Question 1

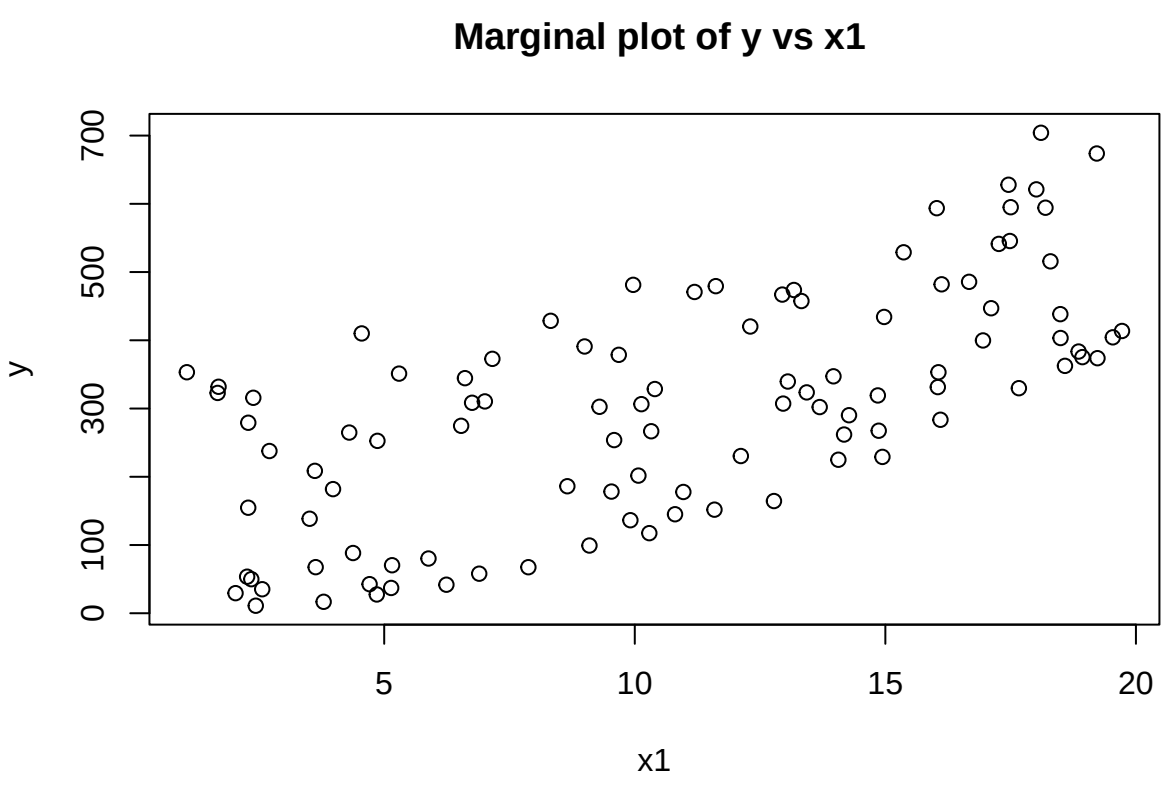
Gifty Osei

2024-11-01

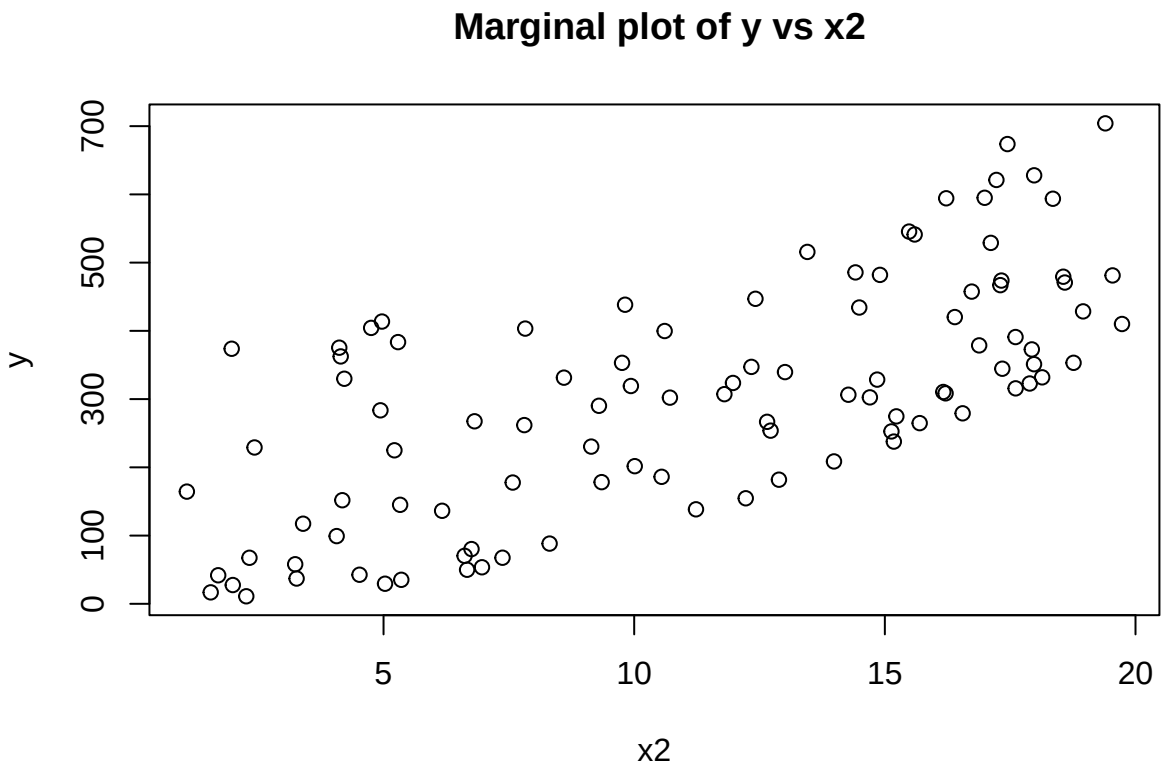
1b.

In general, a nonlinear trend can look fairly linear when data contain noise. For example, the following code plots data generated from $y = f(x1, x2) = x_1^2 + x_2^2 + \epsilon$, where $\epsilon \sim N(0, 0.02^2)$.

```
set.seed(390)
n <- 100
# simulate 20 values from uniform(0,1)
x1 = runif(n,1,20)
x2= runif(n,1,20)

y = x1^2+x2*x2+rnorm(20,sd=0.02)
plot(y~x1, main = " Marginal plot of y vs x1", xlab = "x1", ylab = "y")
```




```
plot(y~x2,main = " Marginal plot of y vs x2", xlab = "x2", ylab = "y")
```



We can see clearly form the marginal plots showing a linear trend but the overall model that generated the y values is nonlinear in x. We can see that from the joint marginal plot that the are jointly nonlinear. We can see that the joint marginal plot shows and captured the quadratic pattern of the form of Y

```
library(plotly)
plot_ly(x = ~x1, y = ~x2, z = ~y, type = "scatter3d", mode = "markers",
        marker = list(size = 3)) %>%
  layout(scene = list(xaxis = list(title = "x1"),
                      yaxis = list(title = "x2"),
                      zaxis = list(title = "y")))
```

Question 2

2a

```
# simulate data
n <- 100
x <- runif(n, 1, 10)
beta_0 <- 3
beta_1 <- 2
sigma <- 3

# Generate y values based on the original model
epsilon <- rnorm(n, mean = 0, sd = sigma * x)
y <- beta_0 + beta_1 * x + epsilon

# transformations
y_prime <- y / x
x_prime <- 1 / x

# Check the variance of y_prime
var_y_prime <- var(y_prime)
print(sqrt(var_y_prime))
```

```
## [1] 3.430146
```

We can see that variance of $y^{'}$ is very close to the initial σ

2c

This is equivalent to OLS estimate because using the weights on the original data is like doing a $\frac{y}{x}$ transformation from part a. We can see from the coefficients that there is not much change although, intercept from OLS is now the slope for WLS and slope now intercept.

```
#  wls model
fit_wls <- lm(y ~ x, weights = 1 / x^2)

# OLS model
fit_ols_trans <- lm(y_prime ~ x_prime)

# Compare the coefficients
summary(fit_wls)$coefficients
```

```
##             Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 4.089974  2.2529295 1.815402 0.0725203966
## x           2.193610  0.6439473 3.406505 0.0009555446
```

```
summary(fit_ols_trans)$coefficients
```

```
##             Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 2.193610  0.6439473 3.406505 0.0009555446
## x_prime     4.089974  2.2529295 1.815402 0.0725203966
```

Question 3

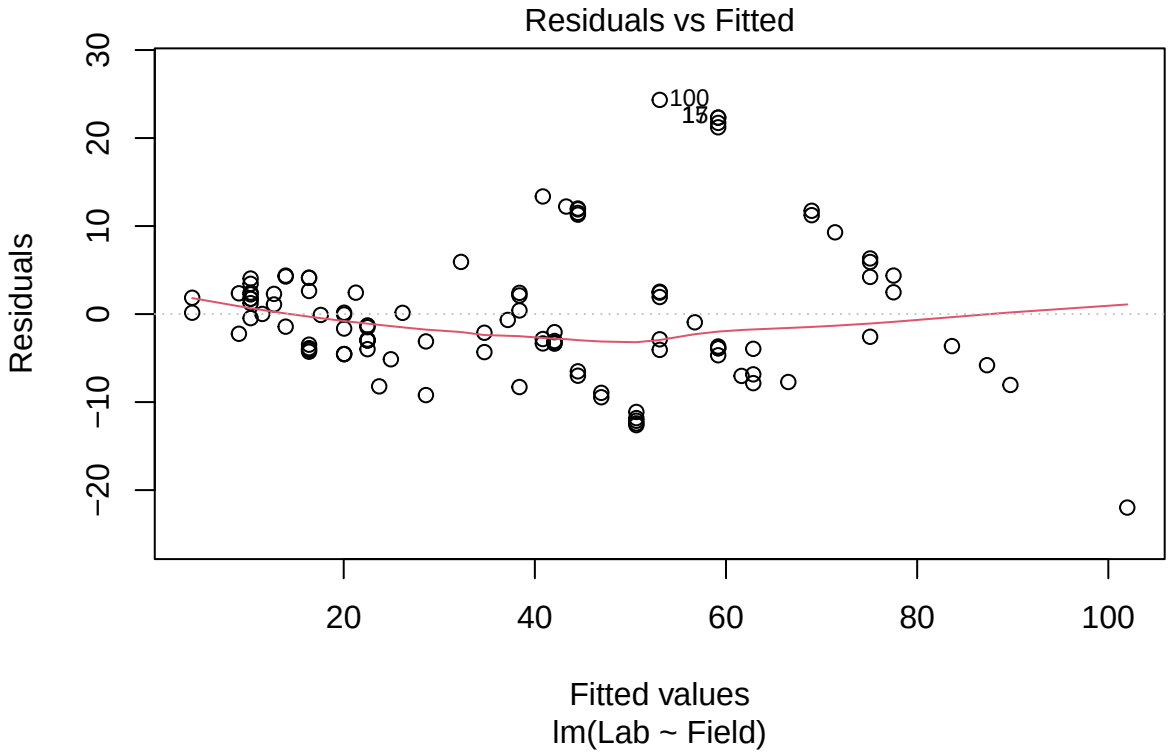
```
library(faraway)
pander::pander(head(pipeline))
```

Field	Lab	Batch
18	20.2	1
38	56	1
15	12.5	1
20	21.2	1
18	15.5	1
36	39	1

3a.

Fitting a linear model by regressing Lab on Field as;

```
fit_lm <- lm( Lab ~ Field, data = pipeline)


plot(fit_lm, which = 1)
```



We can see from the residual vs fitted plot that linearity assumption is checked but the residuals have a funnel like pattern. As x increases the variability increases.

3b

```
i = order(pipeline$Field)
npipe = pipeline[i,]
ff = gl(12,9)[-108]
meanfield = unlist(lapply(split(npipe$Field,ff),mean))
varlab = unlist(lapply(split(npipe$Lab,ff),var))
```

```
# Remove last point
meanfield <- meanfield[-length(meanfield)]
varlab <- varlab[-length(varlab)]

#  Log transform
log_meanfield <- log(meanfield)
log_varlab <- log(varlab)

#  model
var_model <- lm(log_varlab ~ log_meanfield)
tidy_var <- tidy(var_model)
kable(tidy_var, caption = "Log Transform Estimates")
```

Table 2: Log Transform Estimates

term	estimate	std.error	statistic	p.value
(Intercept)	-1.935167	1.0929482	-1.770594	0.1104030
log_meanfield	1.670723	0.3295636	5.069502	0.0006723

```
# Extract coefficients for a0 and a1
## take exp because of the log
a0 <- exp(coef(var_model)[1])

## slope - a1
a1 <- coef(var_model)[2]

#   WLS as the inverse
predicted_variance <- a0 * (pipeline$Field ^ a1)
weights <- 1 / predicted_variance

# Perform WLS regression of Lab on Field using the calculated weights
wls_model <- lm(Lab ~ Field, data = pipeline, weights = weights)
tidy_wls <- tidy(wls_model)
kable(tidy_wls, caption = "Calculated Weights")
```

Table 3: Calculated Weights

term	estimate	std.error	statistic	p.value
(Intercept)	-1.055296	0.6976531	-1.512637	0.1333761
Field	1.189627	0.0340053	34.983536	0.0000000

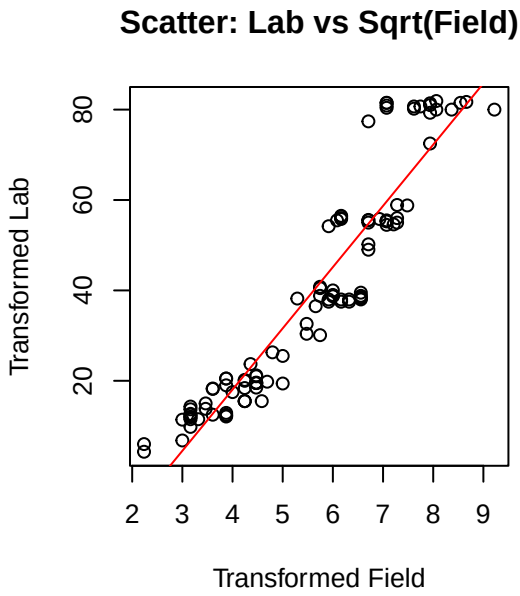
3c

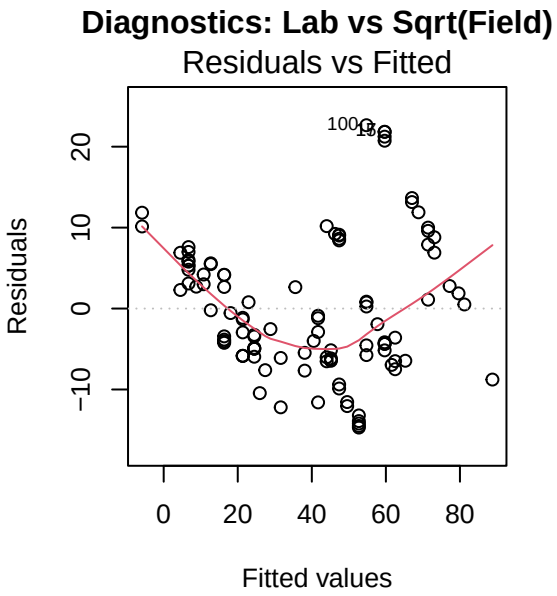
```
## function to automate
linear_const_var_trans <- function(Lab_trans, Field_trans, label) {

#  model with specified transformations
  model <- lm(Lab_trans ~  Field_trans)

# Plot data and model diagnostics
  par(mfrow = c(2, 2))
  plot(Field_trans, Lab_trans, main = paste("Scatter:", label),
       xlab = "Transformed Field", ylab = "Transformed Lab")
  abline(model, col = "red")

  # Plot diagnostics
  plot(model, which = 1:2, main = paste("Diagnostics:", label))
  tidy_model <- tidy(model)
  kable(tidy_model, caption = "Summary of Model", digits = 4)
}

# Different 3 transformations given
sqrt_lab <- sqrt(pipeline$Lab)
sqrt_field <- sqrt(pipeline$Field)
log_lab <- log(pipeline$Lab)
log_field <- log(pipeline$Field)
inv_lab <- 1 / pipeline$Lab
inv_field <- 1 / pipeline$Field

# Apply transformations and evaluate each
linear_const_var_trans(pipeline$Lab,sqrt_field, "Lab vs Sqrt(Field)")
```

Table 4: Summary of Model

term	estimate	std.error	statistic	p.value
(Intercept)	-36.1385	2.8573	-12.6477	0
Field_trans	13.5486	0.4931	27.4771	0

```
linear_const_var_trans(log_lab,pipeline$Field, "Log Lab vs Field")
```





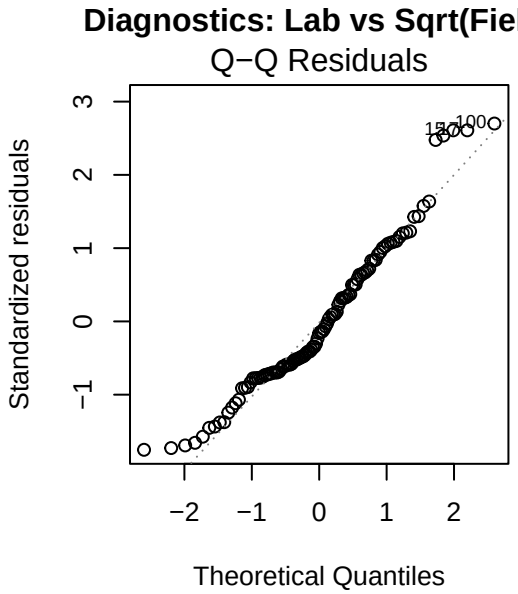
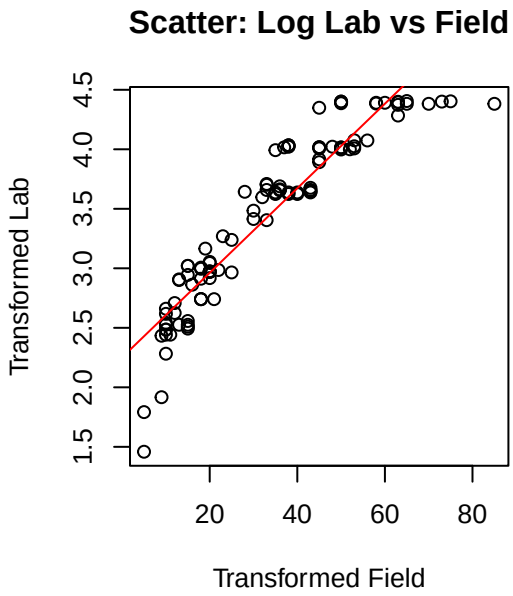
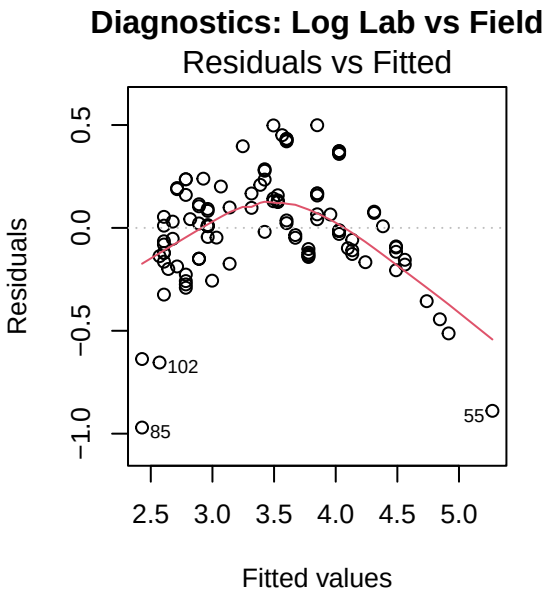


Table 5: Summary of Model

term	estimate	std.error	statistic	p.value
(Intercept)	2.2513	0.0523	43.0718	0
Field_trans	0.0355	0.0014	26.0621	0

```
linear_const_var_trans(sqrt_lab, sqrt_field, "Square Root Transformation")
```





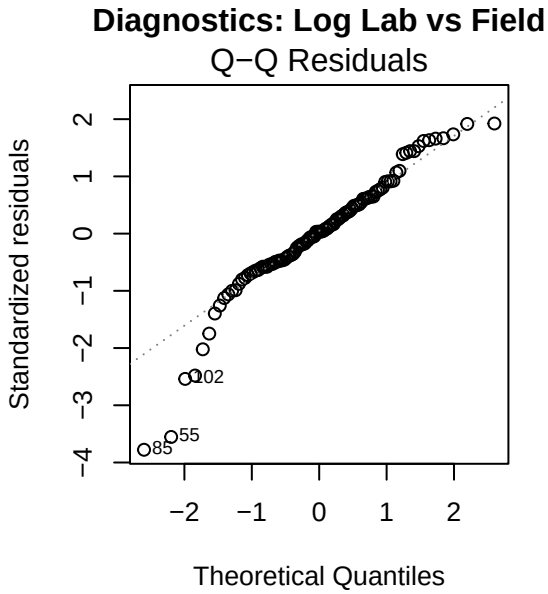
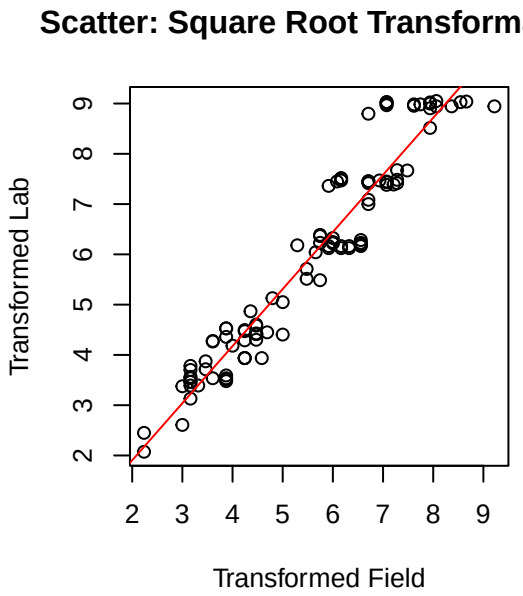
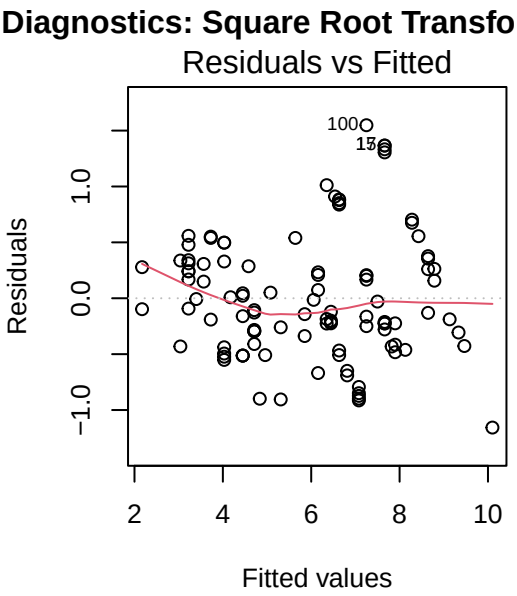


Table 6: Summary of Model

term	estimate	std.error	statistic	p.value
(Intercept)	-0.3677	0.1881	-1.9545	0.0533
Field_trans	1.1355	0.0325	34.9729	0.0000

```
linear_const_var_trans(log_lab, log_field, "Log Transformation")
```





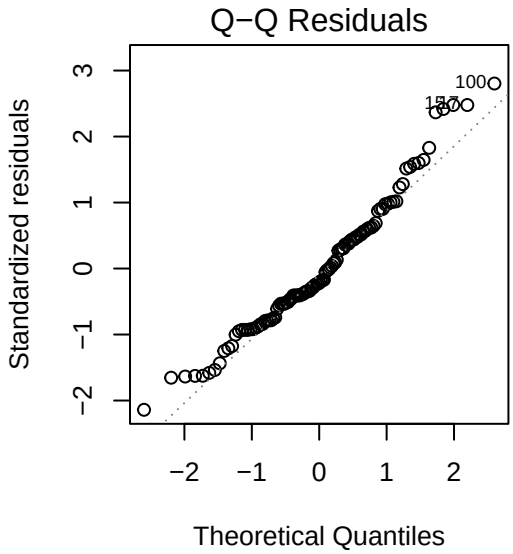
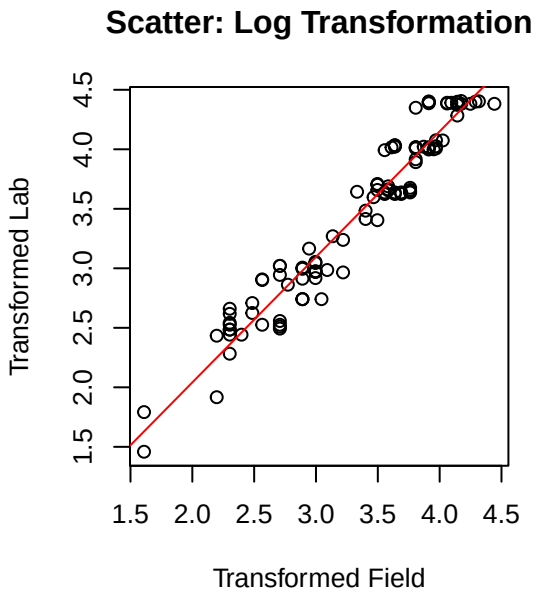
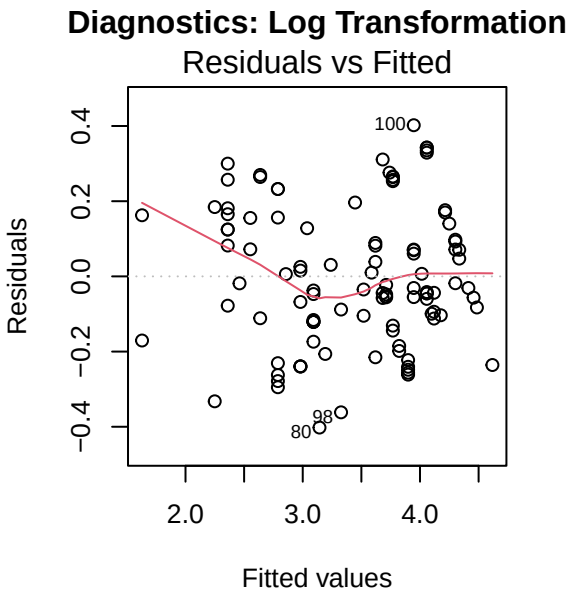


Table 7: Summary of Model

term	estimate	std.error	statistic	p.value
(Intercept)	-0.0685	0.0931	-0.7361	0.4633
Field_trans	1.0548	0.0274	38.4572	0.0000

```
linear_const_var_trans(inv_lab, inv_field, "Inverse Transformation")
```





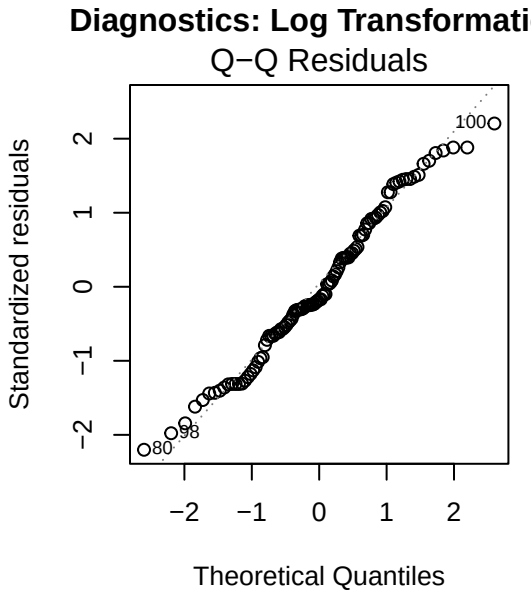
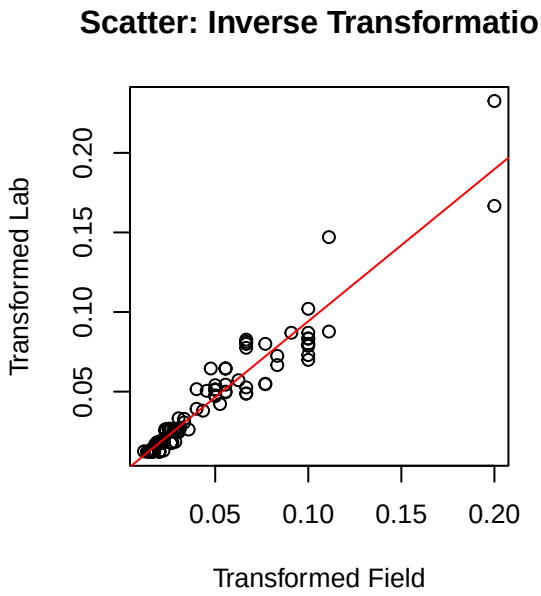
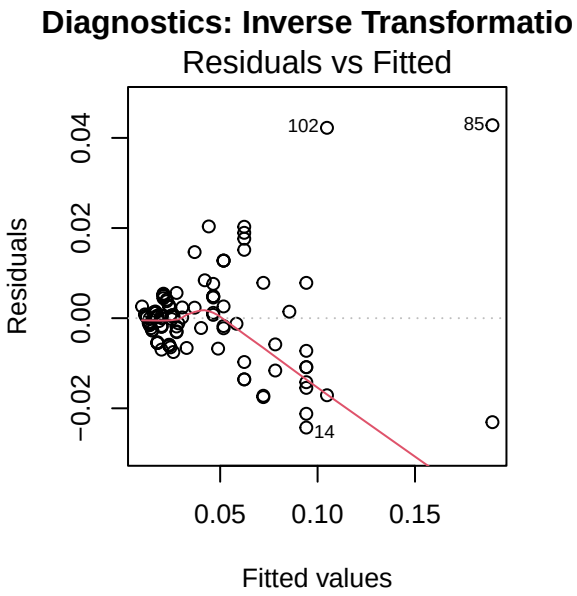


Table 8: Summary of Model

term	estimate	std.error	statistic	p.value
(Intercept)	-0.0013	0.0016	-0.8163	0.4162
Field_trans	0.9554	0.0288	33.1167	0.0000

```
linear_const_var_trans(sqrt_lab, log_field, "Sqrt(Lab), Log(Field)")
```





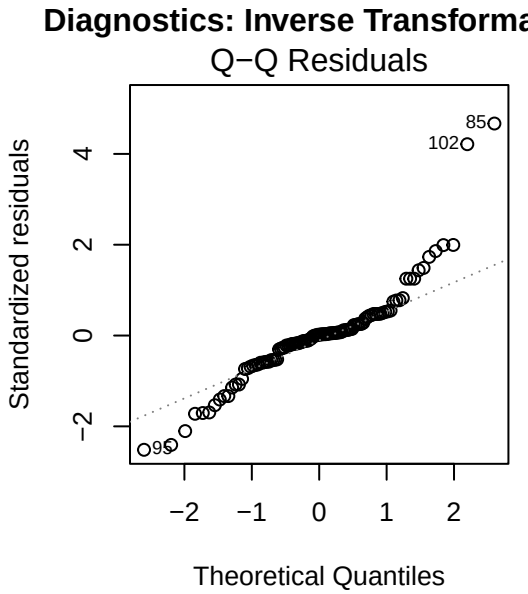
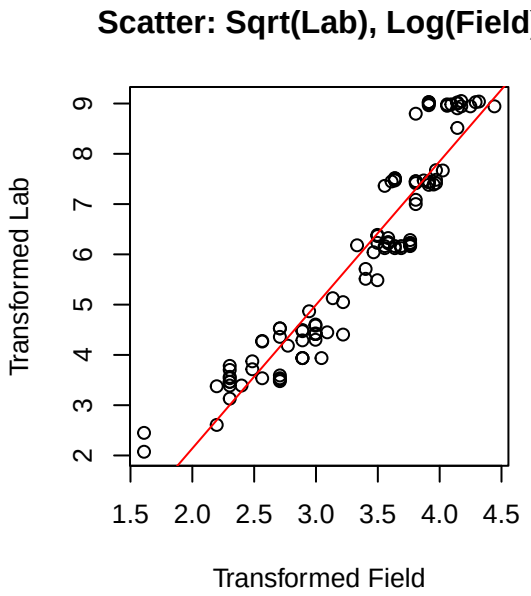
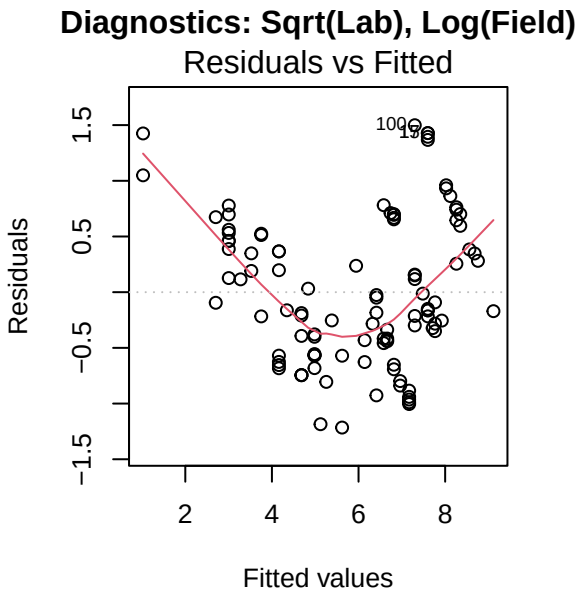


Table 9: Summary of Model

term	estimate	std.error	statistic	p.value
(Intercept)	-3.5714	0.3320	-10.7572	0
Field_trans	2.8555	0.0979	29.1790	0

```
linear_const_var_trans(log_lab, sqrt_field, "Log(Lab), Sqrt(Field)")
```





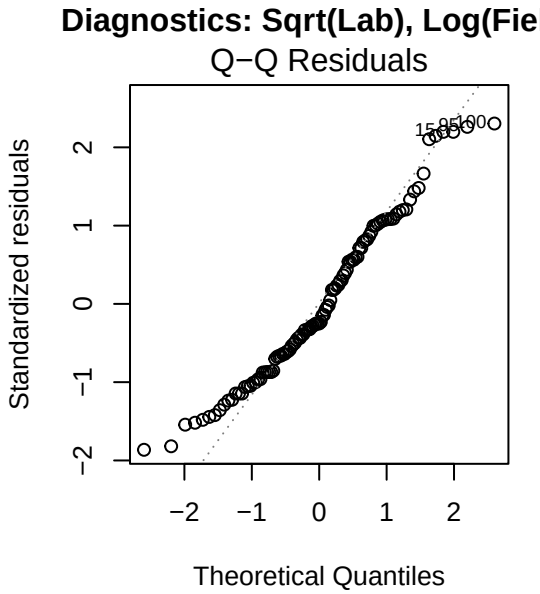
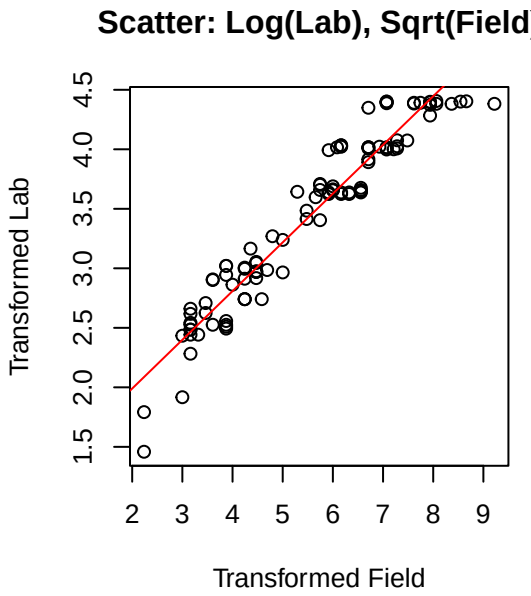
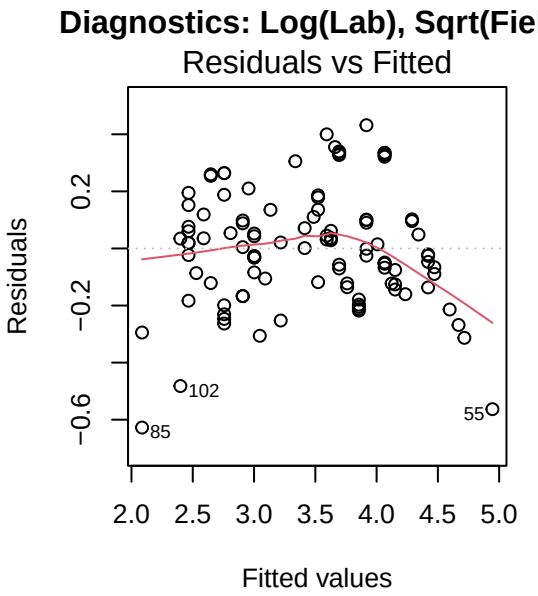
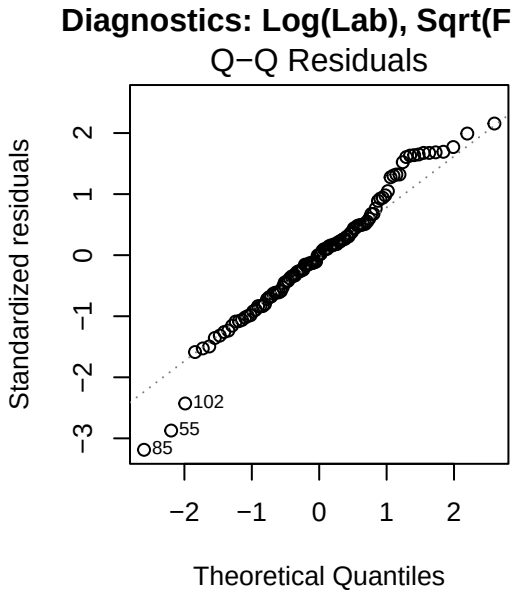


Table 10: Summary of Model

term	estimate	std.error	statistic	p.value
(Intercept)	1.1709	0.0682	17.1583	0
Field_trans	0.4094	0.0118	34.7619	0







Based on these diagnostics plots, the best transformation which shows a linear relationship, no clear pattern and with constant variance is the log transformation.

The Log transformation on both lab and field is the most effective with smallest residual error and also the range scale on the plots is very small.

Inverse transformation also had small range on the residual plot and is less variable.

Square root transforms performs poorly.