Linear Regression, GAM Models, and RMSE Comparison

2025-05-04

Setup Data

```r
# Get data
wineData <- read.csv("~/Downloads/wine-quality-white-and-red.csv")

wineData$type <- as.factor(wineData$type)

# Split to training and testing data
set.seed(1)
train = sample(1:nrow(wineData), 0.7 * nrow(wineData))
train_data = wineData[train, ]
test_data = wineData[-train, ]
```

Best Subset Selection

```r
regfit.best=regsubsets(quality~.,data=train_data, nvmax=12)
test.mat=model.matrix(quality~.,data=test_data) # create an X matrix of test data
val.errors=rep(NA,19)
for(i in 1:12){
   coefi=coef(regfit.best,id=i)
   pred=test.mat[,names(coefi)]%*%coefi
   val.errors[i] <- mean((test_data$quality - pred)^2)
}

summary(regfit.best)
```

```
## Subset selection object
## Call: regsubsets.formula(quality ~ ., data = train_data, nvmax = 12)
## 12 Variables  (and intercept)
##                        Forced in Forced out
## typewhite                  FALSE      FALSE
## fixed.acidity              FALSE      FALSE
## volatile.acidity           FALSE      FALSE
## citric.acid                FALSE      FALSE
## residual.sugar             FALSE      FALSE
## chlorides                  FALSE      FALSE
## free.sulfur.dioxide        FALSE      FALSE
## total.sulfur.dioxide       FALSE      FALSE
## density                    FALSE      FALSE
## pH                         FALSE      FALSE
## sulphates                FALSE      FALSE
## alcohol                  FALSE      FALSE
## 1 subsets of each size up to 12
## Selection Algorithm: exhaustive
##           typewhite fixed.acidity volatile.acidity citric.acid residual.sugar
## 1  ( 1 )  " "       " "           " "              " "         " "
## 2  ( 1 )  " "       " "           "*"              " "         " "
## 3  ( 1 )  " "       " "           "*"              " "         " "
## 4  ( 1 )  " "       " "           "*"              " "         "*"
## 5  ( 1 )  " "       " "           "*"              " "         "*"
## 6  ( 1 )  " "       " "           "*"              " "         "*"
## 7  ( 1 )  " "       " "           "*"              " "         "*"
## 8  ( 1 )  "*"       " "           "*"              " "         "*"
## 9  ( 1 )  "*"       "*"           "*"              " "         "*"
## 10  ( 1 ) "*"       "*"           "*"              " "         "*"
## 11  ( 1 ) "*"       "*"           "*"              " "         "*"
## 12  ( 1 ) "*"       "*"           "*"              "*"         "*"
##           chlorides free.sulfur.dioxide total.sulfur.dioxide density pH
## 1  ( 1 )  " "       " "                 " "                  " "     " "
## 2  ( 1 )  " "       " "                 " "                  " "     " "
## 3  ( 1 )  " "       " "                 " "                  " "     " "
## 4  ( 1 )  " "       " "                 " "                  " "     " "
## 5  ( 1 )  " "       "*"                 "*"                  " "     " "
## 6  ( 1 )  " "       "*"                 "*"                  " "     " "
## 7  ( 1 )  "*"       "*"                 "*"                  " "     " "
## 8  ( 1 )  " "       "*"                 "*"                  "*"     " "
## 9  ( 1 )  " "       "*"                 " "                  "*"     "*"
## 10  ( 1 ) " "       "*"                 "*"                  "*"     "*"
## 11  ( 1 ) "*"       "*"                 "*"                  "*"     "*"
## 12  ( 1 ) "*"       "*"                 "*"                  "*"     "*"
##           sulphates alcohol
## 1  ( 1 )  " "       "*"
## 2  ( 1 )  " "       "*"
## 3  ( 1 )  "*"       "*"
## 4  ( 1 )  "*"       "*"
## 5  ( 1 )  " "       "*"
## 6  ( 1 )  "*"       "*"
## 7  ( 1 )  "*"       "*"
## 8  ( 1 )  "*"       "*"
## 9  ( 1 )  "*"       "*"
## 10  ( 1 ) "*"       "*"
## 11  ( 1 ) "*"       "*"
## 12  ( 1 ) "*"       "*"
```

```r
which.min(val.errors)
```

```
## [1] 12
```

```r
coef(regfit.best,12)
```

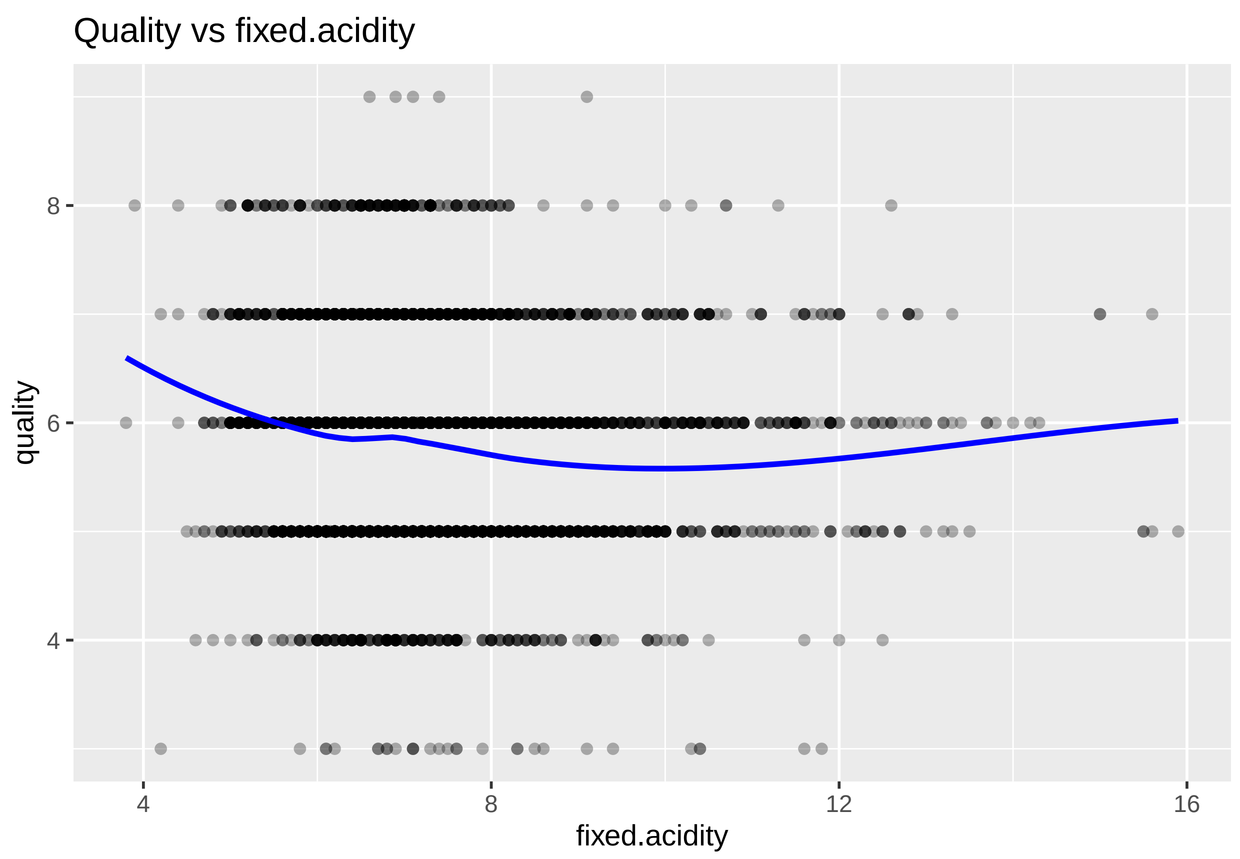
```
##          (Intercept)            typewhite        fixed.acidity
##         1.190237e+02        -3.734753e-01         9.607001e-02
##     volatile.acidity          citric.acid       residual.sugar
##        -1.381747e+00        -1.535991e-02         6.575127e-02
##            chlorides  free.sulfur.dioxide total.sulfur.dioxide
##        -8.970624e-01         5.661084e-03        -1.625646e-03
##              density                   pH            sulphates
##        -1.184895e+02         5.766522e-01         6.741088e-01
##              alcohol
##         2.107583e-01
```

Explore Predictors for GAM

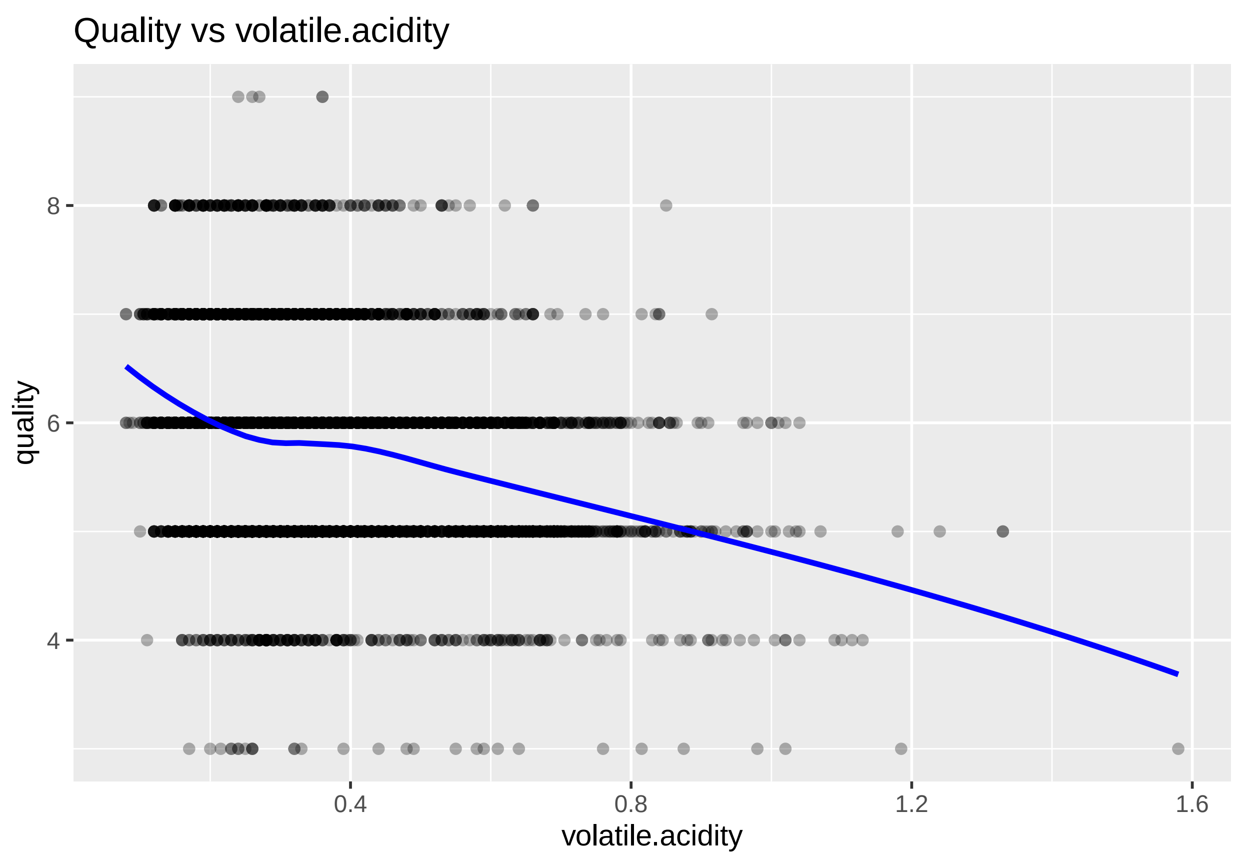
```r
library(ggplot2)

# Continuous variables
predictors <- c("fixed.acidity", "volatile.acidity", "citric.acid", "residual.sugar",
                "chlorides", "free.sulfur.dioxide", "total.sulfur.dioxide",
                "density", "pH", "sulphates", "alcohol")

# Plot each predictor against quality
for (var in predictors) {
  p <- ggplot(wineData, aes_string(x = var, y = "quality")) +
    geom_point(alpha = 0.3) +
    geom_smooth(method = "loess", se = FALSE, color = "blue") +
    labs(title = paste("Quality vs", var))
  print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

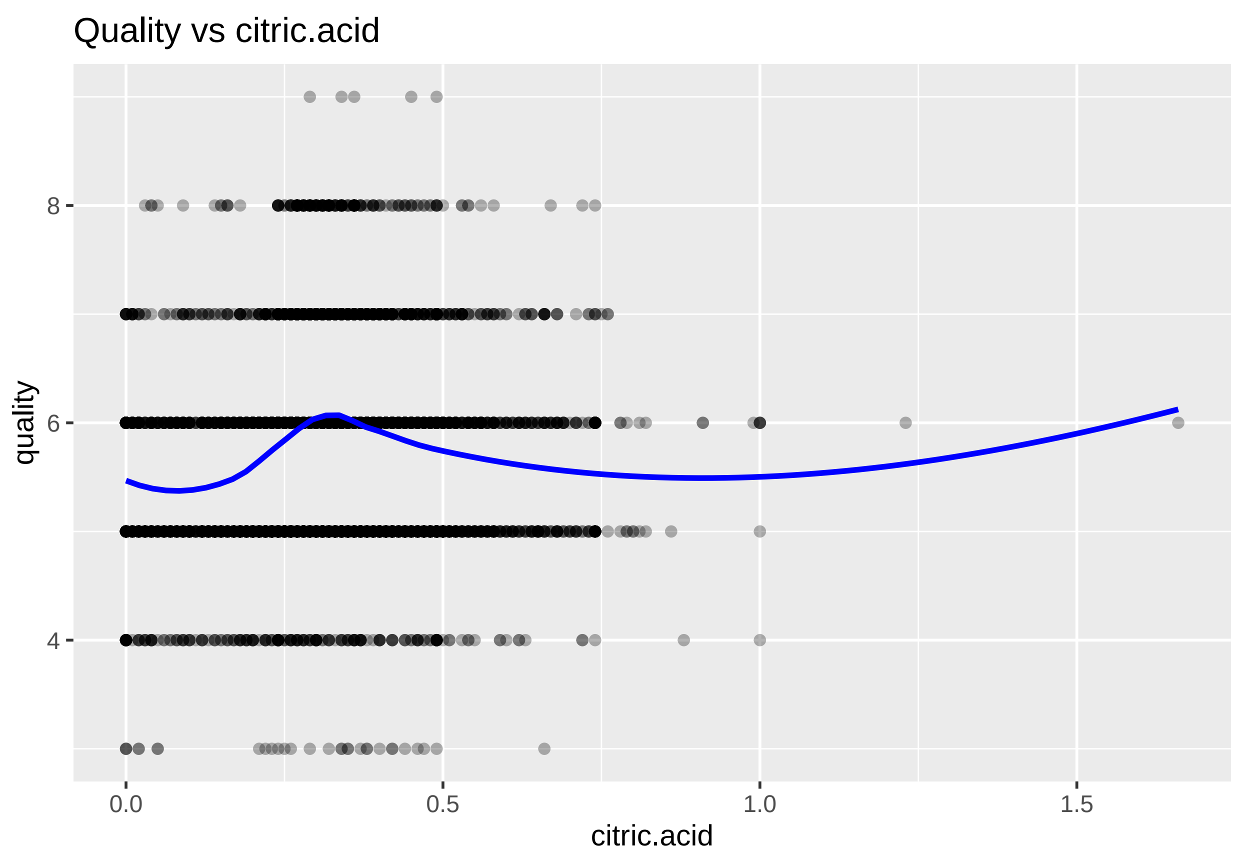
```
## `geom_smooth()` using formula = 'y ~ x'
```



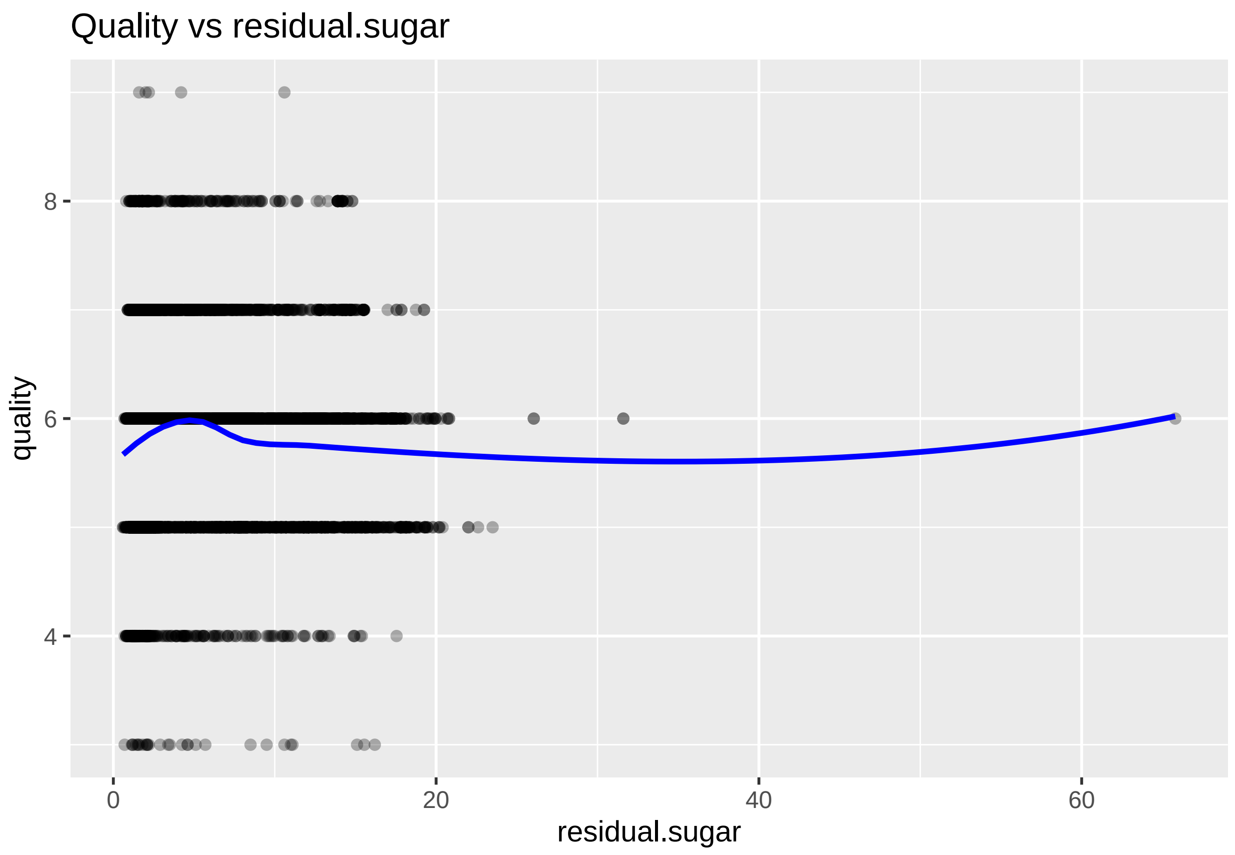
```
## `geom_smooth()` using formula = 'y ~ x'
```



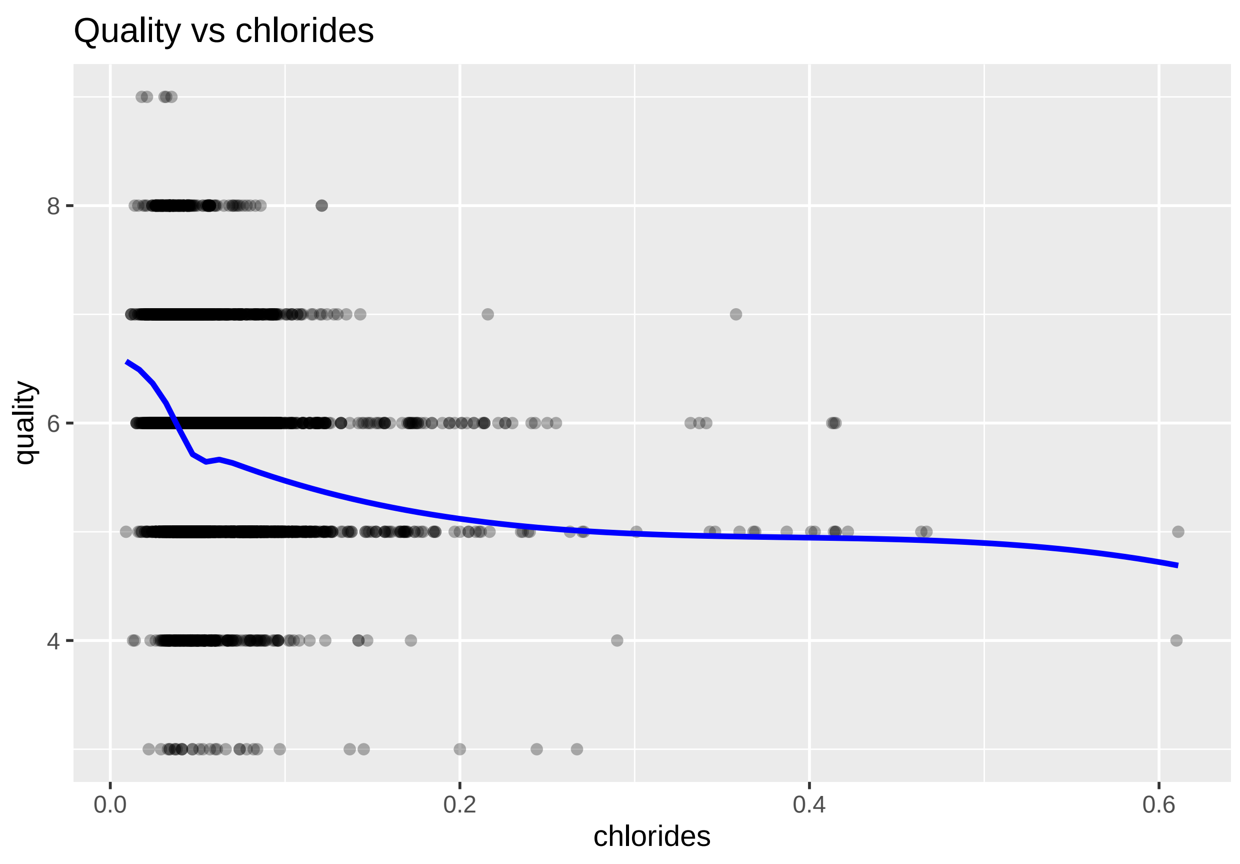
```
## `geom_smooth()` using formula = 'y ~ x'
```



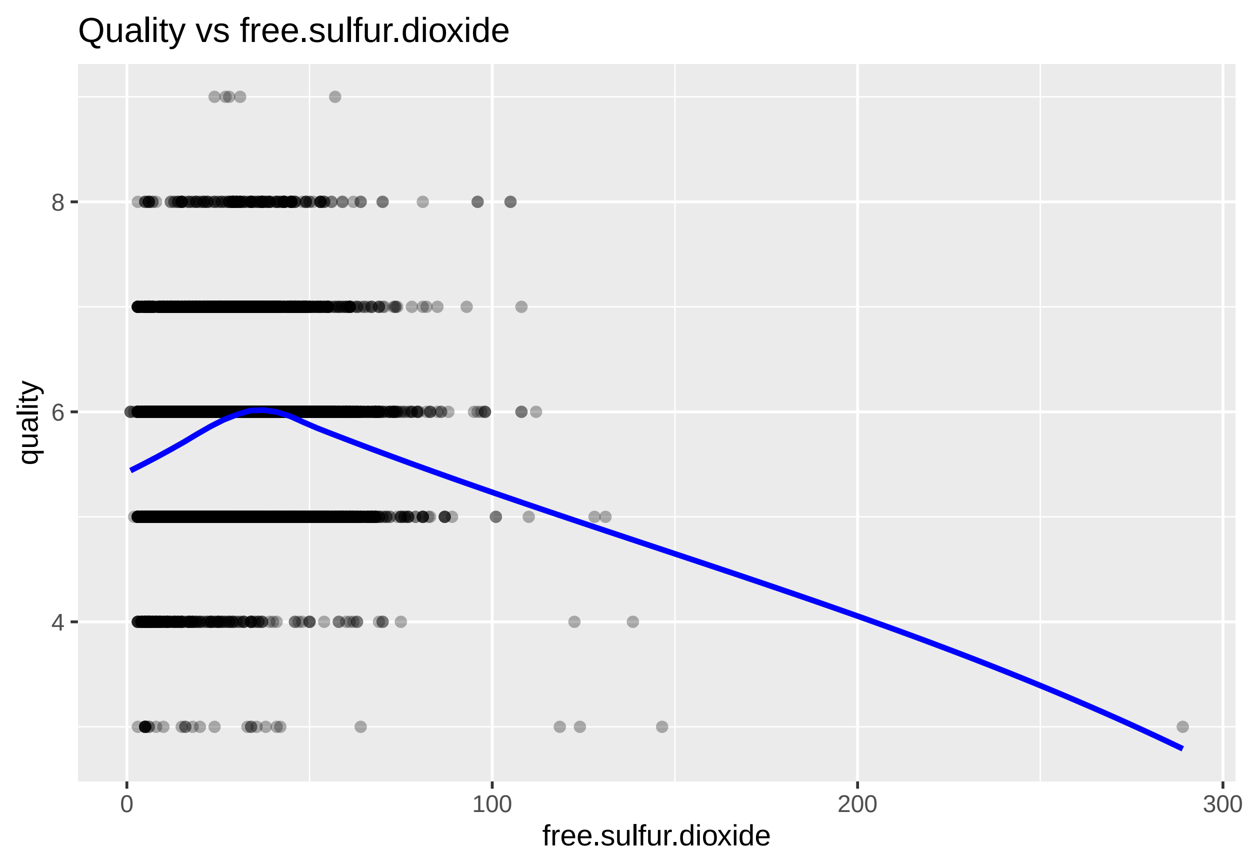
```
## `geom_smooth()` using formula = 'y ~ x'
```



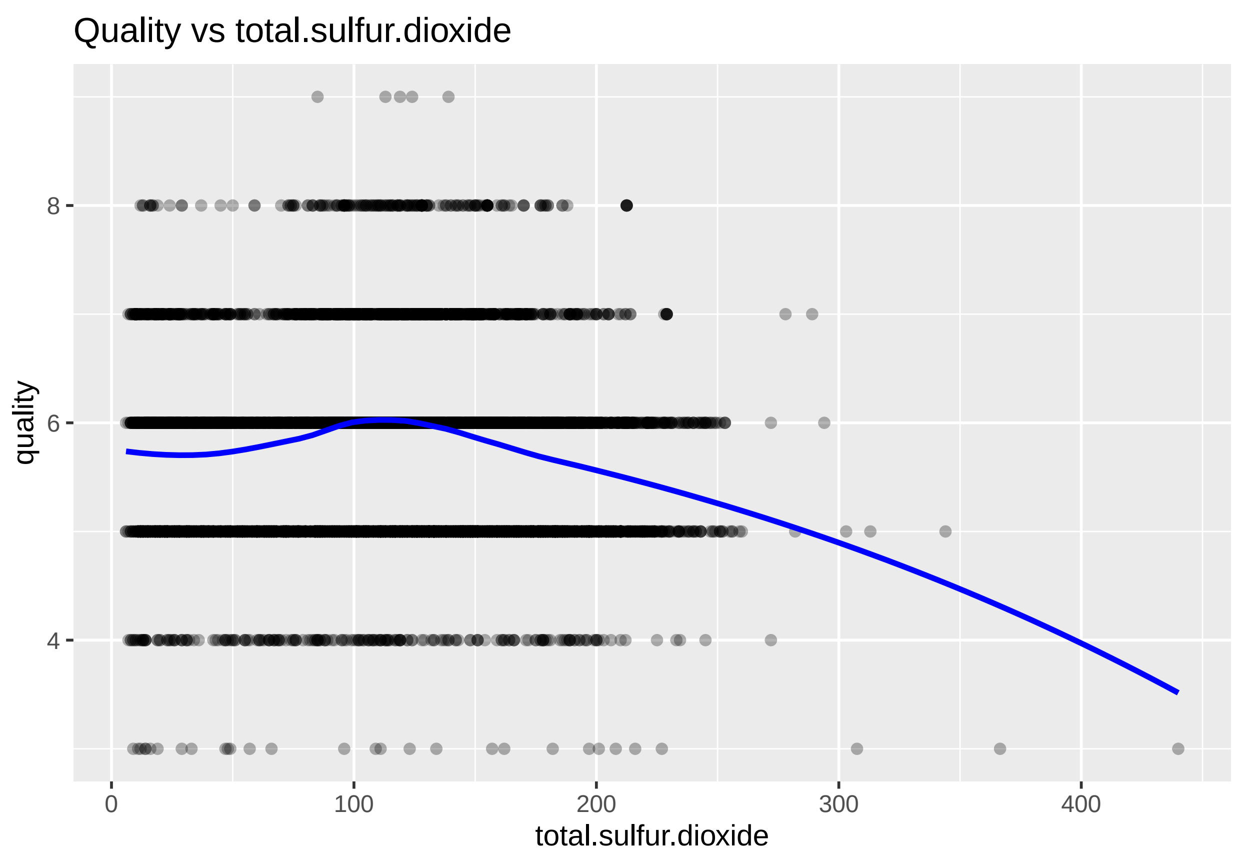
```
## `geom_smooth()` using formula = 'y ~ x'
```



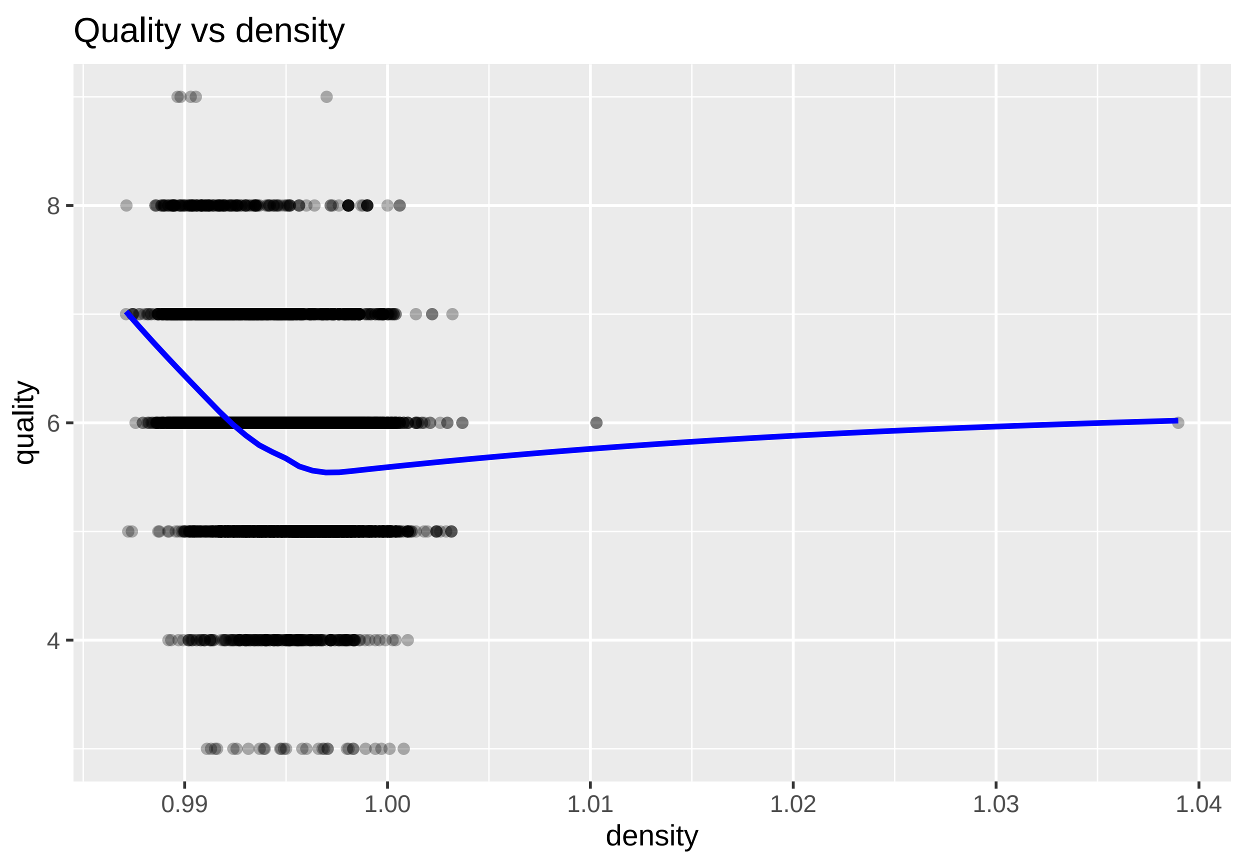
```
## `geom_smooth()` using formula = 'y ~ x'
```



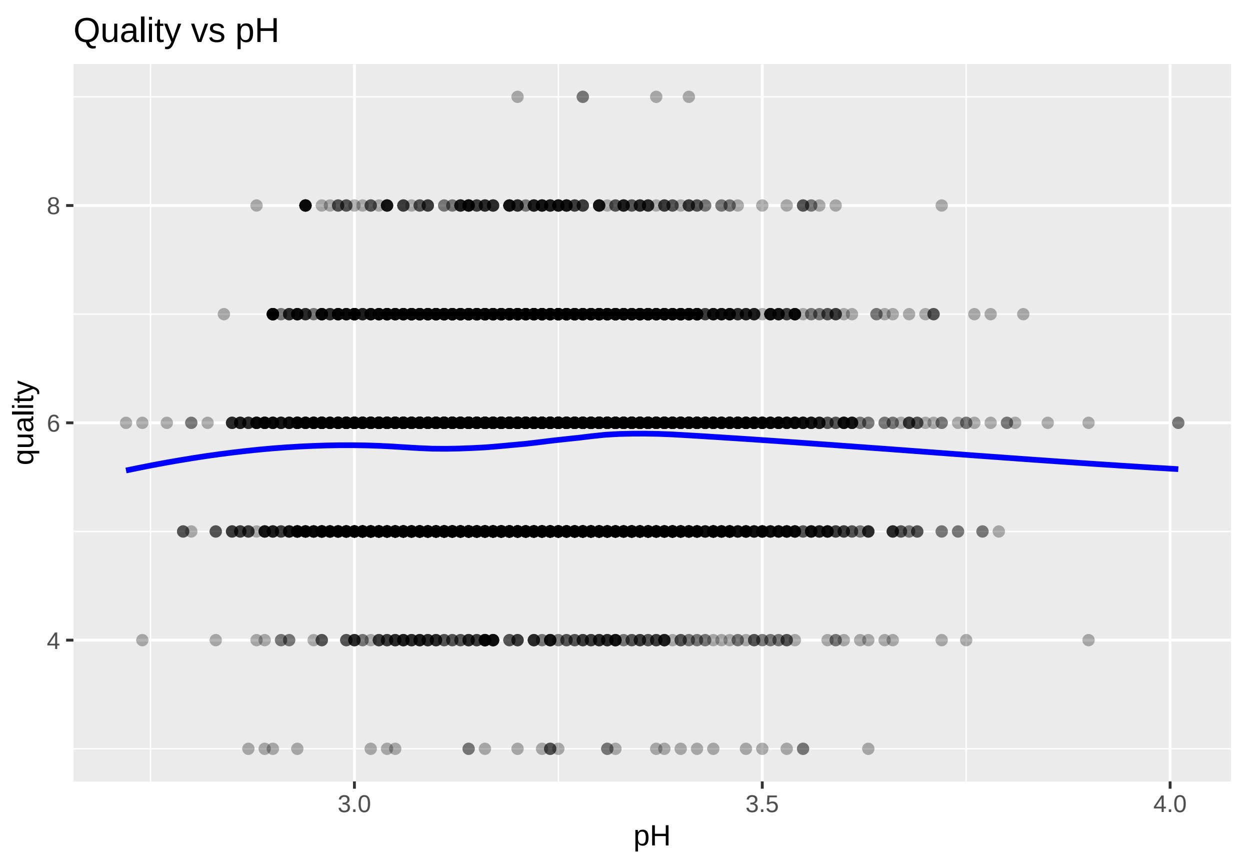
```
## `geom_smooth()` using formula = 'y ~ x'
```



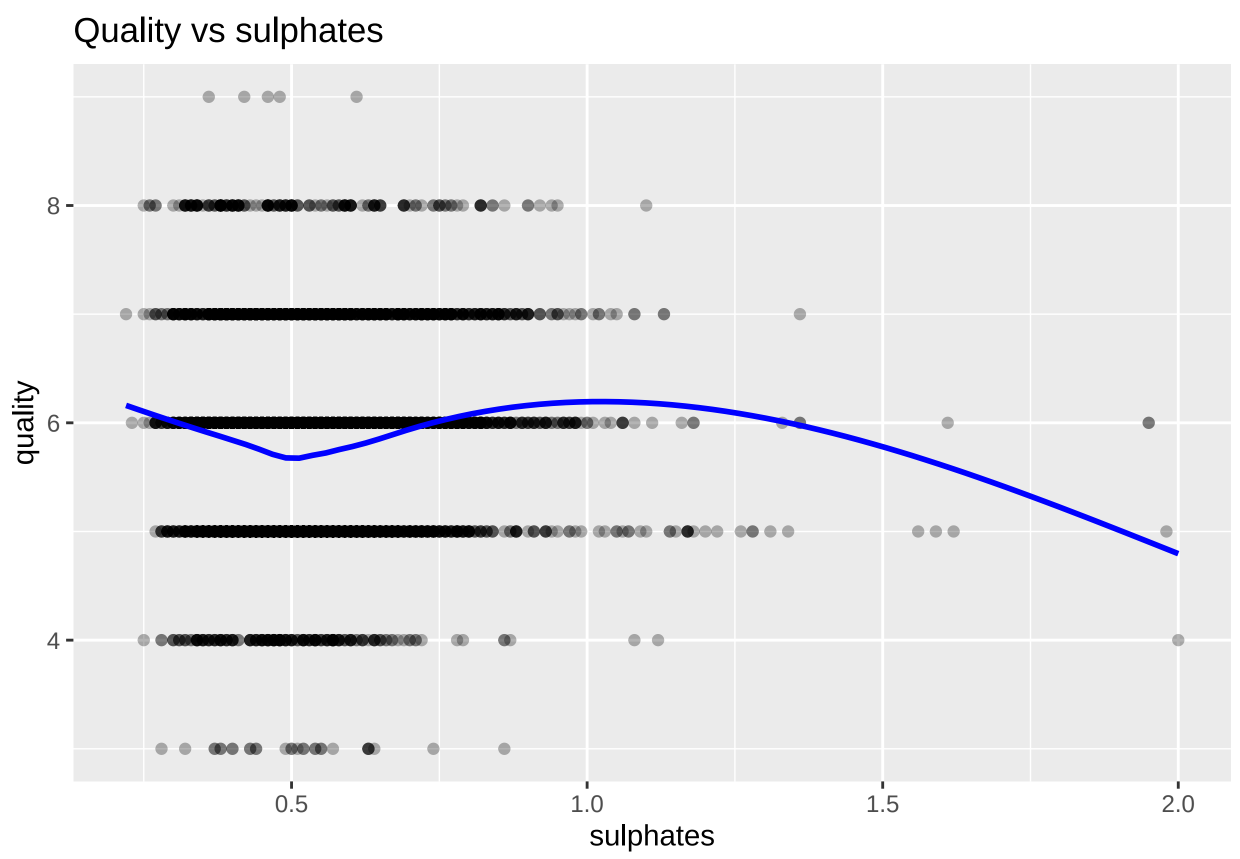
```
## `geom_smooth()` using formula = 'y ~ x'
```



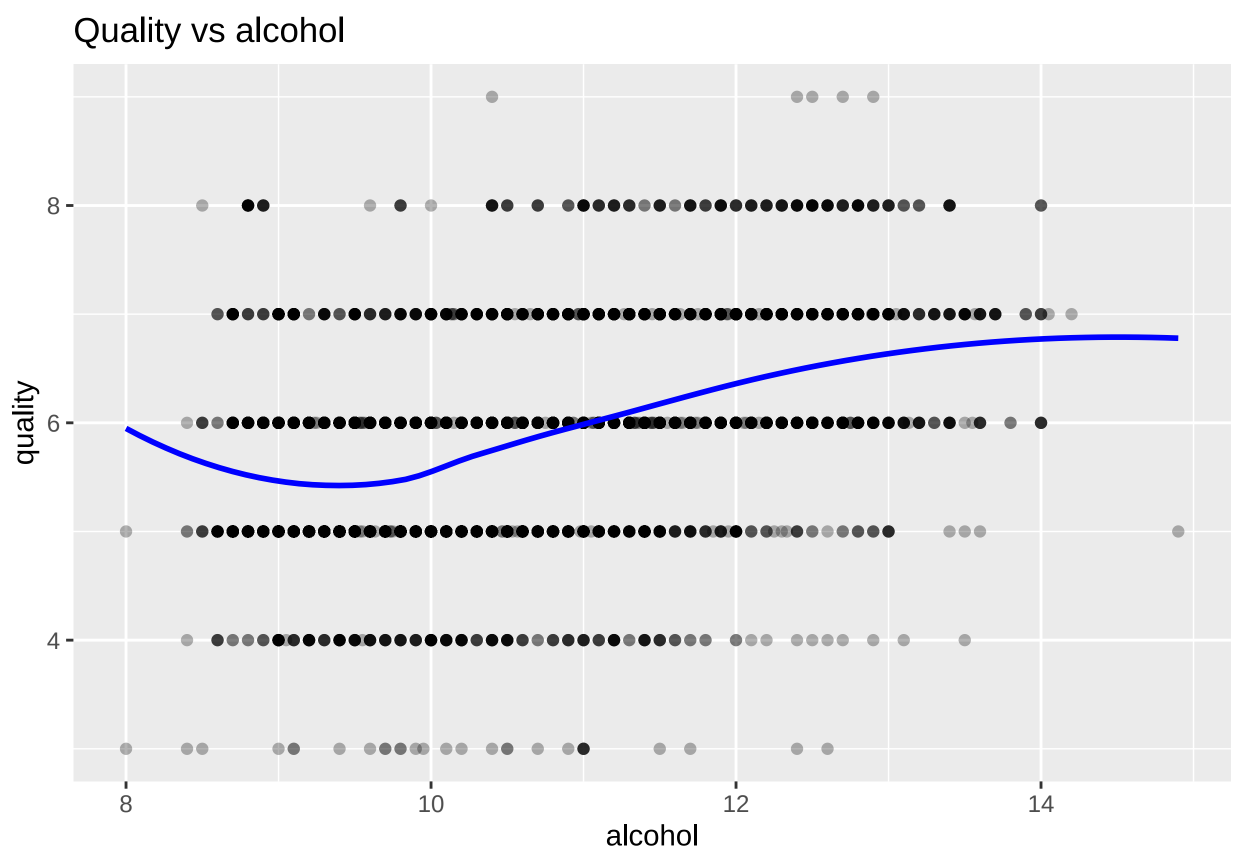
```
## `geom_smooth()` using formula = 'y ~ x'
```



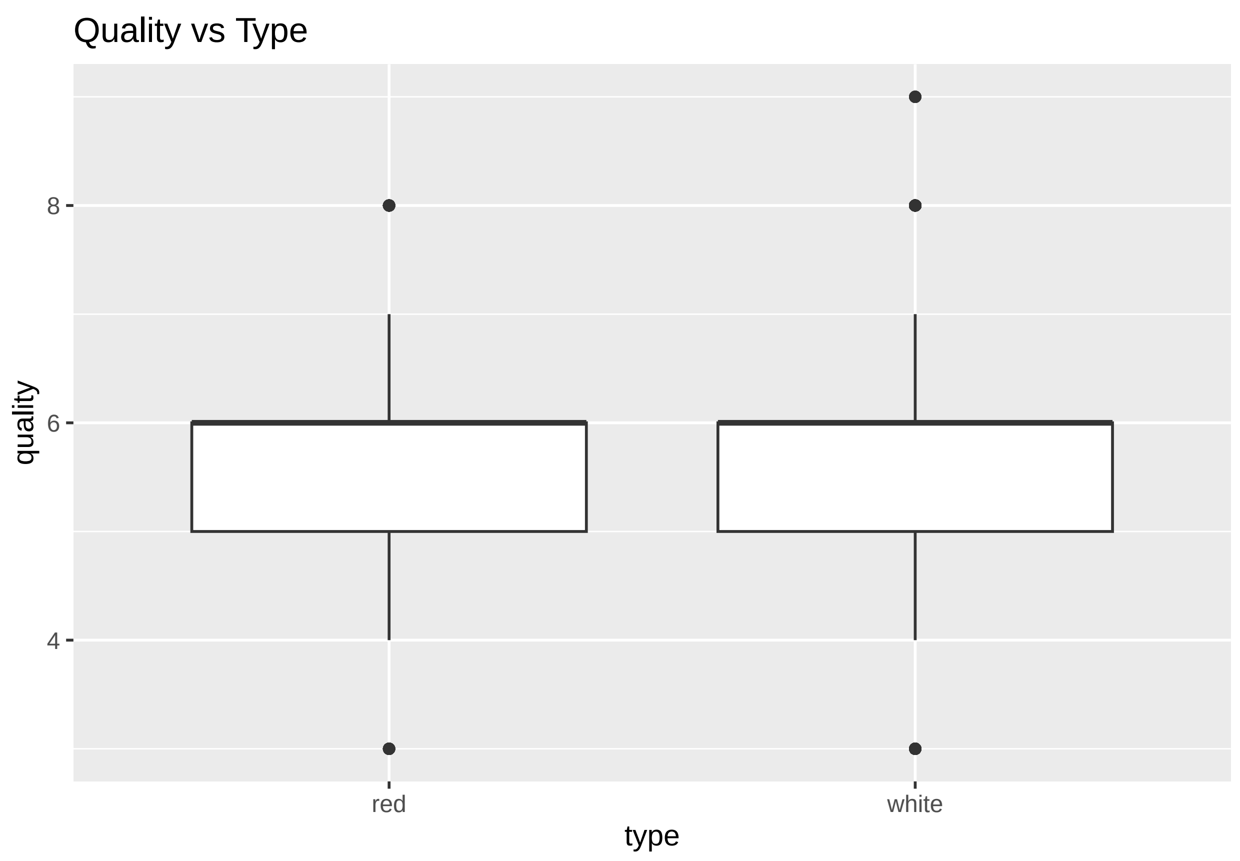
```
## `geom_smooth()` using formula = 'y ~ x'
```



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



```r
# Categorical: type
ggplot(wineData, aes(x = type, y = quality)) +
  geom_boxplot() +
  labs(title = "Quality vs Type")
```



Comparing RMSE using 10-Fold Cross Validation for Linear Regression, GAM1, GAM2, KNN, and Lasso

```
# Get data
wineData <- read.csv("~/Downloads/wine-quality-white-and-red.csv")

wineData$type <- as.factor(wineData$type)

# Split to training and testing data
set.seed(1)
train = sample(1:nrow(wineData), 0.7 * nrow(wineData))
train_data = wineData[train, ]
test_data = wineData[-train, ]

# Normalize function for KNN
normalize <- function(x) {
  return((x - mean(x)) / sd(x))
}


wine <- wineData

# Normalize numeric columns except quality
all_columns <- names(wine)
```

```r
columns_to_normalize <- all_columns[all_columns != "quality" & sapply(wine, is.numeric)]
wine[columns_to_normalize] <- lapply(wine[columns_to_normalize], normalize)

# Convert factor
wine$type <- as.numeric(factor(wine$type))

# Setup cross-validation
k <- 10
folds <- sample(1:k, nrow(wine), replace = TRUE)

# Empty vectors to store RMSE
rmse_linear <- rmse_gam1 <- rmse_gam2 <- rmse_knn <- rmse_lasso <- numeric(k)

for (i in 1:k) {
  ktrain_data <- wine[folds != i, ]
  ktest_data <- wine[folds == i, ]

  # Linear Regression
  lm_model <- lm(quality ~ ., data = ktrain_data)
  linear_preds <- predict(lm_model, newdata = ktest_data)
  rmse_linear[i] <- sqrt(mean((linear_preds - ktest_data$quality)^2))

  # GAM Model 1 (smooth all variables)
  gam_model1 <- gam(quality ~
                     s(fixed.acidity) +
                     s(volatile.acidity) +
                     s(citric.acid) +
                    s(chlorides) +
                     s(residual.sugar) +
                    s(free.sulfur.dioxide) +
                     s(total.sulfur.dioxide) +
                     s(density) +
                     s(sulphates) +
                     s(alcohol) + s(pH), data = ktrain_data, select=TRUE)
    gam_preds1 <- predict(gam_model1, newdata = ktest_data)
    rmse_gam1[i] <- sqrt(mean((gam_preds1 - ktest_data$quality)^2))

    # GAM Model 2 (smooth most variables)
    gam_model2 <- gam(quality ~
                        fixed.acidity +
                     volatile.acidity +
                     s(citric.acid) +
                     residual.sugar +
                     s(chlorides) +
                     free.sulfur.dioxide +
                     total.sulfur.dioxide +
                     density +
                     s(sulphates) +
                     s(alcohol) + s(pH),
                      data = ktrain_data)
    gam_preds2 <- predict(gam_model2, newdata = ktest_data)
    rmse_gam2[i] <- sqrt(mean((gam_preds2 - ktest_data$quality)^2))
  # KNN
  train_knn <- ktrain_data
  test_knn <- ktest_data
  train_knn$quality <- as.factor(train_knn$quality)
  test_knn$quality <- as.factor(test_knn$quality)

  training_X <- dplyr::select(train_knn, -quality)
  testing_X <- dplyr::select(test_knn, -quality)

  knn_preds <- knn.reg(train = training_X, test = testing_X, y = as.numeric(train_knn$quality), k = 7)$
  rmse_knn[i] <- sqrt(mean((knn_preds - as.numeric(test_knn$quality))^2))

  # Lasso
  x_train <- as.matrix(dplyr::select(ktrain_data, -quality))
  y_train <- ktrain_data$quality
  x_test <- as.matrix(dplyr::select(ktest_data, -quality))

  lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1)
  best_lambda <- lasso_cv$lambda.min
  lasso_preds <- predict(lasso_cv, s = best_lambda, newx = x_test)
  rmse_lasso[i] <- sqrt(mean((lasso_preds - ktest_data$quality)^2))
}

# Average RMSE across folds
results <- data.frame(
  Model = c("Linear", "GAM1", "GAM2", "KNN", "Lasso"),
  RMSE = c(mean(rmse_linear), mean(rmse_gam1), mean(rmse_gam2), mean(rmse_knn), mean(rmse_lasso))
)


print(results)
```

```
##    Model      RMSE
## 1 Linear 0.7336196
## 2   GAM1 0.7302731
## 3   GAM2 0.7287078
## 4    KNN 0.6893987
## 5  Lasso 0.7336021
```