Math 289C: Case Study 4: Regression

Saurabh Kulkarni (A53099844)
Mitesh Gadgil (A53095373)
Kyle Kole (A10845644)

Table of Contents

Objective

To provide a procedure to convert gain to density when the gauge is in operation. To rephrase, we have to find a relationship between the density with respect to gain

Introduction

In order to monitor the water supply from yearly snowfall, the United States Department of Agriculture (USDA) uses a gamma transmission snow gauge. These gauges are placed in areas of interest, including the Sierra Nevada Mountains, which supplies water to Northern California. The gauge determines the depth of the snowfall to provide an estimation of the volume of snow as a proxy for the amount of water that will be available for the upcoming year. In addition to the volume of snow, gamma rays are used to approximate density.

Purpose

The main purposes of tracking snowfall, besides approximating the amount of water supplied for the year, include tracking yearly weather patterns, recommending water useage for the area (i.e. during times of droughts), and flood management (dense snow holds less rainwater; therefore during very heavy periods of rain with dense snow, flooding is likely to occur). Guages do not measure the density directly, in the sense that the gauge is not in a location to physically gather snow, because a direct approach may misconstrue actual density readings. Instead, gamma ray emissions are sent to penetrate the snow and converted to a density measurement once a return signal is received. However, as the gauge ages, calibration is required to maintain a level of accuracy in gamma ray readings.

Data

This data is provided by the USDA during one of their calibration sessions in the Sierra Nevada. For our gauge model, different blocks of polyethylene with known densities are placed between the emitting rod and the receiver rod for the gauge. As gamma rays pass through the polyethylene blocks, the gamma ray may be absorbed, bounce away from the receiver rod, or pass through the block to the receiver rod. For each block, 30 measurements are taken, and the middle 10 are reported in the dataset; the measurement from the gauge is called "gain." There were 9 blocks in total.

Background Theory

Correlation

Before there is any fitting of bivariate data to any method, it is important to graph the data in a scatterplot to determine a rough estimate of which model to use. If the data seems to relate in some sort of linear fashion, linear fitting may seem to be a good idea. In order to support this notion, correlation may be calculated. Correlation is a measure of a systematic linear relationship between two sets of data. As long as there is a systematic relationship, the strength may vary between positive and negative 1. The correlation coefficient may be calculated as follows:

$\rho_{X,Y} = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$

where $\italics{cov}$ stands for covariance, and σ stands for the standard deviation of a particular dataset. Of course, as the population standard deviations of the datasets are not known, they may be replaced by the estimated sample standard deviations, and the means μ may be replaced by the sample means in order to estimate the sample correlation coefficient. If the coefficient is strong (i.e. close to 1 in absolute value), then it may make sense to fit the data to a linear regression model, provided a few assumptions are met.

Ordinary Least Squares

A method for estimating unknown parameters in a linear regression model is ordinary least squares. The goal of this method is to minimize the differences between observed responses in the dataset to predicted responses in a linear approximation of the data. Mathematically, we are looking at the following optimization problem with objective function S(b):

S(b) = \sum^n_{i=1} (y_i - x_i^Tb)^2

where b is the candidate value for parameter β. Then what we are looking for in the optimization problem is:

$\hat{\beta_1} = argmin S(b)$

This is the general multivariate optimization problem; linear regression is not limited to only single variable regression. In the multivariate case, what we have is then a matrix of independent variables, and a matrix of parameters.

In terms of simple regression (one variable), the equation for the fitted line is

$y = \beta_0 + \beta_1x$. Formulas for the parameters may be written nicely in the following form:

$\hat{\beta_1} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum{(x_i-\bar{x})^2}$
$\hat{\beta_0} = \bar{y} - \hat{\beta_1}\bar{x}$

However, there are assumptions that go into ordinary least squares before we can simply apply the method of estimation to the data. The data must be exogenous from the error i.e. the errors in the regression should have zero conditional mean:

$E[\epsilon|X] = 0$

The errors must be normally distributed, and the observations must be independent and identically distributed (iid).

It is common to expect any statistical analysis to accompany estimated regression parameters with test statistics. Estimates are only useful if they are statistically significant. In other words, estimates are only useful if they do not occur by pure chance.

The hypothesis test for the slope regression parameter is the following:

$H_0: \hat{\beta_1} = 0$ vs. $H_1: \hat{\beta_1} \neq 0$

The parameters of interest are the slope parameters controlling the explanatory variables. If β for an explanatory variable is significant, we may conclude that the variable is relevant to our regression model. Our test statistic follows a T distribution.

Residuals

As with any model, there are errors to be expected. Errors in the linear model are called residuals, denoted ϵ_i. The residual may be calculated as the difference between the actual, observed value and the predicted value using the OLS fitted model. In other words,

$\epsilon_i = y_i - \hat{y_i}$

It is very important to graph the residuals in order to support the assumption that the residuals are indeed distributed normally. Note that if the plotted residuals take a discernible shape, such as a parabola, then it may indicate that a linear model is not a good fit.

Homo/Heteroscedasticity

Homoscedasticity is a fancy term for the homogeneity of variance. In the case of the OLS estimation, homoscedasticity in the errors is important to ensure that the least-squares estimator is in fact the best linear unbiased estimator (BLUE). If residuals are heteroscedastic, then a weighted least-squares is more prudent in estimation. Fortunately, homoscedasticity is not required to ensure the estimates of parameters are unbiased, consistent, and asymptotically normal; only to ensure the OLS estimator is BLUE. The intuition is that in prediction, if the errors are homoscedastic, then the error terms do not depend on the independent variable and is therefore easily anticipated.

R^2

Most commonly called the R^2 (r-squared) measure of a linear model, the square of the correlation coefficient is commonly used to determine the strength of the fit to a linear model. More precisely, R^2,the coefficient of determination, describes the percent of variability in the response variable described by the linear model. The remaining variability may be attributed to omitted variables. R^2 is calculated as follows:

$R^2 = 1 - \frac{SS_res}{SS_tot}$

Where SS_res is the sum of squares of residuals, $SS_res = \sum(y_i-\hat{y_i})^2$
and SS_tot is the total sum of squares, $SS_tot = \sum(y_i - \bar{y})^2$.

Note that by the way R^2 is defined, it is easily inflated by adding more explanatory variables to the model. Hence, it may not be prudent to use R^2 in multi-variable regression. Caution should also be taken when interpreting R^2 as the coefficient of determination does not accurately indicate if independent variables cause changes in the dependent variable, if omitted variable bias exists, if the correct regression model was used, etc.

Cross Validation

Sometimes referred to rotation estimation, this model validation technique tests for how the results of a statistical analysis generalizes to an independent dataset. This technique is crucial for prediction as the model must be able to generalize, otherwise predictions may be made only within the dataset used to create the model. Cross validation requires a generated dataset to be used as the testing dataset. If done properly, cross validation gives insight to overfitting and how the model generalizes to unknown datasets.
The cross validation method we focus on is called the“ leave-p-out” cross-validation (LpO CV). LpO CV requires p observations to be reserved as part of the validation set and the remaining observations are used as the training set to create the model. This process is repeated multiple times until the original sample is cut into all possible validation sets of p observations and the remaining training set. Thus there are nCp repetitions. This method if costly if the dataset gets to be too large.

Outliers and Odd data

Unfortunately, there is no statistical rule which determines whether or not a data point is a statistical outlier. The best that can be done at the moment is to plot the data point and using intuitive judgment in determining an outlier. The best definition available is that outliers are points that lie away from the cluster of data points. Outliers that are horizontally far away from the center of the cluster are called high leverage points. These may influence the slope of the regression. If this is the case, then the high leverage points are then classified as influential points. To reiterate, these classifications are done by graphing a scatterplot of the data and using intuitive judgment.

Approach

1. Fitting:
    a. Use the given data to fit the gain or a transformation of the gain (function of the gain) as with respect to the density.
    b. Sketch the least square line on the scatter plot
    c. Find correlation, point estimates of slope and intercept
    d. Find the residuals.
2. Analysis of the fit
    a. Check if the residuals should be nearly normal using histogram, Q-Q plots and kurtosis.
    b. Check for homoscedasticity
    c. Find the absolute residual error and least square error
    d. Find the strength of fit using R^2
    e. Comment on the outliers and identify the high leverage and the influential points in the dataset
    f. Find the effect on the estimated line fit, if the density in the data is not exactly reported
3. Prediction
    a. Find the density as a function of the gain
    b. Display the confidence interval on the scatter plot with the fit
4. Cross Validation
    a. Implement cross validation technique, that is omit a block of measurements associated with a particular density 'd = 0.508, 0.001'.
    b. Use the predicted function to find an estimate of the density for that corresponding gain 'g'.
    c. Evaluate the prediction based on the estimation error
    d. Test for different blocks

Data Analysis

Fitting

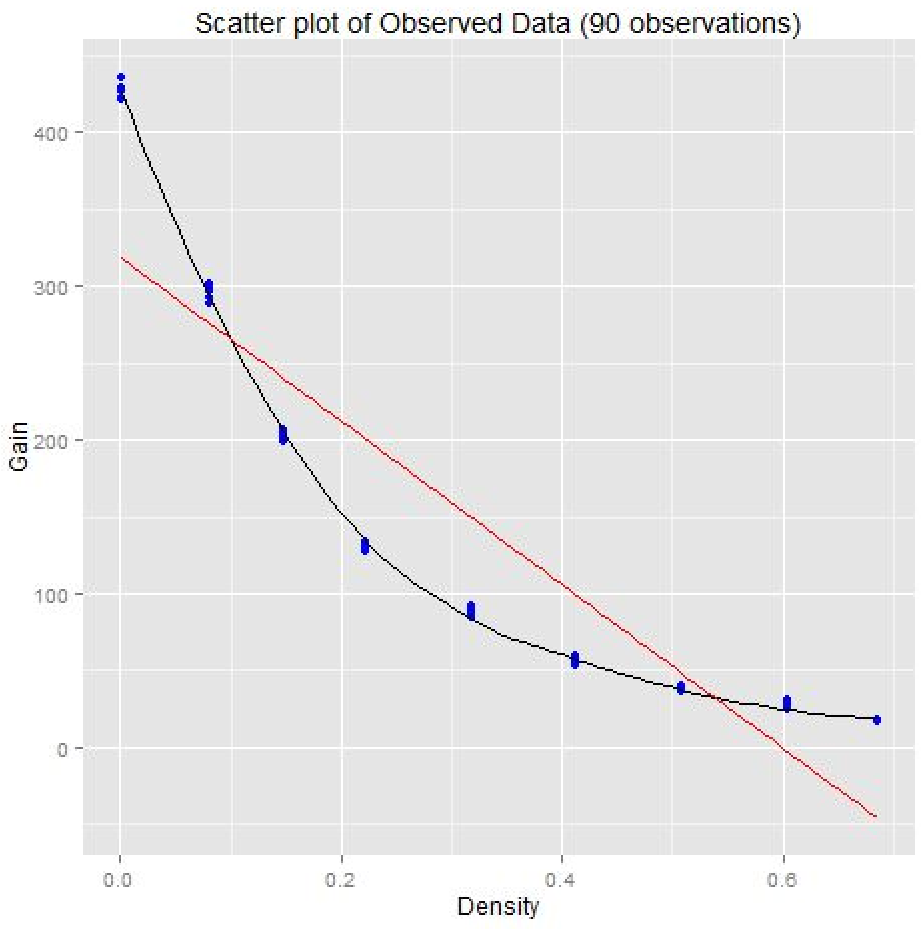
We need to find a linear correlation between the density of the snow with the gain. For this we first calculate a scatterplot of the data. For each of the 9 values of density on the data we have 10 readings. for our linear regression model we will average out the data.

```
rm(list=ls())

# Loading dataset
gauge <- read.csv("gauge.txt", sep="")
head(gauge)
density <- gauge$density
gain <- gauge$gain

# 9 Unique values of density for which observations have been recorded
obs.density <- unique(gauge$density)
# gain observations for each unique density stored as a row vector
obs.gain <- matrix(nrow = length(obs.density),ncol = nrow(gauge)/length(obs.density))
for (i in 1:length(obs.density)){
  obs.gain[i,] <- gauge$gain[which(gauge$density==unique(gauge$density)[i])]
}
# summary statistics for gain observations of each unique density
obs.mean <- numeric(length(obs.density))
obs.sd <- numeric(length(obs.density))
for (i in 1:length(obs.density)){
  obs.mean[i] <- mean(obs.gain[i,])
  obs.sd[i] <- sd(obs.gain[i,])
}
print(matrix(c(obs.density,obs.mean,obs.sd),nrow = 9,ncol=3,dimnames =
list(1:9,c("Density","Mean(gain)"," Std.dev(gain)"))))
```

The readings in table above give the average value of the gains for the corresponding density.

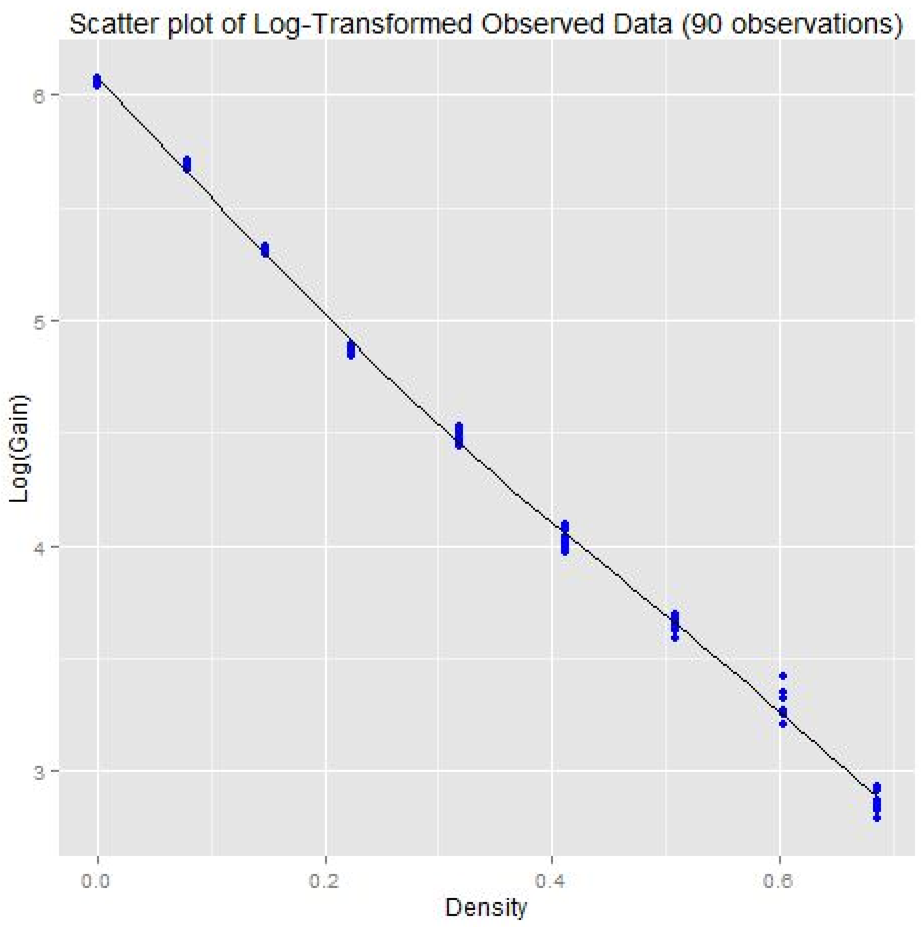
```
str(gauge)
library(ggplot2)
```

```
ggplot(gauge,aes(x = density,
y=gain))+geom_point(col="blue",size=2)+labs(y="Gain",x="Density",title="Scatter plot of
Observed Data (90 observations)")+geom_smooth(mapping = aes(x = density,
y=(gain)),col='black',se=FALSE)+geom_smooth(method = "lm", se = FALSE,col='red')
```



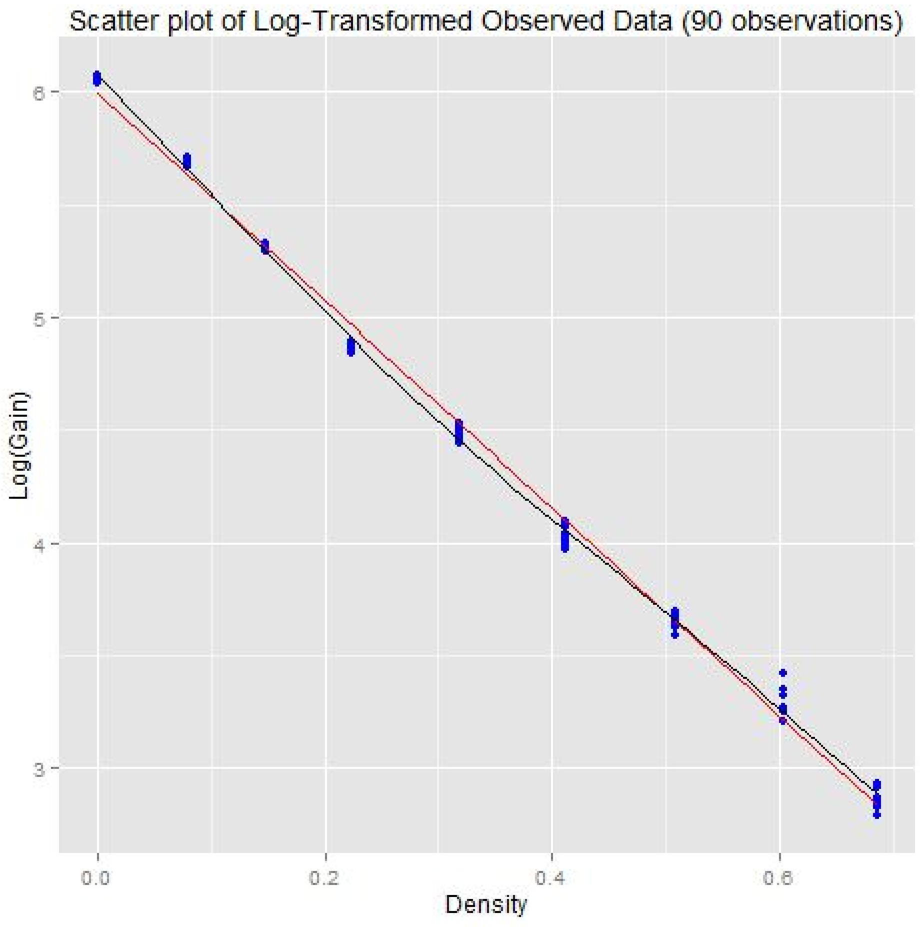
Here by observing the graph we conclude that the relationship between density and gain is non-linear. Hence we can use a transformation of the gain *T(gain)* so that the relationship between Density and *T(gain)* is linear. An obvious choice here would be to use **logarithms**

```
ggplot(gauge,aes(x = density,
y=log(gain)))+geom_point(col="blue",size=2)+labs(y="Log(Gain)",x="Density",title="Scatter plot
of Log-Transformed Observed Data (90 observations)")+geom_smooth(mapping = aes(x =
density, y=log(gain)),col='black',se= FALSE)
```



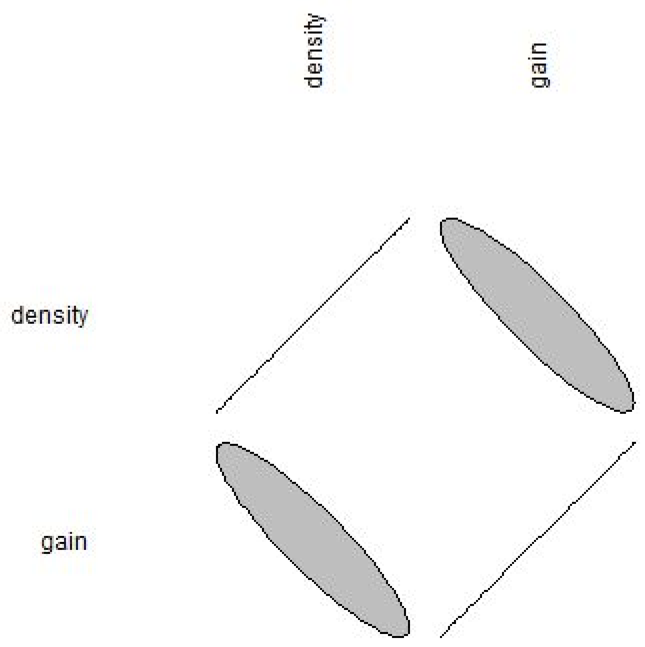
We clearly see that the explanatory and response variables have a **linear correlation**. The data has a **negative correlation.** Using these 9 points we fit a linear line through the data.

```
ggplot(gauge,aes(x = density, y=log(gain)))+geom_point(col="blue",size=2)+labs(y="Log(Gain)",x="Density",title="Scatter plot of Log-Transformed Observed Data (90 observations)")+geom_smooth(method = "lm", se = FALSE,col='red')+geom_smooth(mapping = aes(x = density, y=log(gain)),col='black',se= FALSE)
```



Here we see that the linear regression has worked well in approximating the data. We can use ellipse plots to graphically examine correlation in simple linear fitting. During linear regression, the two variables, X and Y are assumed to follow the bivariate normal distribution. The function *plotcorr()* plots a correlation matrix using ellipse-shaped glyphs for each entry. The ellipse represents a level curve of the density of a bivariate normal with the matching correlation.

```
fit.log <- lm(density~I(log(gain)) )
fit.linear <- lm( density~I(gain) )
library(ellipse)
plotcorr(cor(gauge))
```



We observe that the linear fit without log transformation does not work well. Hence we use the log-linear fit.

```
print("The correlation matrix of the given data is: ")
cor(gauge)
```

We have used the *fit()* function to fit the data. The summary not only tells information about the slope-intercept coefficients and standard errors, it also performs the student-t test and the Fischer test to test for significance of the line.

```
# F-value and R-square
summary(fit.linear)

summary(fit.log)
```

Regression Analysis

We will analyze the summary of the log-linear fit as we clearly see from the summary that *without* log transformations the regression does not work well.

R-squared coefficient

In statistics, the coefficient of determination, denoted R^2 or r^2 and pronounced R squared, is a number that indicates how well data fit a statistical model – sometimes simply a line or a curve.

The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model.
R-squared = Explained variation / Total variation
R-squared is always between 0 and 1

- 0 indicates that the model explains none of the variability of the response data around its mean.
- 1 indicates that the model explains all the variability of the response data around its mean.

In general, the higher the R-squared, the better the model fits your data.

Here our R^2 estimates ~ 1. Hence, or linear model is a good fit.

The Multiple R-squared provides a measure of how well future outcomes are likely to be predicted by the model. As explained in the theory, R-squared measures the proportion of the variation in your dependent variable (Y) explained by your independent variables (X) for a linear regression model. Adjusted R-squared adjusts the statistic based on the number of independent variables in the model.

Student-t test

The test statistic is a t-score (t) defined by the following equation. t = b1 / SE
where b1 is the slope of the sample regression line, and SE is the standard error of the slope. The student t tests are used to conduct hypothesis tests on the regression coefficients obtained in simple linear regression. For our log-linear model we see that the p-value that P(>|t|) ~ 0. This means that the probability of log-linear model fitting well by chance is low. Hence there is no need to reject the null hypothesis which says log-linear model is a good fit. Hence we get desirable p-value

Fisher test

While R-squared provides an estimate of the strength of the relationship between your model and the response variable, it does not provide a formal hypothesis test for this relationship. The overall F-test determines whether this relationship is statistically significant. In general, an F-test in regression compares the fits of different linear models. Unlike t-tests that can assess only one regression coefficient at a time, the F-test can assess multiple coefficients simultaneously.

If the P value for the overall F-test is less than your significance level, you can conclude that the R-squared value is significantly different from zero.

p-Value
If the P value for the F-test of overall significance test is less than your significance level, you can reject the null-hypothesis and conclude that your model provides a better fit than the intercept-only model.In this case, the p-value is very low (~0), it's suggesting that it would be rare to get a result as unusual. Hence our hypothesis to assume that linear regression is a good model is acceptable.

```
# Adjusted R-squared ~ 1 => Good model
# confidence interval for the coefficients
confint(fit.log,level=0.95)
```

Analysis of Residuals

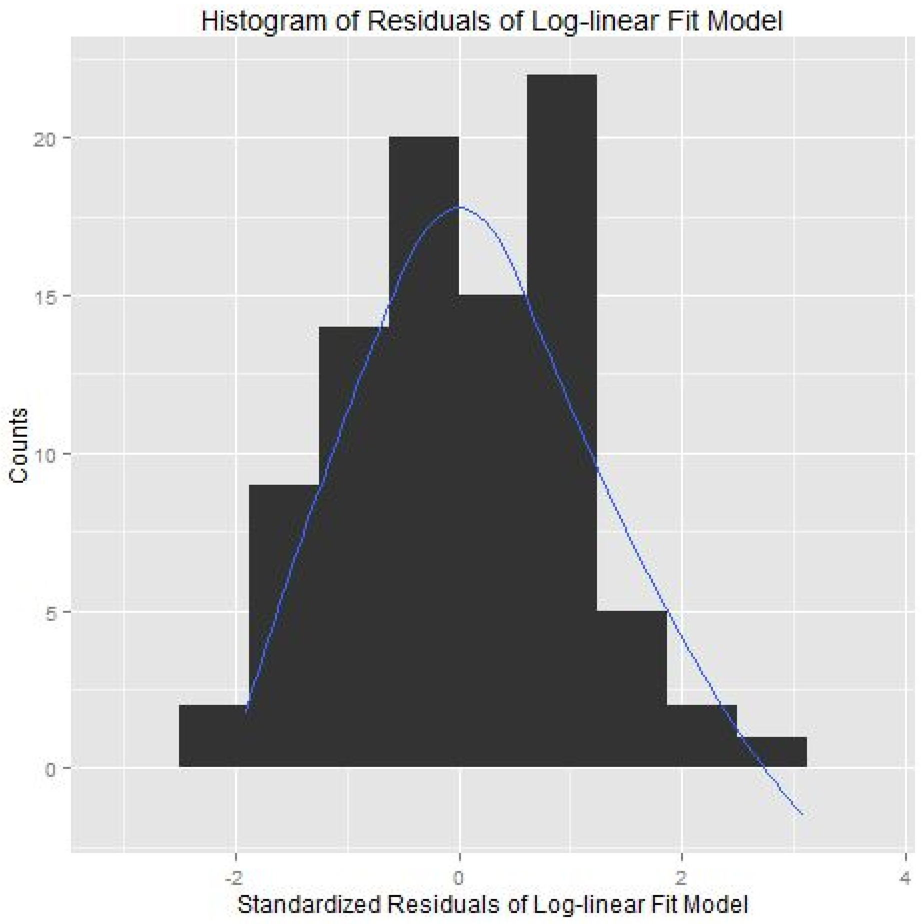
Nearly Normal Residuals

We further analyze the residuals to check if they are nearly normal and show homoscedasticity to further strengthen our regression analysis. Here we will use the all 90 readings (10 for each unique density) so that our analysis will be thorough.

```
res.linear <- as.numeric(residuals(fit.linear))
res.log <- as.numeric(residuals(fit.log))
```

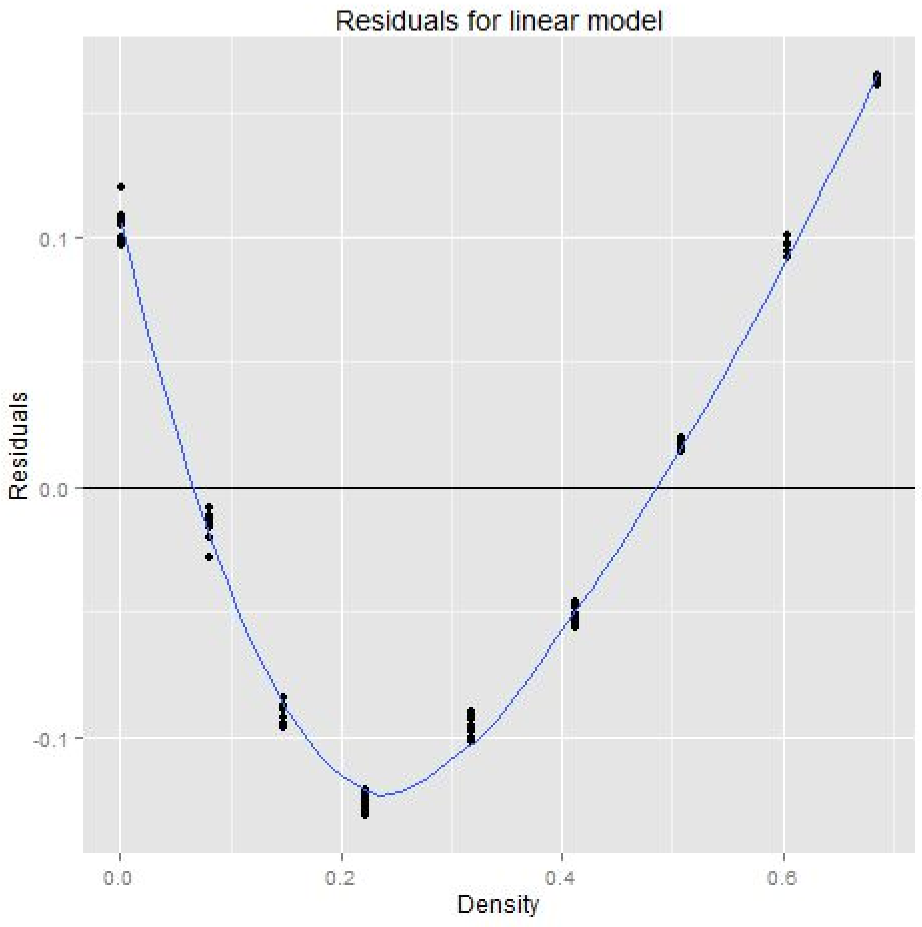
We will now check if the distribution of residuals are normal or not.

```
# To check if residuals are nearly normal
a<-fortify(fit.linear)
sres.linear <- as.numeric(a$.stdresid)


b<-fortify(fit.log)
sres.log <- as.numeric(b$.stdresid)
ggplot(b, aes(sres.log))+geom_histogram(binwidth =
diff(range(sres.log))/8)+labs(x="Standardized Residuals of Log-linear Fit
Model",y="Counts",title="Histogram of Residuals of Log-linear Fit
Model")+geom_smooth(aes(y=45*dnorm(sres.log,mean=mean(sres.log),sd=sd(sres.log))),se =
FALSE)
```

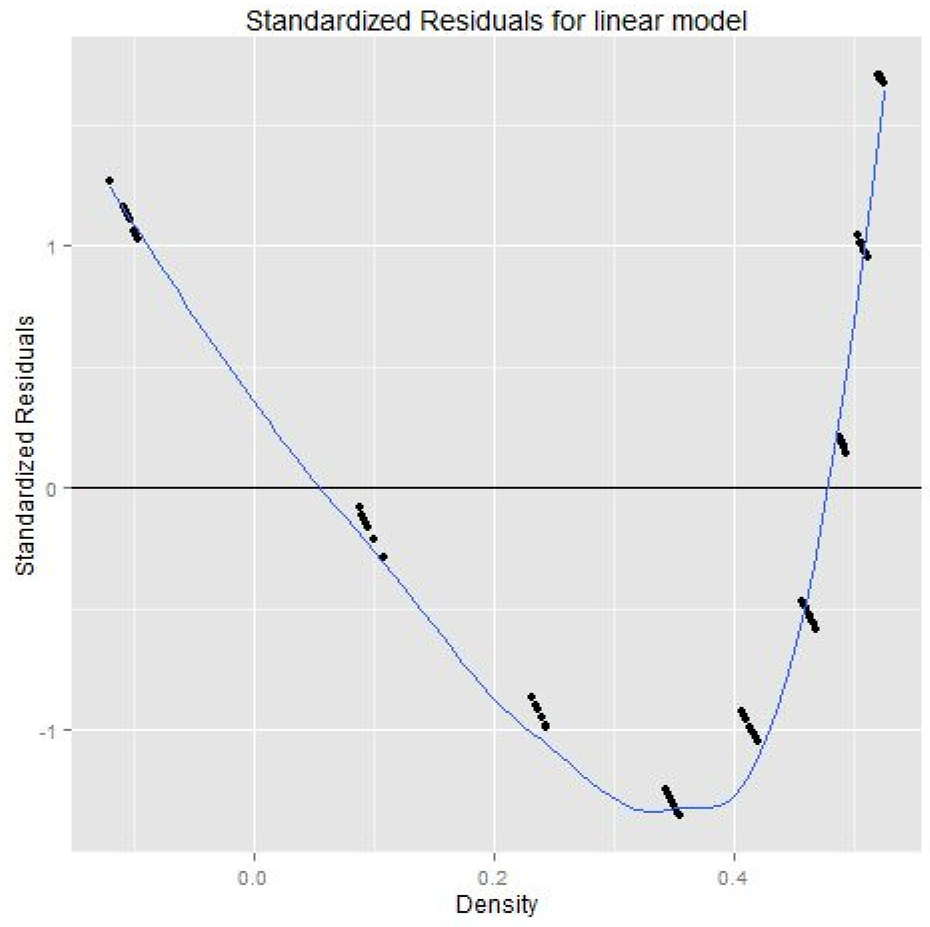


Visual analysis of the histogram implies that the residuals are somewhat normally distributed. We fitted a normal curve through the distribution of the residuals to show that the residuals are ***nearly normal but not completely normal***. In order to verify this, we will calculate the kurtosis and the quantile quantile plots.

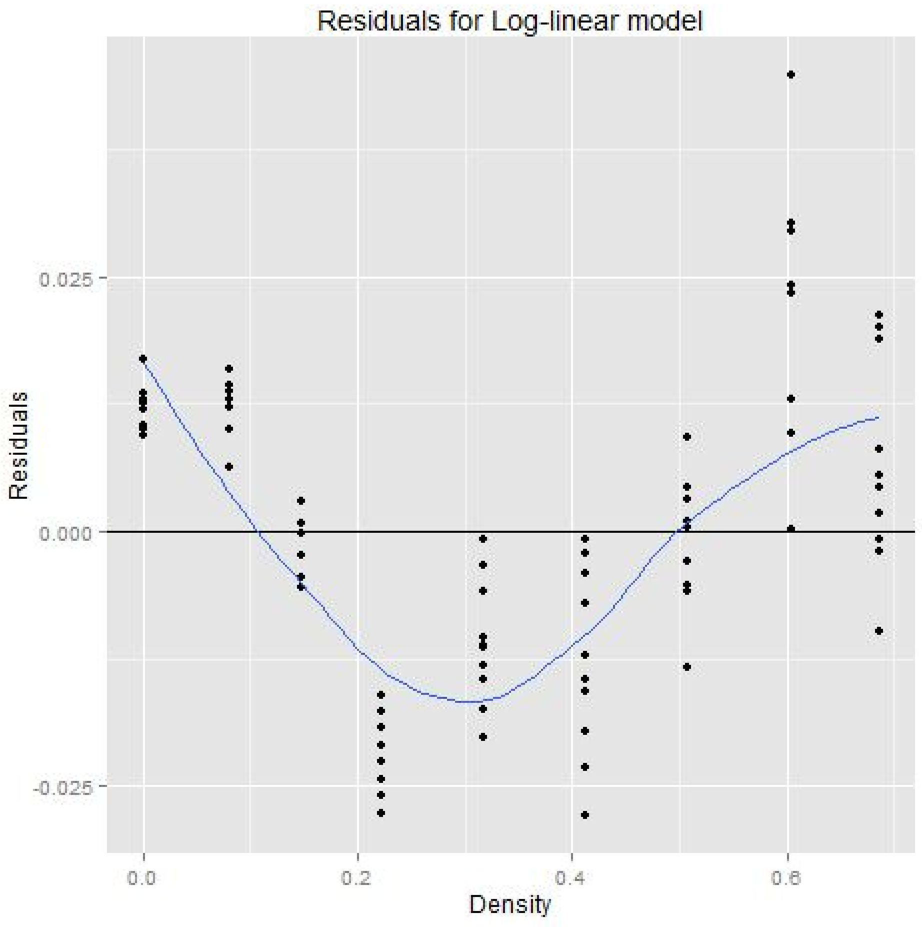
```
#ggplot(fit.linear, aes(density, .resid)) +geom_point() +  geom_hline(yintercept = 0) +
geom_smooth(se = FALSE)+labs(x="Density",y="Residuals",title="Residuals for linear model")
```



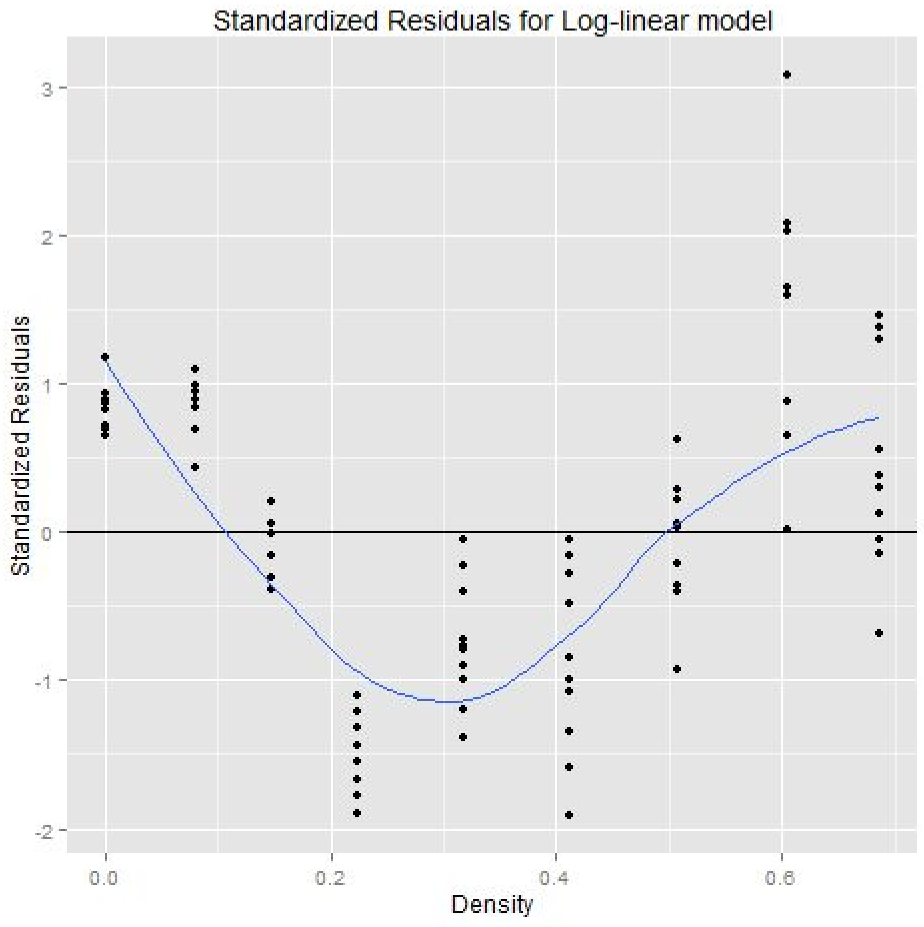
#ggplot(fit.linear, aes(.fitted, .stdresid)) +geom_point() + geom_hline(yintercept = 0) + geom_smooth(se = FALSE)+labs(x="Density",y="Standardized Residuals",title="Standardized Residuals for linear model")



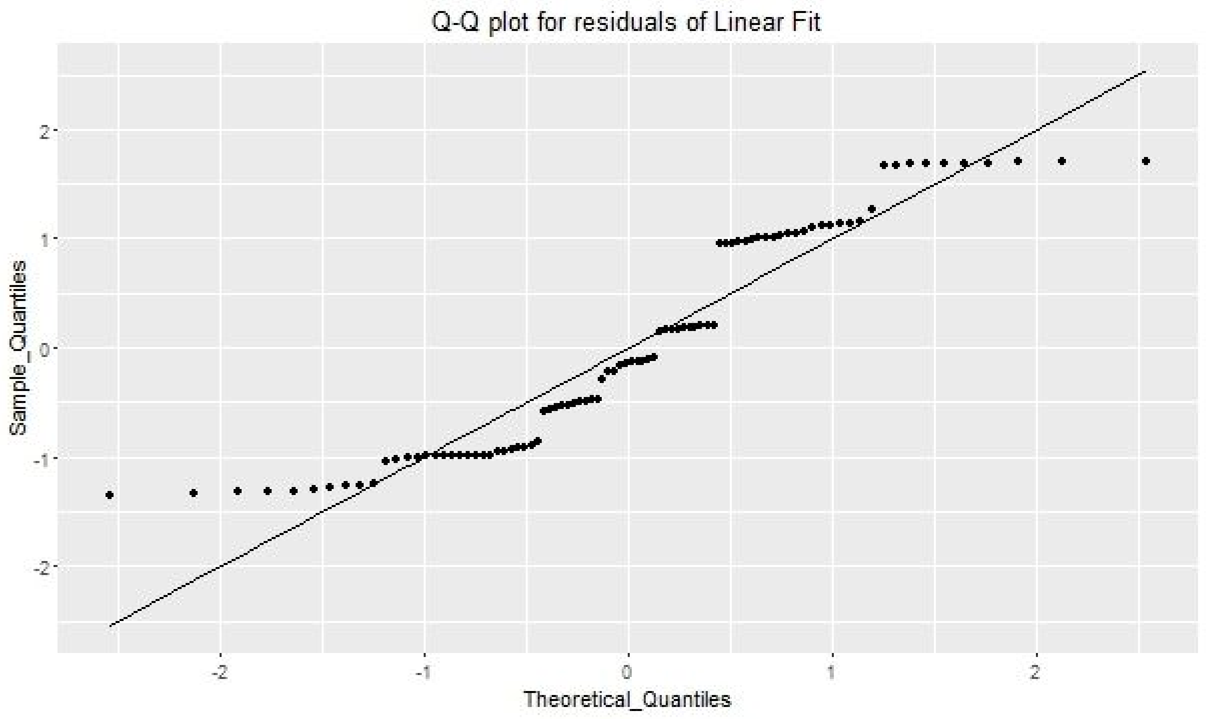
ggplot(fit.log, aes(density, .resid)) +geom_point() + geom_hline(yintercept = 0) + geom_smooth(se = FALSE)+labs(x="Density",y="Residuals",title="Residuals for Log-linear model")

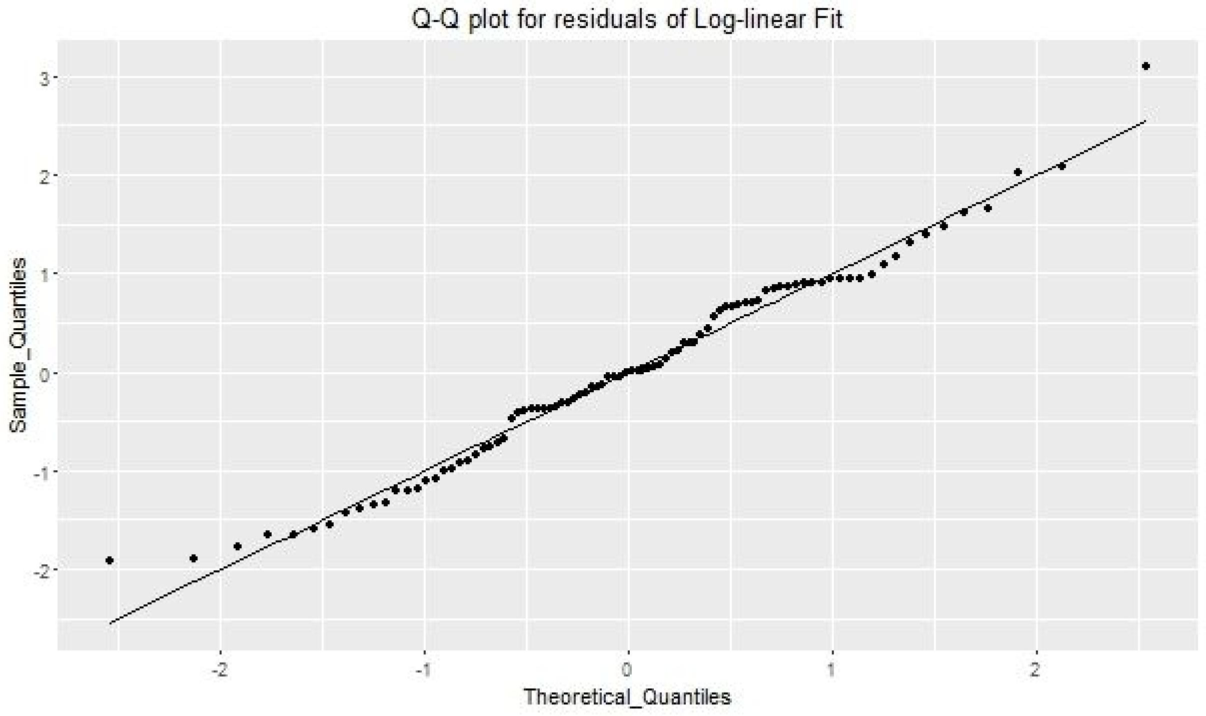


ggplot(fit.log, aes(density, .stdresid)) +geom_point() + geom_hline(yintercept = 0) + geom_smooth(se = FALSE)+labs(x="Density",y="Standardized Residuals",title="Standardized Residuals for Log-linear model")



```
qqplot <- function(y,distribution=qnorm,t) {
  require(ggplot2)
  x <- distribution(ppoints(y))
  d <- data.frame(Theoretical_Quantiles=x, Sample_Quantiles=sort(y))
  p <- ggplot(d, aes(x=Theoretical_Quantiles, y=Sample_Quantiles)) +geom_point() +
geom_line(aes(x=x, y=x)) +labs(title=t)
  return(p)
}
```





```
qqplot(sres.linear,t="Q-Q plot for residuals of Linear Fit")
qqplot(sres.log,t="Q-Q plot for residuals of Log-linear Fit")
```

```
# install.packages('moments')
library(moments)
kurt_res = kurtosis(res.log)
cat("The kurtosis of the distribution of residuals is: ",kurt_res)
```

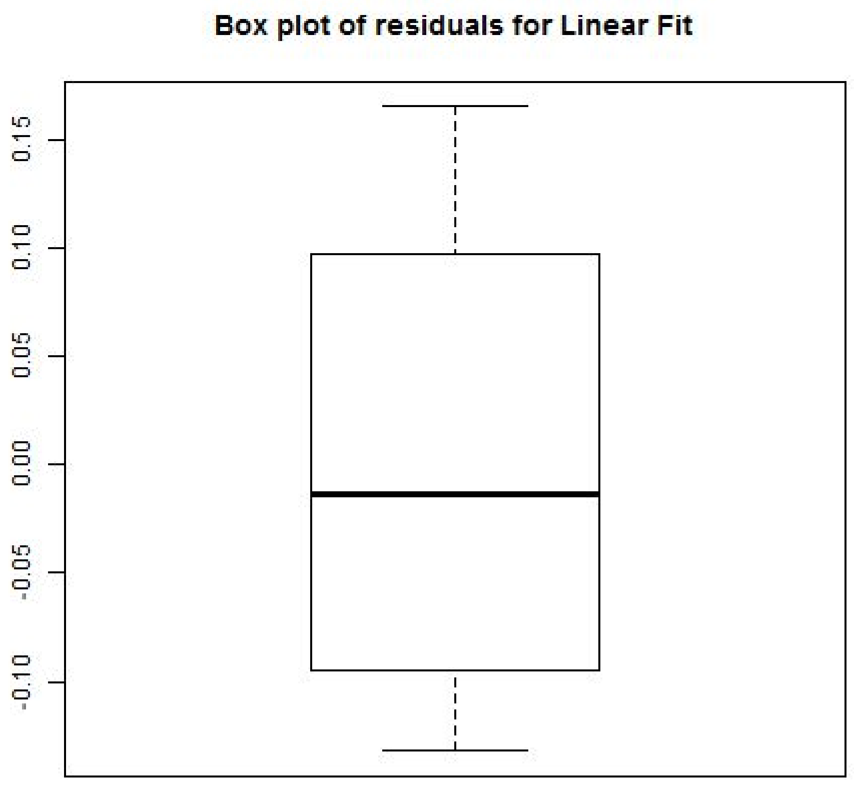
Again, the Q-Q plots show that although the residual distribution is very close to a normal distribution it is not ideally normal. The kurtosis will also be close to 3 but not equal to three. This confirms the nearly normal tendency of the residuals

Homoscedasticity

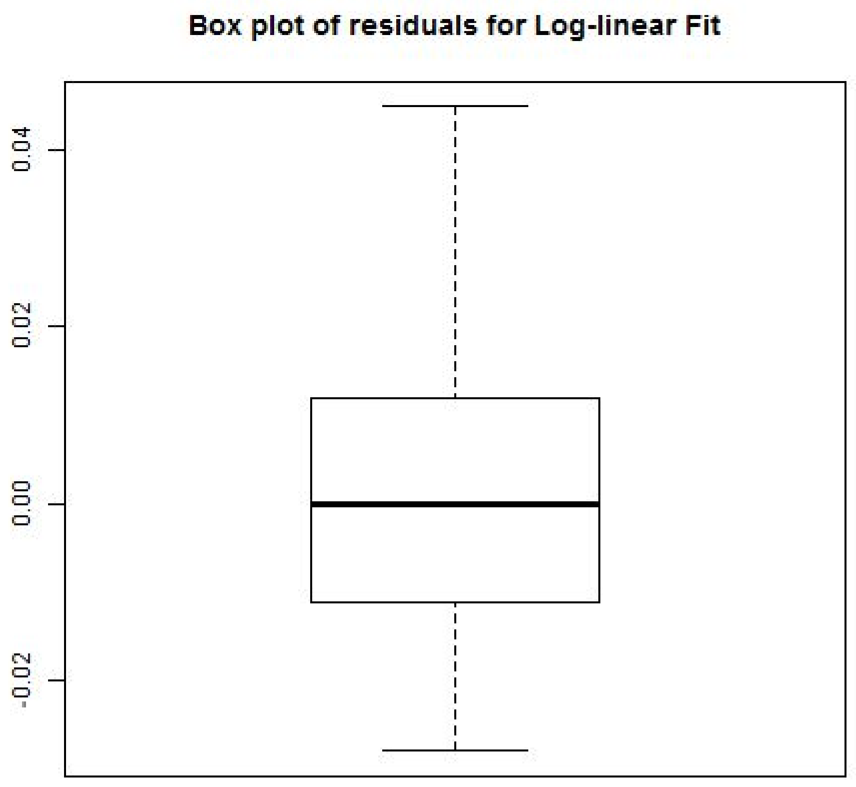
We check for homoscedasticity. We first plot a boxplot of the residuals. We expect the residuals to have the following properties:

- Mean should be zero
- The distribution of residuals should be equal on either ends of the mean

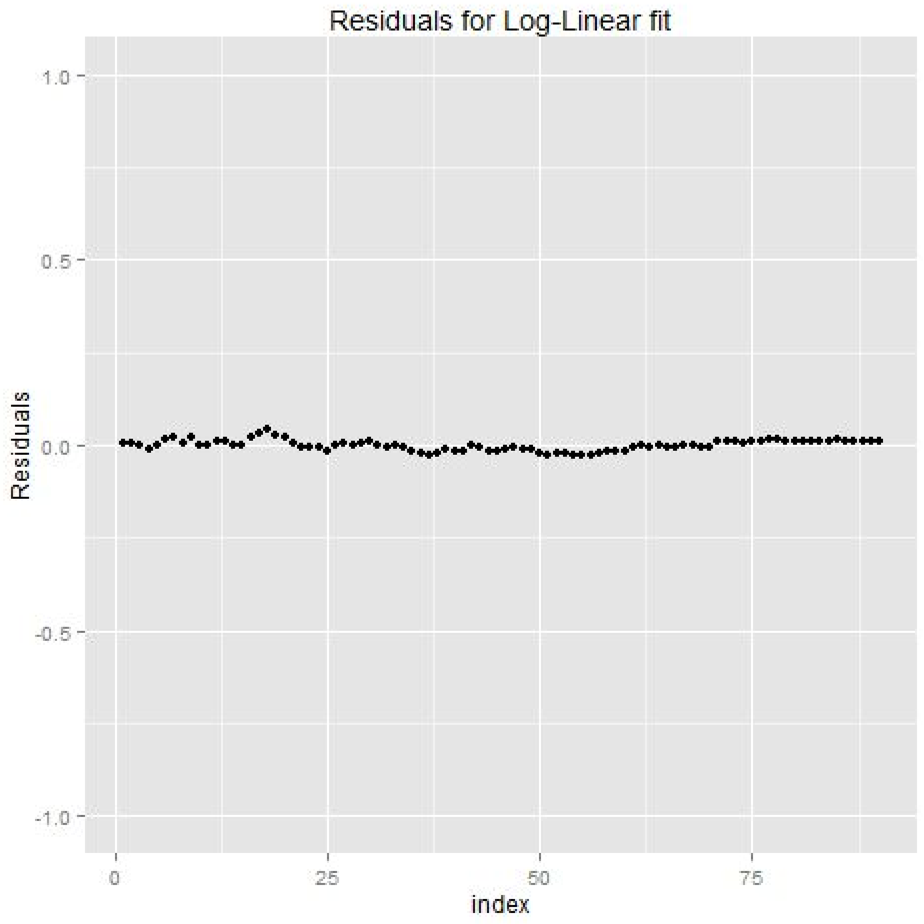
```
boxplot(res.linear,main="Box plot of residuals for Linear Fit")
```



```
boxplot(res.log,main="Box plot of residuals for Log-linear Fit")
```

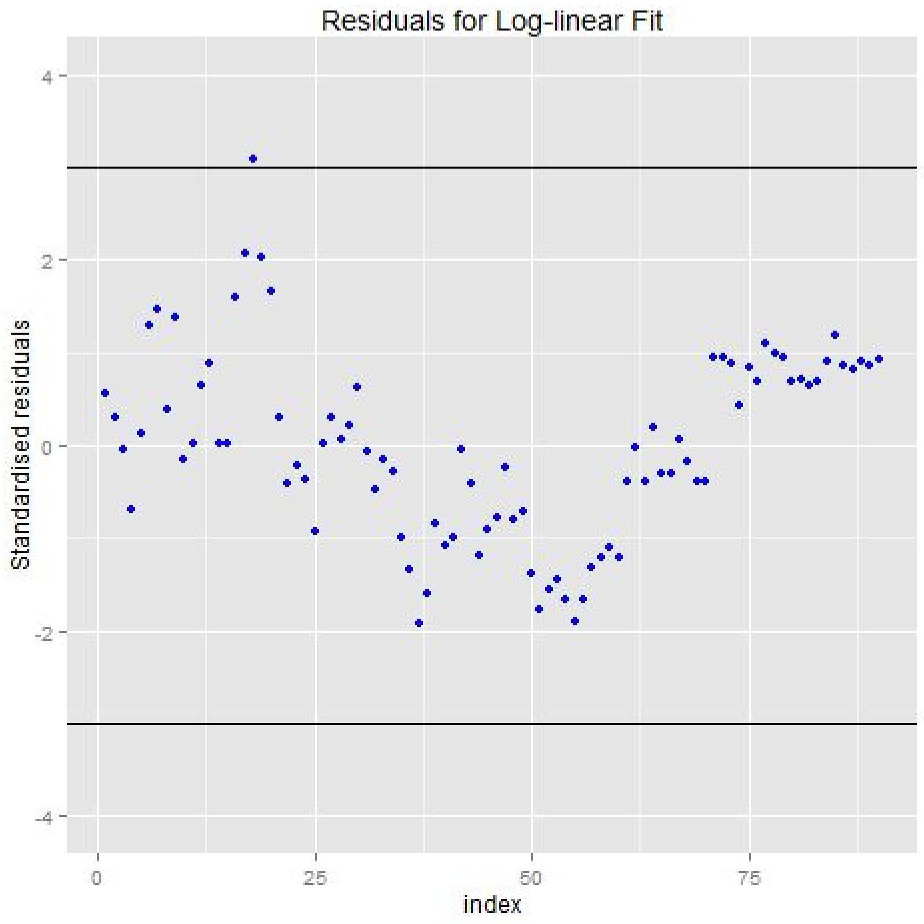


ggplot(b,aes(x=1:90,y=.resid))+geom_point()+labs(title="Residuals for Log-Linear fit",x="index",y="Residuals")+ylim(c(-1,1))

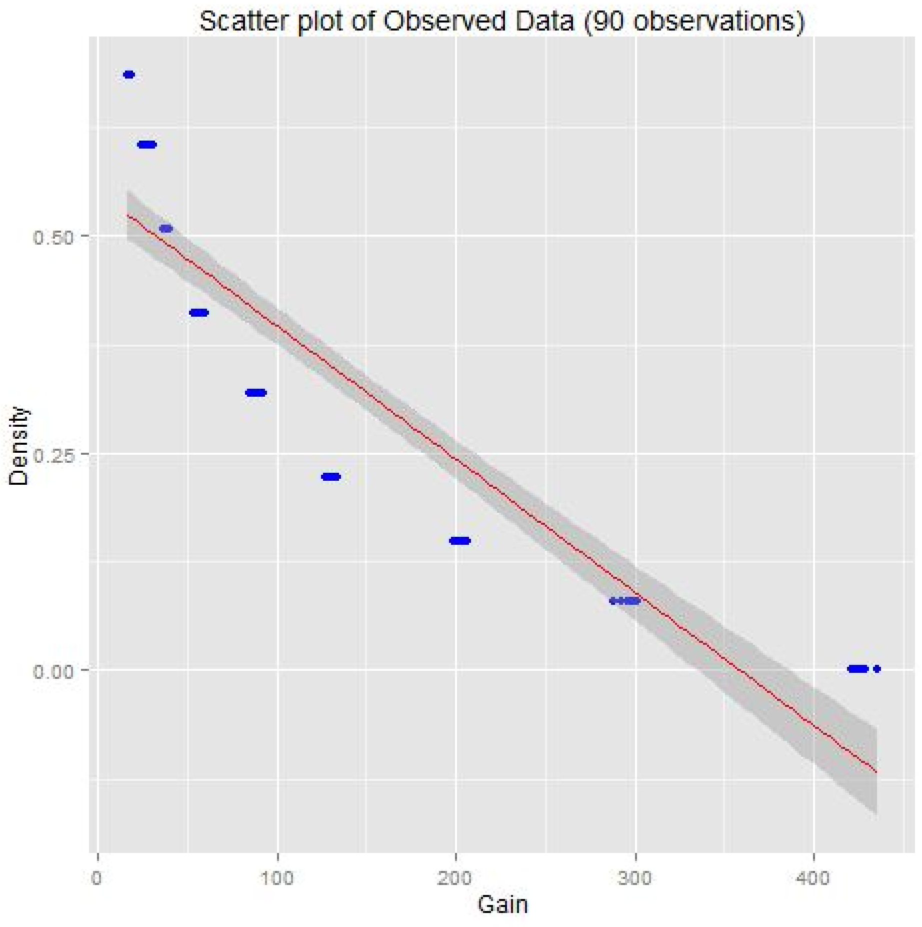


The boxplot is centered at 0 and has equal magnitude on either ends of the mean. That means the variance of the residuals remains constant.
We can further illustrate this on a scatterplot

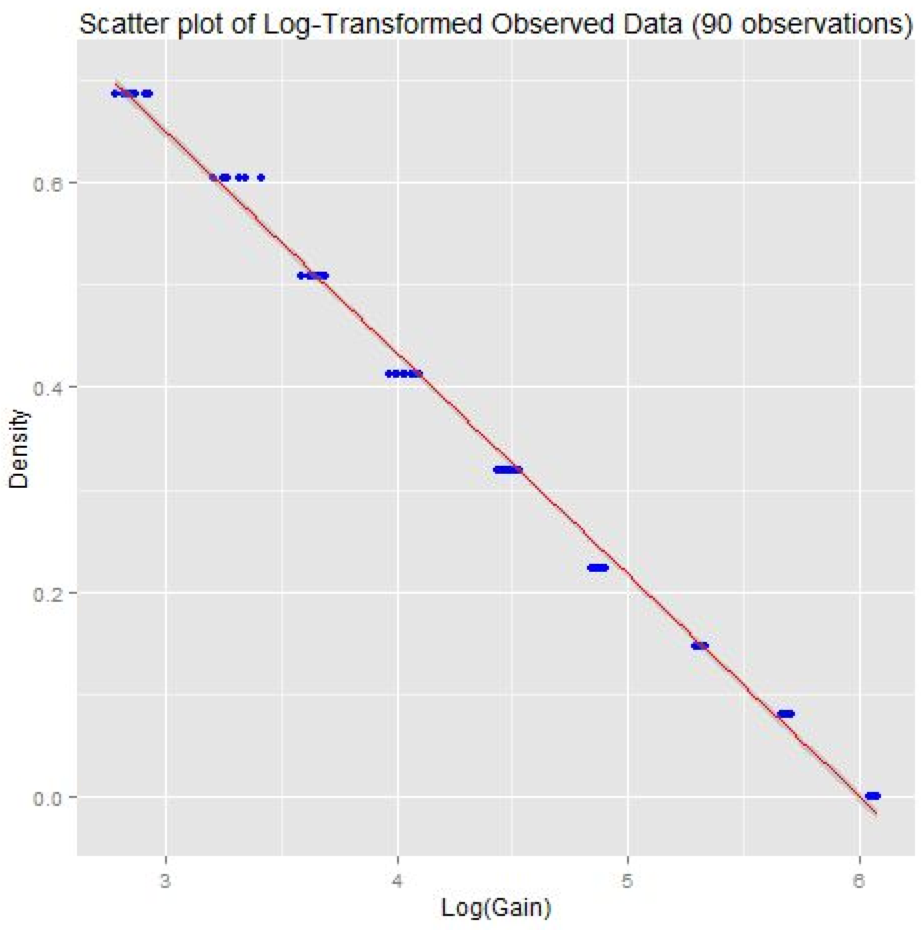
```
ggplot(b,aes(x=1:90,y=.stdresid))+geom_point(col='blue')+ylim(c(-4,4))+geom_hline(yintercept =
3)+geom_hline(yintercept = -3)+labs(title="Residuals for Log-linear Fit",y="Standardised
residuals",x="index")
```



```
ggplot(gauge,aes(y = density,
x=gain))+geom_point(col="blue",size=2)+labs(x="Gain",y="Density",title="Scatter plot of
Observed Data (90 observations)")+stat_smooth(method = "lm", se =
TRUE,col='red',level=0.95)
```



```
ggplot(gauge,aes(y = density,
x=log(gain)))+geom_point(col="blue",size=2)+labs(x="Log(Gain)",y="Density",title="Scatter plot
of Log-Transformed Observed Data (90 observations)")+stat_smooth(method = "lm", se =
TRUE,col='red',level=0.95)
```



```
mean_res = mean(res.log)
var_res = var(res.log)
range_res = c(min(res.log), max(res.log))
```

The scatterplot shows that the variance of the residuals is not changing throughout all data-points. This is a desirable result and tells that the regression analysis we have done is satisfactory.

Errors

We calculate the errors for this process:

RMS: Root mean square error

This is given by the formula: $\sqrt{\frac{1}{N}\sum e_i^2}$

```
# Mean Square Error
rmse <- function(error){
  return(sqrt(mean(error^2)))
}
# RMSE for linear model
RMSE.linear <- rmse(res.log)
cat("The RMS error of the regression fit is: ",RMSE.linear
```

Absolute sum of errors: Given by: $\sum|e_i|$

```
abs_res_error = sum(abs(res.log))
```

Prediction

From the data we find:
sd(density) = 0.2386595
sd(log(gain)) = 1.100421
correlation R = -0.9031

We can calculate the point estimates of the slope = (0.238/1.1)*(-0.90) = -0.194.
Ideal Slope = -0.194
Observed Slope = -0.21

We see that the observed slope (point estimate) is pretty close to the ideal.

Intercept is a bias and not significant here as our explanatory variable never goes to zero. When the explanatory variable (gain) goes to zero we assume the device is off. Hence, slope signifies true relation between the two variables and our estimate is very close to the ideal one.

Still writing this

Cross-Validation and Evaluation of the Log-Linear Model

```
require(plotrix)
```

```
fit_lm<-function(d,t){d.train<-unique(d$density)
g.train <- numeric(length(d.train))
for (i in 1:length(d.train)){
  g.train[i]<-mean(d$gain[which(d$density==d.train[i])])
}
model_x <- g.train
fit.model<-lm(d.train~log(model_x))
pred.density <- predict(fit.model, data.frame(model_x<-t))
r <- list("model"=fit.model,"prediction"=pred.density)
return (r)
}
```

The fit_lm function takes any subset of the data that we have, which we will use to train the model of linear regression called the training data and the remaining portion of the data termed as test data. It outputs the parameters of the fitted model and the prediction values for the explanatory variables in test data.

```
data.train1 <- gauge[which(density!=0.508),]
data.train1 <- cbind.data.frame(data.train1,"id"=rep(x = 1,nrow(data.train1)))
test.data1 <- gain[density==0.508]

data.train2 <- gauge[which(density!=0.001),]
data.train2 <- cbind.data.frame(data.train2,"id"=rep(x = 1,nrow(data.train2)))
test.data2 <- gain[density==0.001]

ret1<-fit_lm(data.train1,test.data1)
ret2<-fit_lm(data.train2,test.data2)
```

In the above code snippet,the dataset has been curated into test dataset and training dataset. We will be running the cross-validation scheme on 2 types of subsets namely using - the sample points with density= 0.508 and density=0.01 as test points while using the rest of the dataset as training dataset. We run these 2 sets of training and test datasets through the fit_lm function to obtain the model parameters and prediction results that are used in the analysis below.

```
model1 <- ret1$model
density.prediction1 <- ret1$prediction
df_temp <-
cbind.data.frame("density"=density.prediction1,"gain"=test.data1,"id"=rep(x=0,length(test.data1)
))
df1<- rbind.data.frame(data.train1,df_temp)
```

```
model2 <- ret2$model
density.prediction2 <- ret2$prediction
df_temp <-
cbind.data.frame("density"=density.prediction2,"gain"=test.data2,"id"=rep(x=0,length(test.data2)
))
df2<- rbind.data.frame(data.train2,df_temp)

ggplot(df1,aes(y = density,
x=log(gain),col=factor(id)))+geom_point(size=3.5)+labs(x="Log(Gain)",y="Density",title="Predict
ed Data (density=0.508) with fitted Log-Linear Model")+geom_smooth(method = "lm", se =
FALSE,col='gray')
ggplot(df2,aes(y = density,
x=log(gain),col=factor(id)))+geom_point(size=3.5)+labs(x="Log(Gain)",y="Density",title="Predict
ed Data (density=0.001) with fitted Log-Linear Model")+geom_smooth(method = "lm", se =
FALSE,col='gray')
```

The above plots show the predicted values for the 2 cases and we can see how the predicted values lie on the fitted linear model exactly. These predictions can only model the deterministic part of the observed density i.e. if we were to measure the actual density for a particular log(gain) value then we wouldn’t get the value predicted here, although it would be very close to that value. Hence, we introduce the concept of a confidence interval, to be more sure about our estimation or prediction.

```
pred.density.CI1 <- predict(model1, data.frame(x.model=test.data1), interval="confidence")
pred.density.CI2 <- predict(model2, data.frame(x.model=test.data2), interval="confidence")

lbd1 <- pred.density.CI1[,2]
ubd1 <- pred.density.CI1[,3]
lbd2 <- pred.density.CI2[,2]
ubd2 <- pred.density.CI2[,3]

true.density1 = 0.508
true.density2 = 0.001

check1 <- (true.density1 >= lbd1) & (true.density1 <= ubd1)
check2 <- (true.density2 >= lbd2) & (true.density2 <= ubd2)
check1
check2
```

We have calculated the lower bounds and upper bounds for the predicted values of density with 95% confidence interval and check if our predicted values of density for the test data points actually lies in the confidence interval around the true density that we know for the test points.

On checking this, we find that we are able to correctly estimate the density for both sets of test data i.e. the predicted value does lie in the confidence interval around the true value.

```
plotCI(test.data1, density.prediction1, ui=ubd1, li=lbd1,main="CI of predicted density & True density (d = 0.508)",sub="Predicted values(Black)   True values(Red)",xlab="Gain",ylab="Density")
points(test.data1, rep(x=true.density1,length(test.data1)), col="red")

plotCI(test.data2, density.prediction2, ui=ubd2, li=lbd2,main="CI of predicted density & True density (d = 0.001)",sub="Predicted values(Black)   True values(Red)",xlab="Gain",ylab="Density")
points(test.data2, rep(x=true.density2,length(test.data2)), col="red")
```

The above plot shows the confidence interval around true value and the predicted values for each of the data points in the test datasets. This gives us a sense of the accuracy with which we can predict density values from gain measurements and why this is a powerful tool.

```
plot(data.train1$density ~ log(data.train1$gain), type = 'n',main="Prediction results for density 0.508",xlab="log(gain)",ylab="Density",sub="shaded region is interval")
x.val <- seq(2.5, 6.5, length.out = 100)
preds1 <- predict(model1,  data.frame(model_x =exp(x.val)),interval = 'prediction')
lines(x.val, preds1[ ,3], lty = 'dashed', col = 'red')
lines(x.val, preds1[ ,2], lty = 'dashed', col = 'red')
polygon(c(rev(x.val), x.val), c(rev(preds1[ ,3]), preds1[ ,2]), col = 'grey80', border = NA)
abline(model1)
points(log(test.data1),rep(x=true.density1,length(test.data1)), col="red")



plot(data.train1$density ~ log(data.train1$gain), type = 'n',main="Prediction results for density 0.001",xlab="log(gain)",ylab="Density",sub="shaded region is interval")
preds2 <- predict(model2,  data.frame(model_x =exp(x.val)), interval = 'prediction')
lines(x.val, preds2[ ,3], lty = 'dashed', col = 'red')
lines(x.val, preds2[ ,2], lty = 'dashed', col = 'red')
polygon(c(rev(x.val), x.val), c(rev(preds2[ ,3]), preds2[ ,2]), col = 'grey80', border = NA)
abline(model2)
points(log(test.data2),rep(x=true.density2,length(test.data2)), col="red")
```

The above plots show a region of confidence interval of predicted density for every point between log(gain) of 2.5 to 6.5. This is a simulation to see how the confidence intervals vary over the range of possible log(gain) values that we might observe. It shows that we are able to predict density values accurately over a range of possible log(gain) values i.e. our model generalizes well to data points that have not been used to train the model. This is one of the cornerstone concepts in modelling data, that the models built should generalize well to data that hasn't been used to train the model.

Further Analysis

We see that the linear model is a good fit for the density vs log(Gain). However, we can go for a non-linear approximation in the first place. By using SV Regression, we can obtain better results. By using different kernels like radial, polynomial we can fine tune our fitting. Hence SVR will lead to superior results. However the disadvantage is that the level of computation substantially increases. Not only do we need to choose an optimal kernel but also fine tune each parameter. This is a major disadvantage as linear regression is a low computation fitting technique which is faster.

One major assumption regression needs is that all samples should be randomly sampled. A non-random input would mean that the slope we get will not work for any future random sample.

Conclusion

Here our objective was to identify a relationship between the density and gain of the snow gauge. In our analysis we identified that there is a linear relationship between log(Gain) and the density of the snow. Here the **explanatory variable was gain and the response variable is density**. (Although, during training, we measured gain for a corresponding density, the nature of the variables during applications get inverted).

We first found the mean value of gain for each density. Then we first tested linear regression model on density v/s gain. The nonlinear relationship between gain and density led to incorrect fitting and high error. Hence we applied linear regression to the density vs log(Gain) plots. We then performed regression analysis on this fit to see if it satisfies all the conditions.

We observed that for the **density v/s log(Gain).** plots had R^2 coefficient ~ 1. TheWe also found F-test p-value to be close to 0. Both these values imply the linear model was a good fit for the plot. We further analyze the effect on all residuals. For this analysis we do not equate the means. We take all 90 points into consideration. The residual distribution was plotted. The QQ plots and the kurtosis show that the distribution is close to normal but not normal. Hence we verify that the plots are **nearly normal.** We also found the mean and variance of the residuals and by visual inspection we see that the residuals are **homoscedastic** in nature surrounded around ~0 mean. Box Plot also illustrates this feature. We found the root mean square error and the absolute error of the fit.

Hence we see that linear model works well for approximation.