R Notebook

Introduction

Unmanned Aerial Vehicles (UAVs) play an important role in agricultural research because they facilitate high-throughput phenotyping (HTP).The ability to identify cotton plant height and boll count across a field can serve as an important tool in predicting plant growth and yield. In order to capture a three-dimensional (3D) view of field plots, which is believed to be helpful in estimating yield and crop development parameters, sensors mounted on UAVs must have access to a view of the ground. However, cotton planted in solid rows can obscure this view. Canopy closure prevents sensors from measuring plant architecture and boll-loads three dimensionally from the midgrowing season until the crop is defoliated. Therefore, this project was initiated to compare solid vs. skip-row planting patterns in terms of predicting yield and fiber quality since skip rows would allow UAV sensors to capture more accurate 3D data from plots. The purposes of this project were to (1) compare the accuracy of UAV-derived data from different row patterns (2) evaluate the ability of UAVs to predict plant yield and (3) characterize genotype x row pattern interaction and how location and year affect that interaction.

Objective 1: Accuracy between different row patterns

The height measured by UAV and human was compared.

```r
df <- read.csv("/Users/wenzhuowu/Desktop/tamu-project/height1.csv", header=TRUE)
head(df)
```

```
##        ID    UAV_h Manual_h row_pattern
## 1 UAV-101 23.08923     15.2       solid
## 2 UAV-102 21.51380     15.2       solid
## 3 UAV-103 22.51454     15.2       solid
## 4 UAV-104 19.85056     15.6       solid
## 5 UAV-105 19.08506     15.8       solid
## 6 UAV-106 16.34745     14.8        skip
```

```r
lm1 = lm(UAV_h~Manual_h, data = df[df$row_pattern=='solid',]) #Create the linear regression
summary(lm1)
```

```
##
## Call:
## lm(formula = UAV_h ~ Manual_h, data = df[df$row_pattern == "solid",
##     ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.4786  -6.1179  -0.4783   5.0431  22.3615
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  5.09666    2.41063   2.114   0.0361 *
## Manual_h     1.00389    0.04343  23.116   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.942 on 158 degrees of freedom
## Multiple R-squared:  0.7718, Adjusted R-squared:  0.7704
## F-statistic: 534.4 on 1 and 158 DF,  p-value: < 2.2e-16
```

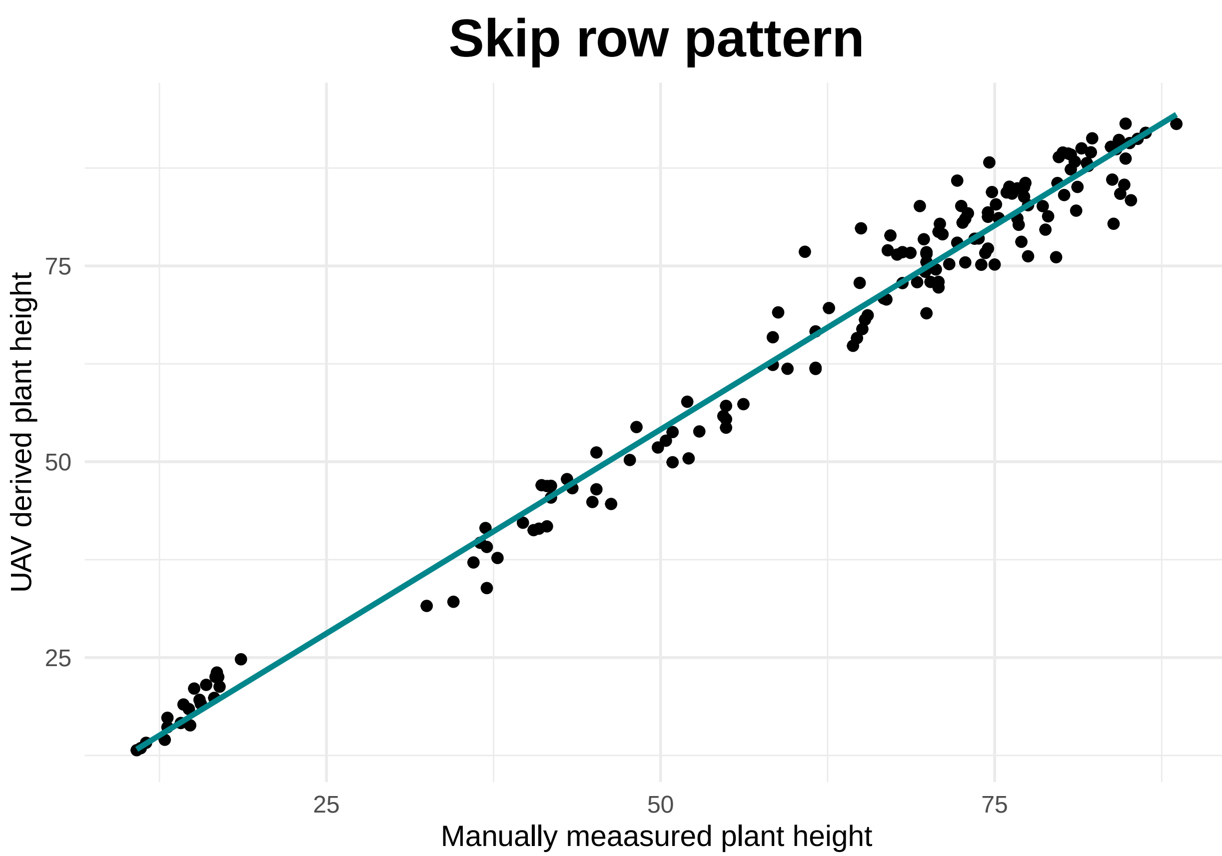
```
library(ggplot2)


ggplot(df[df$row_pattern=='solid',],aes(Manual_h, UAV_h)) +
  geom_point() +
  geom_smooth(method='lm', se=FALSE, color='turquoise4') +
  theme_minimal() +
  labs(x='Manually meaasured plant height', y='UAV derived plant height', title='Solid row pattern') +
  theme(plot.title = element_text(hjust=0.5, size=20, face='bold'))
```


```
lm2 = lm(UAV_h~Manual_h, data = df[df$row_pattern=='skip',]) #Create the linear regression
summary(lm2)
```

```
##
## Call:
## lm(formula = UAV_h ~ Manual_h, data = df[df$row_pattern == "skip",
```

```
##     ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0509 -2.4258 -0.0081  2.6376 11.4343
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.04025    0.81995   2.488   0.0139 *
## Manual_h     1.04166    0.01281  81.300   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.557 on 158 degrees of freedom
## Multiple R-squared:  0.9767, Adjusted R-squared:  0.9765
## F-statistic:  6610 on 1 and 158 DF,  p-value: < 2.2e-16
```

```
ggplot(df[df$row_pattern=='skip',],aes(Manual_h, UAV_h)) +
  geom_point() +
  geom_smooth(method='lm', se=FALSE, color='turquoise4') +
  theme_minimal() +
  labs(x='Manually meaasured plant height', y='UAV derived plant height', title='Skip row pattern') +
  theme(plot.title = element_text(hjust=0.5, size=20, face='bold'))
```



The skip-row planting pattern provided a more accurate plant height (R2 0.97) with lower levels of error (RMSE 3.557) compared to data collected from the solid-row pattern (R2 0.77; RMSE 8.942). Row pattern may have an impact on the accuracy of plant height model based on UAV images.

Objective 2: Yield prediction

Plant height (ph), canopy colume (cv), canopy cover (cc), vegetation index NDVI, ExG were generated from UAV images across multiple dates. Boll count and boll area were processed based on the images taken the day before harvest.

```
library(plyr)
library(readr)
library(dplyr)
library(glmnet)
library(ggplot2)
library(tidyverse)
library(caret)
library(leaps)
library(MASS)
```

```
dat <- read.csv("/Users/wenzhuowu/Desktop/tamu-project/timeline.csv", header=TRUE)
dat = na.omit(dat)
head(dat)
```

```
##   Yield.per.row    ph424    ph514    ph523    ph601    ph606    ph613    ph703
## 1     0.7238045 23.08923 54.43071 64.79951 85.08744 91.08666 88.13228 93.16466
## 2     0.3695207 21.51380 46.89888 55.43674 70.71673 76.67001 72.95586 76.75417
## 4     1.2001809 19.85056 50.22382 61.99820 78.50110 84.90621 81.10504 85.58314
## 5     0.7492547 19.08506 37.71485 44.60903 62.36279 69.06843 65.89498 76.80755
## 6     1.9610077 16.34745 33.87159 44.86051 68.70007 79.80380 76.99425 84.23135
## 7     3.5050858 16.64369 41.53833 49.93660 70.85788 79.35865 76.11390 85.59826
##      ph710     CV0423     CV0514    CV0523    CV0601    CV0606    CV0613
## 1 93.13207 0.05260687 0.10427062 0.1506674 0.2135402 0.2493004 0.2199320
## 2 79.63202 0.03991368 0.08894159 0.1169693 0.1546981 0.1876132 0.1550796
## 4 90.20983 0.05190211 0.11518645 0.1530012 0.1871325 0.2056447 0.1674668
## 5 80.37630 0.05275008 0.11075749 0.1477807 0.1879758 0.1953552 0.1661987
## 6 82.85676 0.04507159 0.09966807 0.1750264 0.3154559 0.4023677 0.3920168
## 7 83.38367 0.04816094 0.08630561 0.1701969 0.2768518 0.3715808 0.3433249
##      CV0703    CV0709     CV0719     CV0730  NDVI0423  NDVI0507  NDVI0514
## 1 0.3768582 0.3786870 0.07653988 0.07887313 0.6414540 0.7310976 0.7623606
## 2 0.2875302 0.2777700 0.05695665 0.06087028 0.6358705 0.7211970 0.7511253
## 4 0.2609449 0.2894993 0.05601709 0.06626179 0.6441107 0.7301496 0.7707717
## 5 0.2943320 0.3203904 0.03871509 0.06586310 0.6307880 0.7184223 0.7598661
## 6 0.4941357 0.4734154 0.12047814 0.15051981 0.6322879 0.7236893 0.7476041
## 7 0.4713480 0.4951644 0.08770911 0.12912089 0.6066887 0.7267454 0.7460610
##    NDVI0523  NDVI0601  NDVI0606  NDVI0613  NDVI0627  NDVI0703  NDVI0709
## 1 0.8141249 0.7782031 0.7955470 0.7708026 0.7638851 0.7683153 0.8042386
## 2 0.8157233 0.7561875 0.7837406 0.7498242 0.7598329 0.7469738 0.8138712
## 4 0.7962466 0.7445925 0.7486963 0.7340292 0.7252100 0.7187384 0.7620830
## 5 0.8073528 0.7381456 0.7380863 0.7344738 0.7474213 0.7449222 0.7946994
## 6 0.8168395 0.7948603 0.8116707 0.8154895 0.8097940 0.7770749 0.8437351
## 7 0.8138227 0.7774627 0.7859246 0.7855760 0.7854951 0.7608642 0.8126085
##    NDVI0719  NDVI0730 CC.NDVI.0423   CC0507   CC0514   CC0523   CC0601   CC0606
## 1 0.6505737 0.5471646     6.065595 19.03157 29.55436 45.79238 48.30467 54.52901
## 2 0.6465960 0.5423009     4.664994 18.87191 27.95609 43.70842 41.58477 45.70461
```

```
## 4 0.6230497 0.5557628     5.477416 23.90442 33.34939 49.21342 42.82393 46.95360
## 5 0.6023094 0.5523218     4.869186 23.37434 35.39653 49.39923 45.50440 47.77592
## 6 0.6586369 0.5554299     4.819057 20.55154 30.20386 50.69988 66.46900 72.52304
## 7 0.6344443 0.5855763     3.819821 17.85646 29.50675 48.21152 57.76120 68.03908
##     CC0613   CC0627   CC0703   CC0709   CC0719     CC0730   ExG0411   ExG0423
## 1 54.47410 65.35066 62.17071 71.96578 46.28016 0.26647220 0.2715594 0.3162469
## 2 44.75978 53.13127 52.31034 60.14174 38.40713 0.06205517 0.2989872 0.2787740
## 4 47.03699 50.96873 43.79587 54.48790 32.82667 0.54754561 0.2724144 0.3004518
## 5 47.27690 58.46413 53.95853 64.08558 28.42791 0.27012250 0.2497029 0.2883167
## 6 75.18226 80.89994 77.49386 83.59526 53.23616 0.90162510 0.2939939 0.2958657
## 7 67.96601 75.67626 75.21950 88.25004 45.51970 3.93137745 0.2964041 0.3074027
##     ExG0507   ExG0514   ExG0523   ExG0601   ExG0606   ExG0613   ExG0703
## 1 0.3794446 0.3489358 0.4617222 0.4282240 0.4041160 0.3574833 0.3684215
## 2 0.3863634 0.3753353 0.4727997 0.3848624 0.3802228 0.3339426 0.3620276
## 4 0.3731964 0.3413482 0.4705526 0.3390341 0.3894267 0.3535366 0.3241440
## 5 0.3840795 0.3402274 0.4568269 0.3504812 0.3854322 0.3642890 0.3567482
## 6 0.3731814 0.3226712 0.4764233 0.4788008 0.4572300 0.4053744 0.4126311
## 7 0.3783647 0.3435644 0.5141149 0.4576412 0.4519874 0.3959878 0.3621281
##     ExG0709   ExG0719   ExG0730 CC0411rgb CC0423rgb CC0507rgb CC0514rgb
## 1 0.3750021 0.3069519 0.2716855  7.105997  8.098282  19.45727  31.93978
## 2 0.3779314 0.3150844 0.2597605  3.674020  5.719051  19.38776  30.43647
## 4 0.3753669 0.3561283 0.2878235  2.691100  7.379075  25.86747  35.99836
## 5 0.3995913 0.3183836 0.2784688  1.656996  7.136868  25.06365  38.02478
## 6 0.3982683 0.3343330 0.2911888  3.653199  6.619501  21.45777  32.61876
## 7 0.3287007 0.3262396 0.3127461  5.894915  6.462973  18.73579  32.30318
##   CC0523rgb CC0601rgb CC0606rgb CC0613rgb CC0703rgb CC0709rgb CC0719rgb
## 1  45.55843  50.28874  49.31391  46.68310  56.89838  67.74638 12.310823
## 2  42.19738  42.18886  39.75439  38.48176  45.72001  55.52296 12.859953
## 4  44.20931  42.97167  42.15443  39.00647  33.77219  48.47658  9.580042
## 5  46.27537  45.18788  41.70990  38.49236  47.84284  60.01452  5.550796
## 6  50.44731  65.58922  69.72807  72.24327  78.97412  81.57193 22.810658
## 7  47.16593  60.41281  61.32903  63.09639  75.08079  83.53423 17.691159
##   CC0730rgb Manual.cotton UAV.cotton      area
## 1 1.4579069      442.5696        420 0.1455842
## 2 0.6977728      492.8616        359 0.1105856
## 4 2.5593429      311.8104        372 0.1650233
## 5 1.5316449      221.2848        398 0.2769855
## 6 4.1765240      422.4528        770 0.6225498
## 7 7.6034479      633.6792        713 0.3705098
```

Data partition

```r
set.seed(100)

index = sample(1:nrow(dat), 0.8*nrow(dat))

train = dat[index,] # Create the training data
test = dat[-index,] # Create the test data
```

Scaling the Numeric Features

```r
cols = colnames(dat)[-1]
pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))
```

```r
train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])
```

stepwise regression

```r
# Set seed for reproducibility
set.seed(123)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Train the model
step.model <- train(Yield.per.row ~., data = dat,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:34),
                    trControl = train.control
                    )
step.model$results
```

```
##    nvmax      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1      1  1.400055 0.3992414  1.064650  0.8376179  0.3492695  0.6223935
## 2      2  1.385521 0.4870007  1.080098  0.8495000  0.3629914  0.6267542
## 3      3  1.436567 0.4663273  1.151763  0.8626869  0.3751979  0.6411067
## 4      4  1.569099 0.4844410  1.243484  0.7925774  0.3119199  0.6152801
## 5      5  1.514464 0.4792848  1.211049  0.7992111  0.3666704  0.6833244
## 6      6  1.570626 0.5160313  1.280873  0.7727055  0.4021671  0.7221285
## 7      7  1.775937 0.4925244  1.419441  0.8865010  0.3815736  0.8434231
## 8      8  1.814445 0.4800011  1.483891  0.8076021  0.3886539  0.7492466
## 9      9  1.800514 0.4663535  1.473391  0.8456117  0.3680168  0.7362490
## 10    10  1.834484 0.4939171  1.434662  0.8973196  0.3665860  0.7425752
## 11    11  1.827496 0.4815026  1.475615  0.8918584  0.3554782  0.6944694
## 12    12  1.820905 0.5307846  1.443824  0.8795850  0.3181045  0.7137283
## 13    13  1.899297 0.5670210  1.531713  0.9287437  0.3473361  0.7385946
## 14    14  2.019330 0.5926622  1.600535  0.9059015  0.3544228  0.7508373
## 15    15  2.204386 0.5895140  1.836050  0.7647980  0.4029096  0.7196949
## 16    16  2.416704 0.5864945  2.006644  0.6477741  0.4240437  0.6749178
## 17    17  2.671186 0.4788828  2.206063  0.6241767  0.4261751  0.6572546
## 18    18  2.849763 0.4943356  2.345923  0.8228487  0.4273665  0.8101966
## 19    19  3.064339 0.4509716  2.581836  1.0066777  0.4389864  1.0065014
## 20    20  3.460947 0.4786664  2.891519  0.9498468  0.3891927  0.9196867
## 21    21  3.968172 0.4230773  3.389232  1.1191569  0.3506543  0.9814143
## 22    22  4.612795 0.4157754  3.938068  1.4205311  0.3506590  1.1980659
## 23    23  5.290047 0.4073425  4.466292  1.9570552  0.3184884  1.7966926
## 24    24  5.910487 0.3600260  4.963332  2.2133064  0.2936551  1.8225266
## 25    25  6.879233 0.3396474  5.775882  3.0098815  0.2933838  2.5223822
## 26    26  7.598395 0.4081613  6.369823  3.1649848  0.3548144  2.4059061
## 27    27  9.711757 0.4073811  8.136517  5.9563941  0.3533064  4.9917642
## 28    28 10.448122 0.4086845  8.556040  6.5439973  0.3514584  5.1555504
## 29    29 10.741302 0.4032084  8.828850  7.4269770  0.3578430  6.0456305
## 30    30 12.474223 0.4002315 10.399944 11.2733660  0.3566163  9.7817758
## 31    31 13.164643 0.3877000 11.093974 12.9212478  0.3507525 11.6892620
## 32    32 14.099071 0.3891766 11.936260 12.6103804  0.3390911 11.1223752
## 33    33 16.339748 0.3820339 13.555980 15.5271180  0.3064891 12.7502540
## 34    34 16.301811 0.4026950 13.521089 15.5510718  0.2991455 12.7678521
```

```r
step.model$bestTune
```

```
##   nvmax
## 2     2
```

```r
summary(step.model$finalModel)
```

```r
m1 = lm(Yield.per.row ~ CV0730+ NDVI0730,
   data = dat)
summary (m1)
```

```
##
## Call:
## lm(formula = Yield.per.row ~ CV0730 + NDVI0730, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8961 -0.6319 -0.1257  0.5315  3.5594
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -36.773      9.088  -4.046 0.000273 ***
## CV0730        12.074      6.737   1.792 0.081773 .
## NDVI0730      67.037     16.719   4.010 0.000304 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.349 on 35 degrees of freedom
## Multiple R-squared:  0.5386, Adjusted R-squared:  0.5123
## F-statistic: 20.43 on 2 and 35 DF,  p-value: 1.321e-06
```

For stepwise regrssion, the RSE is 1.349 and R2 is 53.89 percent.

ridge

```r
cols_reg = colnames(dat)
dummies <- dummyVars(Yield.per.row ~ ., data = dat[,cols_reg])
train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])
```

```r
x = as.matrix(train_dummies)
y_train = train$Yield.per.row

x_test = as.matrix(test_dummies)
y_test = test$Yield.per.row

lambdas <- 10^seq(2, -3, by = -.1)
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0, family = 'gaussian', lambda = lambdas)
summary(ridge_reg)
```

```
##           Length Class     Mode
```

```
## a0          51   -none-    numeric
## beta      3519   dgCMatrix S4
## df          51   -none-    numeric
## dim          2   -none-    numeric
## lambda      51   -none-    numeric
## dev.ratio   51   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         7   -none-    call
## nobs         1   -none-    numeric
```

```
cv_ridge <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas)
optimal_lambda <- cv_ridge$lambda.min
optimal_lambda
```

```
## [1] 0.02511886
```

The optimal lambda value comes out to be 0.01 and will be used to build the ridge regression model. We also create a function for calculating and printing the results, which is done with the eval_results() function in the code below. The next step is to use the predict function to generate predictions on the train and test data. Finally, we use the eval_results function to calculate and print the evaluation metrics.

```
# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)

}

# Prediction and evaluation on train data
predictions_train <- predict(ridge_reg, s = optimal_lambda, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##         RMSE  Rsquare
## 1 0.04156802 0.999493
```

```
# Prediction and evaluation on test data
predictions_test <- predict(ridge_reg, s = optimal_lambda, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##       RMSE   Rsquare
## 1 0.593154 0.9196201
```

The above output shows that the RMSE and R-squared values for the ridge regression model on the training data are 0.0240 and 99.98 percent, respectively. For the test data, the results for these metrics are 0.6105 and 91.48 percent, respectively.

Lasso

```
lambdas <- 10^seq(2, -3, by = -.1)

# Setting alpha = 1 implements lasso regression
lasso_reg <- cv.glmnet(x, y_train, alpha = 1, lambda = lambdas, standardize = TRUE, nfolds = 5)

# Best
lambda_best <- lasso_reg$lambda.min
lambda_best
```

```
## [1] 0.001
```

The optimal lambda value is 0.001, we train the lasso model in the first line of code below. The second through fifth lines of code generate the predictions and print the evaluation metrics for both the training and test datasets.

```
lasso_model <- glmnet(x, y_train, alpha = 1, lambda = lambda_best, standardize = TRUE)

predictions_train <- predict(lasso_model, s = lambda_best, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##          RMSE   Rsquare
## 1 0.002541985 0.9999981
```

```
predictions_test <- predict(lasso_model, s = lambda_best, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##          RMSE   Rsquare
## 1 0.006445168 0.9999905
```

The above output shows that the RMSE and R-squared values on the training data are 0.0025 and 99.99 percent, respectively. The results on the test data are 0.0064 and 99.99 percent, respectively. Lasso regression can also be used for feature selection because the coefficients of less important features are reduced to zero.

```
# Set training control
train_cont <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              search = "random",
                              verboseIter = TRUE)

# Train the model
elastic_reg <- train(Yield.per.row ~ .,
                           data = train,
                           method = "glmnet",
                           preProcess = c("center", "scale"),
                           tuneLength = 10,
                           trControl = train_cont)
```

```r
# Best tuning parameter
elastic_reg$bestTune
```

```
##        alpha    lambda
## 10 0.7689969 0.0089959
```

After we have trained the model, the optimal alpha is 0.86 and lambda is 0.0021.

```r
# Make predictions on training set
predictions_train <- predict(elastic_reg, x)
eval_results(y_train, predictions_train, train)
```

```
##         RMSE   Rsquare
## 1 0.05803351 0.9990117
```

```r
# Make predictions on test set
predictions_test <- predict(elastic_reg, x_test)
eval_results(y_test, predictions_test, test)
```

```
##         RMSE   Rsquare
## 1 0.08225816 0.9984541
```

The above output shows that the RMSE and R-squared values for the elastic net regression model on the training data are 0.0563 and 99.99 percent, respectively. The results for these metrics on the test data are 0.0694 and 99.89 percent, respectively.

Objective 3: The influence of row pattern on yield ranking of 5 varieties

```r
library(tidyverse)
data1 <- read.csv("/Users/wenzhuowu/Desktop/tamu-project/2-year-3-location-split.csv", header=TRUE)
str(data1)
```

```
## 'data.frame':    240 obs. of  15 variables:
##  $ Year       : int  2018 2018 2018 2018 2018 2018 2018 2018 2018 2018 ...
##  $ Loc        : Factor w/ 3 levels "CollSt","CorpCh",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Variety    : Factor w/ 5 levels "Gladdis","T08",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ Row.pattern: Factor w/ 2 levels "skip","solid": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Rep        : int  1 2 3 4 1 2 3 4 1 2 ...
##  $ Env        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Lint_Pct   : Factor w/ 158 levels "",".","26.35869565",..: 157 151 129 155 94 120 48 110 125 150
##  $ Lint_Ac    : Factor w/ 223 levels "",".","1003",..: 142 136 2 128 113 1 115 89 151 122 ...
##  $ Mic        : Factor w/ 105 levels ".","3.1","3.21",..: 99 88 70 95 74 88 62 74 102 104 ...
##  $ Length     : Factor w/ 44 levels ".","0.96","0.97",..: 7 8 16 9 25 23 36 28 12 6 ...
##  $ Unif       : Factor w/ 71 levels ".","79.8","80.4",..: 26 12 15 9 43 35 57 51 40 16 ...
##  $ Strength   : Factor w/ 95 levels ".","26.9","27.1",..: 11 6 23 4 58 63 75 74 43 16 ...
##  $ Elongation : Factor w/ 40 levels ".","3.1","3.3",..: 28 22 18 20 27 16 23 24 27 21 ...
##  $ X          : logi  NA NA NA NA NA NA ...
##  $ X.1        : logi  NA NA NA NA NA NA ...
```

This considers a ficticious series of yield trials. There are 2 treatment factors:

-Variety with 5 different genotype with levels Gladdis T08 Tamcot73 WK11L X263 and -Row.pattern with levels skip and solid.

The trials were conducted at 3 locations (Loc with levels Weslaco, CollSt and CorpCh). Moreover, the these trials were repeated across 2 years (Year with levels 2018 and 2017) . Thus, there are 3 trials with repeated measures across 2 years, respectively. Similar experimental designs (with different randomizations) were used at each location and in each year.

Before anything, the columns Year, Rep should be encoded as factors, since R by default encoded them as integer. Also lint_Ac, Mic, Length, Unif, strength and wlongation should be encoded as integer. Lastly remove the last two columns.

```
data1 <- data1 %>%
  mutate_at(vars(Year, Rep, Env), as.factor)

data1 <- data1 %>%
  mutate_at(vars(Lint_Pct:Elongation), as.integer)

data1 <- subset (data1, select = -c(X:X.1 ))

head(data1)
```

```
##   Year    Loc Variety Row.pattern Rep Env Lint_Pct Lint_Ac Mic Length Unif
## 1 2018 CorpCh Gladdis        skip   1   1      157     142  99      7   26
## 2 2018 CorpCh Gladdis        skip   2   1      151     136  88      8   12
## 3 2018 CorpCh Gladdis        skip   3   1      129       2  70     16   15
## 4 2018 CorpCh Gladdis        skip   4   1      155     128  95      9    9
## 5 2018 CorpCh     T08        skip   1   1       94     113  74     25   43
## 6 2018 CorpCh     T08        skip   2   1      120       1  88     23   35
##   Strength Elongation
## 1       11         28
## 2        6         22
## 3       23         18
## 4        4         20
## 5       58         27
## 6       63         16
```

We grouped the locations and years and classified them as 6 different environments For the first environment:

```
library(agricolae)
data <- data1[data1$Env==1,]
attach(data)
model <- sp.plot(
                 block = Rep,
                 pplot = Variety,
                 splot = Row.pattern,
                 Y = Lint_Ac)
```

```
##
## ANALYSIS SPLIT PLOT:  Lint_Ac
## Class level information
##
```

```
## Variety  :  Gladdis T08 Tamcot73 WK11L X263
## Row.pattern  :  skip solid
## Rep  :  1 2 3 4
##
## Number of observations:  40
##
## Analysis of Variance Table
##
## Response: Lint_Ac
##                     Df  Sum Sq Mean Sq F value Pr(>F)
## Rep                  3   308.6  102.87
## Variety              4 10598.1 2649.53  1.3079 0.3218
## Ea                  12 24308.9 2025.74
## Row.pattern          1  2433.6 2433.60  1.7634 0.2041
## Variety:Row.pattern  4  7182.9 1795.73  1.3012 0.3140
## Eb                  15 20701.5 1380.10
##
## cv(a) = 46.1 %, cv(b) = 38.1 %, Mean = 97.6
```

```r
data <- data1[data1$Env==2,]
attach(data)
model <- sp.plot(
                 block = Rep,
                 pplot = Variety,
                 splot = Row.pattern,
                 Y = Lint_Ac)
```

```
##
## ANALYSIS SPLIT PLOT:  Lint_Ac
## Class level information
##
## Variety  :  Gladdis T08 Tamcot73 WK11L X263
## Row.pattern  :  skip solid
## Rep  :  1 2 3 4
##
## Number of observations:  40
##
## Analysis of Variance Table
##
## Response: Lint_Ac
##                     Df Sum Sq Mean Sq F value Pr(>F)
## Rep                  3   4564  1521.3
## Variety              4  50648 12661.9  2.2697 0.1220
## Ea                  12  66944  5578.6
## Row.pattern          1   3010  3010.2  0.5541 0.4682
## Variety:Row.pattern  4   5080  1270.0  0.2338 0.9150
## Eb                  15  81490  5432.7
##
## cv(a) = 85.7 %, cv(b) = 84.6 %, Mean = 87.175
```

```r
data <- data1[data1$Env==3,]
attach(data)
model <- sp.plot(
```

```r
                 block = Rep,
                 pplot = Variety,
                 splot = Row.pattern,
                 Y = Lint_Ac)
```

```
##
## ANALYSIS SPLIT PLOT:  Lint_Ac
## Class level information
##
## Variety  :  Gladdis T08 Tamcot73 WK11L X263
## Row.pattern  :  skip solid
## Rep  :  1 2 3 4
##
## Number of observations:  40
##
## Analysis of Variance Table
##
## Response: Lint_Ac
##                     Df Sum Sq Mean Sq F value  Pr(>F)
## Rep                  3   3754  1251.2
## Variety              4  55982 13995.5  4.0421 0.02656 *
## Ea                  12  41550  3462.5
## Row.pattern          1   9120  9120.4  2.6938 0.12153
## Variety:Row.pattern  4  30519  7629.8  2.2536 0.11187
## Eb                  15  50785  3385.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## cv(a) = 45.4 %, cv(b) = 44.9 %, Mean = 129.7
```

```r
data <- data1[data1$Env==4,]
attach(data)
model <- sp.plot(
                 block = Rep,
                 pplot = Variety,
                 splot = Row.pattern,
                 Y = Lint_Ac)
```

```
##
## ANALYSIS SPLIT PLOT:  Lint_Ac
## Class level information
##
## Variety  :  Gladdis WK11L T08 Tamcot73 X263
## Row.pattern  :  skip solid
## Rep  :  1 2 3 4
##
## Number of observations:  40
##
## Analysis of Variance Table
##
## Response: Lint_Ac
##                     Df Sum Sq Mean Sq F value    Pr(>F)
## Rep                  3   7272    2424
```

```
## Variety              4  46875   11719  4.4462 0.0195997 *
## Ea                  12  31629    2636
## Row.pattern          1  77881   77881 26.7122 0.0001145 ***
## Variety:Row.pattern  4  22477    5619  1.9273 0.1580908
## Eb                  15  43733    2916
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## cv(a) = 44.2 %, cv(b) = 46.5 %, Mean = 116.025
```

```
data <- data1[data1$Env==5,]
attach(data)
model <- sp.plot(
                 block = Rep,
                 pplot = Variety,
                 splot = Row.pattern,
                 Y = Lint_Ac)
```

```
##
## ANALYSIS SPLIT PLOT:  Lint_Ac
## Class level information
##
## Variety  :  Gladdis WK11L T08 Tamcot73 X263
## Row.pattern  :  skip solid
## Rep  :  1 2 3 4
##
## Number of observations:  40
##
## Analysis of Variance Table
##
## Response: Lint_Ac
##                     Df Sum Sq Mean Sq F value  Pr(>F)
## Rep                  3  11123  3707.8
## Variety              4  17722  4430.6  2.3506 0.11296
## Ea                  12  22619  1884.9
## Row.pattern          1   6150  6150.4  2.4071 0.14162
## Variety:Row.pattern  4  25684  6421.1  2.5131 0.08559 .
## Eb                  15  38326  2555.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## cv(a) = 33.4 %, cv(b) = 38.9 %, Mean = 130.1
```

```
data <- data1[data1$Env==6,]
attach(data)
model <- sp.plot(
                 block = Rep,
                 pplot = Variety,
                 splot = Row.pattern,
                 Y = Lint_Ac)
```

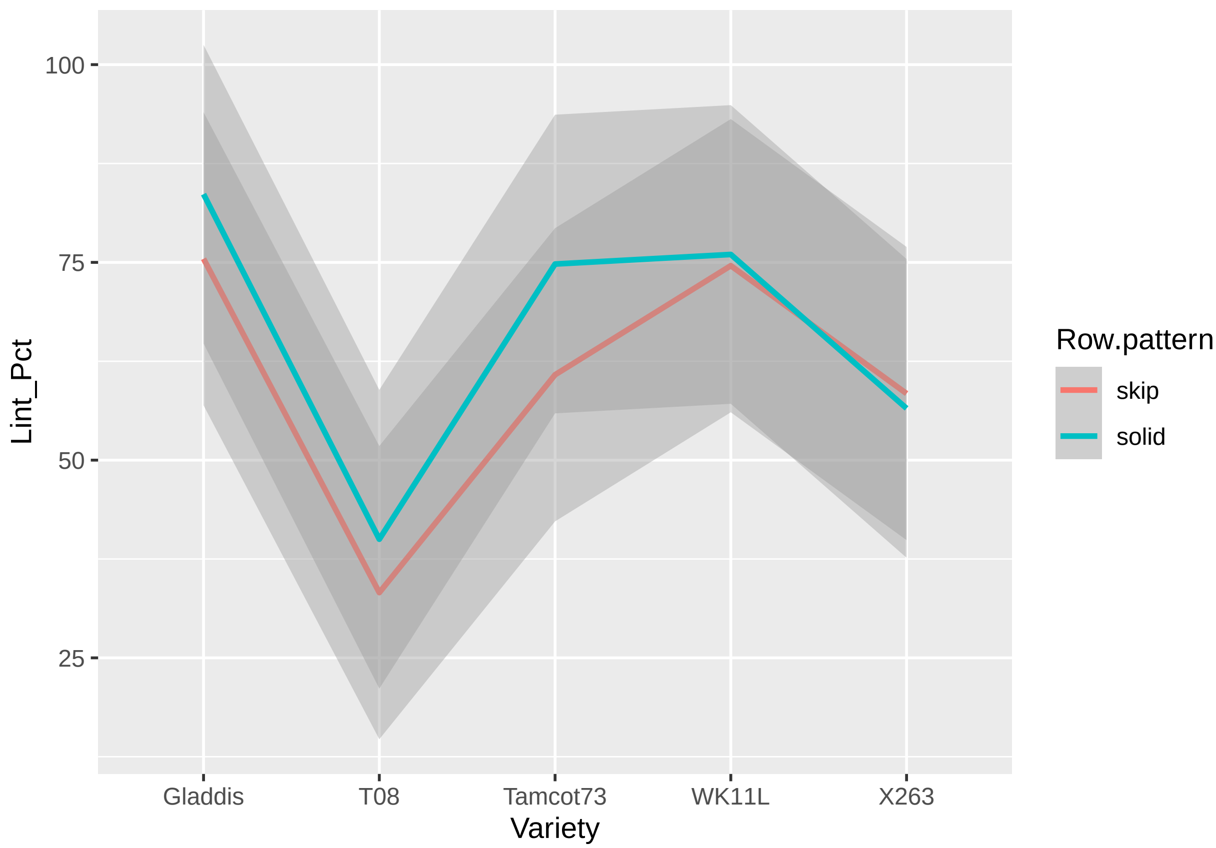
```
##
## ANALYSIS SPLIT PLOT:  Lint_Ac
```

```
## Class level information
##
## Variety  :  Gladdis WK11L T08 Tamcot73 X263
## Row.pattern  :  skip solid
## Rep  :  1 2 3 4
##
## Number of observations:  40
##
## Analysis of Variance Table
##
## Response: Lint_Ac
##                     Df Sum Sq Mean Sq F value Pr(>F)
## Rep                  3   5292  1764.0
## Variety              4   8908  2227.0  0.4600 0.7638
## Ea                  12  58100  4841.7
## Row.pattern          1   6401  6400.9  1.3940 0.2561
## Variety:Row.pattern  4  14100  3524.9  0.7676 0.5627
## Eb                  15  68879  4591.9
##
## cv(a) = 82.6 %, cv(b) = 80.4 %, Mean = 84.25
```

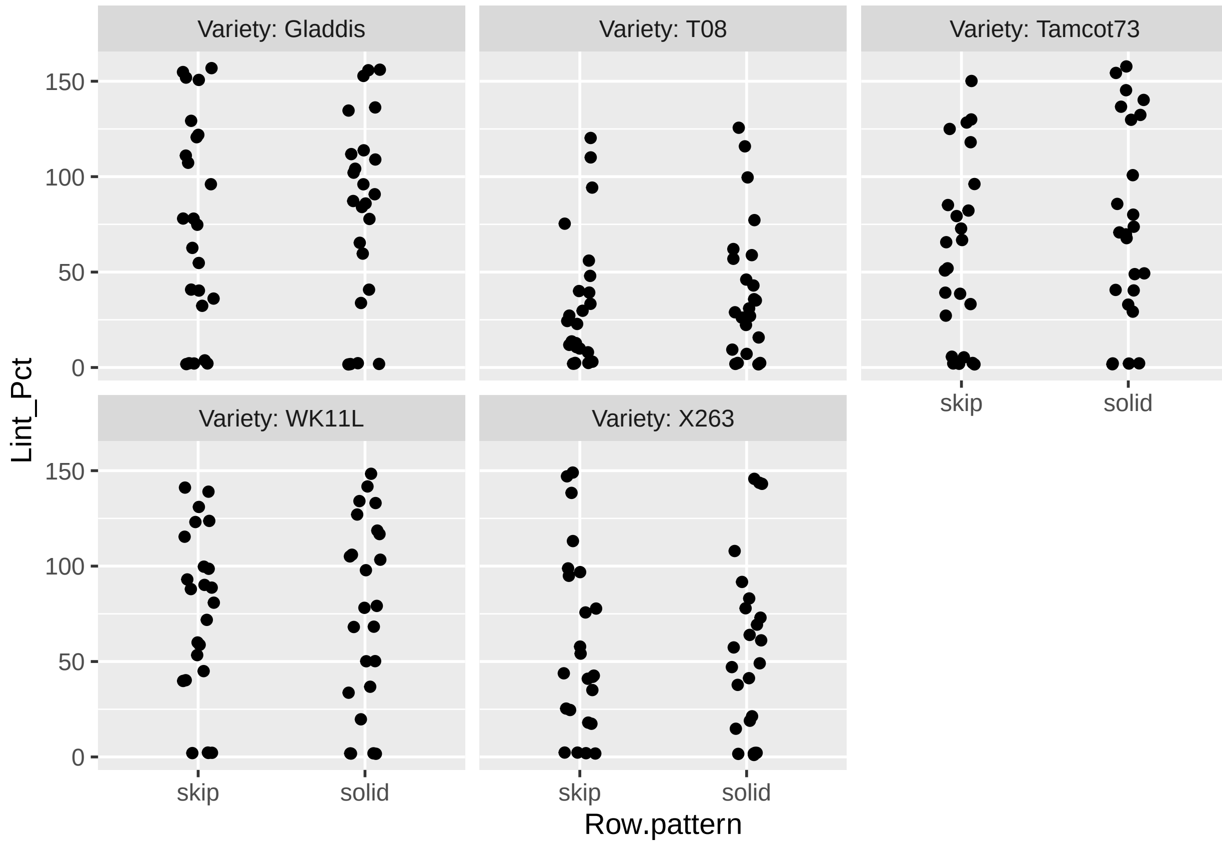
Based on the result from 6 environments, it shows there is no interaction between variety and row pattern, which means row pattern will not influence variety's yield ranking.

library(ggplot2) ggplot(aes(x = Row.pattern, y = Lint_Pct, group= Variety, colour = Variety), data = data1) + geom_line() + facet_wrap(~ Env) + theme_bw()

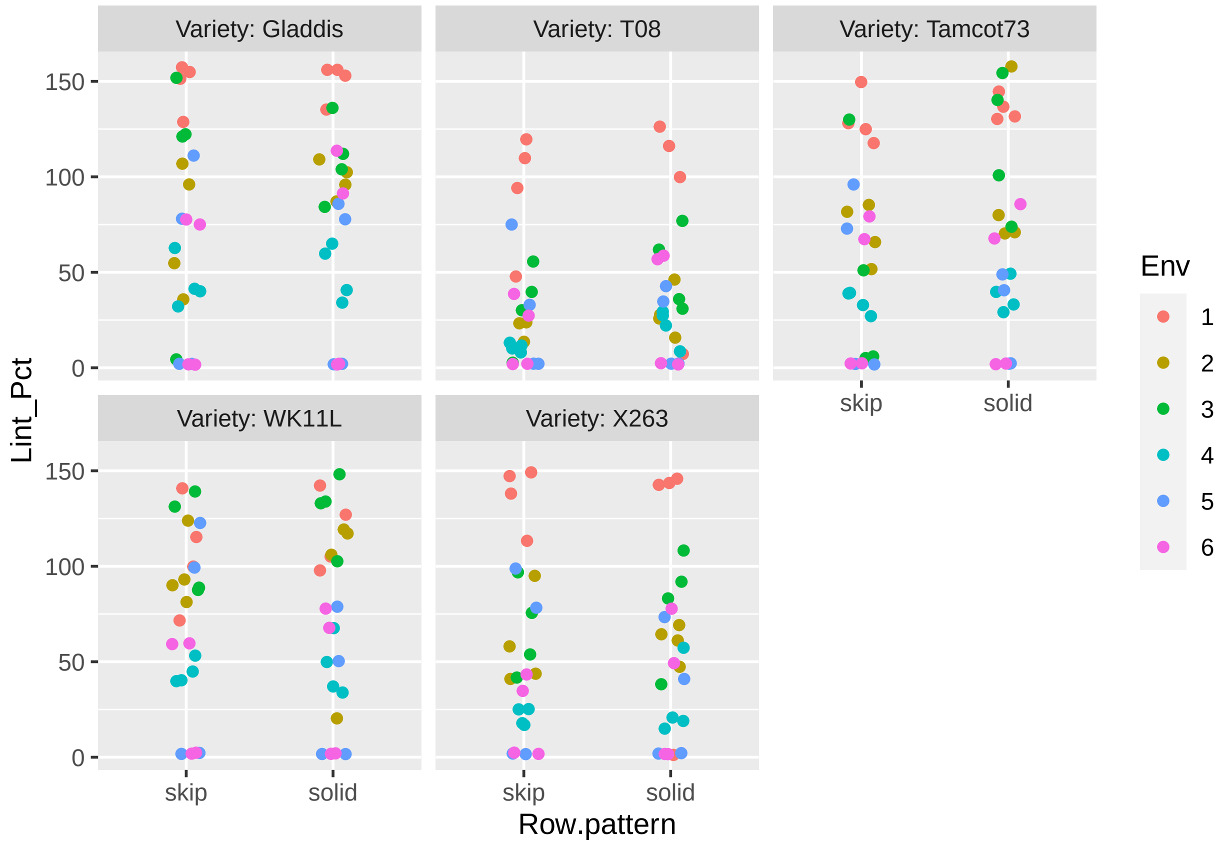
```
ggplot(data1,aes(x=Variety,y=Lint_Pct)) +
  geom_smooth(aes(group = Row.pattern,color = Row.pattern),method = 'lm',formula = 'y~factor(x)')
```




```
ggplot(data1,aes(x=Row.pattern,y=Lint_Pct)) +
  geom_jitter(width = 0.1) +
  facet_wrap(~Variety,labeller = label_both) # This line separates the plots into separate plots for ea
```




```
ggplot(data1,aes(x=Row.pattern,y=Lint_Pct)) +
  geom_jitter(aes(color = Env),width = 0.1) +
  facet_wrap(~Variety,labeller = label_both) # This line separates the plots into separate plots for ea
```



For combined analysis of split-plot design acorss locations and years, I used SAS v.9.4 (SAS v.9.4, SAS Institute, 2015). Location is fixed and year is random effect. The combined analysis can provide information about how treatmenta or combinations of treatments react to different soil types and weather etc. Firstly, we checked the homogeneity of the error variance at various locations by using Hartley's HOV Maximum F-test. Result showed cariances are homogenenous, we can proceed with the combined analysis from all locations. Here is how SAS calculated Mean sqaure for each variance.

SOV	DF	MS	F-TESTS		
			Random year loc	Random year; fixed loc	Fixed year loc
Years	y-1	M1	M1/M3	M1/M4	M1/M4
Locations	l-1	M2	M2/M3	M2/M3	M2/M4
Years x Locations	(y-1)(l-1)	M3	M3/M4	M3/M4	M3/M4
Blocks (Locations x Years)	(r -1)yl	M4			
A	a-1	M5	(M5+M8)/(M6+M7)	M5/M6	M5/M9
A x Years	(a-1)(y-1)	M6	M6/M8	M6/M9	M6/M9
A x Locations	(a -1)(l-1)	M7	M7/M8	M7/M8	M7/M9
A x Years x Locations	(a-1)(y-1)(l-1)	M8	M8/M9	M8/M9	M8/M9
Pooled error a	yl(a-1)(r-1)	M9			
B	b-1	M10	(M10+M13)/(M11+M12)	M10/M11	M10/M18
B x Years	(b-1)(y-1)	M11	M11/M13	M11/M18	M11/M18
B x Locations	(b-1)(l-1)	M12	M12/M13	M12/M13	M12/M18
B x Years x Locations	(b-1)(y-1)(l-1)	M13	M13/M16	M13/M18	M13/M18
A x B	(a-1)(b-1)	M14	(M14+M17)/(M15+M16)	M14/M15	M14/M18
A x B x Years	(a-1(b-1)(y-1)	M15	M15/M17	M15/M18	M15/M18
A x B x Locations	(a-1)(b-1)(l-1)	M16	M16/M17	M16/M17	M16/M18
A x B x Years x Locations	(a-1)(b-1)(y-1)(l-1)	M17	M17/M18	M17/M18	M17/M18
Pooled error b	yla(b-1)(r-1)	M18			
Corrected total	abryl-1				

My result is showed below, which shows row pattern does not influence lint yield.

SOV	DF	Mean square						
		Lint percent	Lint yield(kg/ha)	MIC	UHML	Uniformity	Strength	Elongation
Year	1	161.66***	13,458,508.21***	0.0344	0.1073**	117.7829**	71.0587**	8.5299**
Location	2	138.03	13,904,890.16**	5.2542	0.0787	13.0299	2.2960	11.6901
Year*Location	2	38.59**	461,743.26***	5.4350***	0.1231***	27.2201*	26.8092	5.1424**
Rep (Year*Location)	18	5.69	33,635.46	0.0906	0.0075***	5.6113***	7.3878***	0.3349
Row	1	24.65	12,110,872.16	0.2026	0.0133	22.0777	9.8870	1.6577*
Year*Row	1	19.10*	3,586,086.24***	0.2054	0.0075*	3.8435	0.8089	0.0027
Location*Row	2	29.79	1,607,140.08	0.0137	0.0013	1.0341	0.0864	0.2238
Year*Location*Row	2	7.32	53,281.11	0.2631*	0.0012	1.5426	5.4714	0.7394*
Rep*Row (Year*Location)	18	2.82	17,919.49	0.0628	0.0015	1.1029	1.8299	0.1440
Genotype	4	74.37*	794,579.21*	6.2409***	0.2771***	34.2105***	241.7301**	13.3946*
Year*Genotype	4	7.40	94,055.29	0.1031	0.0023	0.1481	3.9826	0.3775
Location*Genotype	8	6.52	195,654.52	0.0960	0.0014	3.5743**	3.0831	0.3035
Year*Location*Genotype	8	5.42	160,393.18***	0.2090***	0.0034*	1.3541	1.0661	0.3639
Row*Genotype	4	4.27	83,405.99	0.1024*	0.0031*	0.2323	0.5626	0.8924
Year*Row*Genotype	4	6.67	107,929.86	0.0120	0.0003	0.6013	1.6168	0.2410
Location*Row*Genotype	8	2.60	78,632.19	0.0509	0.0004	0.8853	1.8598	0.6698*
Year*Location*Row*Genotype	8	2.51	77,024.70	0.1004	0.0006	0.7356	1.8473	0.0928
Residual	144	3.78	38,705.88	0.0528	0.0012	1.1007	1.7658	0.1908

*, ** and *** represent significant difference at 0.05, 0.01 and 0.001, respectively, and NS represent no significant difference