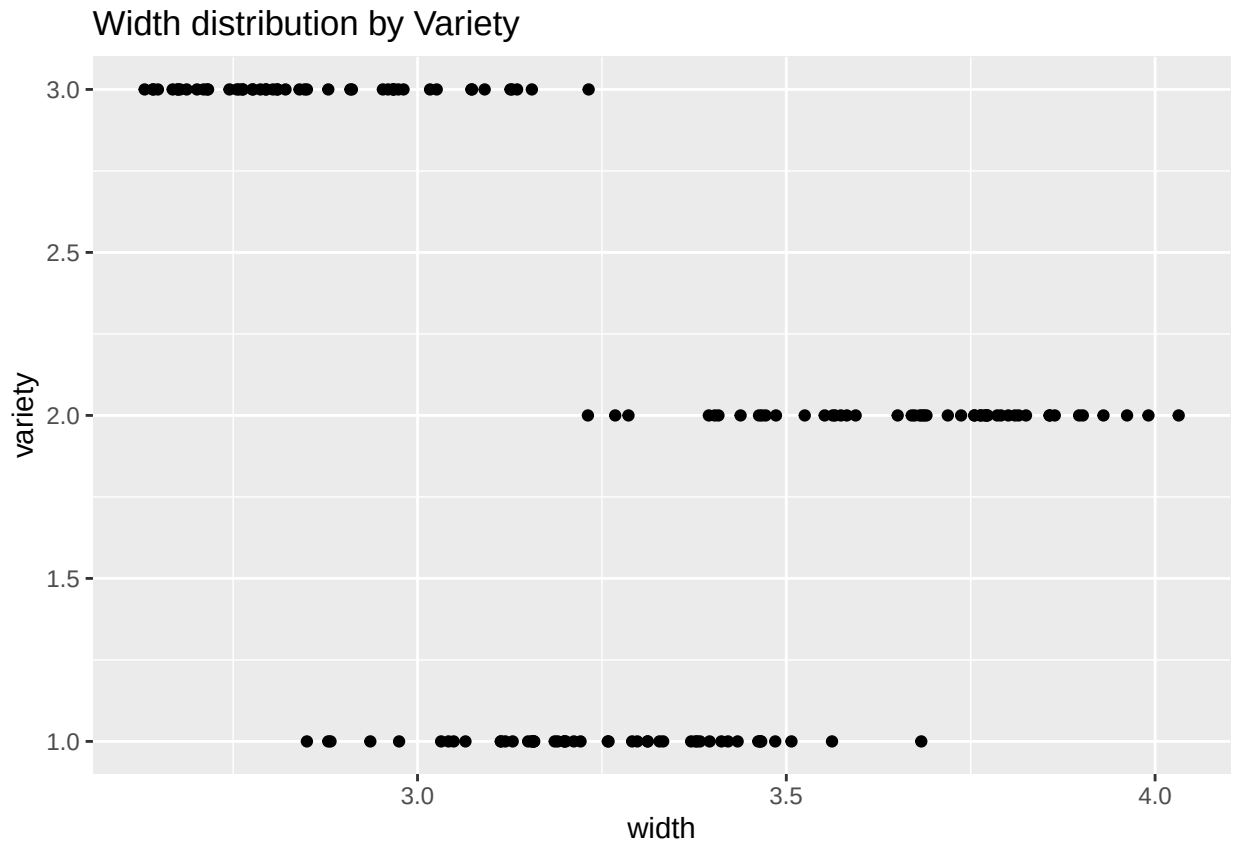

```
cor(seeds_train$length, seeds_train$variety)
```

```
## [1] -0.2594278
```

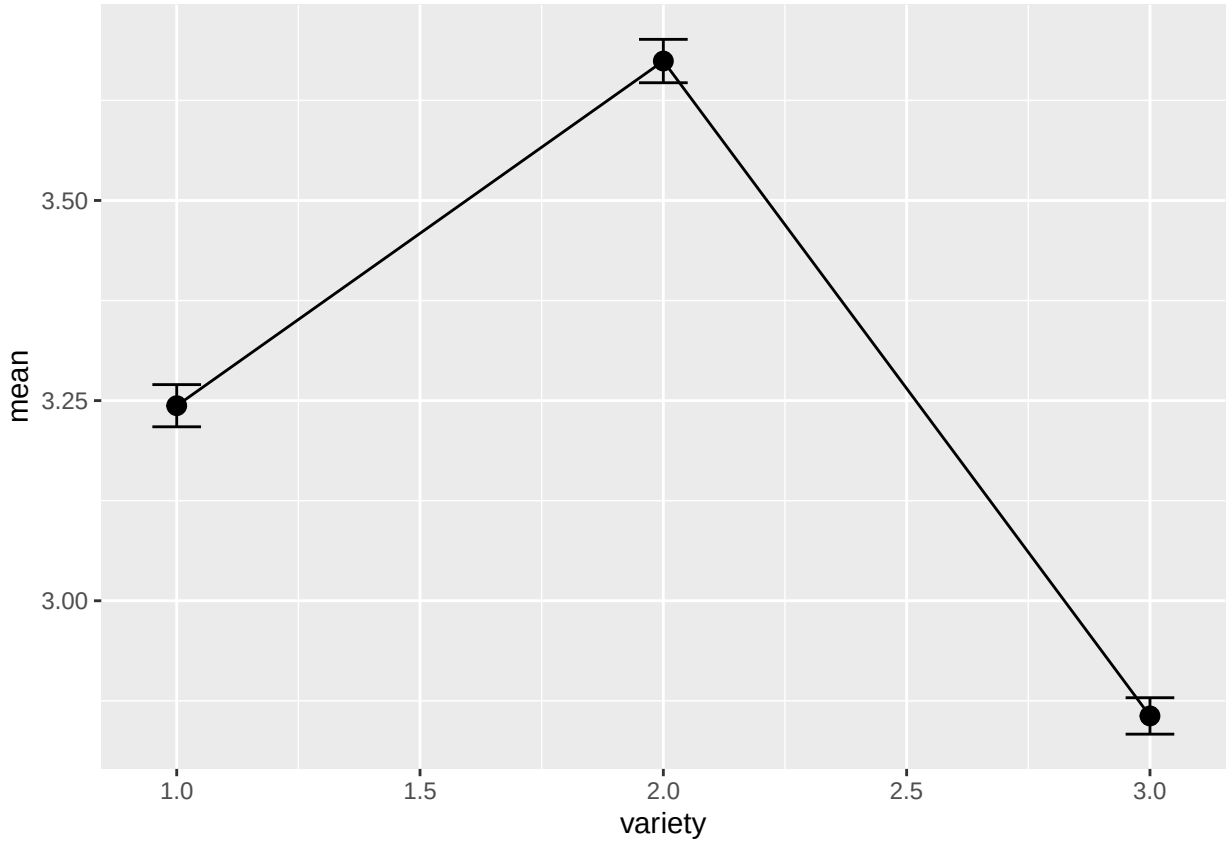
```
fit_length <- lm(variety ~ length, data=seeds_train)
summary(fit_length)
```

```
##
## Call:
## lm(formula = variety ~ length, data = seeds_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3483 -0.9598  0.2523  0.7646  0.9614
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.7237     0.8360   5.651 7.96e-08 ***
## length       -0.4846     0.1483  -3.268  0.00135 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7939 on 148 degrees of freedom
## Multiple R-squared:  0.0673, Adjusted R-squared:  0.061
## F-statistic: 10.68 on 1 and 148 DF,  p-value: 0.001347
```

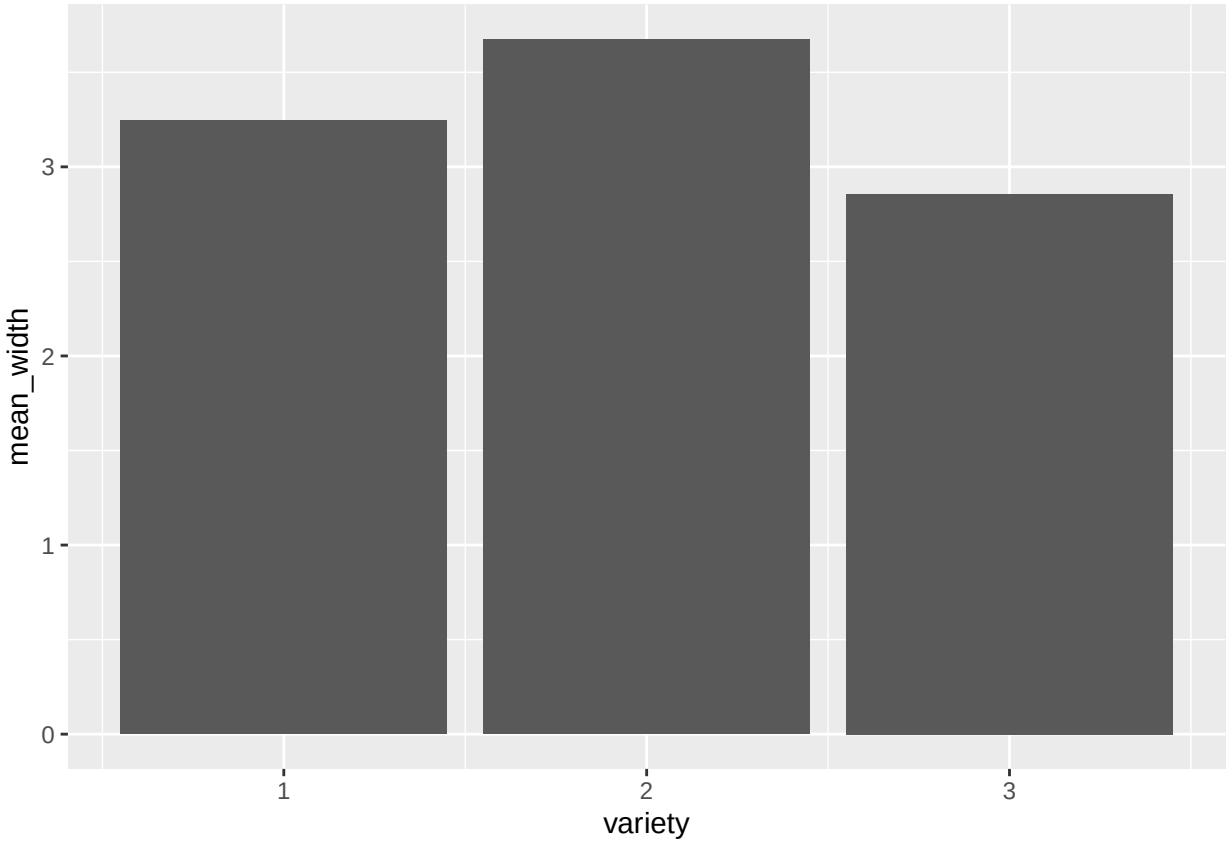
```
### width
ggplot(seeds_train,  # strip plot
       aes(y = variety,
           x = width)) +
  geom_point() +
  labs(title = "Width distribution by Variety")
```




```
plotdata_width <- seeds_train %>% # mean/sem plot
  group_by(variety) %>%
  summarize(n = n(),
          mean = mean(width),
          sd = sd(width),
          se = sd / sqrt(n),
          ci = qt(0.975, df = n - 1) * sd / sqrt(n))
ggplot(plotdata_width,
       aes(x = variety,
           y = mean,
           group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  geom_errorbar(aes(ymin = mean - se,
                    ymax = mean + se),
                width = .1)
```




```
plotdata_width_1 <- seeds_train %>% # bar plot
  group_by(variety) %>%
  summarize(mean_width = mean(width))
ggplot(plotdata_width_1,
       aes(x = variety,
           y = mean_width)) +
  geom_bar(stat = "identity")
```



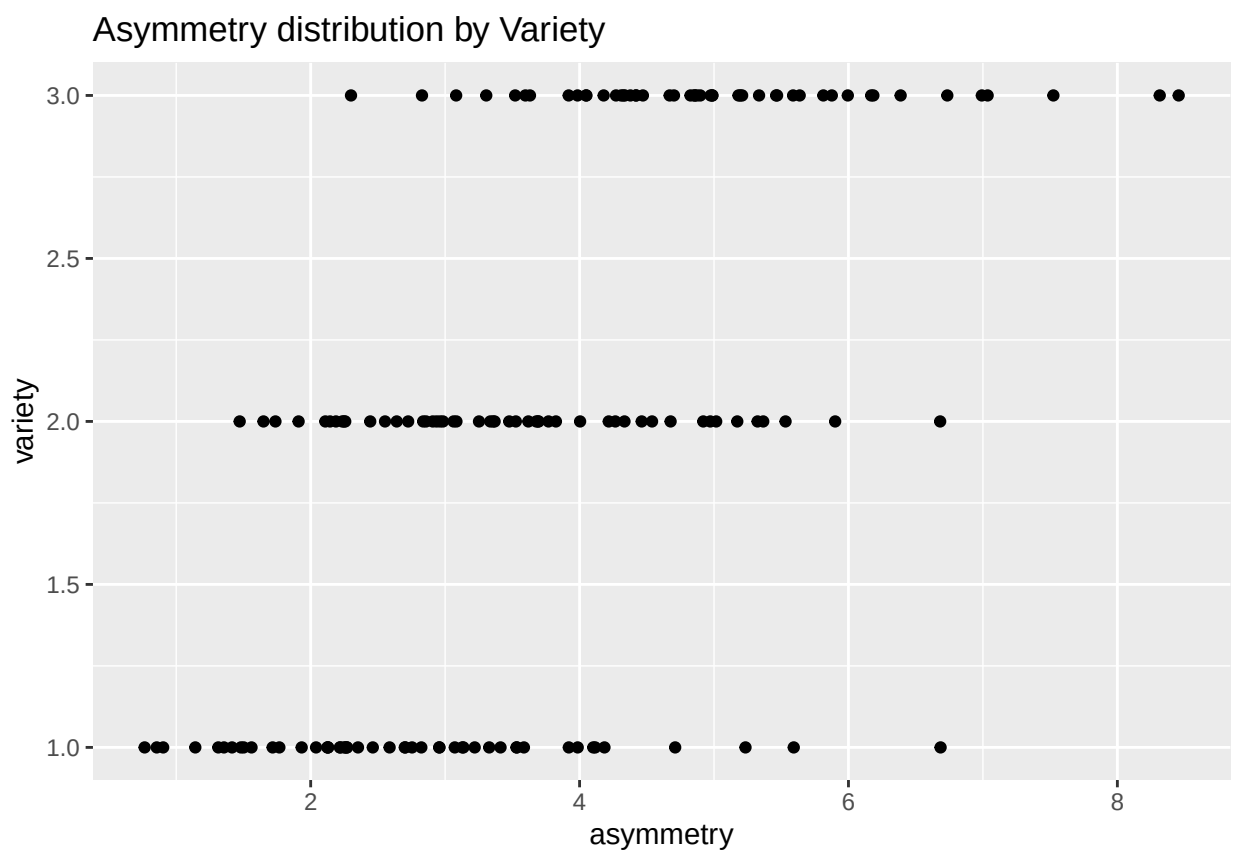

```
cor(seeds_train$width, seeds_train$variety)
```

```
## [1] -0.4177482
```

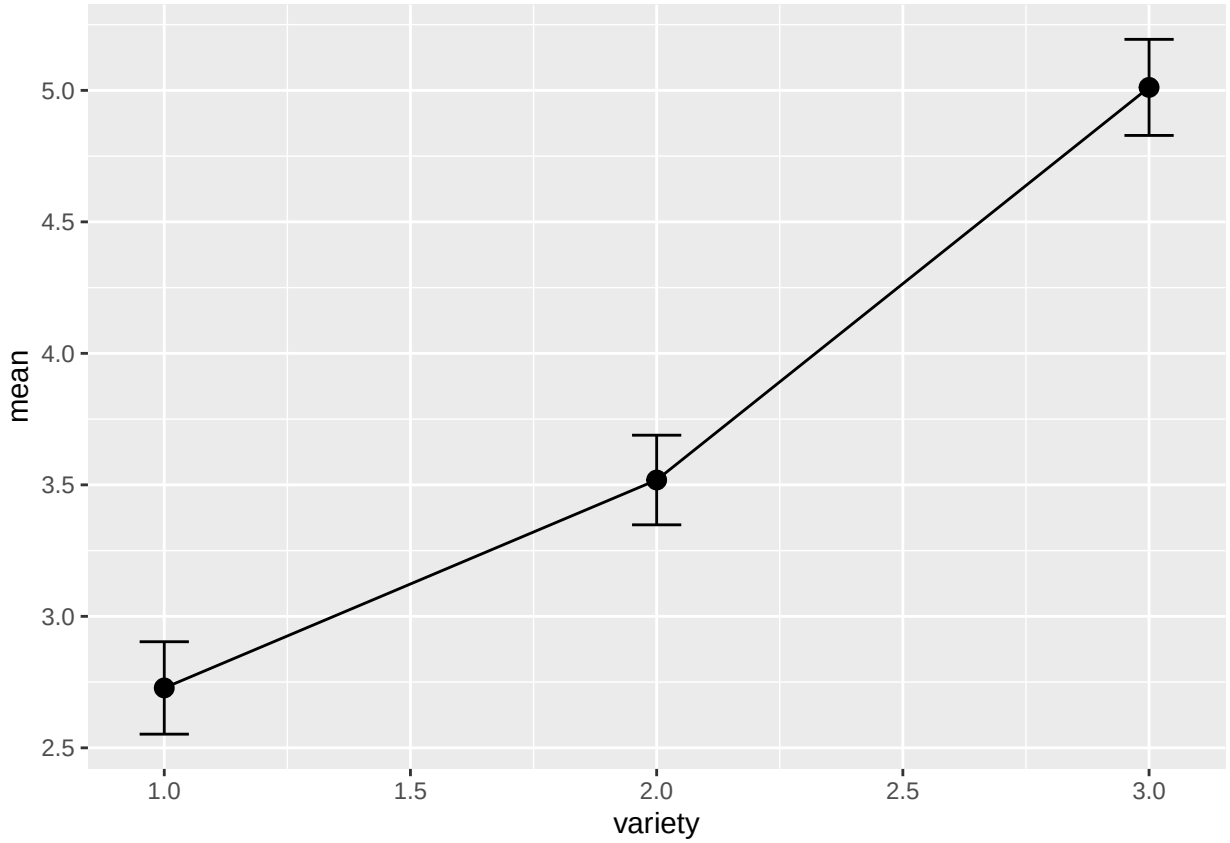
```
fit_width <- lm(variety ~ width, data=seeds_train)
summary(fit_width)
```

```
##
## Call:
## lm(formula = variety ~ width, data = seeds_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3675 -0.8845  0.4022  0.5630  0.9766
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.9344     0.5281   9.343  < 2e-16 ***
## width        -0.9007     0.1610  -5.594 1.04e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7468 on 148 degrees of freedom
## Multiple R-squared:  0.1745, Adjusted R-squared:  0.1689
## F-statistic: 31.29 on 1 and 148 DF,  p-value: 1.045e-07
```

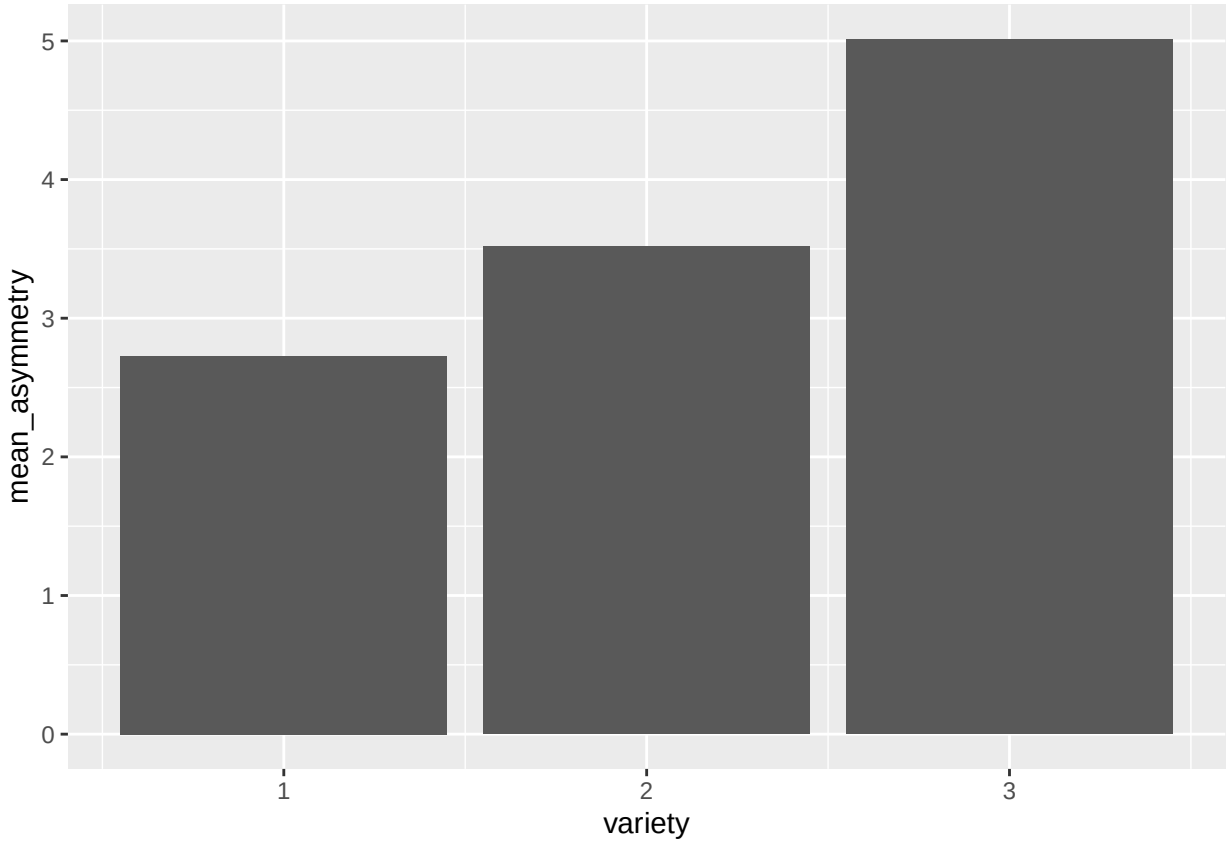
```
### asymmetry
ggplot(seeds_train,  # strip plot
       aes(y = variety,
           x = asymmetry)) +
  geom_point() +
  labs(title = "Asymmetry distribution by Variety")
```




```
plotdata_asymmetry <- seeds_train %>% # mean/sem plot
  group_by(variety) %>%
  summarize(n = n(),
            mean = mean(asymmetry),
            sd = sd(asymmetry),
            se = sd / sqrt(n),
            ci = qt(0.975, df = n - 1) * sd / sqrt(n))
ggplot(plotdata_asymmetry,
       aes(x = variety,
           y = mean,
           group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  geom_errorbar(aes(ymin = mean - se,
                    ymax = mean + se),
                width = .1)
```




```
plotdata_asymmetry_1 <- seeds_train %>% # bar plot
  group_by(variety) %>%
  summarize(mean_asymmetry = mean(asymmetry))
ggplot(plotdata_asymmetry_1,
       aes(x = variety,
           y = mean_asymmetry)) +
  geom_bar(stat = "identity")
```



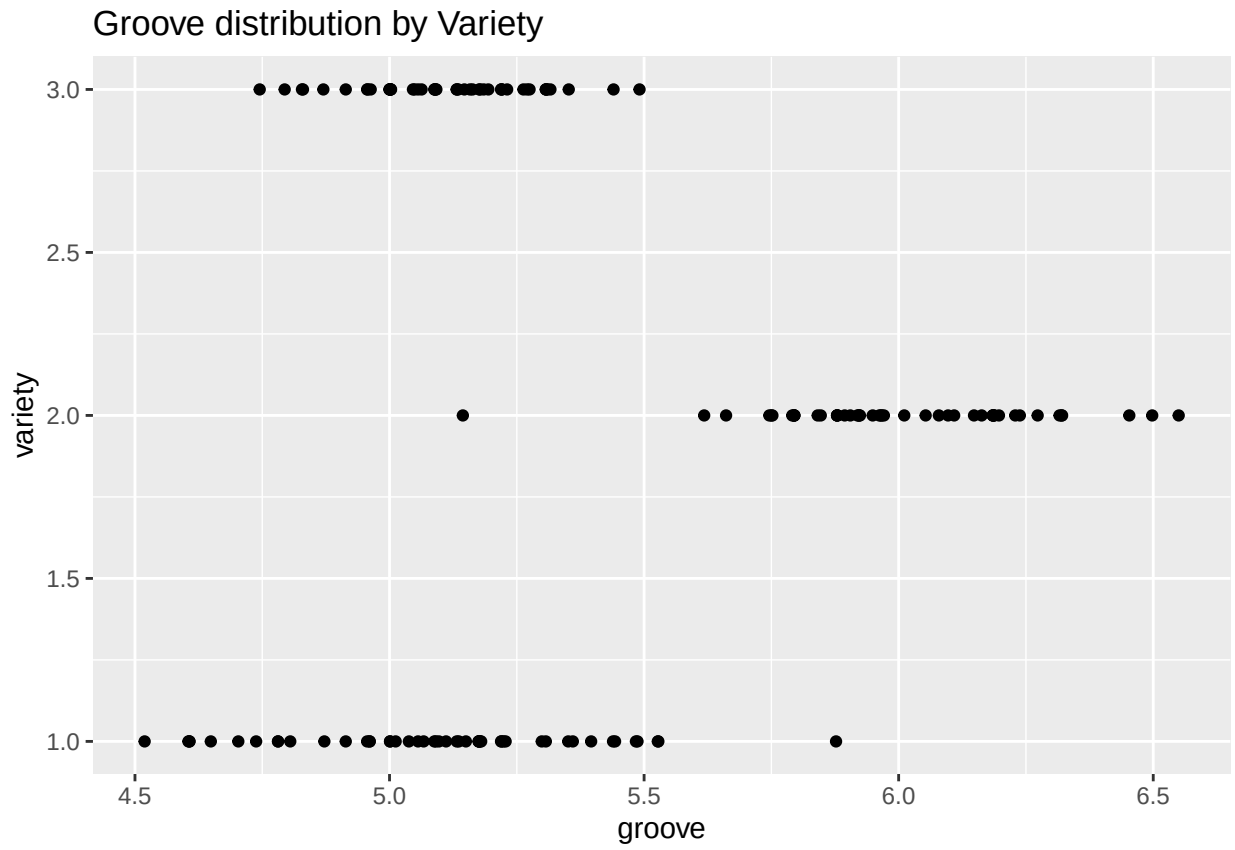

```
cor(seeds_train$asymmetry, seeds_train$variety)
```

```
## [1] 0.599249
```

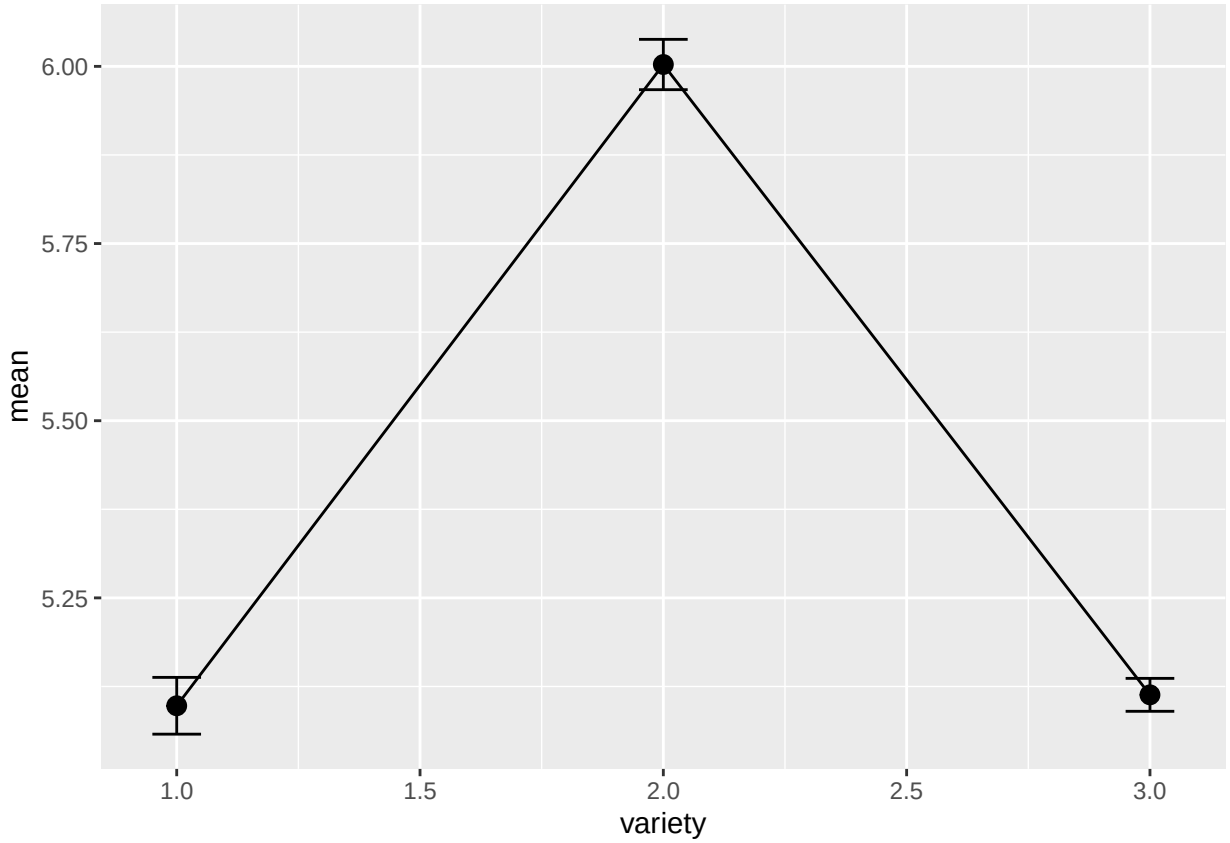
```
fit_asymmetry <- lm(variety ~ asymmetry, data=seeds_train)
summary(fit_asymmetry)
```

```
##
## Call:
## lm(formula = variety ~ asymmetry, data = seeds_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.92230 -0.49304  0.02139  0.51415  1.45683
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.81979    0.14030   5.843 3.14e-08 ***
## asymmetry    0.31451    0.03454   9.106 5.44e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6581 on 148 degrees of freedom
## Multiple R-squared:  0.3591, Adjusted R-squared:  0.3548
## F-statistic: 82.93 on 1 and 148 DF,  p-value: 5.444e-16
```

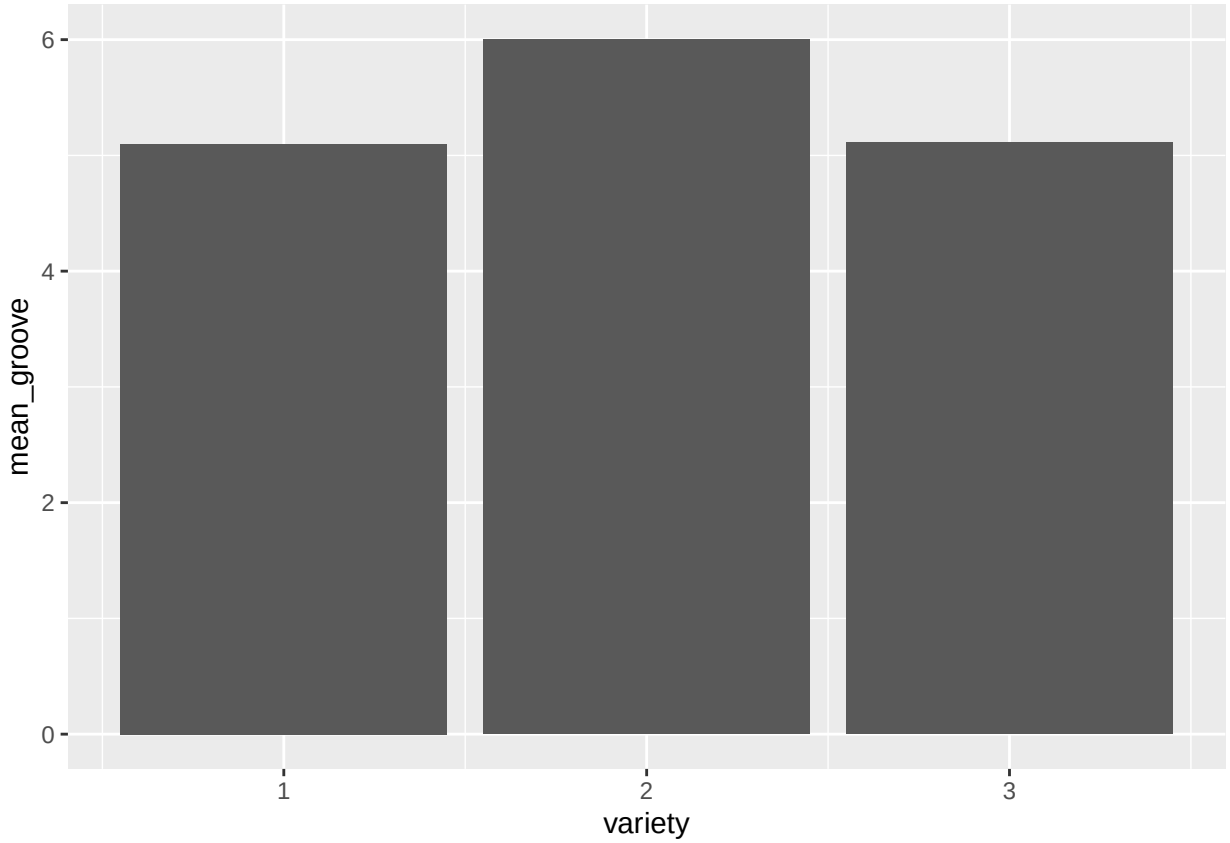
```
### groove
ggplot(seeds_train,  # strip plot
       aes(y = variety,
           x = groove)) +
  geom_point() +
  labs(title = "Groove distribution by Variety")
```




```
plotdata_groove <- seeds_train %>% # mean/sem plot
  group_by(variety) %>%
  summarize(n = n(),
          mean = mean(groove),
          sd = sd(groove),
          se = sd / sqrt(n),
          ci = qt(0.975, df = n - 1) * sd / sqrt(n))
ggplot(plotdata_groove,
       aes(x = variety,
           y = mean,
           group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  geom_errorbar(aes(ymin = mean - se,
                    ymax = mean + se),
                width = .1)
```




```
plotdata_groove_1 <- seeds_train %>% # bar plot
  group_by(variety) %>%
  summarize(mean_groove = mean(groove))
ggplot(plotdata_groove_1,
       aes(x = variety,
           y = mean_groove)) +
  geom_bar(stat = "identity")
```




```
cor(seeds_train$groove, seeds_train$variety)
```

```
## [1] 0.01299946
```

```
fit_groove <- lm(variety ~ groove, data=seeds_train)
summary(fit_groove)
```

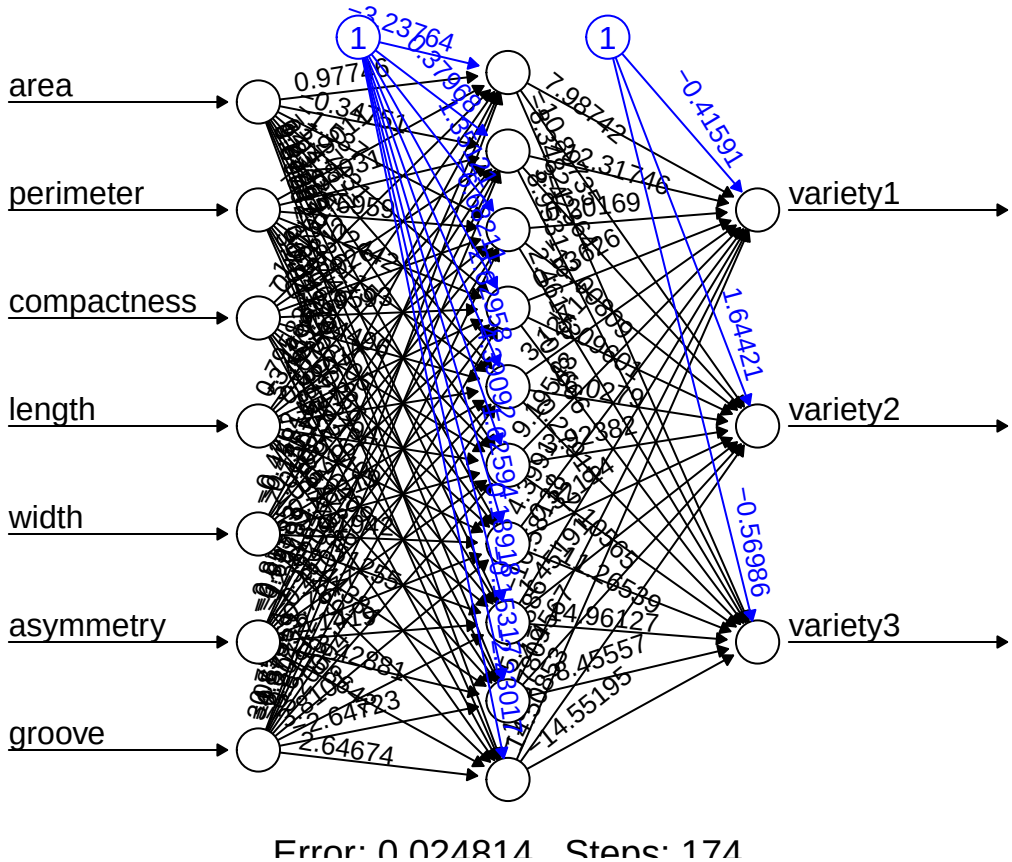
```
##
## Call:
## lm(formula = variety ~ groove, data = seeds_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01036 -0.98994 -0.01225  1.00397  1.01446
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.88155    0.75196   2.502   0.0134 *
## groove       0.02192    0.13858   0.158   0.8745
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8219 on 148 degrees of freedom
## Multiple R-squared:  0.000169,   Adjusted R-squared:  -0.006587
## F-statistic: 0.02501 on 1 and 148 DF,  p-value: 0.8745
```

```r
# Feature Engineering
## set up factors and standardize the training data
variety <- as.data.frame(class.ind(as.factor(seeds_train$variety)))
names(variety) <- c("variety1", "variety2", "variety3")
seeds_train_std <- scale(seeds_train[, c(1:7)])
## put factors and continuous variables back together
seeds_train_ind <- cbind(seeds_train_std, variety)
## set up factors and standardize the testing data
variety_1 <- as.data.frame(class.ind(as.factor(seeds_test$variety)))
names(variety_1) <- c("variety1", "variety2", "variety3")
seeds_test_std <- scale(seeds_test[, c(1:7)])
## put factors and continuous variables back together
seeds_test_ind <- cbind(seeds_test_std, variety_1)
```

```r
# Hyperparameter Selection
## a)
### write the formula
nms <- names(seeds_train_ind)
frmla <- as.formula(paste("variety1 + variety2 + variety3 ~", paste(nms[!nms %in% c("variety1", "variet
frmla
```

```
## variety1 + variety2 + variety3 ~ area + perimeter + compactness +
##     length + width + asymmetry + groove
```

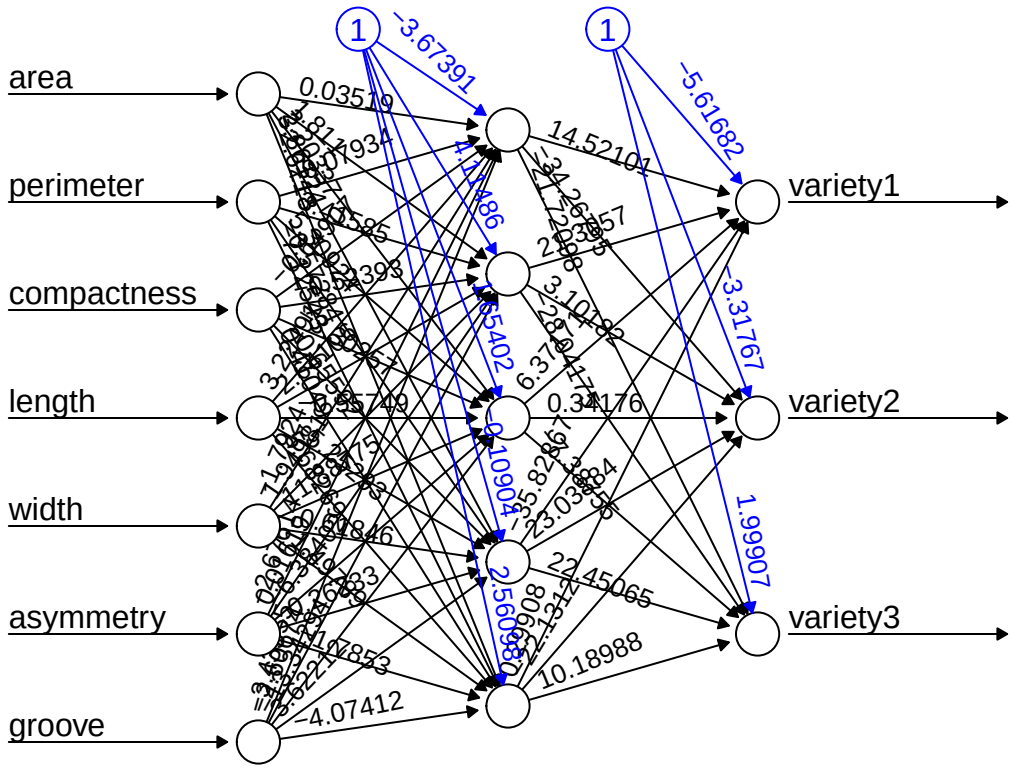
```r
### build a neural network: with 7 input features, I chose 10 nodes on a single hidden layer (hidden=10
mods <- neuralnet(frmla, data = seeds_train_ind,
                  rep = 10, act.fct = "logistic",
                  linear.output = FALSE, lifesign = "minimal",
                  hidden = 10)
plot(mods,rep = "best")
```




```
### look at the distribution of error -- is 10 enough repetitions?
mods$result.matrix[1,]
```

```
##   [1] 0.03162932 0.02536668 0.02481396 0.02888876 0.03722856 0.07437261
##   [7] 0.04012092 0.02506785 0.05065402 0.02779406
```

```
### try with decreased hidden layers because our data may not be that complicated
mods1 <- neuralnet(frmla, data = seeds_train_ind,
                   rep = 10, act.fct = "logistic",
                   linear.output = FALSE, lifesign = "minimal",
                   hidden = 5)
plot(mods1,rep = "best")
```



Error: 0.013631 Steps: 426

```
mods1$result.matrix[1,]
```

```
##  [1] 0.01363086 0.05087728 0.02209623 0.83254192 0.04942626 0.03285960
##  [7] 0.02804780 0.02881116 0.04515643 0.03375543
```

```
### print the best model weights and error
mods1$result.matrix[,which.min(mods1$result.matrix[1,])]
```

```
##                    error        reached.threshold                    steps
##             0.013630855             0.009721567           426.000000000
##   Intercept.to.1layhid1       area.to.1layhid1  perimeter.to.1layhid1
##            -3.673914614             0.035190418            -0.079335794
## compactness.to.1layhid1    length.to.1layhid1     width.to.1layhid1
##            -0.649074450             3.224913024            -1.792404607
##   asymmetry.to.1layhid1    groove.to.1layhid1  Intercept.to.1layhid2
##            -2.676140759            -3.433552549             4.114858231
##        area.to.1layhid2 perimeter.to.1layhid2 compactness.to.1layhid2
##             1.810999739             3.415851396            -0.523929658
##      length.to.1layhid2      width.to.1layhid2  asymmetry.to.1layhid2
##             2.641059617             1.949806072             0.016194413
##      groove.to.1layhid2  Intercept.to.1layhid3       area.to.1layhid3
##            -2.606740724             1.654017727             2.203367889
##   perimeter.to.1layhid3 compactness.to.1layhid3    length.to.1layhid3
##             5.309237561            -3.302572175            -0.557489405
##       width.to.1layhid3  asymmetry.to.1layhid3     groove.to.1layhid3
```

```
##            11.887753432            -6.384016747           -13.342942291
##   Intercept.to.1layhid4       area.to.1layhid4   perimeter.to.1layhid4
##            -0.109037367             3.861237397            -0.572844380
## compactness.to.1layhid4     length.to.1layhid4       width.to.1layhid4
##            -0.655917736            -5.252926368             0.618455005
##   asymmetry.to.1layhid4     groove.to.1layhid4   Intercept.to.1layhid5
##             0.262329389             3.622122170             2.560982555
##        area.to.1layhid5   perimeter.to.1layhid5 compactness.to.1layhid5
##            -2.621801945            -7.125433520             0.410487469
##      length.to.1layhid5      width.to.1layhid5   asymmetry.to.1layhid5
##             5.682694024             1.497887071            -1.785303182
##      groove.to.1layhid5   Intercept.to.variety1    1layhid1.to.variety1
##            -4.074116232            -5.616818010            14.521005087
##    1layhid2.to.variety1    1layhid3.to.variety1    1layhid4.to.variety1
##            21.305703241             6.371713057           -35.828674087
##    1layhid5.to.variety1   Intercept.to.variety2    1layhid1.to.variety2
##             0.299080742            -3.317665075           -34.267949939
##    1layhid2.to.variety2    1layhid3.to.variety2    1layhid4.to.variety2
##             3.101817095             0.341764862            23.038844122
##    1layhid5.to.variety2   Intercept.to.variety3    1layhid1.to.variety3
##           -22.131200296             1.999068712           -21.720983142
##    1layhid2.to.variety3    1layhid3.to.variety3    1layhid4.to.variety3
##           -28.041748114            -7.387546086            22.450647465
##    1layhid5.to.variety3
##            10.189877624
```

```
### here, I get the rep number for the rep with lowest error
which.min(mods1$result.matrix[1,])
```

```
## [1] 1
```

```
### compare predicted and observed on the training data
bestmod1.train <- as.data.frame(predict(mods1, rep = which.min(mods1$result.matrix[1,]),newdata = seeds
bestmod1.train <- cbind(bestmod1.train, seeds_train_ind[, c(8:10)])
table(round(bestmod1.train[, 1], 0), bestmod1.train[, 4])
```

```
##
##       0   1
##   0 100   0
##   1   0  50
```

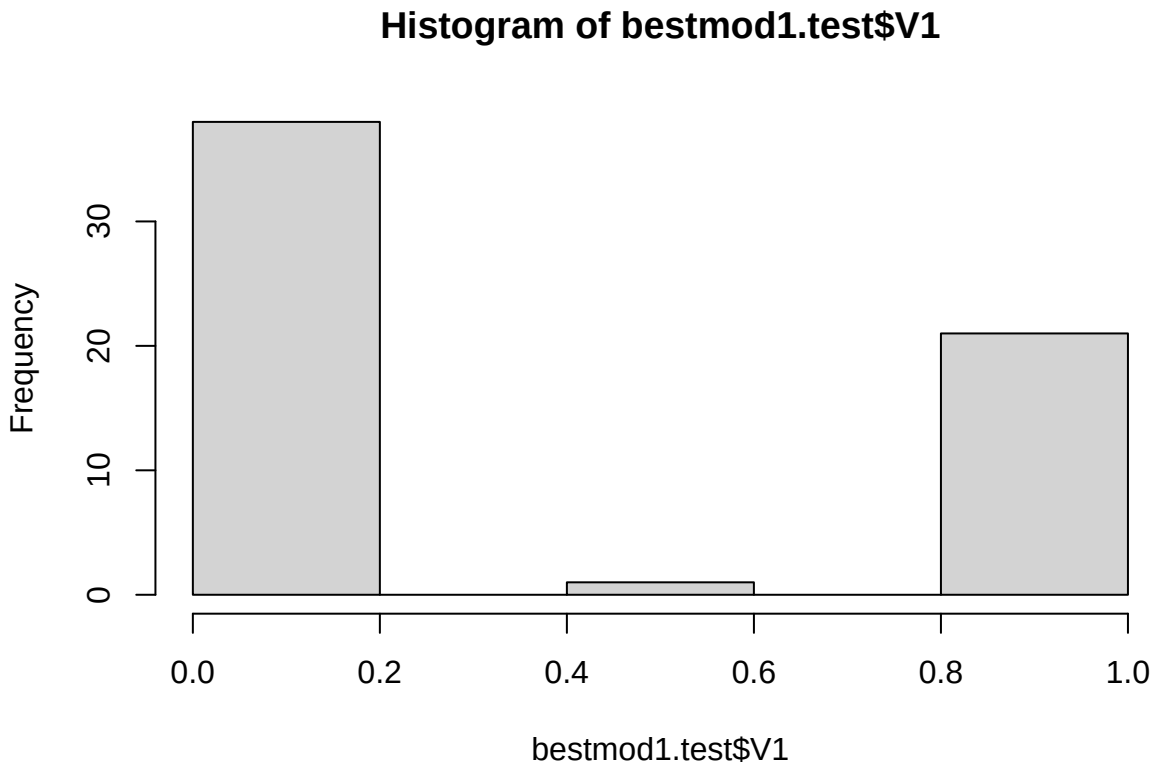
```
table(round(bestmod1.train[, 2], 0), bestmod1.train[, 5])
```

```
##
##       0   1
##   0 100   0
##   1   0  50
```

```
table(round(bestmod1.train[, 3], 0), bestmod1.train[, 6])
```

```
##
##       0   1
##   0 100   0
##   1   0  50
```

```
### now create predictions on the test data and compare to observed
seeds_test_noy <- subset(seeds_test_ind, select = c("area", "perimeter", "compactness", "length", "widt
bestmod1.test <- as.data.frame(predict(mods1, newdata = seeds_test_noy), rep = which.min(mods1$result.ma
bestmod1.test <- cbind(bestmod1.test, seeds_test_ind[, c(8:10)])
hist(bestmod1.test$V1)
```




```
table(round(bestmod1.test$V1, 0), bestmod1.test$variety1)
```

```
##
##      0  1
##   0 38  0
##   1  2 20
```

```
table(round(bestmod1.test$V2, 0), bestmod1.test$variety2)
```

```
##
##      0  1
##   0 40  0
##   1  0 20
```

```
table(round(bestmod1.test$V3, 0), bestmod1.test$variety3)
```

```
##
```

```
##      0  1
##   0 40  1
##   1  0 19
```

```
### compute overall accuracy
#### first, just check to see if any cases get more than one range predicted
bestmod1.test$test.sum <- bestmod1.test$variety1 + bestmod1.test$variety2 + bestmod1.test$variety3
table(bestmod1.test$test.sum)
```

```
##
##  1
## 60
```

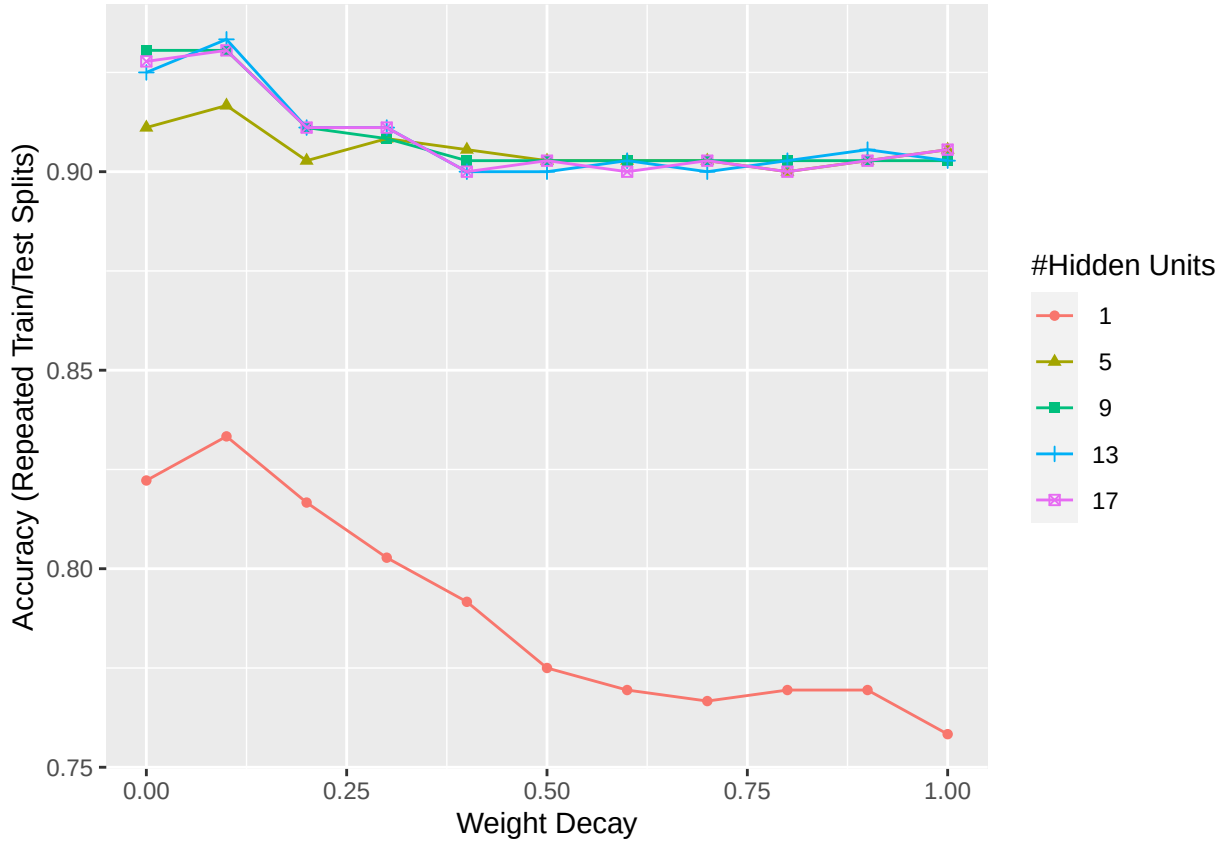
```
#### second, create categorical version
bestmod1.test$pred_variety[round(bestmod1.test$V1, 0) == 1] <- 1
bestmod1.test$pred_variety[round(bestmod1.test$V2, 0) == 1] <- 2
bestmod1.test$pred_variety[round(bestmod1.test$V3, 0) == 1] <- 3
#### add the original variables
bestmod1.test$obs_variety <- seeds_test$variety
#### now look at accuracy using confusionMatrix function from caret package
xtab <- table(bestmod1.test$pred_variety,bestmod1.test$obs_variety)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##
##      1  2  3
##   1 20  0  1
##   2  0 20  0
##   3  0  0 19
##
## Overall Statistics
##
##                Accuracy : 0.9833
##                  95% CI : (0.9106, 0.9996)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.975
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   1.0000   0.9500
## Specificity            0.9750   1.0000   1.0000
## Pos Pred Value         0.9524   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9756
## Prevalence             0.3333   0.3333   0.3333
## Detection Rate         0.3333   0.3333   0.3167
## Detection Prevalence   0.3500   0.3333   0.3167
## Balanced Accuracy      0.9875   1.0000   0.9750
```

```r
## b)
### now use caret train-control to identify best size and weight decay; I will search a grid of possibl
### need to choose a limited number of options for the grid
### here I have 5 options for size and 11 options for weight decay, which 5x11=55 different options to
#without much loss. Very little change occured after 50 iterations
#(the function outputs the value after every 10 iterations, so you can see therate of convergence.)
#Using k-fold (k=10) cross-validation to pick best size and weight decay.
nnetTunegrid <- expand.grid(.size = seq(1,20,4),
                            .decay = seq(0,1,0.1))
cv_count <- 10
### define cross-validation experiment
numFolds = trainControl(method = "LGOCV", #Leave-group out cross-validation
                        number = cv_count)
### format the output layer as a factor with levels 1-3
train.Variety <- factor(seeds_train$variety, levels = c(1, 2, 3))
train.Variety
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3
## Levels: 1 2 3
```

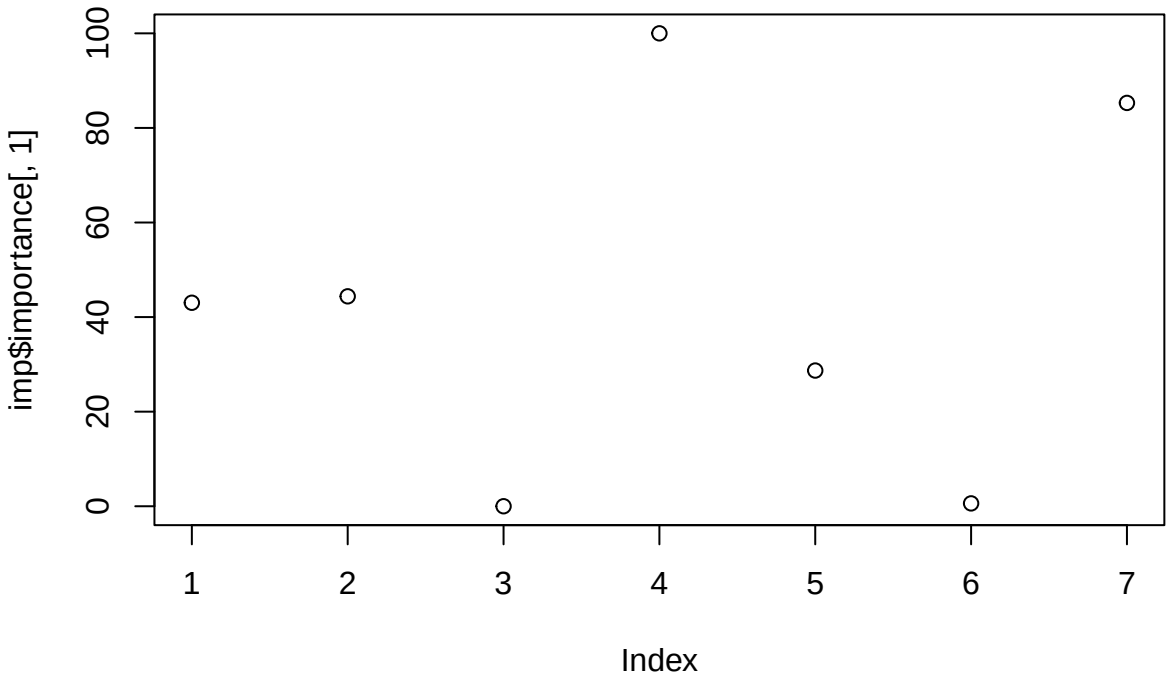
```r
nnetFit <- train(x = seeds_train_ind[, c(1:7)], y = train.Variety,
                 method = "nnet",
                 trControl = numFolds,
                 tuneGrid = nnetTunegrid,
                 maxit = 500, #max iterations for nnet only, set lower (50) to reduce computation
                 metric = "Accuracy",
                 trace = FALSE)
### reduce maxit to 60
nnetFit <- train(x = seeds_train_ind[, c(1:7)], y = train.Variety,
                 method = "nnet",
                 trControl = numFolds,
                 tuneGrid = nnetTunegrid,
                 maxit = 60, #max iterations for nnet only, set lower (50) to reduce computation
                 metric = "Accuracy",
                 trace = FALSE)
ggplot(nnetFit)
```




```
### return the maximum accuracy then maximum kappa
acc <- nnetFit$results[which.max(nnetFit$results$Accuracy),]
## finally, a method for looking at variable importance.
## varimp function (from the caret package, requires nnet object)
imp <- varImp(nnetFit)
imp
```

```
## nnet variable importance
##
##   variables are sorted by maximum importance across the classes
##              Overall        1        2        3
## length      100.0000 100.0000 100.0000 100.0000
## groove       85.2875  85.2875  85.2875  85.2875
## perimeter    44.3887  44.3887  44.3887  44.3887
## area         43.0320  43.0320  43.0320  43.0320
## width        28.6943  28.6943  28.6943  28.6943
## asymmetry     0.6035   0.6035   0.6035   0.6035
## compactness   0.0000   0.0000   0.0000   0.0000
```

```
plot(imp$importance[, 1])
```




```
## c)
## use ordered logistic regression: predicts probabilities for an ordinal scale, to see how close the r
seeds_train$variety <- factor(seeds_train$variety, levels = c("1", "2", "3"), ordered = TRUE)
o.logit <- polr(variety ~ .,data = seeds_train)
summary(o.logit)
```

```
## Call:
## polr(formula = variety ~ ., data = seeds_train)
##
## Coefficients:
##                  Value Std. Error t value
## area           11.4848     1.3268   8.656
## perimeter     -25.4165     2.3980 -10.599
## compactness  -266.2269    13.9165 -19.130
## length        -15.7015     3.8005  -4.131
## width           6.3742     5.8554   1.089
## asymmetry       0.6181     0.1874   3.298
## groove         14.0214     2.2251   6.301
##
## Intercepts:
##     Value     Std. Error t value
## 1|2 -423.2765    8.9478   -47.3053
## 2|3 -418.9378    8.9617   -46.7477
##
## Residual Deviance: 133.458
## AIC: 151.458
```

```r
# Evaluate Predictions
## b)
#Now make predictions on the test data (our holdout sample)
#Predict the test and examine accuracy
test.Variety <- as.numeric(predict(nnetFit, seeds_test_ind[, c(1:7)]))
cv_xtab<-table(test.Variety, seeds_test$variety)
confusionMatrix(cv_xtab)
```

```
## Confusion Matrix and Statistics
##
##
## test.Variety  1  2  3
##            1 19  0  1
##            2  0 20  0
##            3  1  0 19
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95% CI : (0.8847, 0.9959)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9500   1.0000   0.9500
## Specificity            0.9750   1.0000   0.9750
## Pos Pred Value         0.9500   1.0000   0.9500
## Neg Pred Value         0.9750   1.0000   0.9750
## Prevalence             0.3333   0.3333   0.3333
## Detection Rate         0.3167   0.3333   0.3167
## Detection Prevalence   0.3333   0.3333   0.3333
## Balanced Accuracy      0.9625   1.0000   0.9625
```

```r
## c) ordered logistic regression
seeds_test$olog_pred <- predict(o.logit, newdata = seeds_test)
xtab2 <- table(seeds_test$variety, seeds_test$olog_pred)
xtab2
```

```
##
##      1  2  3
##   1 19  1  0
##   2  0 20  0
##   3  0  2 18
```