HOMEWORK 3

Yiqiao Yin [YY2502]

Contents

PROBLEM 1 (Ch2, Q2) **1**

PROBLEM 2 (Ch2, Q66) **1**

PROBLEM 3 (Ch2, Q68) **5**

PROBLEM 4 (Ch3, Q9) **6**

PROBLEM 5 (Ch3, Q16) **7**

PROBLEM 6 (Ch3, Q31) **11**

PROBLEM 1 (Ch2, Q2)

As the question stated: $H_0 : \beta_1 \leq 0$ and $H_1 : \beta_1 > 0$. If the conclusion is null, it implies that there is no sufficient evidence to say that the linear coefficient β_1 is positive. This does not imply that there is no linear association at all. In other words, there may still be some linear association. It is just we are unsure that it is positive.

PROBLEM 2 (Ch2, Q66)

Given linear model

$$\mathbb{E}(Y) = 20 + 4X$$

where $\epsilon_i \sim \mathcal{N}(0, 25)$.

1) Generate five normal random numbers

```
set.seed(2020)
epsilon = rnorm(5, 0, 5); epsilon
```

```
## [1]   1.884861   1.507742  -5.490116  -5.652030 -13.982672
```

From the above randomness generated and give ndata X, we can compute Y_i:

```r
X <- c(4, 8, 12, 16, 20)
Y = 20 + 4*X + epsilon
Y
```

```
## [1] 37.88486 53.50774 62.50988 78.34797 86.01733
```

Using above data of Y and X, let us run regression model

```r
LM1 <- lm(Y~X); summary(LM1)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      1      2      3      4      5
## -1.548  1.965 -1.144  2.584 -1.857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   27.322      2.547   10.73 0.001732 **
## X              3.028      0.192   15.77 0.000554 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.428 on 3 degrees of freedom
## Multiple R-squared:  0.9881, Adjusted R-squared:  0.9841
## F-statistic: 248.7 on 1 and 3 DF,  p-value: 0.0005541
```

and we obtain regression model to be

$$Y = 27.322 + 3.028 \cdot X$$

This means when $X = 10$, we can compute estimate of Y:

```r
yhat = 27.322 + (3.182446 * 10); yhat
```

```
## [1] 59.14646
```

and the answer is 57.602; the 95% confidence interval is computed based on the following critical value:

```r
critical = qt(1-0.05/2, 5-2); print(paste0("critical-Value = ", critical))
```

```
## [1] "critical-Value = 3.18244630528371"
```

and since

```r
n = length(Y)
sse <- sum((Y - LM1$fitted.values)^2)
mse <- sse / (n - 2)
se <- sqrt(
```

```
  sum((Y - LM1$fitted.values)^2) /
    (n - 2)) *
  sqrt(1 / n + (10 - mean(X))^2 / sum((10 - mean(X))^2))
SD_Y <- se; print(paste0("Standard Error for Y_hat: ", SD_Y))
```

```
## [1] "Standard Error for Y_hat: 2.66001843997834"
```

we have

$$95\%\ \text{CI} = [59.146 - 3.18 \cdot 2.66, 59.146 + 3.18 \cdot 2.66] = [50.6872, 67.6048]$$

2) Let us repeat part 1) 200 times:

```
# Define
Ys <- c()
betas = c()
CIs <- rbind()

set.seed(2021)
for (i in 1:200) {
  epsilon = rnorm(5, 0, 5); epsilon
  Y = 20 + 4*X + epsilon
  currLM <- lm(Y~X); summary(currLM)
  n = length(Y)
  sse <- sum((Y - currLM$fitted.values)^2)
  mse <- sse / (n - 2)
  se <- sqrt(
    sum((Y - currLM$fitted.values)^2) /
      (n - 2)) *
    sqrt(1 / n + (10 - mean(X))^2 / sum((X - mean(X))^2))
  currY = coef(summary(currLM))[1,1] + coef(summary(currLM))[2,1]*10

  betas <- c(betas, coef(summary(currLM))[2,1])
  Ys <- c(Ys, currY)
  CIs <- rbind(CIs, c(currY - critical*se, currY + critical*se))
}

# View
print(paste0("Summary as below:"))
```

```
## [1] "Summary as below:"
```

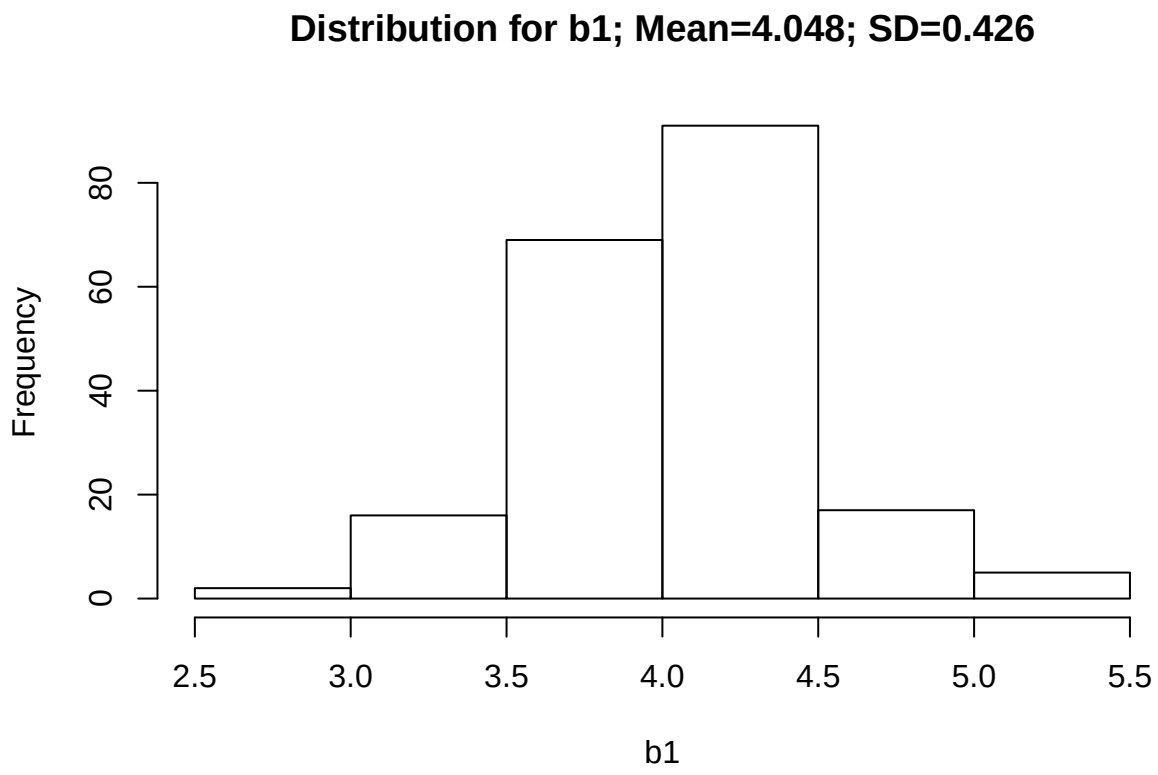
```
summary(betas)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.748   3.820   4.062   4.048   4.283   5.314
```

The above we provided a summary table with five summary statistics for intercept and coefficient, that is, β_0 and β_1 for 200 linear models, respectively.

3) Make a frequency table for β_1. We compute that the mean is approximately 4 and the standard deviation of β_1 is approximately 0.4. Let us plot the following histogram.

```
b1 = betas
hist(
  b1,
  main=paste0(
    "Distribution for b1; Mean=",
    round(mean(b1), 3),
    "; SD=", round(sd(b1), 3)),
  xlab = "b1")
```



The result is very close to the approximates from part 1). It is also consistent with theoretical results because we are given $\sigma^2 = 25$ for ϵ_i. We can also compute SSX

```
ssx = sum((X - mean(X))^2); ssx
```

```
## [1] 160
```

and we obtain SSX $= 160$. This means that we can compute $\sqrt{25/160} = 0.3953 \approx 0.40$. This is very close to simulated results.

4) Let us compute

```
print("Recall from part 1) that our estimated Yhat is:")
```

```
## [1] "Recall from part 1) that our estimated Yhat is:"
```

```r
print(yhat)
```

```
## [1] 59.14646
```

```r
print("Let us compute the proportion of intervals that this Yhat falls within:")
```

```
## [1] "Let us compute the proportion of intervals that this Yhat falls within:"
```

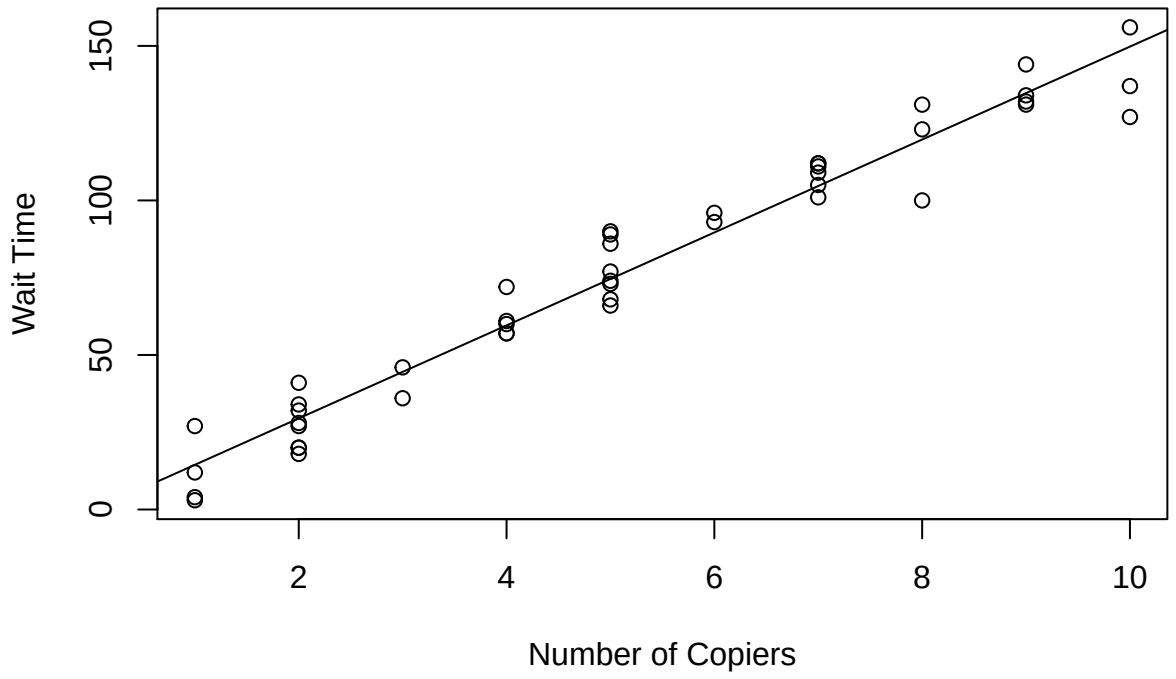
```r
print(sum(apply(t(CIs), 2, function(ci) {(yhat > ci[1]) && (yhat < ci[2])}))/200)
```

```
## [1] 0.95
```

PROBLEM 3 (Ch2, Q68)

Referring to data in Ch1, Q20, let us load the data here

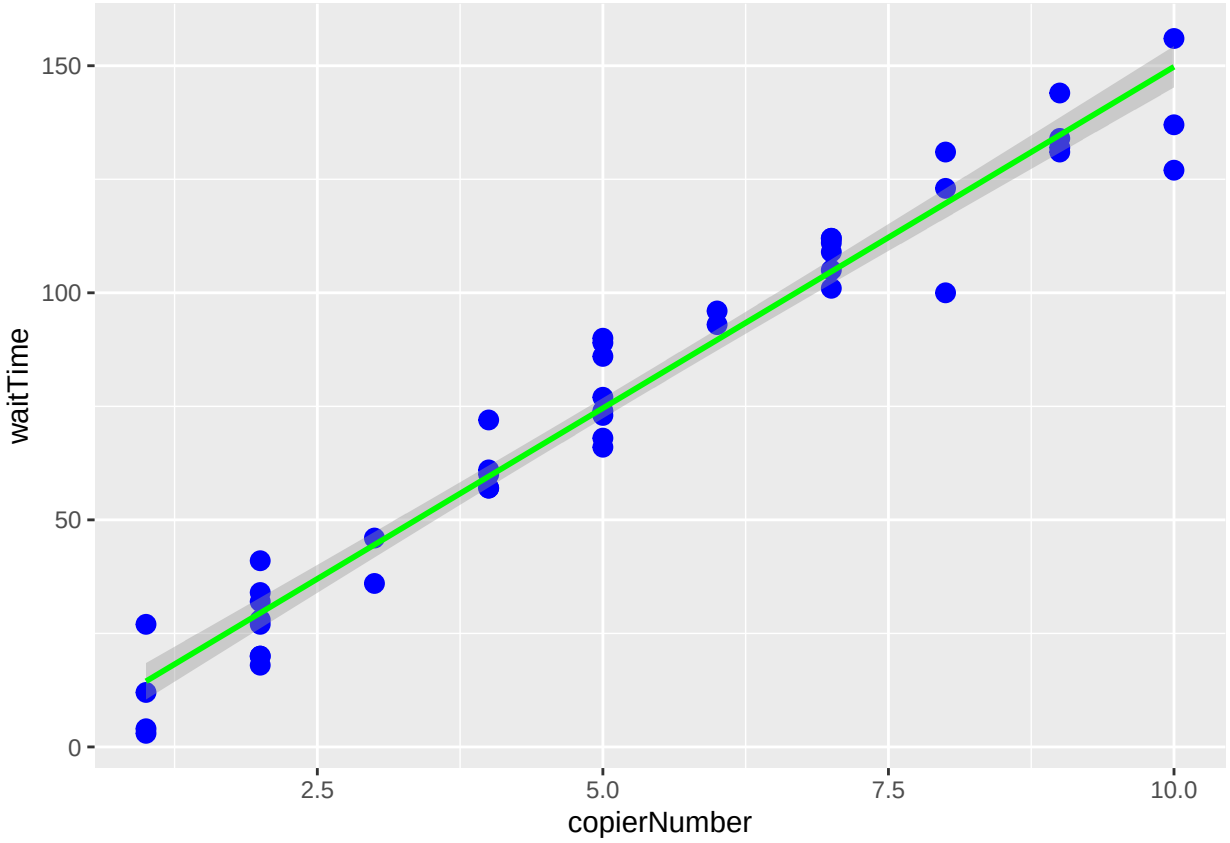
```r
setwd("C:/Users/eagle/OneDrive/Course/CU Stats/STATS GR6101 - Applied Statistics I/Data")
data = read.csv("CH01PR20.csv", header = FALSE)
colnames(data) <- c("waitTime", "copierNumber")
copierNumber = data[,2]
waitTime = data[,1]
copierLM = lm(waitTime~copierNumber)
plot(copierNumber, waitTime, xlab = "Number of Copiers", ylab = "Wait Time")
abline(copierLM)
```



THe linear model seems like to be a good fit of the data.

```
library(ggplot2)
ggplot(data, aes(x=copierNumber, y=waitTime)) +
  geom_point(color="blue", size = 3) +
  geom_smooth(method=lm, level=0.90, color="green")
```

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

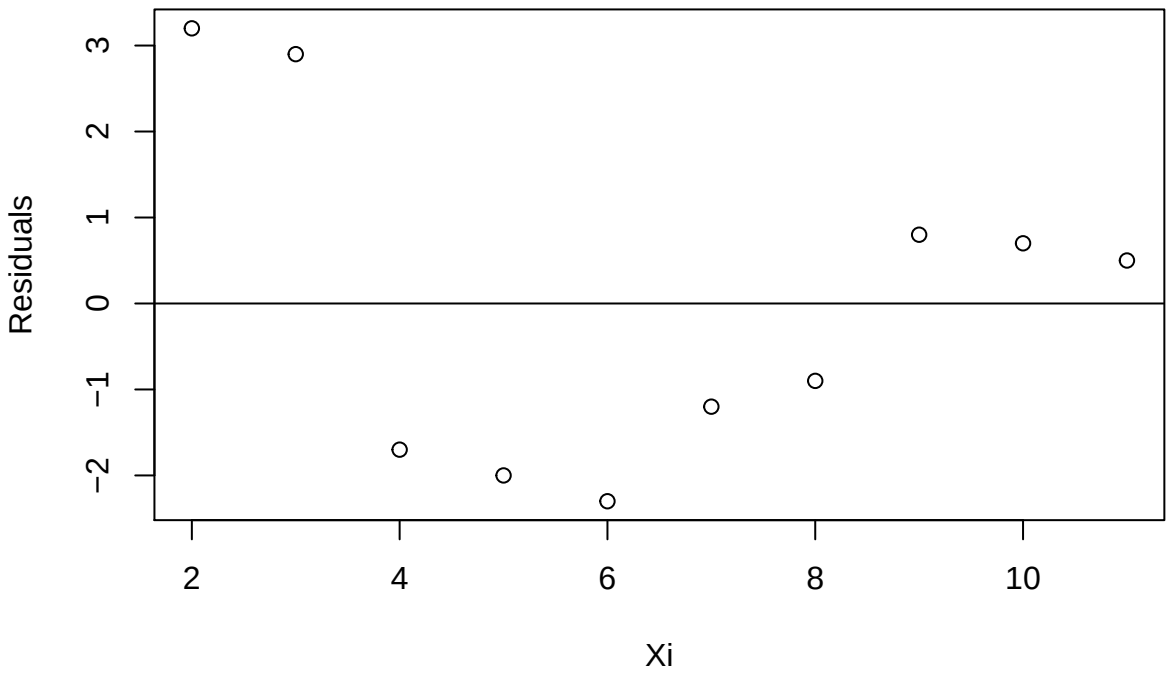


The model somewhat shows good fit and the samples fall on the plot with linear association which is captured by the model. But then we do observe a quite a few observations fall outside of the band, this means there could still be a lot of variance not captured by the model.

PROBLEM 4 (Ch3, Q9)

Let us load the data first.

```
X = c(2,3,4,5,6,7,8,9,10,11)
resi = c(3.2,2.9,-1.7,-2.0,-2.3,-1.2,-0.9,0.8,0.7,0.5)
plot(X, resi, xlab = "Xi", ylab = "Residuals")
abline(a=0, b=0)
```



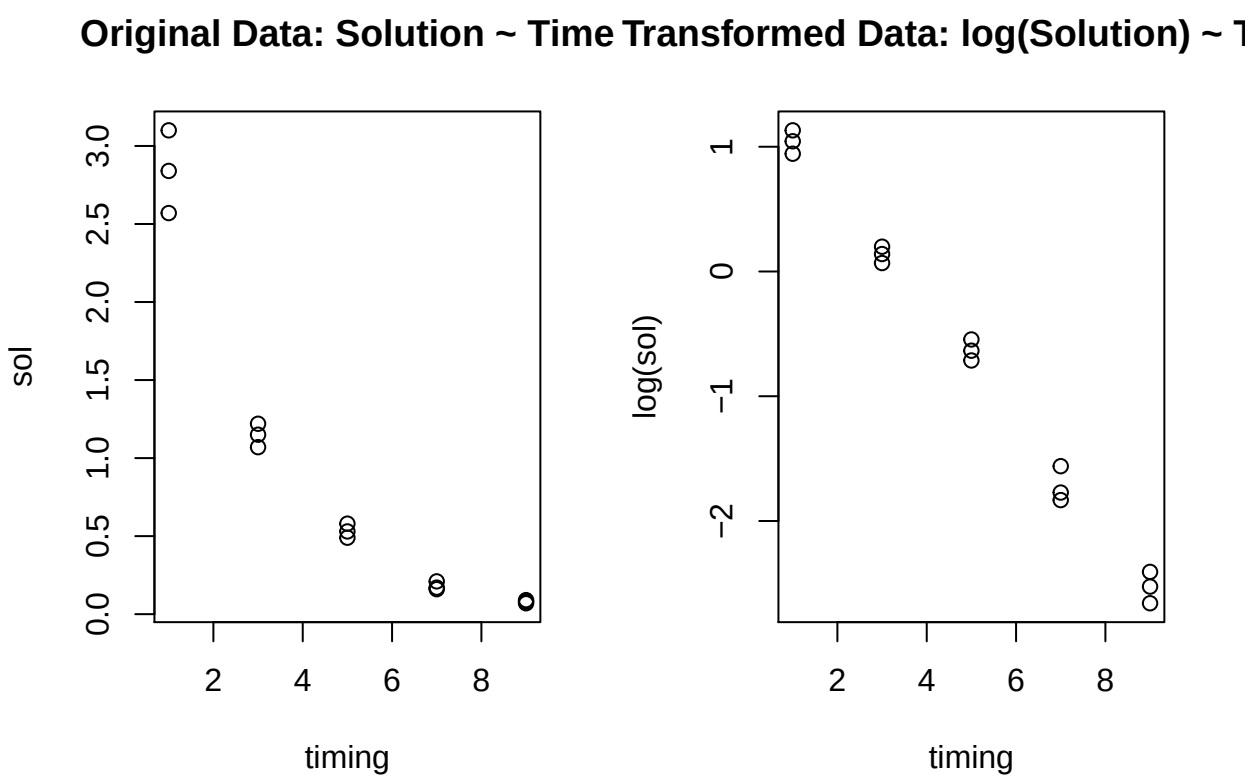
From reuslts above, it is not a very good fit. Moreover, we observe that when X is small there is a much larger residual than when X is larger. In other words, $\epsilon_i = Y_i - \hat{Y}_i$ might be better visualized if we transform using some sort of logarithm function. Though $\log(\cdot)$ is not the only method we may use, it is common practice to plot transformation and linear regression model to obtain visualization of goodness of fit as well as computing mean square error before we can say with conviction what the situation is in this data.

PROBLEM 5 (Ch3, Q16)

Let us load the data first:

```
setwd("C:/Users/eagle/OneDrive/Course/CU Stats/STATS GR6101 - Applied Statistics I/Data")
data = read.csv("CH03PR15.csv", header = FALSE)
sol = data[,1]
timing = data[,2]

# Plot
par(mfrow=c(1,2))
plot(timing, sol, main = "Original Data: Solution ~ Time")
plot(timing, log(sol), main = "Transformed Data: log(Solution) ~ Time")
```



1) With visualizations above, we present transformation that is the logarithm of the original data for Solution, i.e. Y. We can observe non-linear pattern from the original data (plot on the left). This can be "fixed" using logarithm function (plot on the right).

2) Box-Cox Transformation

```
library(MASS)

# Package
# print(paste0("From MASS package, we see that the most optimal lambda is the following: "))
# boxCoxResult = boxcox(sol~timing)
# lambda <- boxCoxResult$x[which.max(boxCoxResult$y)]
# print(paste0("Best lambda is ", lambda))

# Confirm
print(paste0("Let us confirm by using SSE."))
```

```
## [1] "Let us confirm by using SSE."
```

```
powerTransform <- function(y, lambda1, method = "boxcox") {
  boxcoxTrans <- function(x, lam1) {
    if (lam1 == 0L) {
      K2 = (prod(sol))^(1/length(sol))
      return(K2*log(y))
    } else {
```

```
      K2 = (prod(sol))^(1/length(sol))
      K1 = 1/(lam1*K2^(lam1-1))
      return(K1*(sol^lam1 - 1))
    }
  }

  switch(
    method,
    boxcox = boxcoxTrans(y, lambda1),
    tukey = y^lambda1
  )
}

for (ll in c(-.2, -.1, 0, .1, .2)) {
  currModel <- lm(powerTransform(sol, ll) ~ timing)
  print(paste0("lambda:", ll, "; SSE=", sum((powerTransform(sol, ll) - currModel$fitted.values)^2)))
}
```

```
## [1] "lambda:-0.2; SSE=0.123530470975412"
## [1] "lambda:-0.1; SSE=0.0650506677553841"
## [1] "lambda:0; SSE=0.0389730346883832"
## [1] "lambda:0.1; SSE=0.0439606177062632"
## [1] "lambda:0.2; SSE=0.0813179301952558"
```

We observe from above results that the SSE is at the lowest at 0.

3) Run linear model using $Y' := \log(Y)$ transformation:

```
logModel = lm(log10(sol)~timing)
summary(logModel)
```

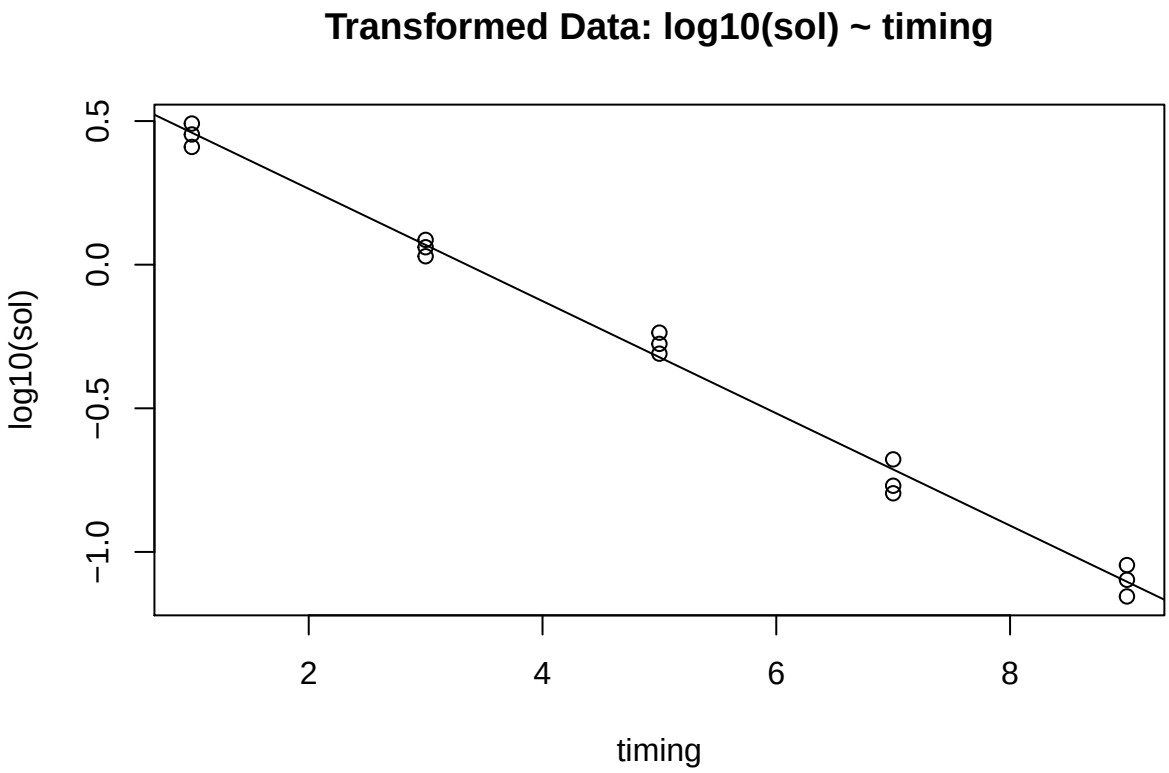
```
##
## Call:
## lm(formula = log10(sol) ~ timing)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.082958 -0.044421  0.006813  0.033512  0.085550
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.654880   0.026181   25.01 2.22e-12 ***
## timing      -0.195400   0.004557  -42.88 2.19e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04992 on 13 degrees of freedom
## Multiple R-squared:  0.993,  Adjusted R-squared:  0.9924
## F-statistic:  1838 on 1 and 13 DF,  p-value: 2.188e-15
```

Using above results, we obtain the following model

$$\log_{10}(Y) = 0.654880 - 0.1954 \cdot X$$

4) Plot Regression Line and Transformed Data

```
plot(timing, log10(sol), main = "Transformed Data: log10(sol) ~ timing")
abline(logModel)
```



The model for transformed data using log10 analogue appears to be almost perfect fit.

5) Let us examine the model by comparing Residuals and Fitted Values together in two columnns:

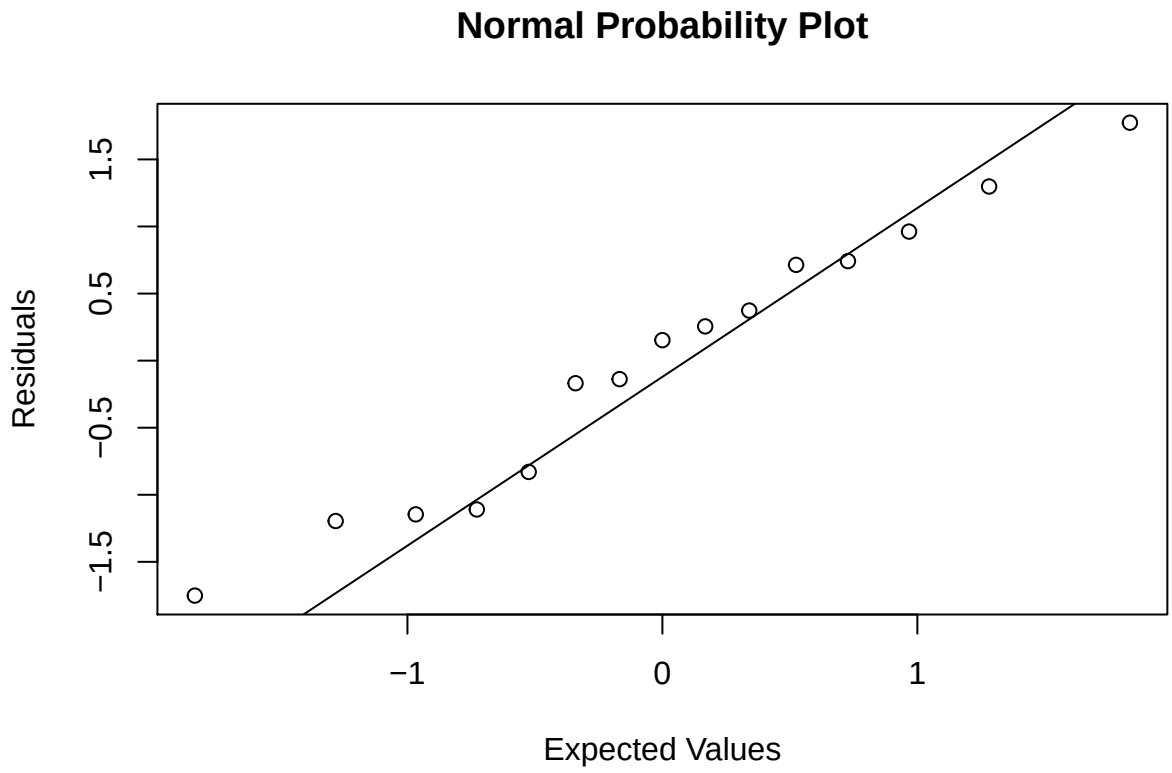
```
cbind(Residual=logModel$residuals, Fit=logModel$fitted.values)
```

```
##         Residual         Fit
## 1  -0.051178946 -1.10372301
## 2   0.057965523 -1.10372301
## 3   0.006813001 -1.10372301
## 4  -0.082957620 -0.71292240
## 5  -0.056628681 -0.71292240
## 6   0.035141692 -0.71292240
## 7   0.012317861 -0.32212178
## 8   0.085549775 -0.32212178
## 9   0.046397651 -0.32212178
## 10  0.017680995  0.06867884
## 11 -0.007980995  0.06867884
## 12 -0.039295058  0.06867884
## 13 -0.006161112  0.45947945
```

```
## 14 -0.049546328  0.45947945
## 15  0.031882242  0.45947945
```

Let us plot normal probability plot (QQ-Plot).

```
qqnorm(
  rstandard(logModel),
  main = "Normal Probability Plot",
  xlab = "Expected Values",
  ylab = "Residuals");
qqline(rstandard(logModel))
```



6) Let us express the model in the following by using what we obtained in part 1) $\log_{10}(Y) = 0.654880 - 0.1954 \cdot X$.

$$\begin{aligned} \log_{10}(Y) &= 0.654880 - 0.1954 \cdot X \\ Y &= 10^{0.654880 - 0.1954 \cdot X} \\ &= 10^{0.654880} \cdot 10^{-0.1954^X} \\ &= 4.5173 \times 0.6377^X \end{aligned}$$

PROBLEM 6 (Ch3, Q31)

Let us load the data first

```r
setwd("C:/Users/eagle/OneDrive/Course/CU Stats/STATS GR6101 - Applied Statistics I/Data")
data = read.csv("APPENC07.csv", header = FALSE)
colnames(data) <- c(
  "ID",
  "SalesPrice",
  "FinishedSqFt",
  "NumBedrooms",
  "NumBathrooms",

  "AC",
  "GarageSize",
  "Pool",
  "YearrBuilt",
  "Quality",

  "Style",
  "LotSize",
  "AdjToHighway"
)
summary(data)
```

```
##        ID          SalesPrice      FinishedSqFt   NumBedrooms
##  Min.   :  1.0   Min.   : 84000   Min.   : 980   Min.   :0.000
##  1st Qu.:131.2   1st Qu.:180000   1st Qu.:1701   1st Qu.:3.000
##  Median :261.5   Median :229900   Median :2061   Median :3.000
##  Mean   :261.5   Mean   :277894   Mean   :2261   Mean   :3.471
##  3rd Qu.:391.8   3rd Qu.:335000   3rd Qu.:2636   3rd Qu.:4.000
##  Max.   :522.0   Max.   :920000   Max.   :5032   Max.   :7.000
##   NumBathrooms         AC           GarageSize       Pool
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0   Min.   :0.00000
##  1st Qu.:2.000   1st Qu.:1.0000   1st Qu.:2.0   1st Qu.:0.00000
##  Median :3.000   Median :1.0000   Median :2.0   Median :0.00000
##  Mean   :2.642   Mean   :0.8314   Mean   :2.1   Mean   :0.06897
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:2.0   3rd Qu.:0.00000
##  Max.   :7.000   Max.   :1.0000   Max.   :7.0   Max.   :1.00000
##    YearrBuilt      Quality          Style           LotSize
##  Min.   :1885   Min.   :1.000   Min.   : 1.000   Min.   : 4560
##  1st Qu.:1956   1st Qu.:2.000   1st Qu.: 1.000   1st Qu.:17205
##  Median :1966   Median :2.000   Median : 2.000   Median :22200
##  Mean   :1967   Mean   :2.184   Mean   : 3.345   Mean   :24370
##  3rd Qu.:1981   3rd Qu.:3.000   3rd Qu.: 7.000   3rd Qu.:26787
##  Max.   :1998   Max.   :3.000   Max.   :11.000   Max.   :86830
##   AdjToHighway
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.02107
##  3rd Qu.:0.00000
##  Max.   :1.00000
```

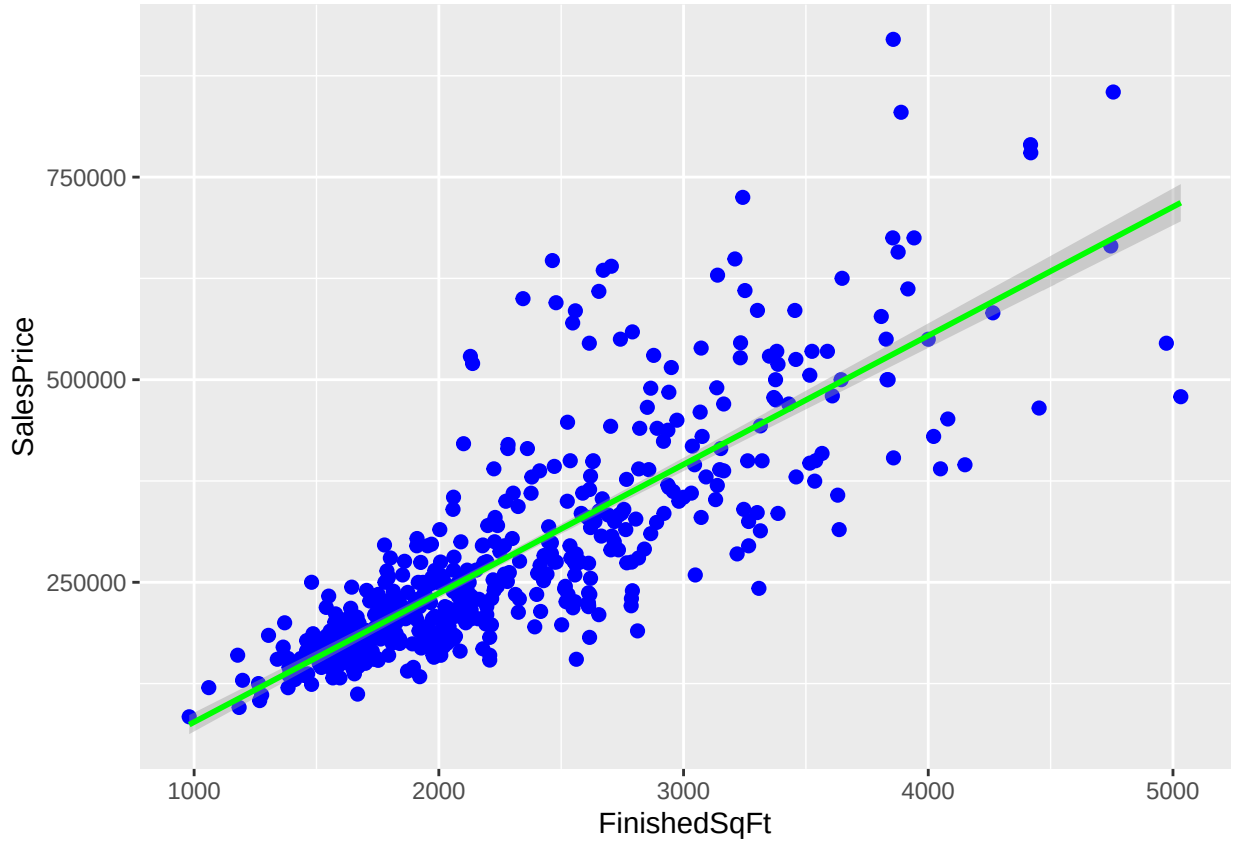
Build a model to predict Sales Price as a function of Finished Square Feet. Let us define dependent and independnt variables and run linear regression model on these variables.

```r
set.seed(2020)
idx = sample(1:522,200)
Y = data$SalesPrice[idx]
X = data$FinishedSqFt[idx]
housingModel = lm(Y~X)
summary(housingModel)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -217327  -41820   -6839   24809  345488
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.232e+05  2.159e+04  -5.708 4.13e-08 ***
## X            1.809e+02  9.231e+00  19.596  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 87300 on 198 degrees of freedom
## Multiple R-squared:  0.6598, Adjusted R-squared:  0.6581
## F-statistic:   384 on 1 and 198 DF,  p-value: < 2.2e-16
```

```r
ggplot(data, aes(x=FinishedSqFt, y=SalesPrice)) +
  geom_point(color="blue", size = 2) +
  geom_smooth(method=lm, level=0.90, color="green")
```

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



We observe scatter plot of Sales Price Y depending on FInished Squre Feet X. It is apparent the homoskedasticity assumption fails to check, because the data points are getting further apart from each other when X increases. In other words, we expect noncontant variance across this data. The fitted linear regression model is not going to be a good fit.

Based on above model, let us predict newcoming houses (but be aware the estimated housing prices will not be precise):

```
house1EstPrice = coef(summary(housingModel))[1,1] + coef(summary(housingModel))[2,1]*1100
house2EstPrice = coef(summary(housingModel))[1,1] + coef(summary(housingModel))[2,1]*4900
print(paste0('House 1 Sales Price: ', house1EstPrice))
```

```
## [1] "House 1 Sales Price: 75777.7691708542"
```

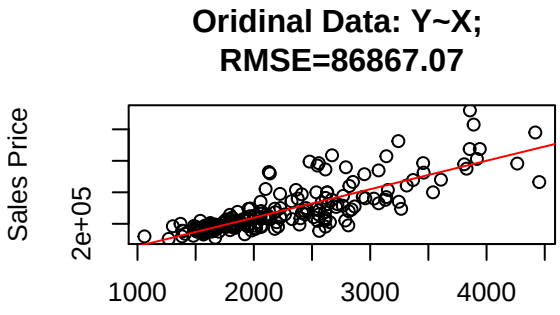
```
print(paste0('House 2 Sales Price: ', house2EstPrice))
```

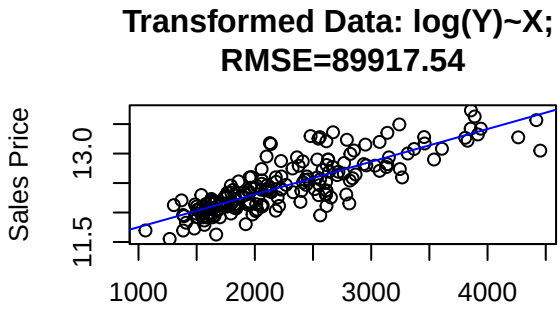
```
## [1] "House 2 Sales Price: 763187.821201884"
```

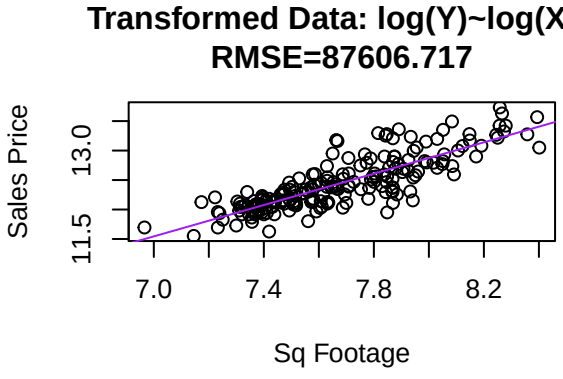
Let us examine the data by plotting Y against X.

```
# Compare models:
housingModel1 = lm(Y~X)
housingModel2 = lm(log(Y)~X)
housingModel3 = lm(log(Y)~log(X))
```

```r
# Plot
par(mfrow=c(2,2))
plot(
  X, Y,
  main = paste0(
    "Oridinal Data: Y~X; \nRMSE=",
    round(sqrt(mean((Y - housingModel1$fitted.values)^2)), 3)),
  xlab = "Sq Footage", ylab = "Sales Price");
abline(housingModel1, col = "red", lty = 1)
plot(
  X, log(Y),
  main = paste0(
    "Transformed Data: log(Y)~X; \nRMSE=",
    round(sqrt(mean((Y - exp(housingModel2$fitted.values))^2)), 3)),
  xlab = "Sq Footage", ylab = "Sales Price");
abline(housingModel2, col = "blue", lty = 1)
plot(
  log(X), log(Y),
  main = paste0(
    "Transformed Data: log(Y)~log(X); \nRMSE=",
    round(sqrt(mean((Y - exp(housingModel3$fitted.values))^2)), 3)),
  xlab = "Sq Footage", ylab = "Sales Price");
abline(housingModel3, col = "purple", lty = 1)
```

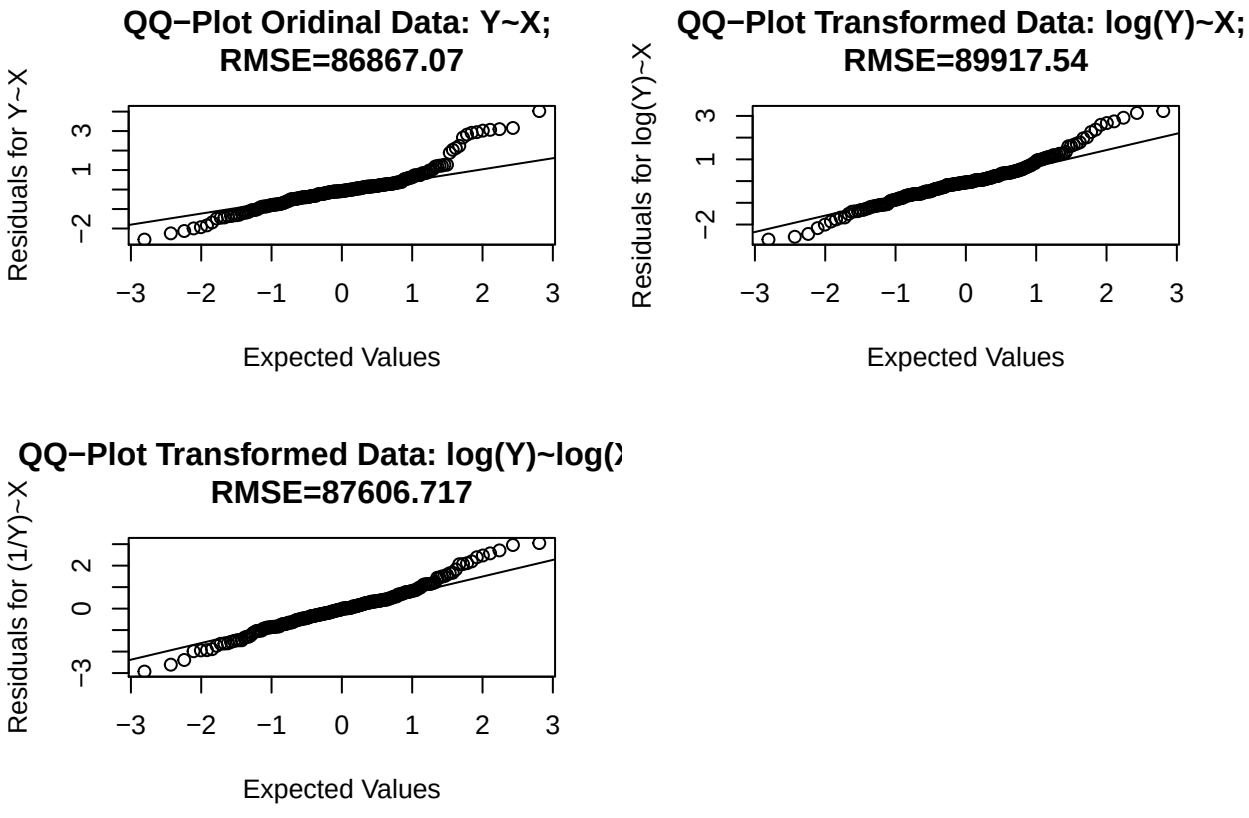






Let us examine normal probability plot on residuals:

```r
par(mfrow=c(2,2))
qqnorm(
  rstandard(housingModel1),
  main = paste0(
    "QQ-Plot Oridinal Data: Y~X; \nRMSE=",
    round(sqrt(mean((Y - housingModel1$fitted.values)^2)), 3)),
  xlab = "Expected Values",
  ylab = "Residuals for Y~X"); qqline(rstandard(housingModel1))
qqnorm(
  rstandard(housingModel2),
  main = paste0(
    "QQ-Plot Transformed Data: log(Y)~X; \nRMSE=",
    round(sqrt(mean((Y - exp(housingModel2$fitted.values))^2)), 3)),
  xlab = "Expected Values",
  ylab = "Residuals for log(Y)~X"); qqline(rstandard(housingModel2))
qqnorm(
  rstandard(housingModel3),
  main = paste0(
    "QQ-Plot Transformed Data: log(Y)~log(X); \nRMSE=",
    round(sqrt(mean((Y - exp(housingModel3$fitted.values))^2)), 3)),
  xlab = "Expected Values",
  ylab = "Residuals for (1/Y)~X"); qqline(rstandard(housingModel3))
```



From above result, we observe that *housingModel3* is the best because it has the least Root Mean Square

Error. The model is

$$\log(Y) = \beta_0 + \beta_1 \cdot log(X) = 2.233 + 1.33 \cdot \log(X) \Rightarrow Y = \exp(\beta_0) \cdot X^{\beta_1} = \exp(2.233) \cdot X^{1.33}$$

Hence, let us use *housingModel3* to predict new coming houses.

```
house1EstPrice = exp(2.233+1.33*log(1100))
house2EstPrice = exp(2.233+1.33*log(4900))
print(paste0('House 1 Sales Price: ', house1EstPrice))
```

```
## [1] "House 1 Sales Price: 103474.145669335"
```

```
print(paste0('House 2 Sales Price: ', house2EstPrice))
```

```
## [1] "House 2 Sales Price: 754640.94369871"
```

Conclusion: We conclude that overall the model still delivered a somewhat good fit for this data. We have a reasonable root mean square error and we have a rather linear association between dependent variable and independent variable after the transformation using logarithm function. However, we still observe some noise in the data. We can also observe from QQ-plot that on the tails on both ends the observations do not quite follow the straight line. This could lead to heavy tail in reality (i.e. if we are provided test data set, our model may deliver poor performance). The fact is worsened before the transformation.