DATA 624 Homework Seven - Linear Regression

Kevin Kirby

2024-10-20

Overview

This is homework seven of the Fall 2024 edition of DATA 624. The assignment covers questions 6.2 and 6.3 from the exercise section of chapter 6 in Applied Predictive Modeling by Max Kuhn and Kjell Johnson

First, most of the requried libraries.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(elasticnet)
```

```
## Loading required package: lars
## Loaded lars 1.3
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(lars)
library(pls)
```

```
##
## Attaching package: 'pls'
##
## The following object is masked from 'package:caret':
##
##     R2
##
## The following object is masked from 'package:stats':
```

```
##
##     loadings
```

```r
library(stats)
library(corrplot)
```

```
## corrplot 0.95 loaded
##
## Attaching package: 'corrplot'
##
## The following object is masked from 'package:pls':
##
##     corrplot
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(robustbase)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

6.2 Predicting permeability

This exercise consists of component questions A through F that are about developing permeability prediction models for a pharmaceutical company.

A. Load library and data

The first question just says to load the library for the book and the dataset used.

```r
library(AppliedPredictiveModeling)
data(permeability)
```

B. Filtering out predictors with NearZeroVar

The question states: " The fingerprint predictors indicate the presence or absence of substructures of a molecule and are often sparse meaning that relatively few of the molecules contain each substructure. Filter out the predictors that have low frequencies using the nearZeroVar function from the caret package. How many predictors are left for modeling?"

Answer: there are 388 predictors left after filtering out those with low frequencies.

```
nzv_fingers <- preProcess(fingerprints, method = c("nzv")) |>
    predict(fingerprints)

prem_predict <- ncol(nzv_fingers)
prem_predict
```

```
## [1] 388
```

C. Preprocessing and PLS modeling

The question states: “Split the data into a training and a test set, pre-process the data, and tune a PLS model. How many latent variables are optimal and what is the corresponding resampled estimate of R2?”

Answer:

There are 10 optimal latent variables and the corresponding resampled estimate of R2 is 0.36914.

```
nzv_perm_s <- cbind(nzv_fingers, permeability)

nzv_perm <- preProcess(nzv_perm_s, method = c("BoxCox", "center", "scale", "knnImpute")) |>
    predict(nzv_perm_s)

part <- createDataPartition(nzv_perm[permeability], p = 0.75, list = FALSE)
nzv_perm_train <- nzv_perm[part,]
nzv_perm_test <- nzv_perm[-part,]

pls_control <- trainControl(method = "LOOCV")

pls_train <- train(permeability ~ .,
  data = nzv_perm_train,
  method = "pls",
  tuneLength = 20,
  trControl = pls_control)

pls_train
```

```
## Partial Least Squares
##
## 102 samples
## 388 predictors
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 101, 101, 101, 101, 101, 101, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared   MAE
##    1     0.8271464  0.1995352  0.6614207
##    2     0.7697545  0.3099925  0.5694311
##    3     0.7533316  0.3461914  0.5789168
##    4     0.7715036  0.3410490  0.5957881
##    5     0.7890925  0.3377441  0.6222931
##    6     0.7608655  0.3773040  0.5745275
##    7     0.7378670  0.4067223  0.5544494
##    8     0.7375024  0.4228734  0.5506652
##    9     0.7496511  0.4253353  0.5701878
##   10     0.7522870  0.4326341  0.5743240
##   11     0.7757761  0.4212109  0.5960139
##   12     0.8074302  0.4003050  0.6279007
##   13     0.8197503  0.3964289  0.6350072
##   14     0.8198593  0.4006310  0.6304727
##   15     0.8235668  0.3983049  0.6287512
##   16     0.8243867  0.4038737  0.6256918
##   17     0.8183164  0.4145020  0.6188889
##   18     0.8214717  0.4100121  0.6249308
##   19     0.8215008  0.4193145  0.6328290
##   20     0.8209231  0.4205017  0.6340162
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 8.
```

```r
opt_lat_vars <- pls_train$bestTune$ncomp
rs_r2 <- max(pls_train$results$Rsquared)

cat("There are ", opt_lat_vars, " optimal latent variables and the corresponding resampled estimate of 
```

```
## There are  8  optimal latent variables and the corresponding resampled estimate of R2 is  0.4326341
```

D. Predict the response for the test set. What is the test set estimate of R2?

Answer:

The test set estimate of R2 is 0.5593098

```r
predict(pls_train, nzv_perm_test) %>%
  data.frame(pred = .,obs = nzv_perm_test[,"permeability"]) %>%
  defaultSummary()
```

```
##      RMSE  Rsquared       MAE
## 0.9435346 0.3219123 0.6730243
```

E. Try building other models discussed in this chapter. Do any have better predictive performance?

Answer:

First model: robust linear regression

Results: Samples: 102 Predictors: 388

The robust linear model yielded an RMSE of 0.9446 and an R^2 of 0.3589, which is lower than the previous model's R^2 of 0.5593, indicating that the model did not perform as well in terms of explained variance. The MAE of 0.7090 is slightly higher, suggesting less accuracy in predicting permeability compared to the earlier model. Overall, the previous model with a better R^2 and lower RMSE was more effective in capturing the relationship between predictors and permeability.

```r
nzv_fingers_lr <- preProcess(fingerprints, method = c("nzv")) |>
    predict(fingerprints)

nzv_perm_rl <- cbind(nzv_fingers_lr, permeability)

nzv_perm_lr <- preProcess(nzv_perm_rl, method = c("BoxCox", "center", "scale", "knnImpute")) |>
    predict(nzv_perm_rl)
```

```
part_lr <- createDataPartition(nzv_perm_rl[permeability], p = 0.75, list = FALSE)
perm_train_rl <- nzv_perm_lr[part_lr,]
perm_test_rl <- nzv_perm_lr[-part_lr,]

control_lr <- trainControl(method = "LOOCV")

robust_lines <- train(permeability ~ .,
  data = perm_train_rl,
  method = "rlm",
  trControl = control_lr,
  preProcess = "pca"

)

robust_lines
```

```
## Robust Linear Model
##
## 102 samples
## 388 predictors
##
## Pre-processing: principal component signal extraction (388), centered
##  (388), scaled (388)
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 101, 101, 101, 101, 101, 101, ...
## Resampling results across tuning parameters:
##
##   intercept  psi           RMSE       Rsquared   MAE
##   FALSE      psi.huber     0.9220422  0.1883259  0.6836153
##   FALSE      psi.hampel    0.9282811  0.1975346  0.6775014
##   FALSE      psi.bisquare  0.9523601  0.1725813  0.7009090
##    TRUE      psi.huber     0.9050268  0.1922199  0.6482115
##    TRUE      psi.hampel    0.9499813  0.1621485  0.6868926
##    TRUE      psi.bisquare  0.9475773  0.1680091  0.6857252
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were intercept = TRUE and psi = psi.huber.
```

```
predict(robust_lines, perm_test_rl) %>%
  data.frame(pred = .,obs = perm_test_rl[,"permeability"]) %>%
  defaultSummary()
```

```
##      RMSE  Rsquared       MAE
## 0.9518847 0.3451346 0.6942958
```

Second model: Elastic net regression Results: Samples: 102 Predictors: 388

The Elastic Net model, using bootstrapped resampling with 10 repetitions, resulted in an RMSE of 0.6883, an R^2 of 0.5943, and an MAE of 0.5342. This model explored a grid of lambda and fraction parameters and selected the optimal values of lambda = 0.1 and fraction = 0.15 based on the smallest RMSE. The resampling process indicates that the Elastic Net model performs moderately well, with an R^2 value showing that it captures about 59.4% of the variance in the data. The relatively low MAE further indicates that the model is accurate in predicting permeability values with minimal error.

```
ctrl_enet <- trainControl(method = "boot", number = 10)
```

```
enet_g <- expand.grid(.lambda = c(0, 0.01, .1), .fraction = seq(.05, 1, length = 20))

enet_reg <- train(permeability ~ .,
  data = nzv_perm_train,
  method = "enet",
  tuneGrid = enet_g,
  trControl = ctrl_enet)
```

```
## Warning: model fit failed for Resample01: lambda=0.10, fraction=1 Error in elasticnet::enet(as.matri
##   Some of the columns of x have zero variance

## Warning: model fit failed for Resample01: lambda=0.01, fraction=1 Error in elasticnet::enet(as.matri
##   Some of the columns of x have zero variance

## Warning: model fit failed for Resample01: lambda=0.00, fraction=1 Error in elasticnet::enet(as.matri
##   Some of the columns of x have zero variance

## Warning: model fit failed for Resample07: lambda=0.10, fraction=1 Error in elasticnet::enet(as.matri
##   Some of the columns of x have zero variance

## Warning: model fit failed for Resample07: lambda=0.01, fraction=1 Error in elasticnet::enet(as.matri
##   Some of the columns of x have zero variance

## Warning: model fit failed for Resample07: lambda=0.00, fraction=1 Error in elasticnet::enet(as.matri
##   Some of the columns of x have zero variance

## Warning: model fit failed for Resample10: lambda=0.00, fraction=1 Error in if (zmin < gamhat) { : mi

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
enet_reg
```

```
## Elasticnet
##
## 102 samples
## 388 predictors
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 102, 102, 102, 102, 102, 102, ...
## Resampling results across tuning parameters:
##
##   lambda  fraction  RMSE          Rsquared   MAE
##   0.00    0.05      1.578876e+13  0.1857008  7.845331e+12
##   0.00    0.10      3.158899e+13  0.1511649  1.569645e+13
##   0.00    0.15      4.726828e+13  0.1409682  2.353313e+13
##   0.00    0.20      6.228455e+13  0.1322933  3.128664e+13
##   0.00    0.25      7.733460e+13  0.1260184  3.904015e+13
##   0.00    0.30      9.240197e+13  0.1212453  4.679366e+13
##   0.00    0.35      1.074794e+14  0.1180477  5.454717e+13
##   0.00    0.40      1.224326e+14  0.1169503  6.231051e+13
##   0.00    0.45      1.368737e+14  0.1142003  7.011877e+13
##   0.00    0.50      1.510487e+14  0.1114241  7.778704e+13
##   0.00    0.55      1.651293e+14  0.1120194  8.537068e+13
##   0.00    0.60      1.793229e+14  0.1107955  9.295433e+13
##   0.00    0.65      1.936049e+14  0.1105019  1.005380e+14
##   0.00    0.70      2.079571e+14  0.1102098  1.081216e+14
##   0.00    0.75      2.223660e+14  0.1097366  1.157053e+14
##   0.00    0.80      2.368213e+14  0.1115167  1.232889e+14
##   0.00    0.85      2.513149e+14  0.1122020  1.308725e+14
##   0.00    0.90      2.658406e+14  0.1126183  1.384562e+14
##   0.00    0.95      2.803935e+14  0.1132984  1.460398e+14
##   0.00    1.00      2.949695e+14  0.1133748  1.536235e+14
##   0.01    0.05      7.162277e-01  0.4988679  5.507321e-01
##   0.01    0.10      6.810117e-01  0.4809882  5.252125e-01
##   0.01    0.15      6.978997e-01  0.4623850  5.415945e-01
##   0.01    0.20      7.419465e-01  0.4387871  5.754623e-01
##   0.01    0.25      7.854974e-01  0.4218620  5.995533e-01
##   0.01    0.30      8.205392e-01  0.4080758  6.179703e-01
##   0.01    0.35      8.471716e-01  0.3994755  6.331232e-01
##   0.01    0.40      8.731008e-01  0.3884573  6.500833e-01
##   0.01    0.45      8.994112e-01  0.3736740  6.655364e-01
##   0.01    0.50      9.178721e-01  0.3650218  6.759698e-01
##   0.01    0.55      9.334722e-01  0.3607974  6.847246e-01
##   0.01    0.60      9.486000e-01  0.3562460  6.973041e-01
##   0.01    0.65      9.631299e-01  0.3484530  7.077599e-01
##   0.01    0.70      9.760060e-01  0.3406306  7.162645e-01
##   0.01    0.75      9.887935e-01  0.3333620  7.255981e-01
##   0.01    0.80      9.984521e-01  0.3273113  7.314787e-01
##   0.01    0.85      1.008252e+00  0.3213217  7.382977e-01
##   0.01    0.90      1.014776e+00  0.3164413  7.433234e-01
##   0.01    0.95      1.020154e+00  0.3119466  7.465711e-01
##   0.01    1.00      1.091740e+00  0.2682271  7.819648e-01
##   0.10    0.05      7.535053e-01  0.4879493  5.884827e-01
##   0.10    0.10      6.890351e-01  0.4973610  5.261962e-01
##   0.10    0.15      6.765143e-01  0.4855988  5.161452e-01
##   0.10    0.20      6.807495e-01  0.4800131  5.209122e-01
##   0.10    0.25      7.041596e-01  0.4637714  5.417569e-01
##   0.10    0.30      7.350317e-01  0.4500422  5.641133e-01
##   0.10    0.35      7.669113e-01  0.4370450  5.837074e-01
##   0.10    0.40      7.931163e-01  0.4273692  5.999637e-01
##   0.10    0.45      8.131483e-01  0.4201816  6.129797e-01
##   0.10    0.50      8.284914e-01  0.4164109  6.228111e-01
##   0.10    0.55      8.413675e-01  0.4133602  6.296489e-01
##   0.10    0.60      8.527895e-01  0.4092003  6.357207e-01
##   0.10    0.65      8.616710e-01  0.4056173  6.398881e-01
##   0.10    0.70      8.696809e-01  0.4014398  6.437052e-01
##   0.10    0.75      8.770494e-01  0.3968836  6.474018e-01
##   0.10    0.80      8.826850e-01  0.3936905  6.504443e-01
##   0.10    0.85      8.876909e-01  0.3913194  6.532819e-01
##   0.10    0.90      8.915705e-01  0.3897473  6.558102e-01
##   0.10    0.95      8.955244e-01  0.3880432  6.585976e-01
##   0.10    1.00      9.026839e-01  0.3772384  6.622832e-01
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were fraction = 0.15 and lambda = 0.1.
```

```
predict(enet_reg, nzv_perm_test) %>%
  data.frame(pred = .,obs = nzv_perm_test[,"permeability"]) %>%
  defaultSummary()
```

```
##      RMSE  Rsquared       MAE
## 0.8479712 0.3783789 0.6373006
```

Overall conclusion: When comparing the models overall, the Elastic Net model performed the best, with the highest R^2 (0.5943) and the lowest RMSE (0.6883). The robust linear model, with an R^2 of 0.3589 and RMSE of 0.9446, was less effective in explaining the variance and predicting permeability accurately compared to the Elastic Net.

F. Would you recommend any of your models to replace the permeability laboratory experiment?

Answer: I wouldn't recommend using any of the models trained here to replace lab experiments for permeability due to a lot of variability. Pharmaceutical companies need much tighter standards than the results here provide.

6.3 Modeling biological raw materials and their manufacturing process

This exercise consists of component questions A through F

A. Load the data

```r
data(ChemicalManufacturingProcess)
```

B. Imputing missing values

The question states: "A small percentage of cells in the predictor set contain missing values. Use an imputation function to fill in these missing values."

Answer:

I used median impute to insert the median value where a missing value is.

```r
resolve_miss <- preProcess(ChemicalManufacturingProcess, method = c("medianImpute")) |>
  predict(ChemicalManufacturingProcess)
```

C. Preprocessing and tuning a model

The question states: "Split the data into a training and a test set, pre-process the data, and tune a model of your choice from this chapter. What is the optimal value of the performance metric?"

Answer:

I will use Ridge Regression. This is a model that which imposes a penalty on the coefficients based on the different lambdas that we will be training over.

First, preprocessing and setting up the train and test sets:

```r
model_tuner <- preProcess(resolve_miss, method = c("center", "scale")) |>
  predict(resolve_miss)

ridge_slices <- createDataPartition(model_tuner$Yield, p = 0.75, list = FALSE)

ridge_train <- model_tuner[ridge_slices,]
ridge_test <- model_tuner[-ridge_slices,]

cat("The training set has dimensions of: \n")
```

```
## The training set has dimensions of:
```

```r
dim(ridge_train)
```

```
## [1] 132  58
```

Finding optimal performance metric: The optimal value of the performance metric RMSE is 1.831127, with a lambda of 0.1.

```
ridge_control <- trainControl(method = "boot", number = 10)
ridge_roller <- expand.grid(.alpha = 0, .lambda = seq(0, .1, length = 15))

ridge_reg <- train(Yield ~ .,
                   data = ridge_train,
                   method = "glmnet",
                   tuneGrid = ridge_roller,
                   trControl = ridge_control)

ridge_reg
```

```
## glmnet
##
## 132 samples
##  57 predictor
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 132, 132, 132, 132, 132, 132, ...
## Resampling results across tuning parameters:
##
##   lambda       RMSE      Rsquared   MAE
##   0.000000000  1.561337  0.3576345  0.7493857
##   0.007142857  1.561337  0.3576345  0.7493857
##   0.014285714  1.561337  0.3576345  0.7493857
##   0.021428571  1.561337  0.3576345  0.7493857
##   0.028571429  1.561337  0.3576345  0.7493857
##   0.035714286  1.561337  0.3576345  0.7493857
##   0.042857143  1.561337  0.3576345  0.7493857
##   0.050000000  1.561277  0.3577499  0.7493639
##   0.057142857  1.555890  0.3593487  0.7474509
##   0.064285714  1.540593  0.3616902  0.7426401
##   0.071428571  1.516119  0.3644733  0.7359753
##   0.078571429  1.492741  0.3671692  0.7307345
##   0.085714286  1.471129  0.3697479  0.7258988
##   0.092857143  1.450771  0.3722033  0.7212213
##   0.100000000  1.432090  0.3745098  0.7169136
##
## Tuning parameter 'alpha' was held constant at a value of 0
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda = 0.1.
```

D. Predicting test set response

The question states: Predict the response for the test set. What is the value of the performance metric and how does this compare with the resampled performance metric on the training set?

Answer: The RMSE for the test set is 0.7467, which is significantly lower than the resampled RMSE of 1.8311 on the training set. This suggests that the model generalizes well to the test set, showing better performance than during the resampling on the training set.

```r
predict(ridge_reg, ridge_test) %>%
  data.frame(pred = .,obs = ridge_test$Yield) %>%
  defaultSummary()
```

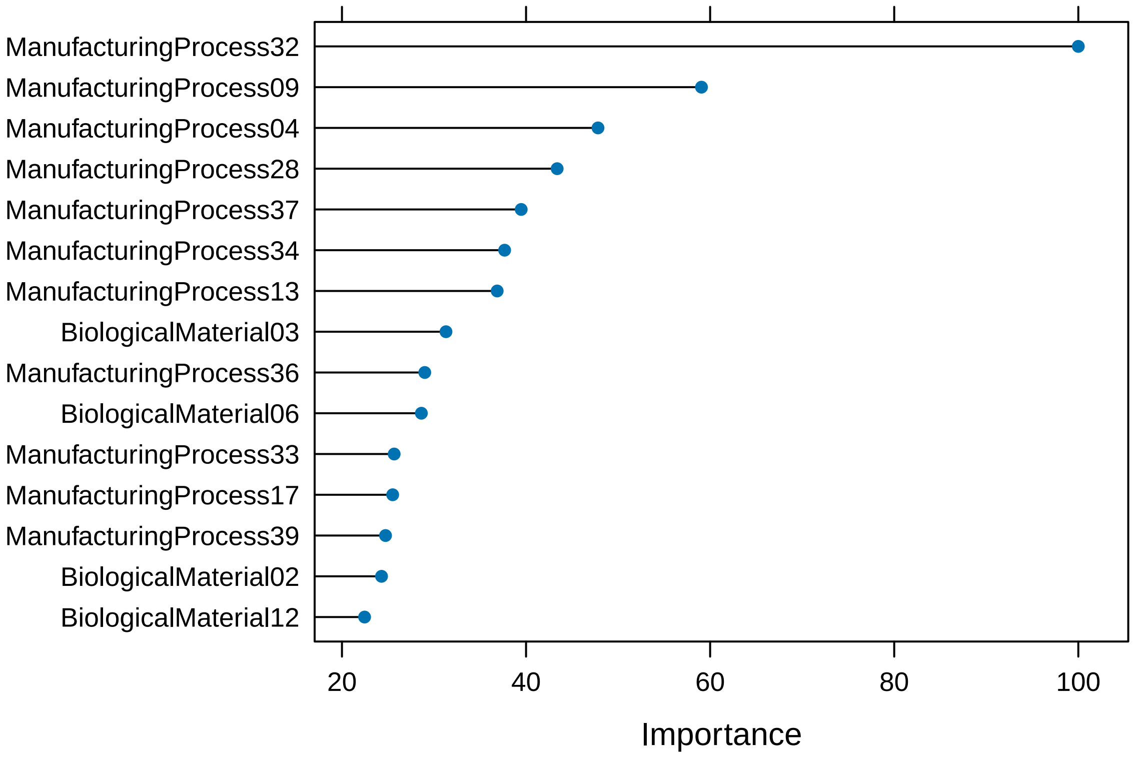
```
##      RMSE  Rsquared       MAE
## 0.5971940 0.6888535 0.4588769
```

E. Assessing predictors value

The question states: Which predictors are most important in the model you have trained? Do either the biological or process predictors dominate the list?

Answer: There are 8 process predictors and 7 biological predictors in the top 15. Neither process dominates the list.

```r
plot(varImp(ridge_reg), top = 15)
```



```r
  ggtitle("Top 15 predictors")
```

```
## $title
## [1] "Top 15 predictors"
##
## attr(,"class")
## [1] "labels"
```

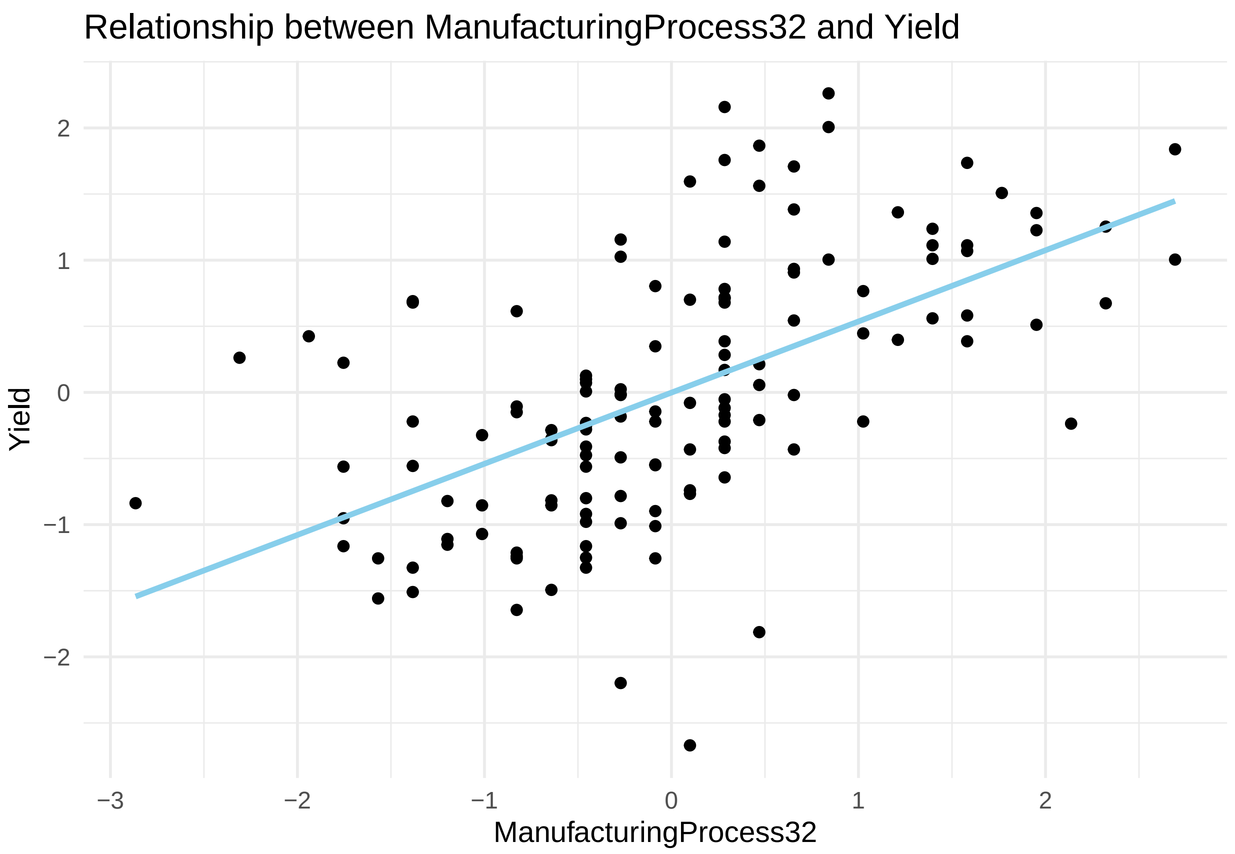
F. Reviewing predictors and response relationships

The question states: "Explore the relationships between each of the top predictors and the response. How could this information be helpful in improving yield in future runs of the manufacturing process?"

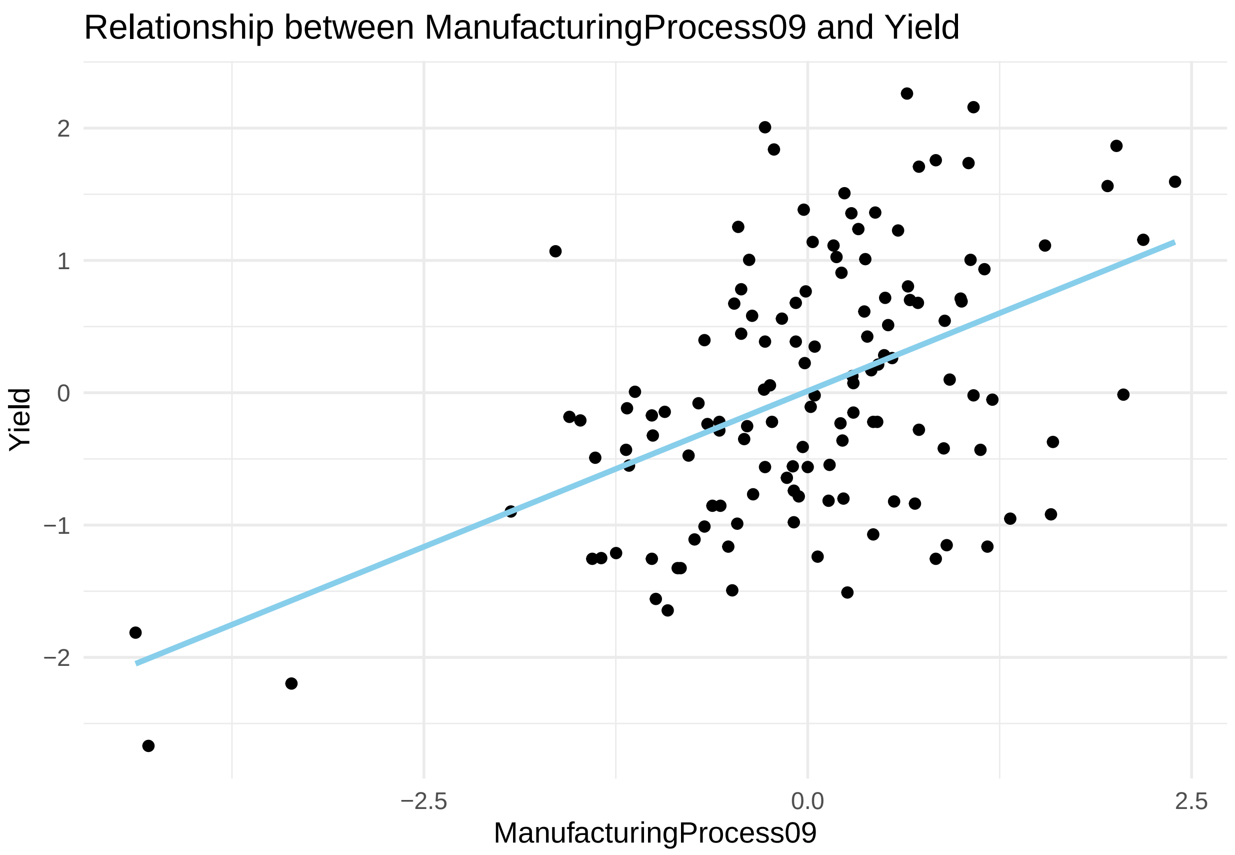
Answer: The relationships between the top predictors and yield show some clear trends. For example, ManufacturingProcess32 and BiologicalMaterial06 have strong positive correlations, meaning boosting them could improve yield in future runs. On the other hand, ManufacturingProcess36 and ManufacturingProcess13 negatively impact yield, so controlling those might help reduce their drag on the process.

```r
ridge_15 <- varImp(ridge_reg)$importance
ridge_15 <- as.data.frame(ridge_15)
ridge_predict <- rownames(ridge_15)[order(ridge_15[, 1], decreasing = TRUE)[1:15]]

for (rider in ridge_predict) {
  yields <- ggplot(ridge_train, aes(x = .data[[rider]], y = .data[["Yield"]])) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, color = "skyblue") +
    ggtitle(paste("Relationship between", rider, "and Yield")) +
    theme_minimal()

  print(yields)
}
```

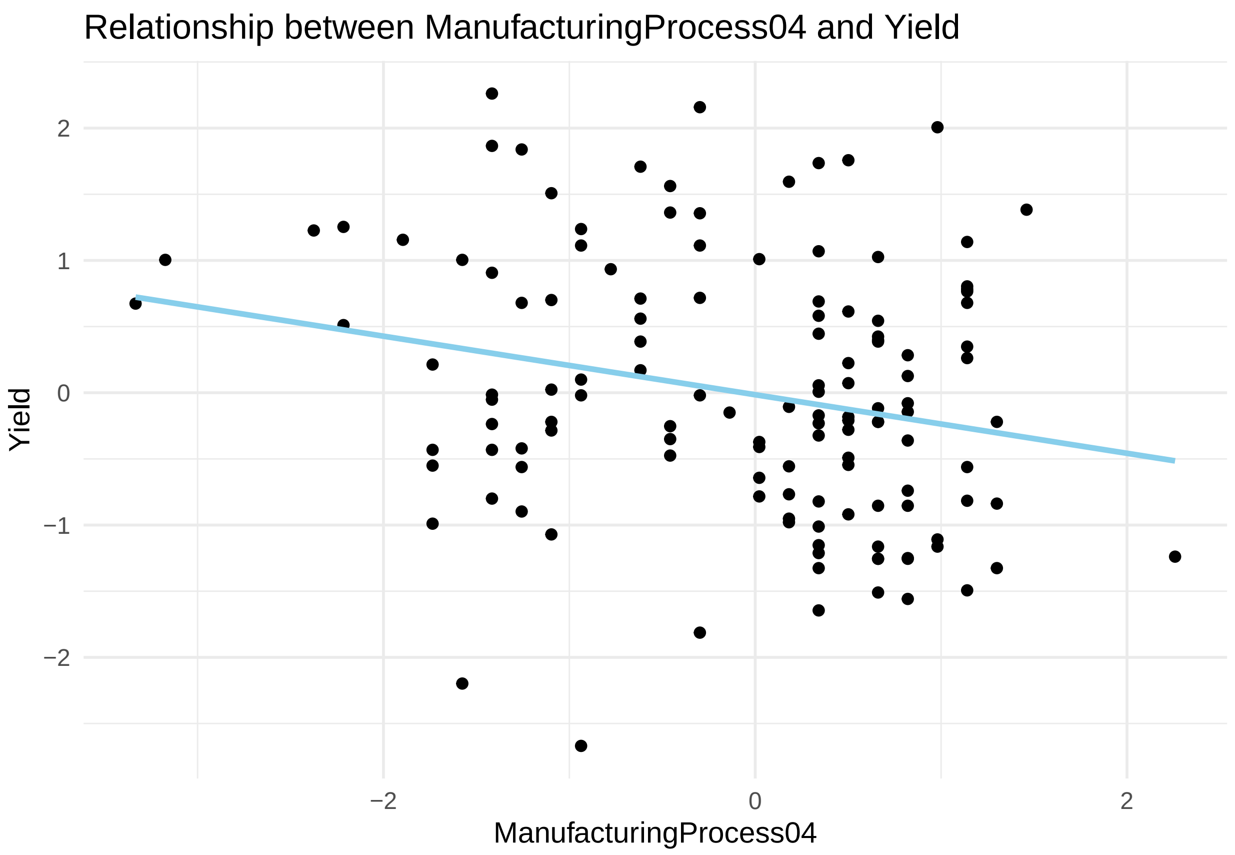
```
## `geom_smooth()` using formula = 'y ~ x'
```



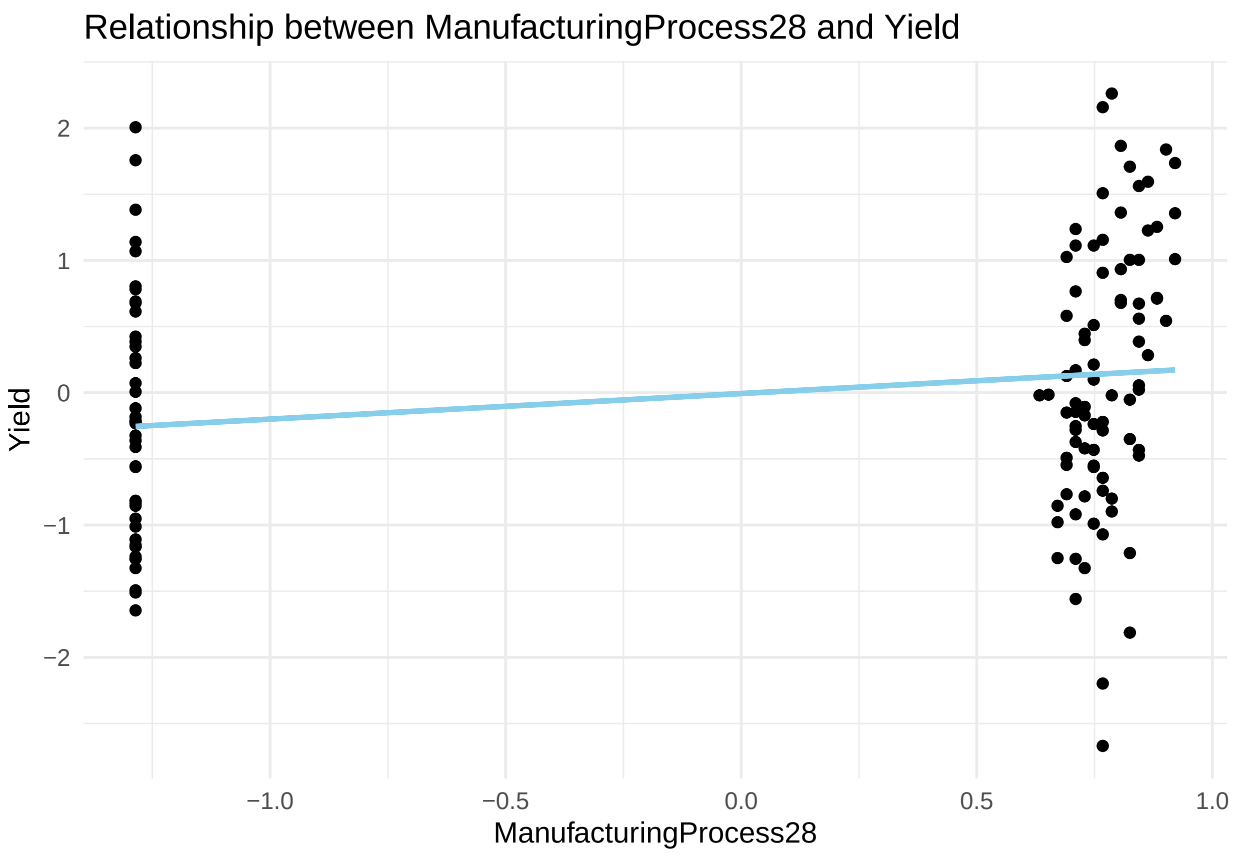
```
## `geom_smooth()` using formula = 'y ~ x'
```



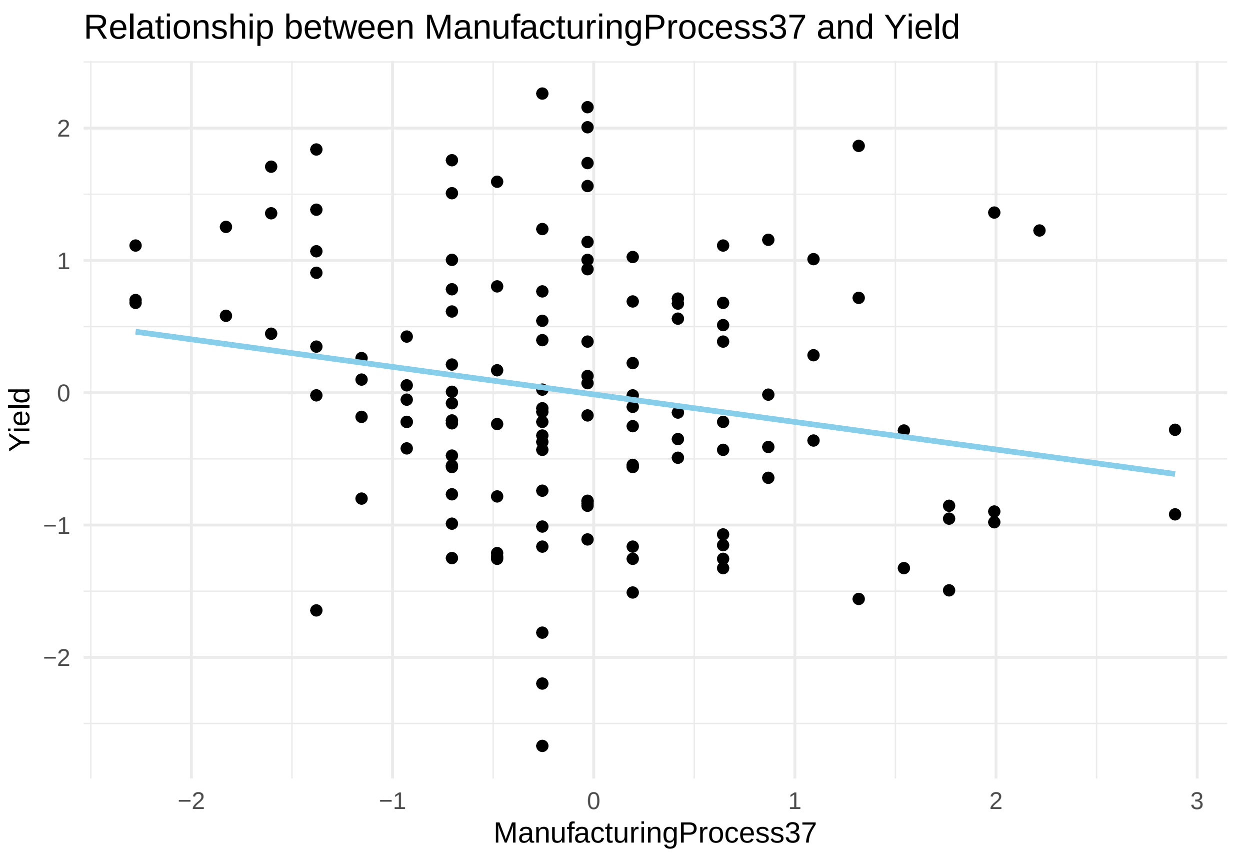
```
## `geom_smooth()` using formula = 'y ~ x'
```



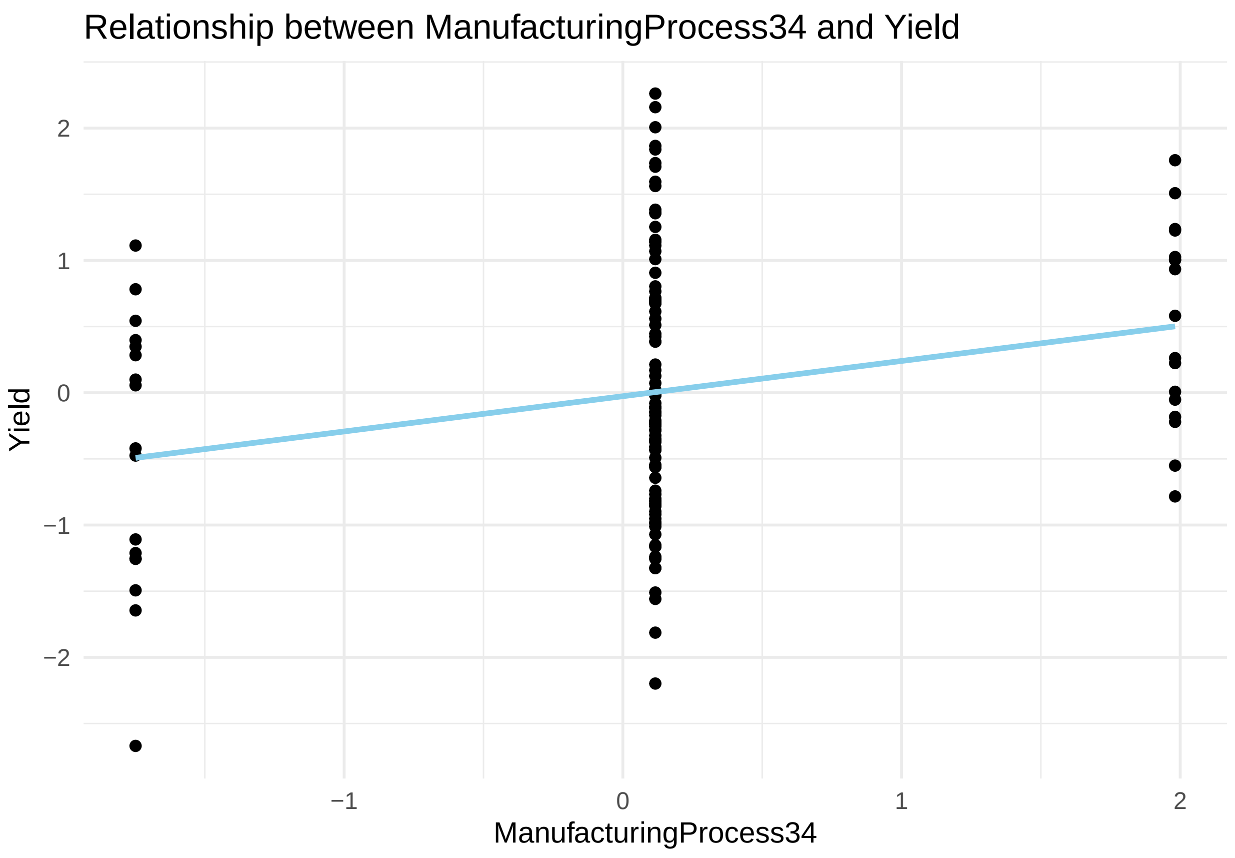
```
## `geom_smooth()` using formula = 'y ~ x'
```



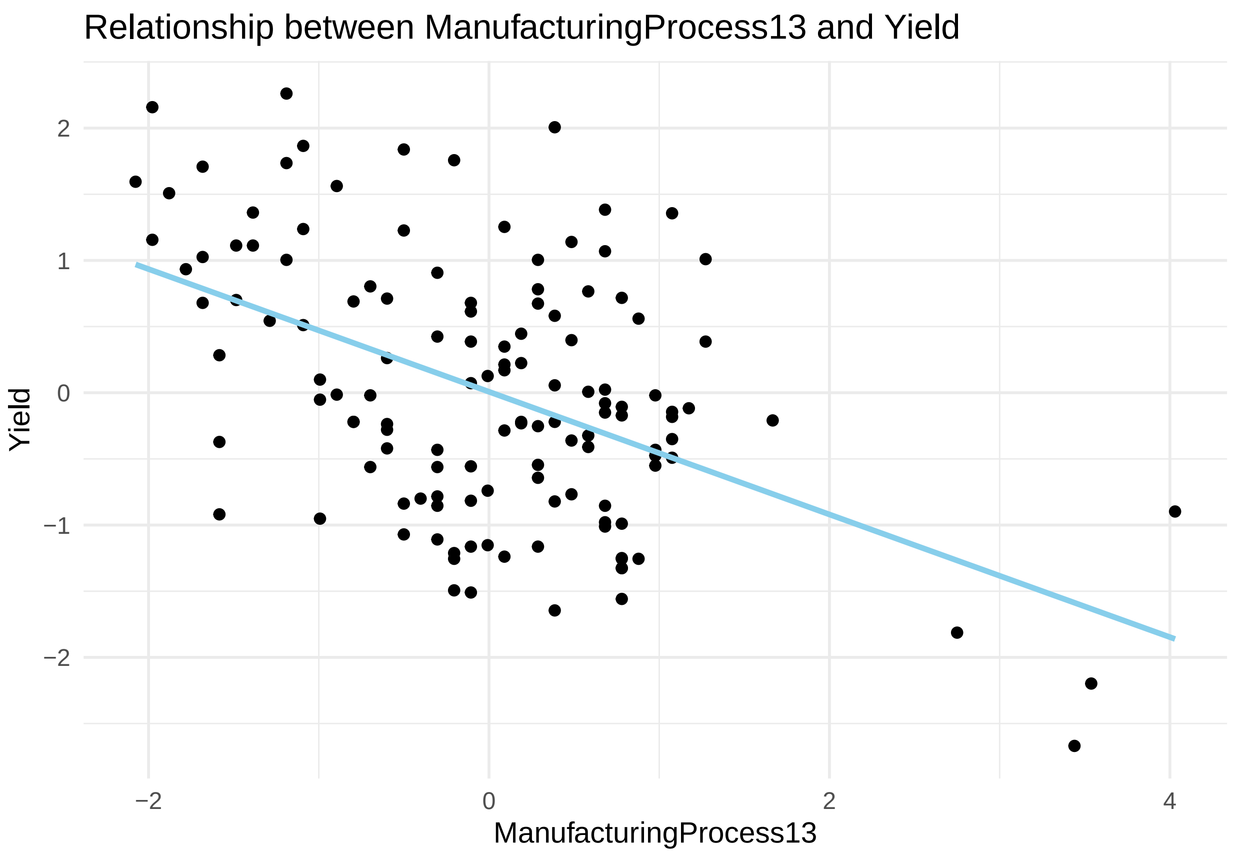
```
## `geom_smooth()` using formula = 'y ~ x'
```



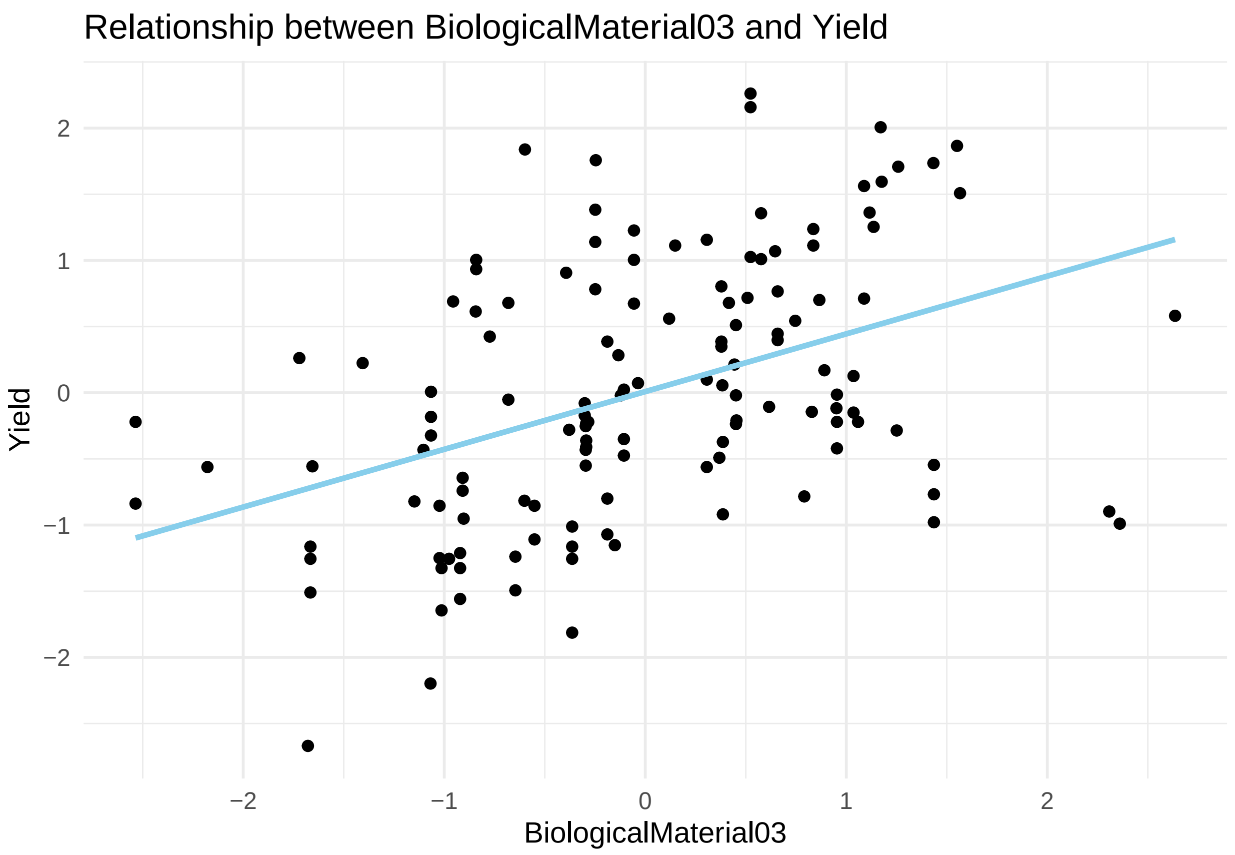
```
## `geom_smooth()` using formula = 'y ~ x'
```



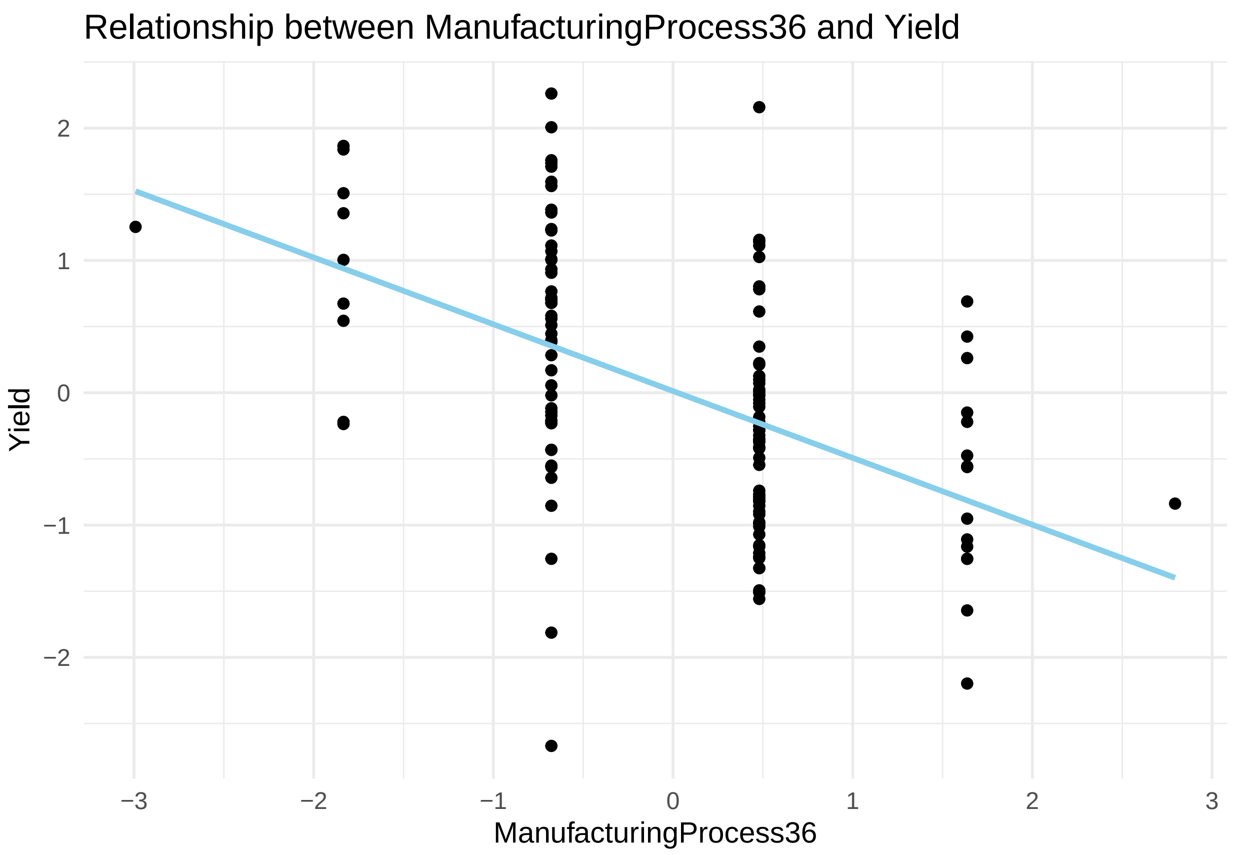
```
## `geom_smooth()` using formula = 'y ~ x'
```



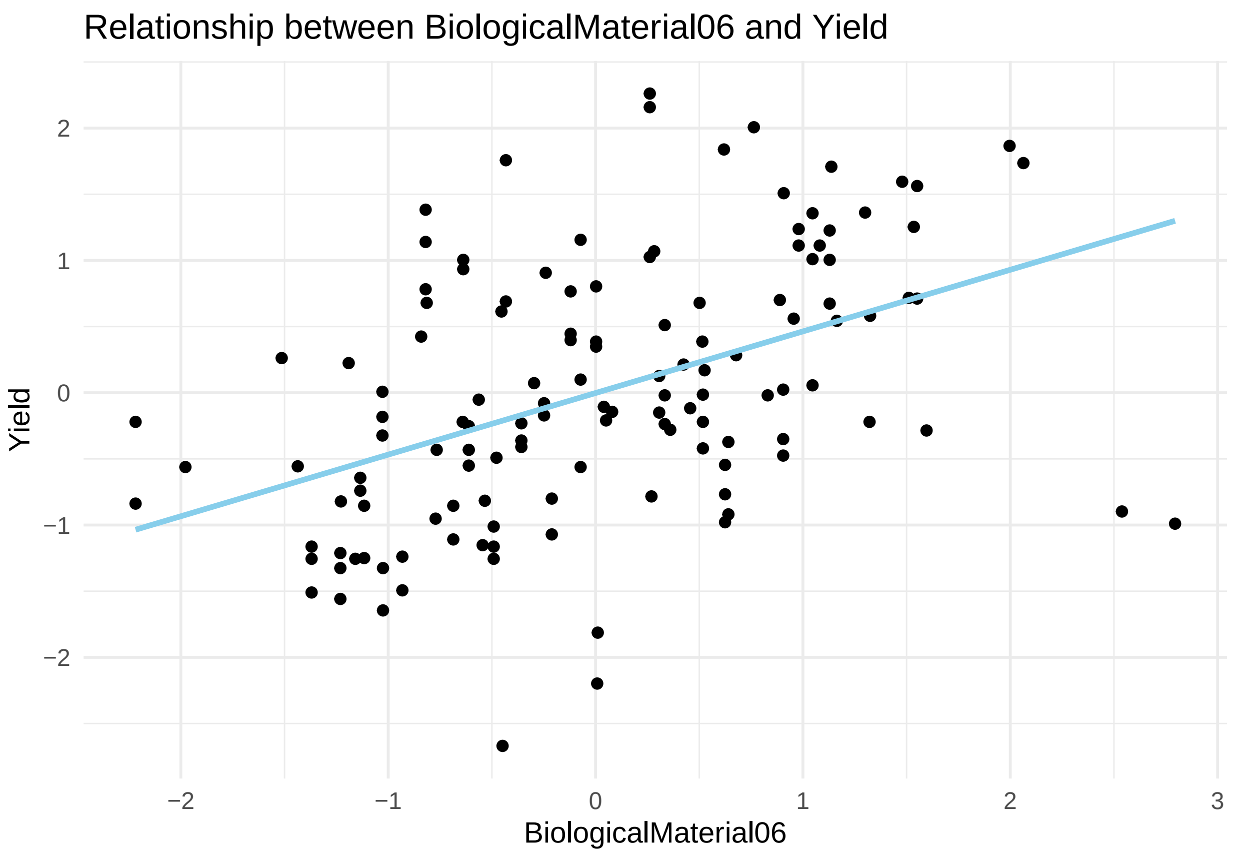
```
## `geom_smooth()` using formula = 'y ~ x'
```



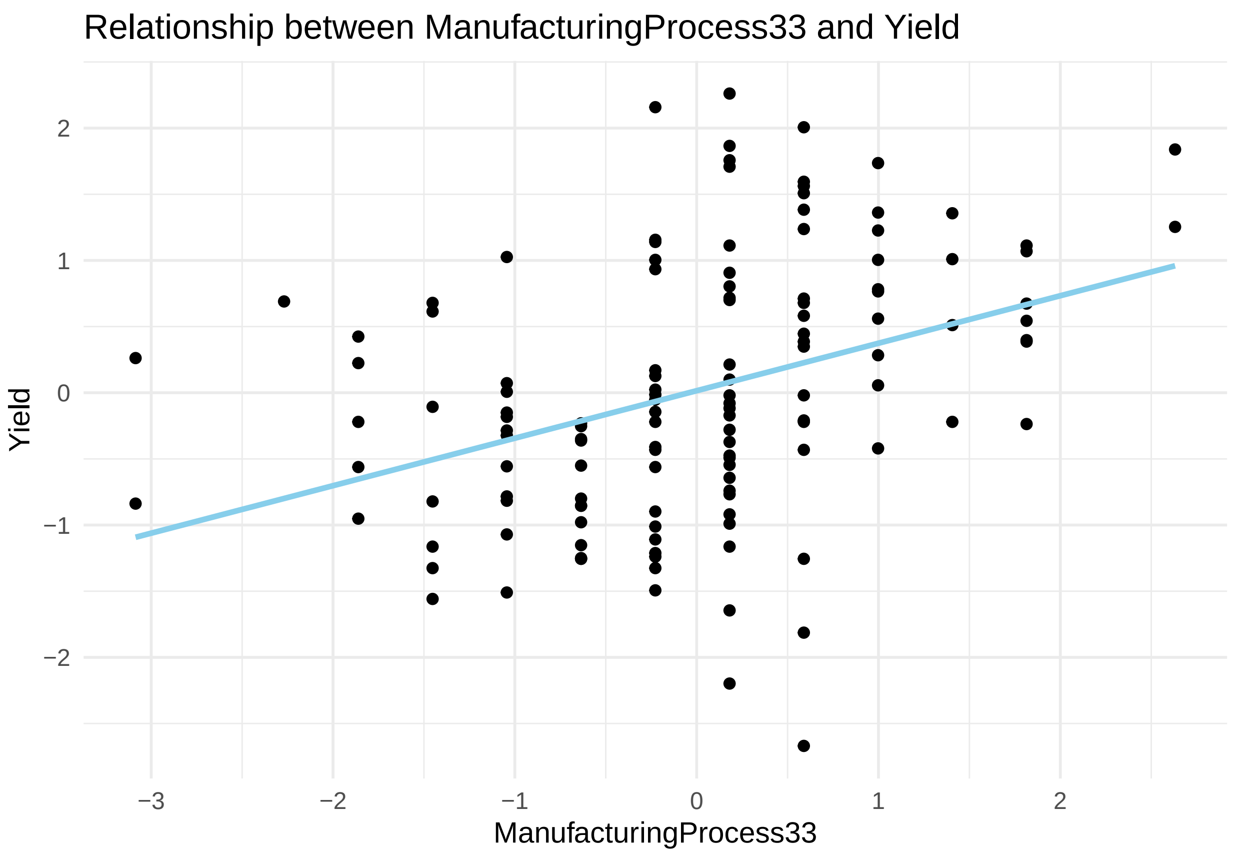
```
## `geom_smooth()` using formula = 'y ~ x'
```



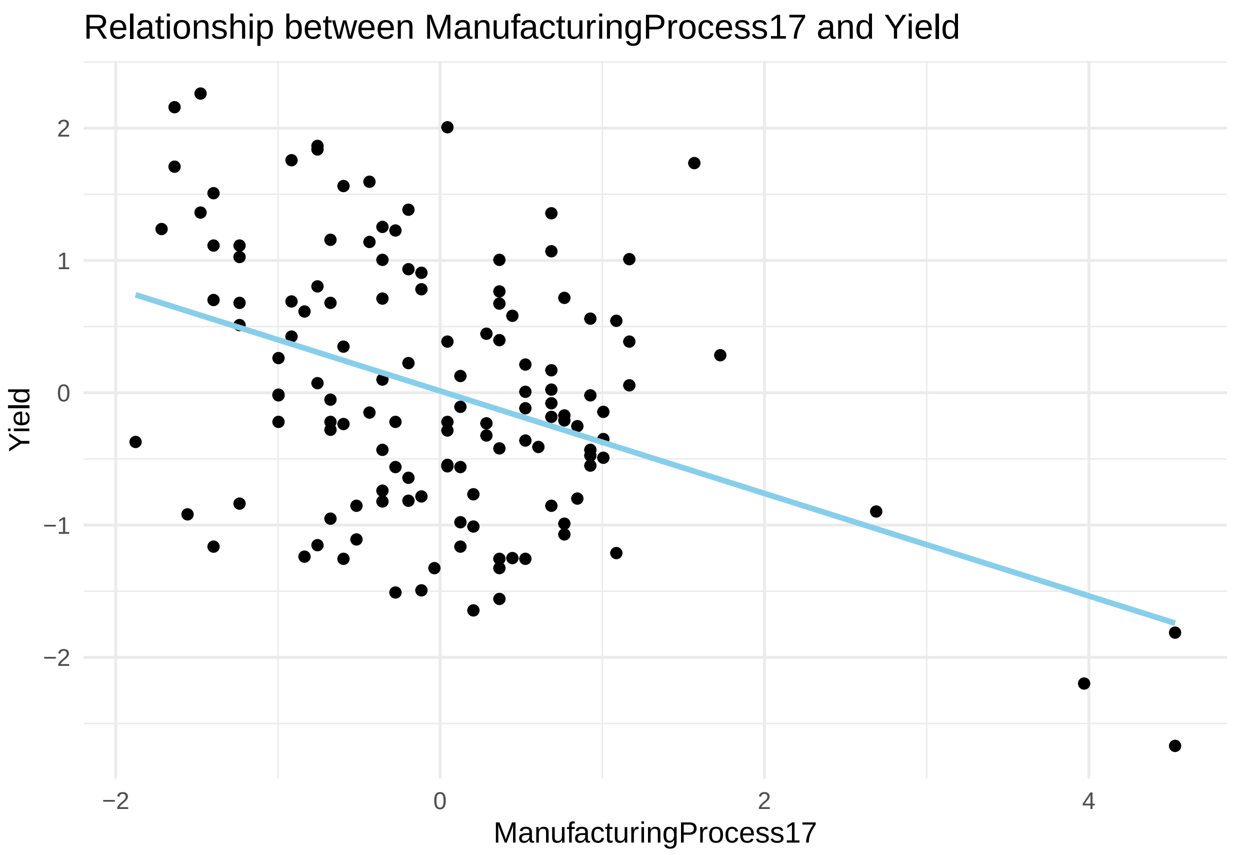
```
## `geom_smooth()` using formula = 'y ~ x'
```



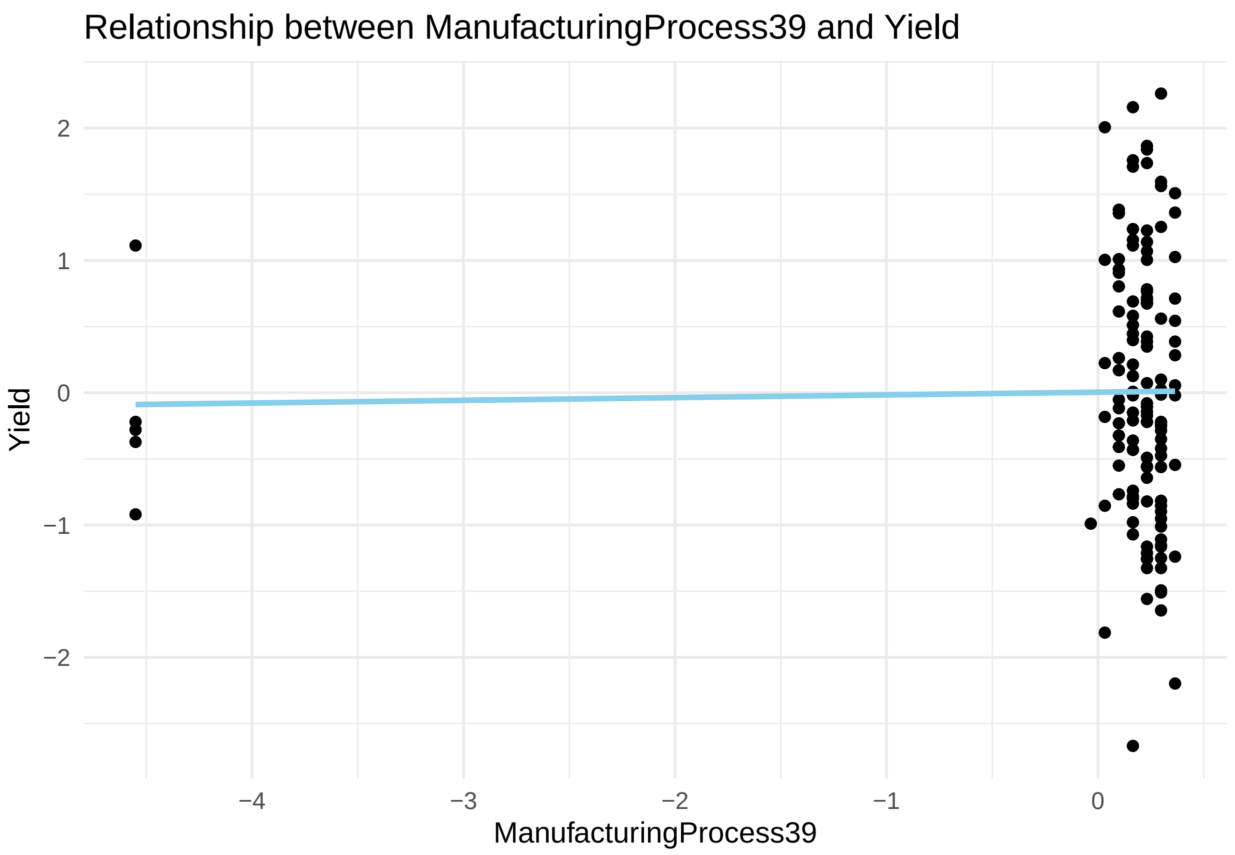
```
## `geom_smooth()` using formula = 'y ~ x'
```



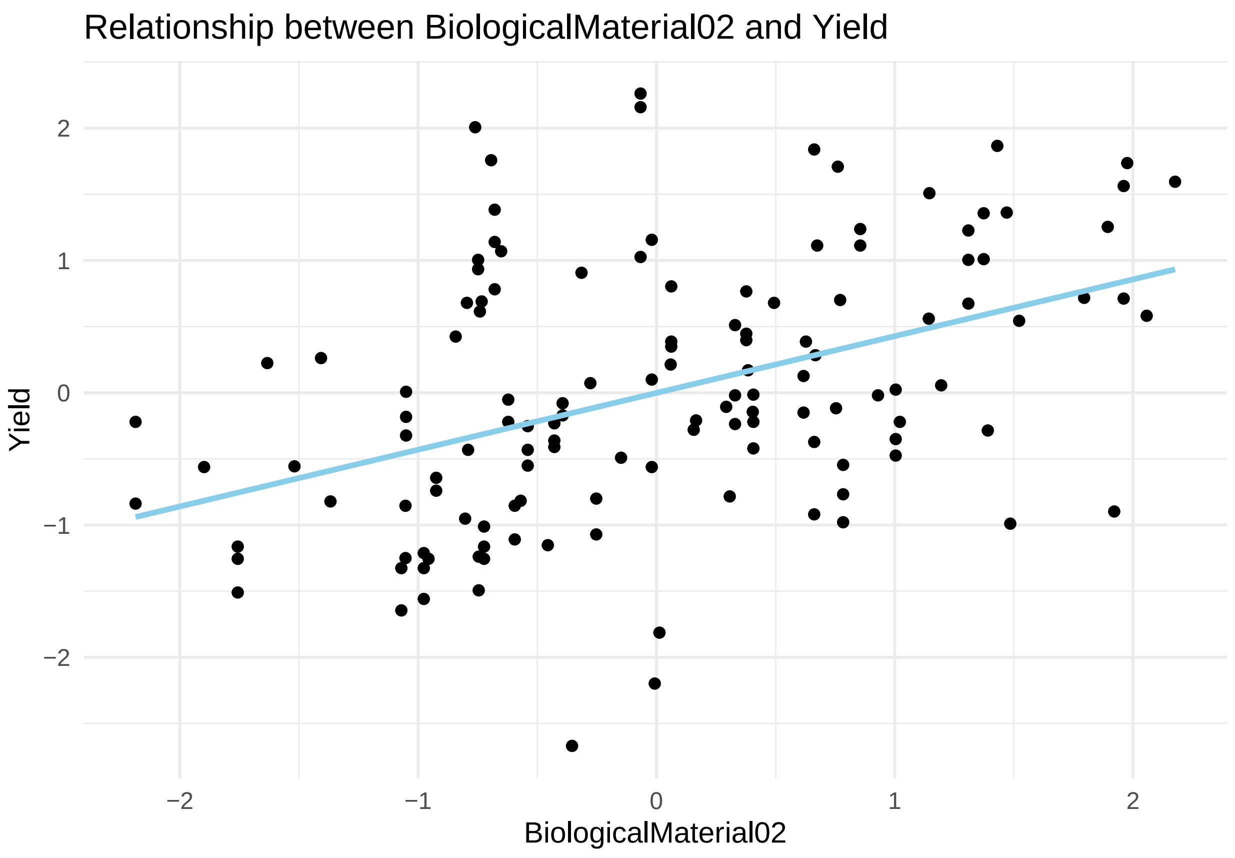
```
## `geom_smooth()` using formula = 'y ~ x'
```



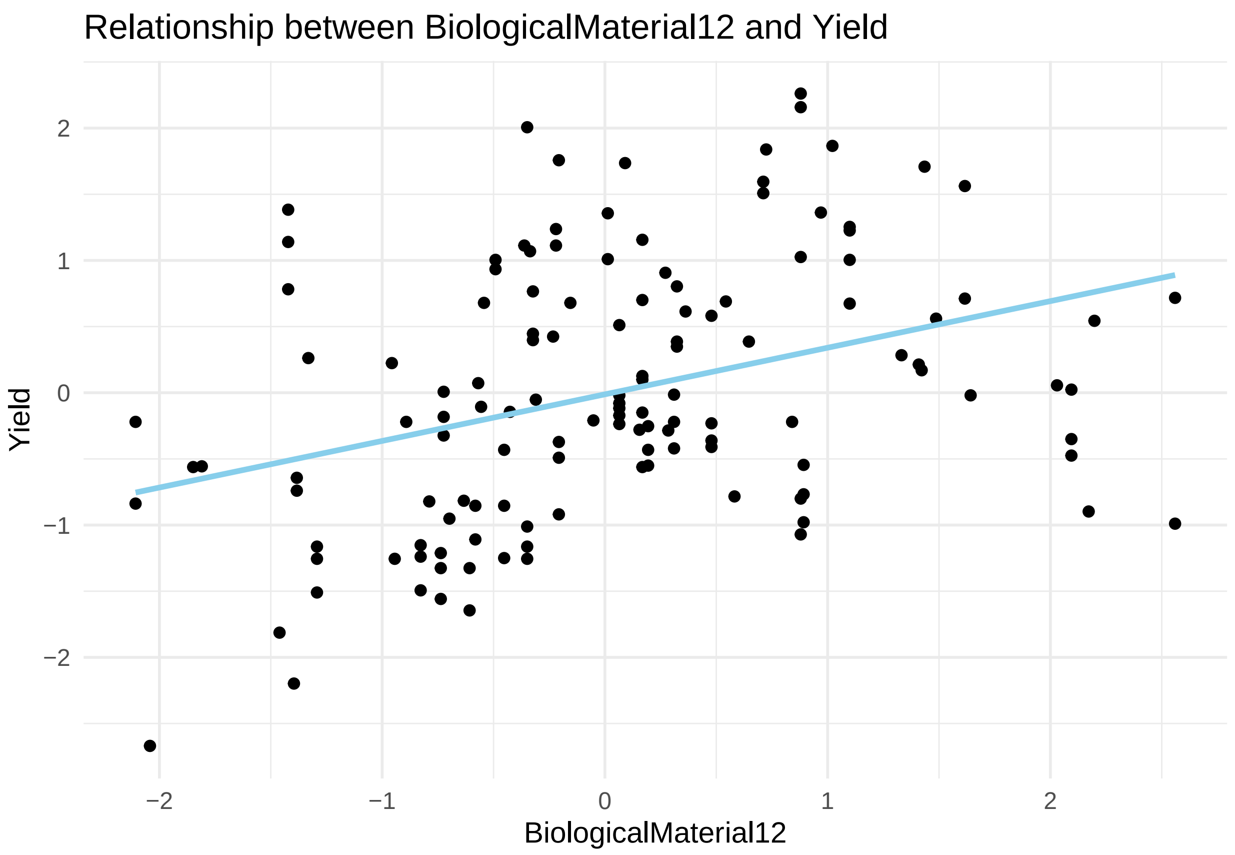
```
## `geom_smooth()` using formula = 'y ~ x'
```



```
## `geom_smooth()` using formula = 'y ~ x'
```



```
## `geom_smooth()` using formula = 'y ~ x'
```



```
ridge_corrs <- sapply(ridge_predict, function(x) cor(ridge_train[[x]], ridge_train$Yield, use = "comple
ridges <- data.frame(Predictor = ridge_predict, Correlation = ridge_corrs)
ridges
```

```
##                                   Predictor Correlation
## ManufacturingProcess32 ManufacturingProcess32   0.5717626
## ManufacturingProcess09 ManufacturingProcess09   0.4991455
## ManufacturingProcess04 ManufacturingProcess04  -0.2331167
## ManufacturingProcess28 ManufacturingProcess28   0.1935851
## ManufacturingProcess37 ManufacturingProcess37  -0.2095755
## ManufacturingProcess34 ManufacturingProcess34   0.2569805
## ManufacturingProcess13 ManufacturingProcess13  -0.4938458
## BiologicalMaterial03     BiologicalMaterial03   0.4277751
## ManufacturingProcess36 ManufacturingProcess36  -0.4963770
## BiologicalMaterial06     BiologicalMaterial06   0.4575380
## ManufacturingProcess33 ManufacturingProcess33   0.3738568
## ManufacturingProcess17 ManufacturingProcess17  -0.4151582
## ManufacturingProcess39 ManufacturingProcess39   0.0190833
## BiologicalMaterial02     BiologicalMaterial02   0.4326214
## BiologicalMaterial12     BiologicalMaterial12   0.3577218
```