Homework 4

View it online: http://acsweb.ucsd.edu/~djc035/Assignment4.html (http://acsweb.ucsd.edu/~djc035/Assignment4.html)

Objective

The aim of this lab is to provide a simple procedure for converting gain into density when the gauge is in operation. Keep in mind that the experiment was conducted by varying density and measuring the response in gain, but when the gauge is ultimately in use, the snow-pack density is to be estimated from the measured gain.

1.

Fitting: Use the data to fit the gain, or a transformation of gain, to density. Try sketching the least squares line on a scatter plot.

** Do the residuals indicate any problems with the fit?

** If the densities of the polyethylene blocks are not reported exactly, how might this affect the fit?

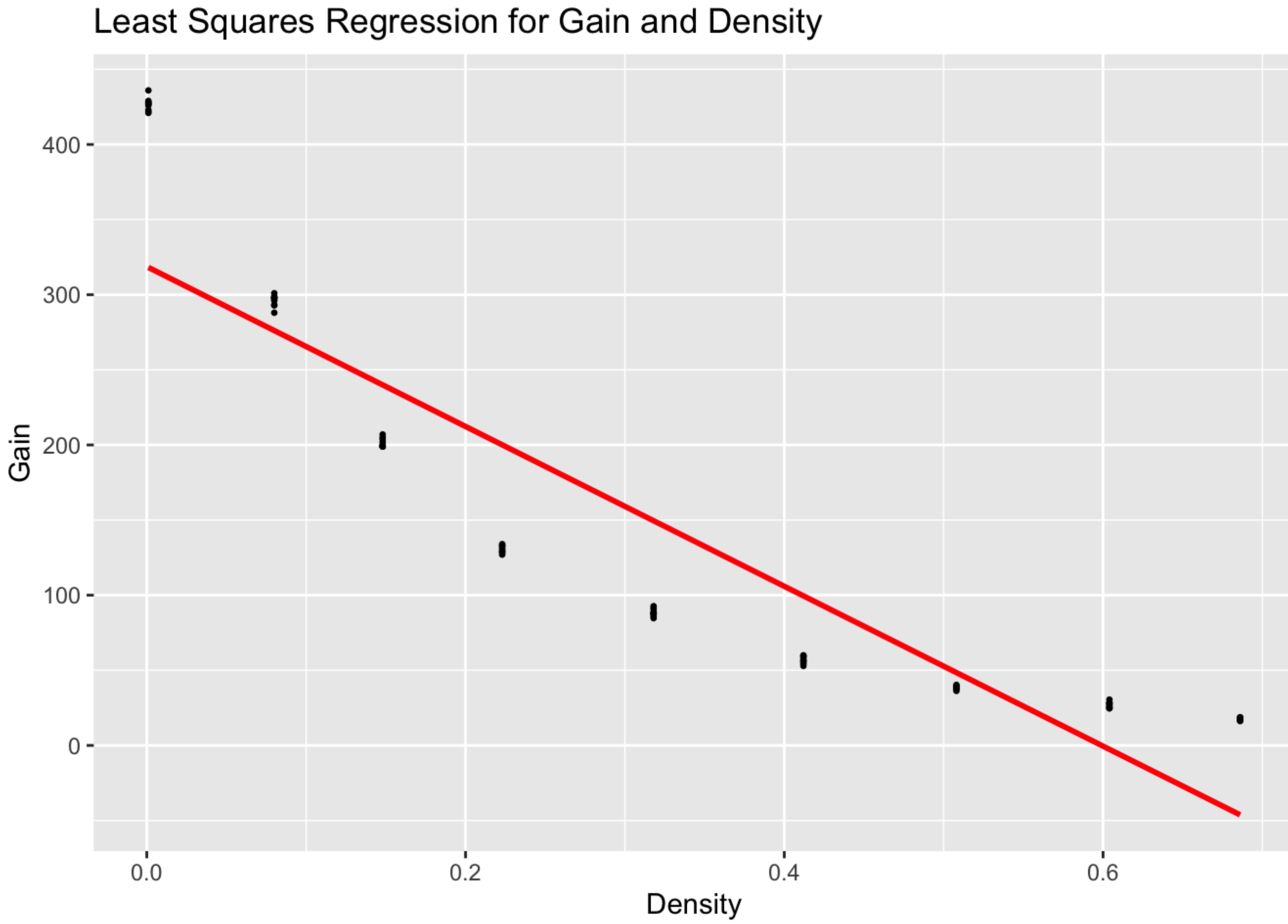
```
data <- read.csv("gauge.txt", sep="")
head(data)
```

```
##   density gain
## 1   0.686 17.6
## 2   0.686 17.3
## 3   0.686 16.9
## 4   0.686 16.2
## 5   0.686 17.1
## 6   0.686 18.5
```

```
density <- data$density
gain <- data$gain
```

```
library(ggplot2)

ggplot(data,aes(x = density, y = gain)) +
geom_point(col = "black", size = 0.5) +
labs(y = "Gain", x = "Density", title = "Least Squares Regression for Gain and Density")
+
geom_smooth(method = "lm", se = FALSE, col = 'red')
```

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




```
ggplot(data, aes(x = density, y = log(gain))) +
geom_point(col = "black", size = 0.5) +
labs(y = "Log(Gain)", x = "Density", title = "Least Squares Regression for Log-Transform
ed Gain and Density") +
geom_smooth(method = "lm", se = FALSE, col = 'red')
```

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

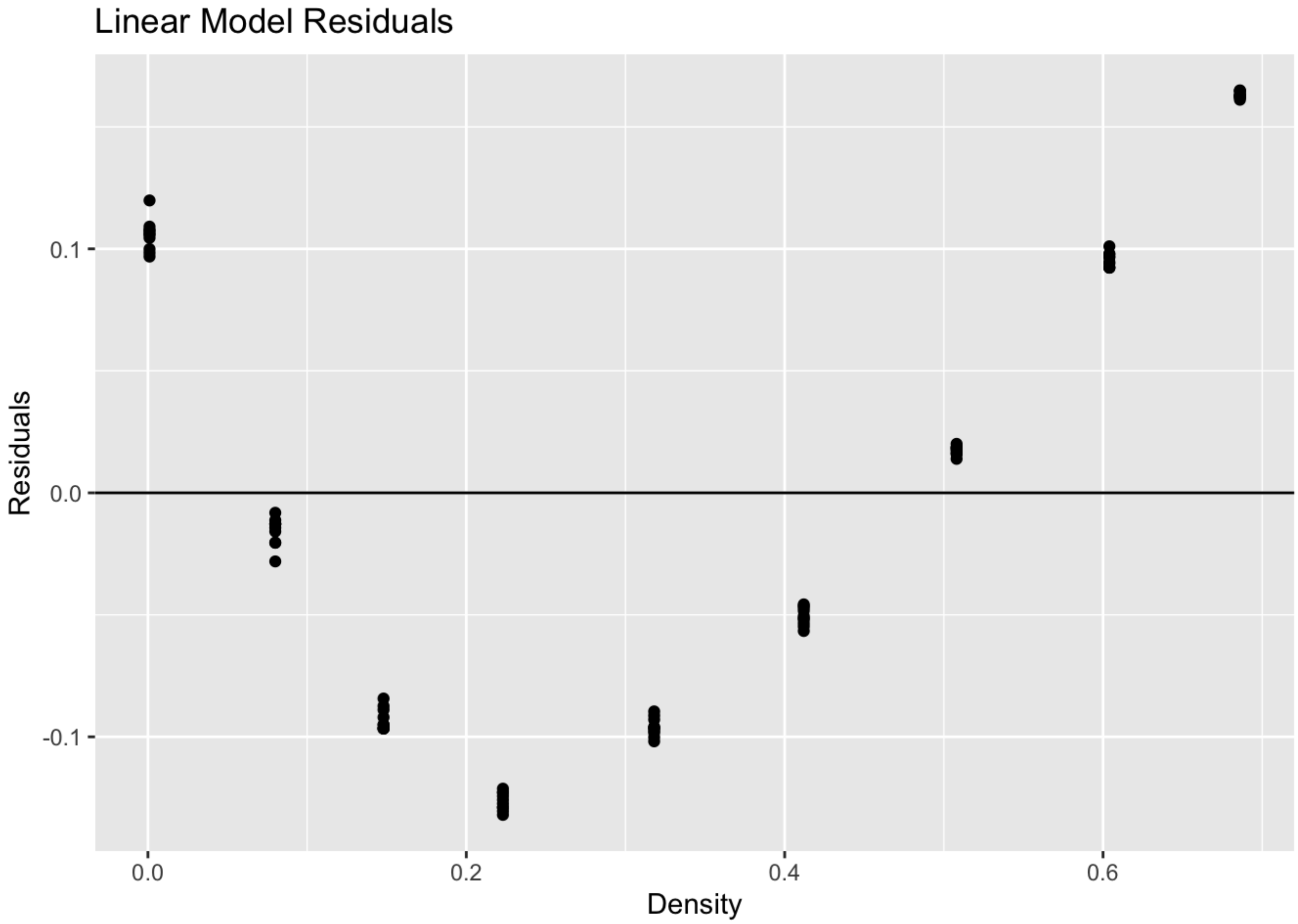

```
fit.linear <- lm(density~I(gain))
fit.log <- lm(density~I(log(gain)))

summary(fit.linear)
```

```
##
## Call:
## lm(formula = density ~ I(gain))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13198 -0.09452 -0.01354  0.09682  0.16495
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5497239  0.0151243   36.35   <2e-16 ***
## I(gain)     -0.0015334  0.0000777  -19.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09769 on 88 degrees of freedom
## Multiple R-squared:  0.8157, Adjusted R-squared:  0.8136
## F-statistic: 389.5 on 1 and 88 DF,  p-value: < 2.2e-16
```

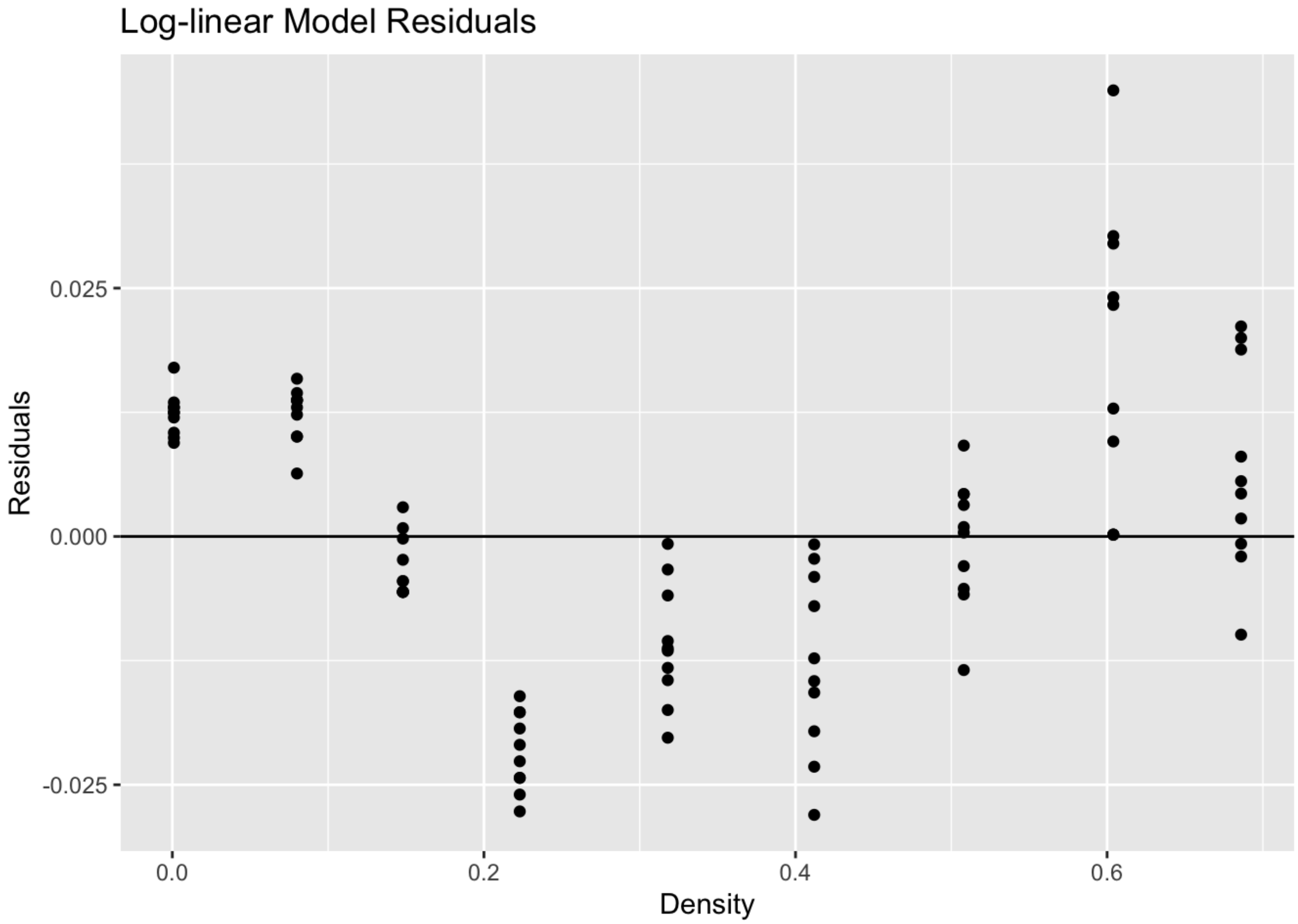
```
summary(fit.log)
```

```
##
## Call:
## lm(formula = density ~ I(log(gain)))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.028031 -0.011079 -0.000018  0.011595  0.044911
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.298013   0.006857   189.3   <2e-16 ***
## I(log(gain)) -0.216203   0.001494  -144.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01471 on 88 degrees of freedom
## Multiple R-squared:  0.9958, Adjusted R-squared:  0.9958
## F-statistic: 2.096e+04 on 1 and 88 DF,  p-value: < 2.2e-16
```

```
ggplot(fit.linear, aes(density, .resid)) +
geom_point() +
geom_hline(yintercept = 0) +
labs(x = "Density", y = "Residuals", title = "Linear Model Residuals")
```



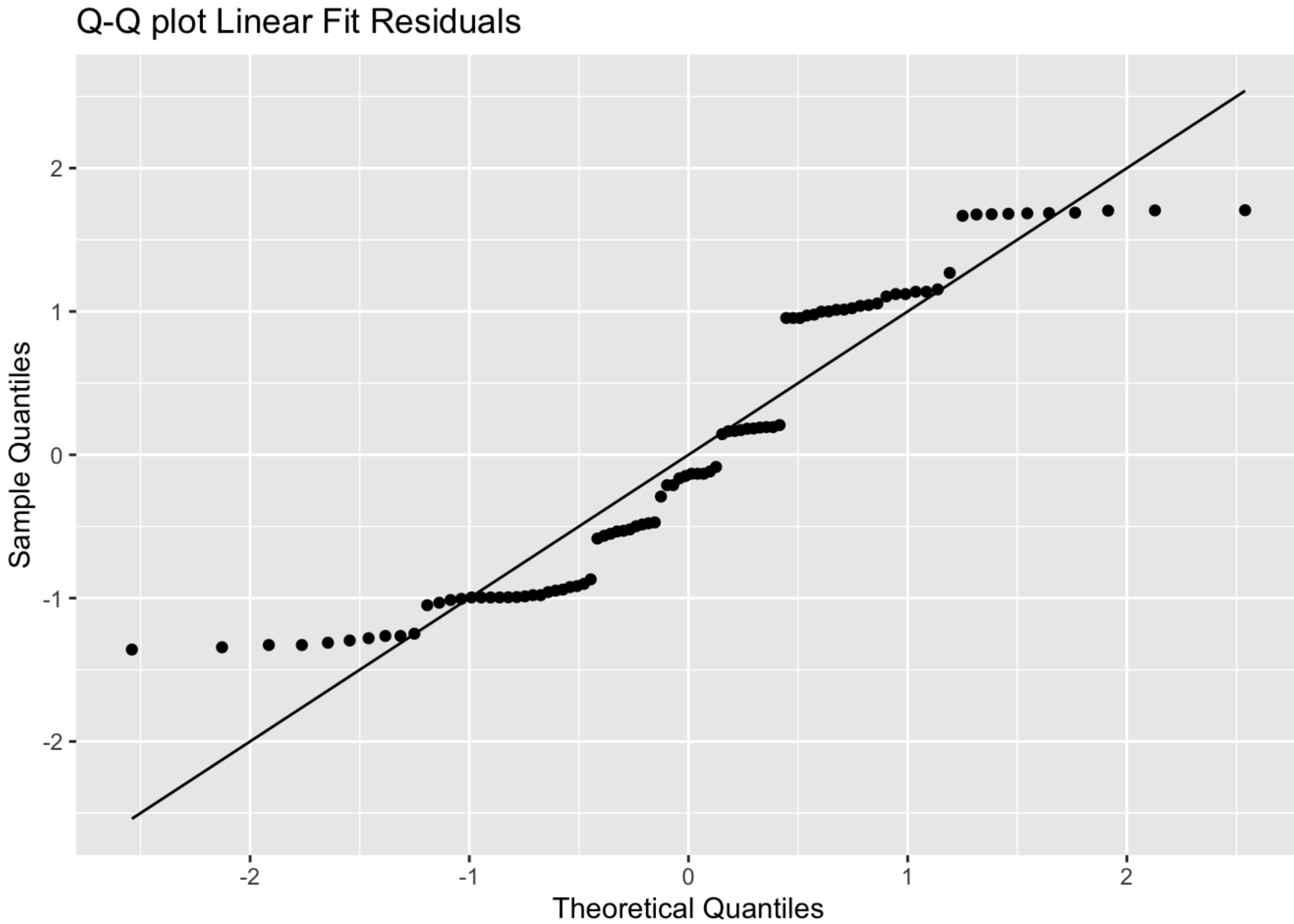

```
ggplot(fit.log, aes(density, .resid)) +
geom_point() +
geom_hline(yintercept = 0) +
labs(x = "Density", y = "Residuals", title = "Log-linear Model Residuals")
```



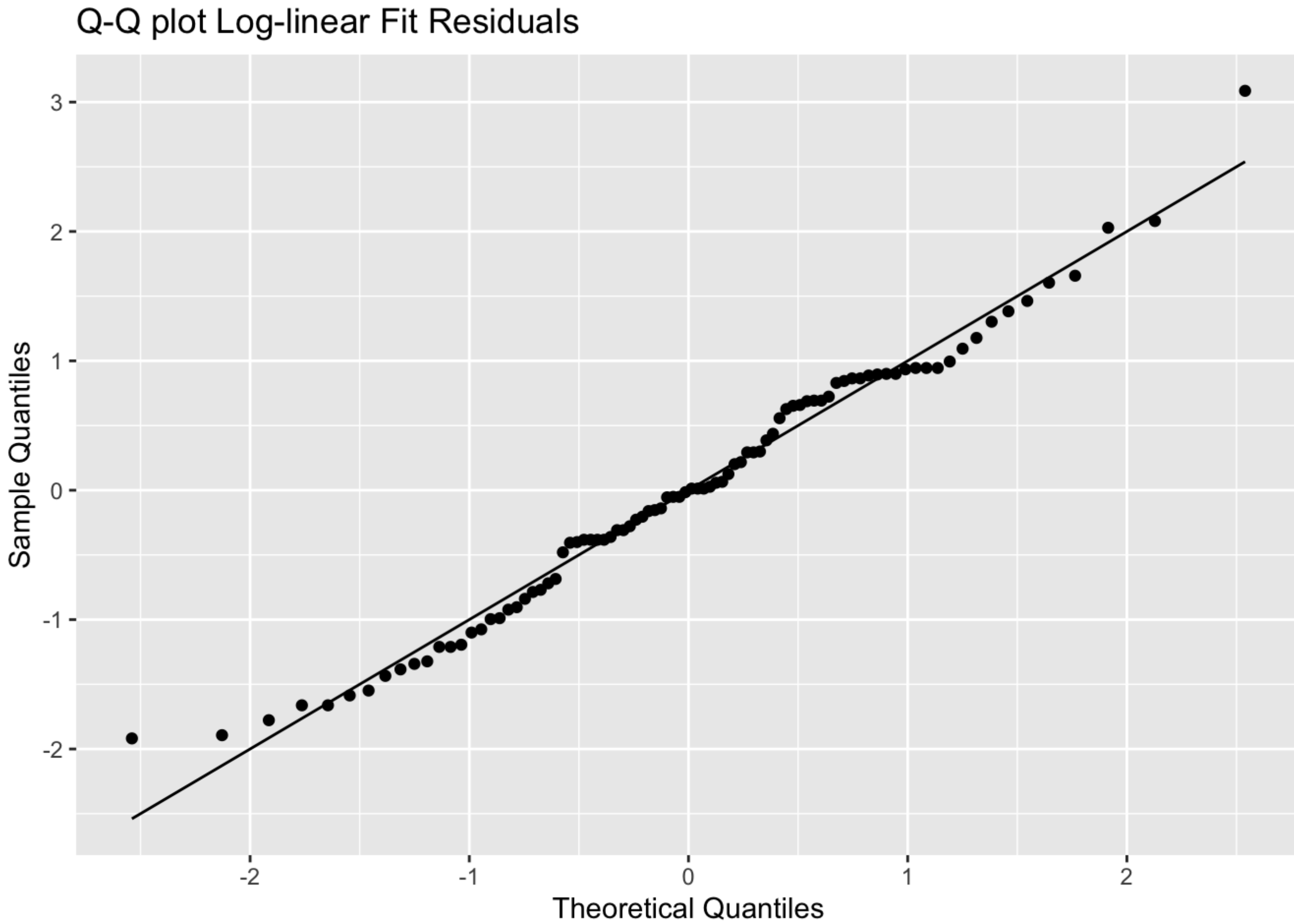

```
res.linear <- as.numeric(residuals(fit.linear))
res.log <- as.numeric(residuals(fit.log))

lin <- fortify(fit.linear)
sres.linear <- as.numeric(lin$.stdresid)

log <- fortify(fit.log)
sres.log <- as.numeric(log$.stdresid)
```

```
qqplot <- function(y, distribution = qnorm, tit) {
  require(ggplot2)
  x <- distribution(ppoints(y))
  df <- data.frame(Theoretical = x, Sample = sort(y))
  plot <- ggplot(df, aes(x = Theoretical, y = Sample)) +
    geom_point() +
    geom_line(aes(x = x, y = x)) +
    labs(title = tit, x = 'Theoretical Quantiles', y = 'Sample Quantiles')
  return(plot)
}
```

```
qqplot(sres.linear, tit = "Q-Q plot Linear Fit Residuals")
```




```
qqplot(sres.log, tit = "Q-Q plot Log-linear Fit Residuals")
```



2.

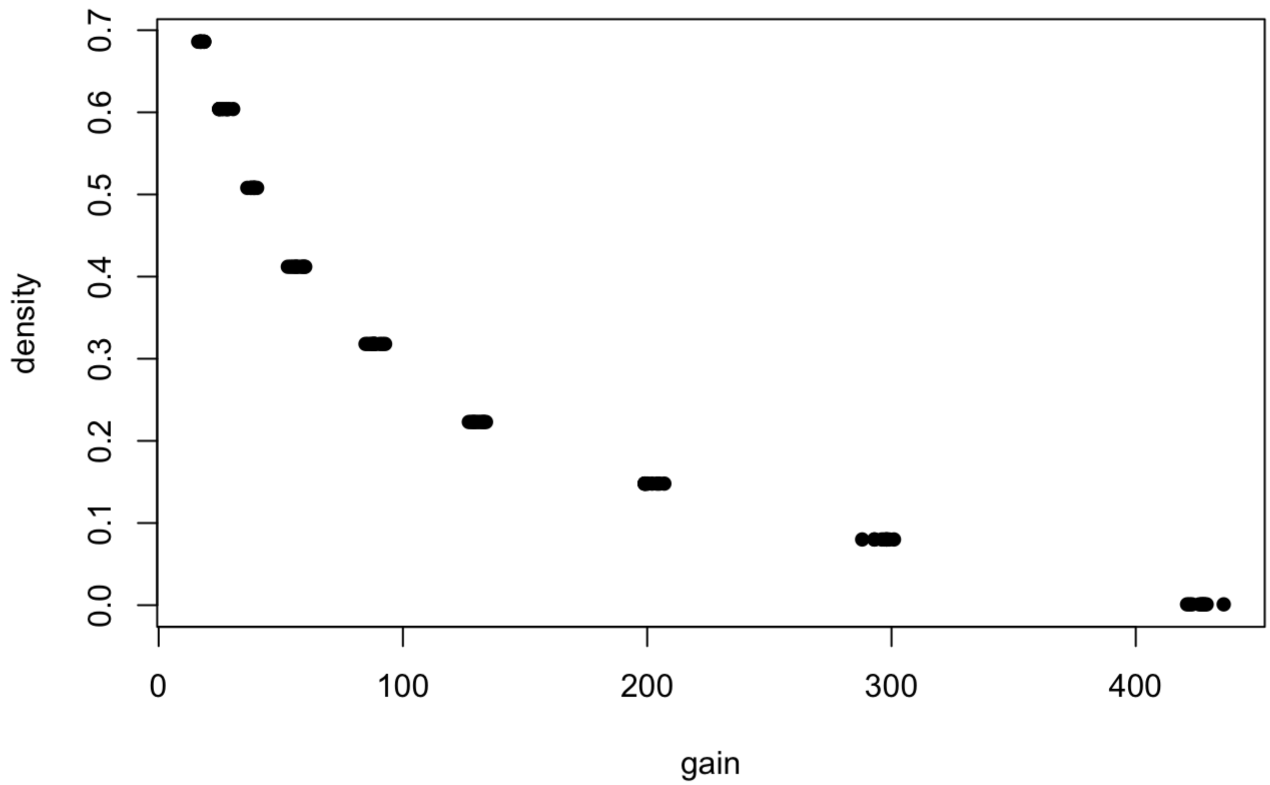
Predicting: Ultimately we are interested in answering questions such as: Given a gain reading of 38.6, what is the density of the snow-pack? or Given a gain reading of 426.7, what is the density of snow-pack? These two numeric values, 38.6 and 426.7, were chosen because they are the average gains for the 0.508 and 0.001 densities, respectively.

** Develop a procedure for adding CIs around your least squares line that can be used to make interval estimates for the snow-pack density from gain measurements. Keep in mind how the data were collected: several measurements of gain were taken for polyenthylene blocks of known density.

```
data <- read.table("gauge.txt", header=TRUE)
head(data)
```

```
##   density gain
## 1   0.686 17.6
## 2   0.686 17.3
## 3   0.686 16.9
## 4   0.686 16.2
## 5   0.686 17.1
## 6   0.686 18.5
```

```
gain = data$gain
density = data$density
plot(gain, density, pch=16)
```

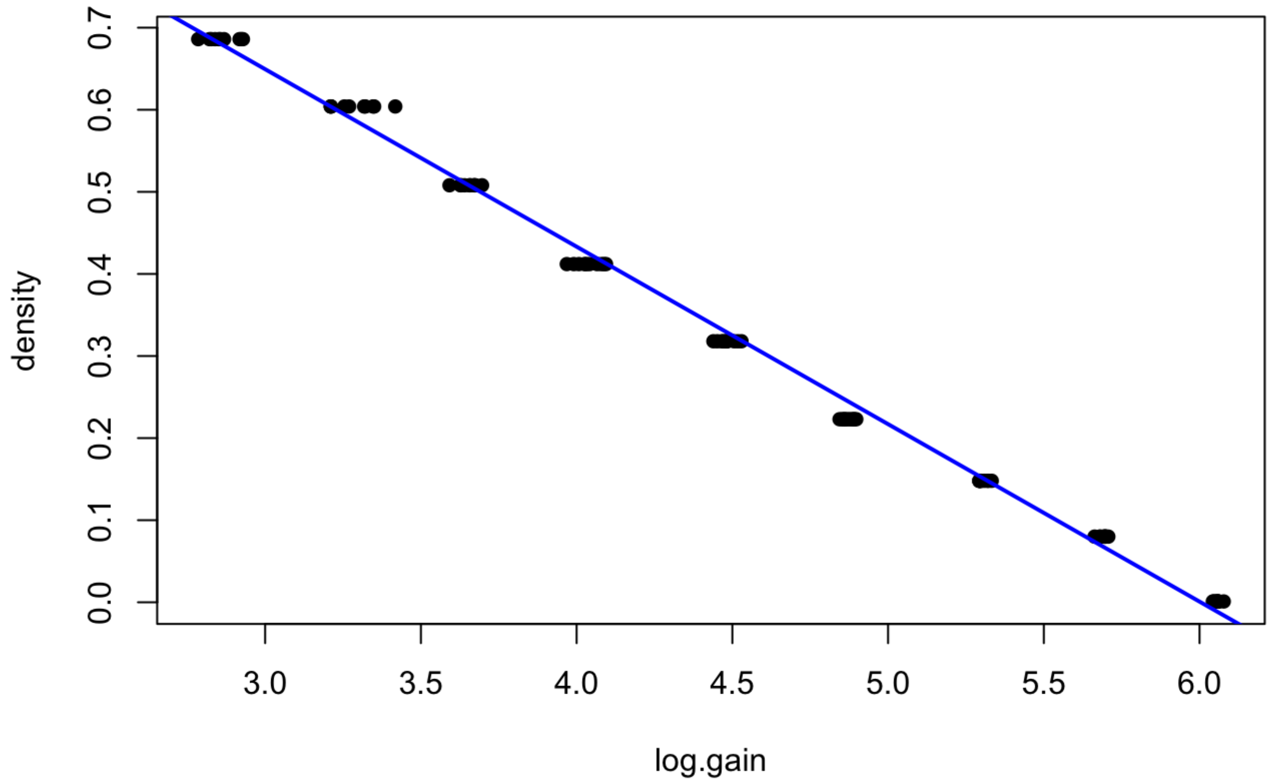



```
log.gain = log(data$gain)
lm2 = lm(density~log.gain, data = data)
summary(lm2)
```

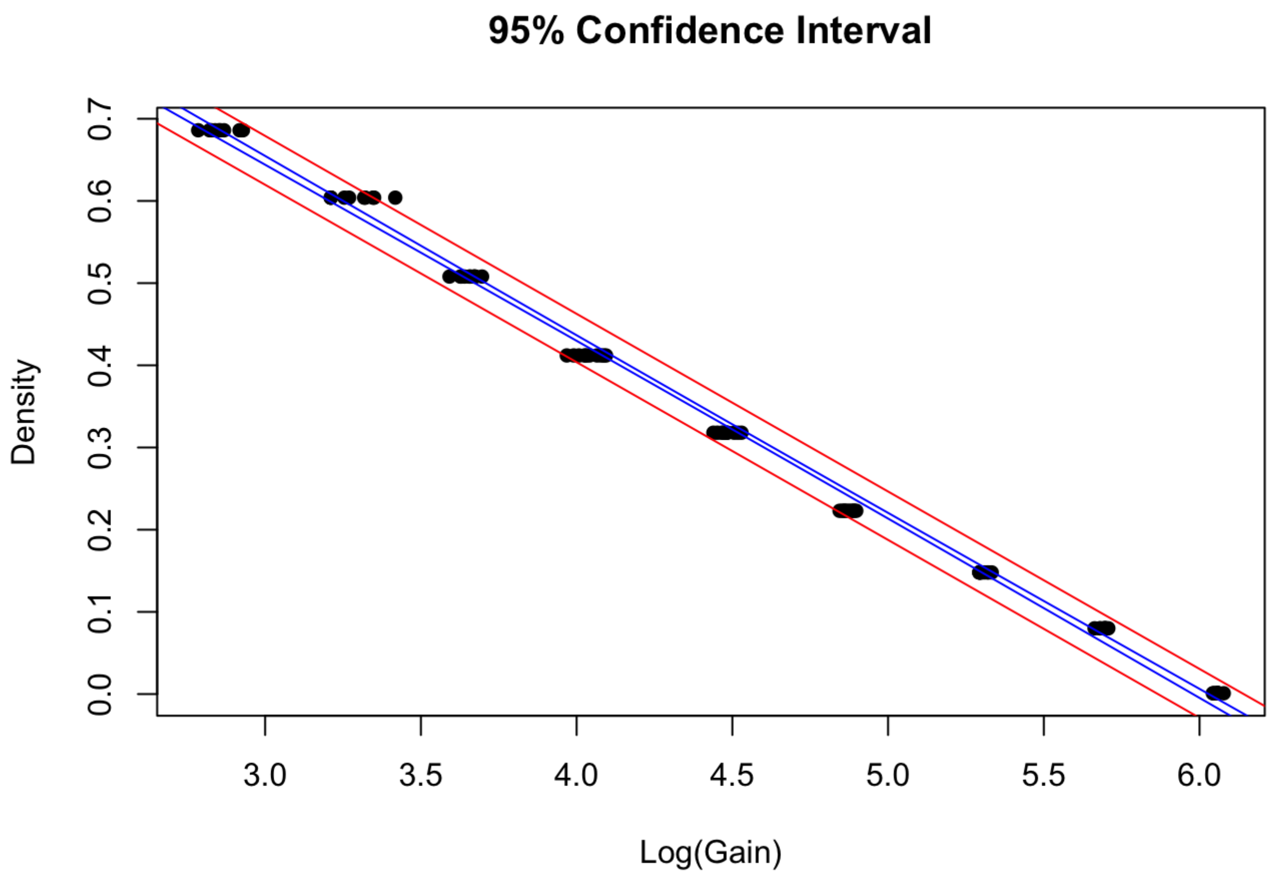
```
##
## Call:
## lm(formula = density ~ log.gain, data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.028031 -0.011079 -0.000018  0.011595  0.044911
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.298013   0.006857   189.3   <2e-16 ***
## log.gain    -0.216203   0.001494  -144.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01471 on 88 degrees of freedom
## Multiple R-squared:  0.9958, Adjusted R-squared:  0.9958
## F-statistic: 2.096e+04 on 1 and 88 DF,  p-value: < 2.2e-16
```

In such case, our remedy is polynomial regression. To start, we will first look at a degree 1 and 2 examples.

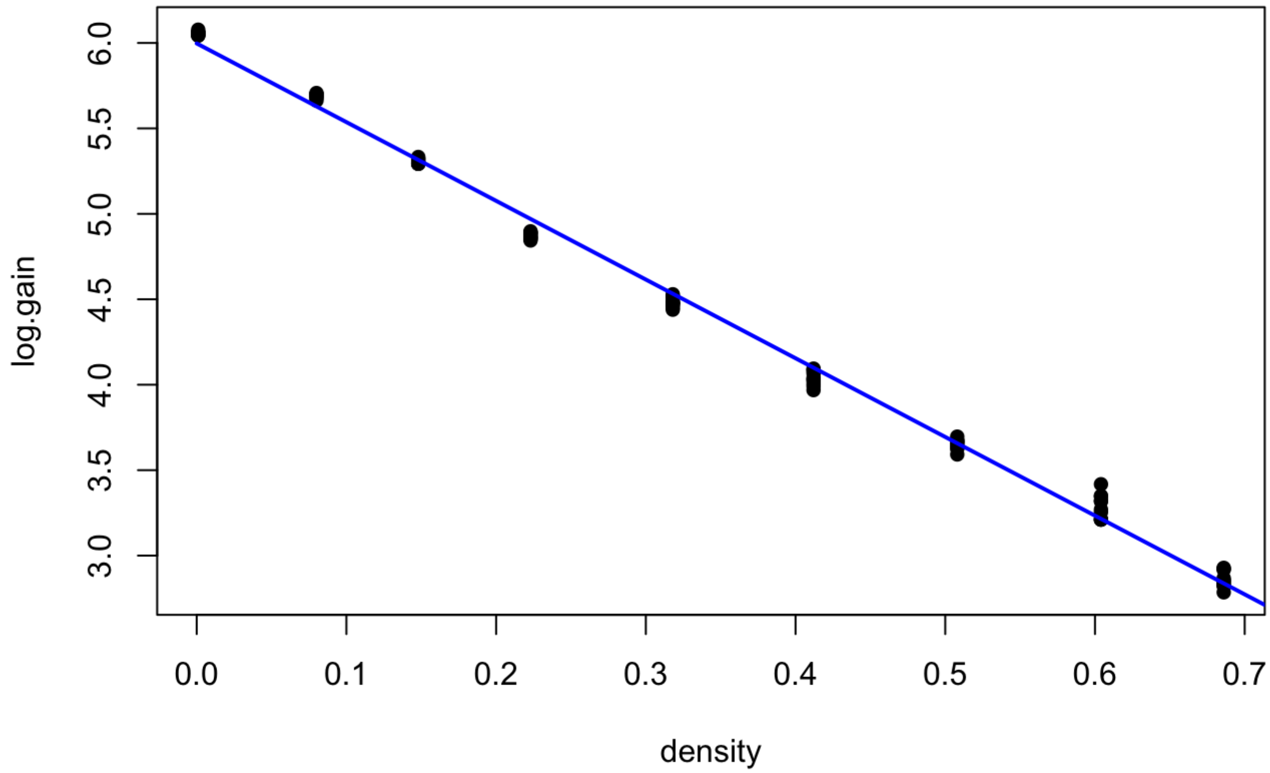
```
fit.d1 <- lm(density~log.gain, data = data)
pts <- seq(0, 8, length.out=90)
val.d1 <- predict(fit.d1, data.frame(log.gain=pts))
plot(log.gain, density, pch=16)
lines(pts, val.d1, col="blue", lwd=2)
```



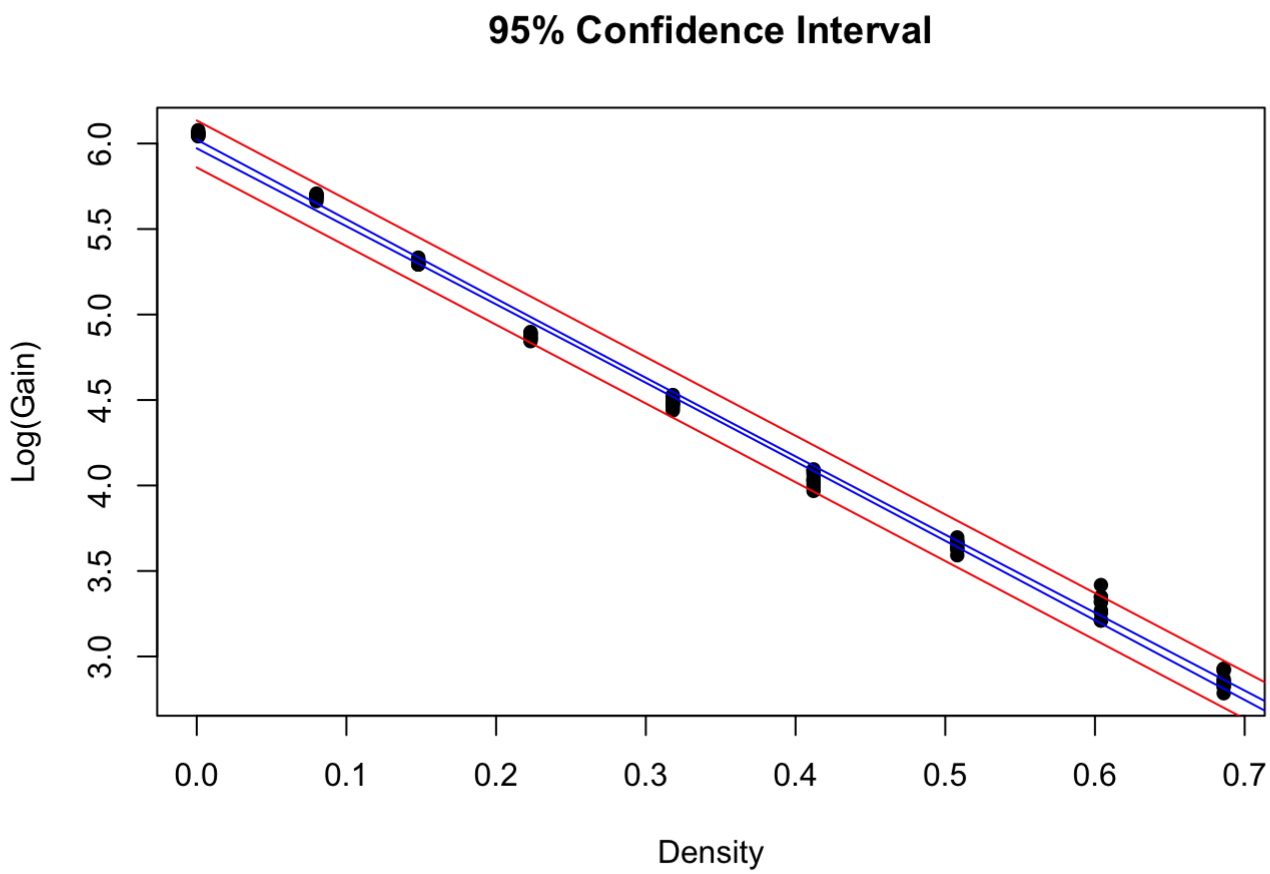

```
CI.conf <- predict(fit.d1, data.frame(log.gain=pts), interval = "confidence") #confidenc
e interval
CI.pred <- predict(fit.d1, data.frame(log.gain=pts), interval = "predict") #prediction i
nterval
plot(log.gain, density, pch=16, xlab='Log(Gain)', ylab='Density', main='95% Confidence I
nterval')
#lines(pts, CI.conf[,"fit"], col="black", lwd=2)
lines(pts, CI.conf[,"lwr"], col="blue", lwd=1)
lines(pts, CI.conf[,"upr"], col="blue", lwd=1)

lines(pts, CI.pred[,"lwr"], col="red", lwd=1)
lines(pts, CI.pred[,"upr"], col="red", lwd=1)
```




```
fit.d1 <- lm(log.gain~density, data = data)
pts <- seq(0, 8, length.out=90)
val.d1 <- predict(fit.d1, data.frame(density=pts))
plot(density, log.gain, pch=16)
lines(pts, val.d1, col="blue", lwd=2)
```




```
CI.conf <- predict(fit.d1, data.frame(density=pts), interval = "confidence") #confidence 
interval
CI.pred <- predict(fit.d1, data.frame(density=pts), interval = "predict") #prediction in
terval
plot(density, log.gain, pch=16, xlab='Density', ylab='Log(Gain)', main='95% Confidence I
nterval')
#lines(pts, CI.conf[,"fit"], col="black", lwd=2)
lines(pts, CI.conf[,"lwr"], col="blue", lwd=1)
lines(pts, CI.conf[,"upr"], col="blue", lwd=1)

lines(pts, CI.pred[,"lwr"], col="red", lwd=1)
lines(pts, CI.pred[,"upr"], col="red", lwd=1)
```




```
newdata = data.frame(density=0.6860)
predict(fit.d1, newdata, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 2.837593 2.811023 2.864163
```

```
newdata = data.frame(density=0.6040)
predict(fit.d1, newdata, interval = "confidence")
```

```
##       fit      lwr      upr
## 1 3.21528 3.192916 3.237644
```

```
newdata = data.frame(density=0.5080)
predict(fit.d1, newdata, interval = "confidence")
```

```
##       fit      lwr      upr
## 1 3.65745 3.639352 3.675547
```

```
newdata = data.frame(density=0.4120)
predict(fit.d1, newdata, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 4.099619 4.084501 4.114738
```

```
newdata = data.frame(density=0.3180)
predict(fit.d1, newdata, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 4.532578 4.518326 4.546829
```

```
newdata = data.frame(density=0.2230)
predict(fit.d1, newdata, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 4.970142 4.954357 4.985926
```

```
newdata = data.frame(density=0.1480)
predict(fit.d1, newdata, interval = "confidence")
```

```
##        fit      lwr     upr
## 1 5.315587 5.297244 5.33393
```

```
newdata = data.frame(density=0.0800)
predict(fit.d1, newdata, interval = "confidence")
```

```
##        fit      lwr     upr
## 1 5.628791 5.607471 5.65011
```

```
newdata = data.frame(density=0.0010)
predict(fit.d1, newdata, interval = "confidence")
```

```
##       fit      lwr     upr
## 1 5.99266 5.967399 6.01792
```

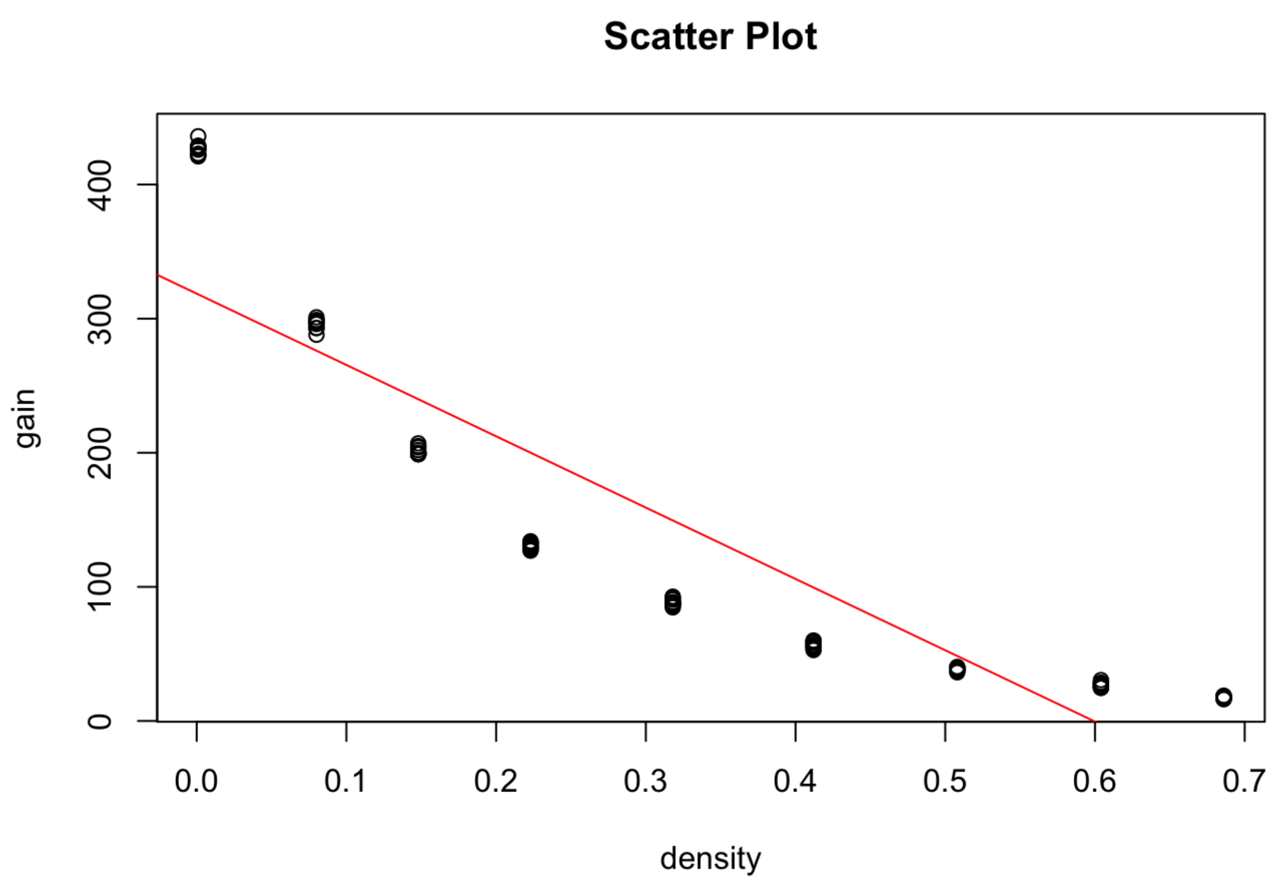
Advanced Analysis

```
data <- read.table("gauge.txt",header=TRUE)
head(data)
```

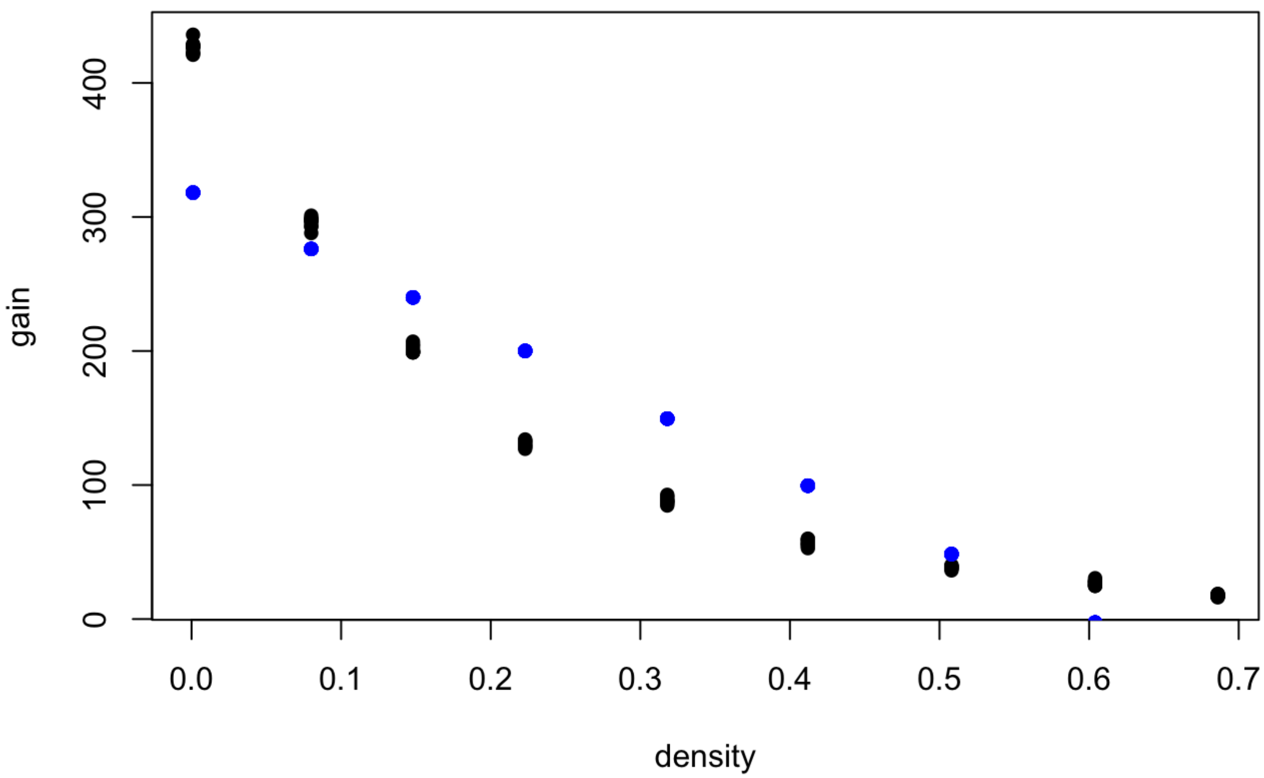
```
##   density gain
## 1   0.686 17.6
## 2   0.686 17.3
## 3   0.686 16.9
## 4   0.686 16.2
## 5   0.686 17.1
## 6   0.686 18.5
```

Including Plots

```
plot(data, main ="Scatter Plot")

#Fit linear model using OLS
model=lm(data$gain~data$density,data)

#Overlay best-fit line on scatter plot
abline(model,col='red')
```



```
#Scatter Plot
plot (data, pch=16)

#Predict Y using Linear Model
predY <- predict (model, data)

#Overlay Predictions on Scatter Plot
points (data$density, predY, col = "blue", pch=16)
```




```
#Install Package
library(hydroGOF)
```

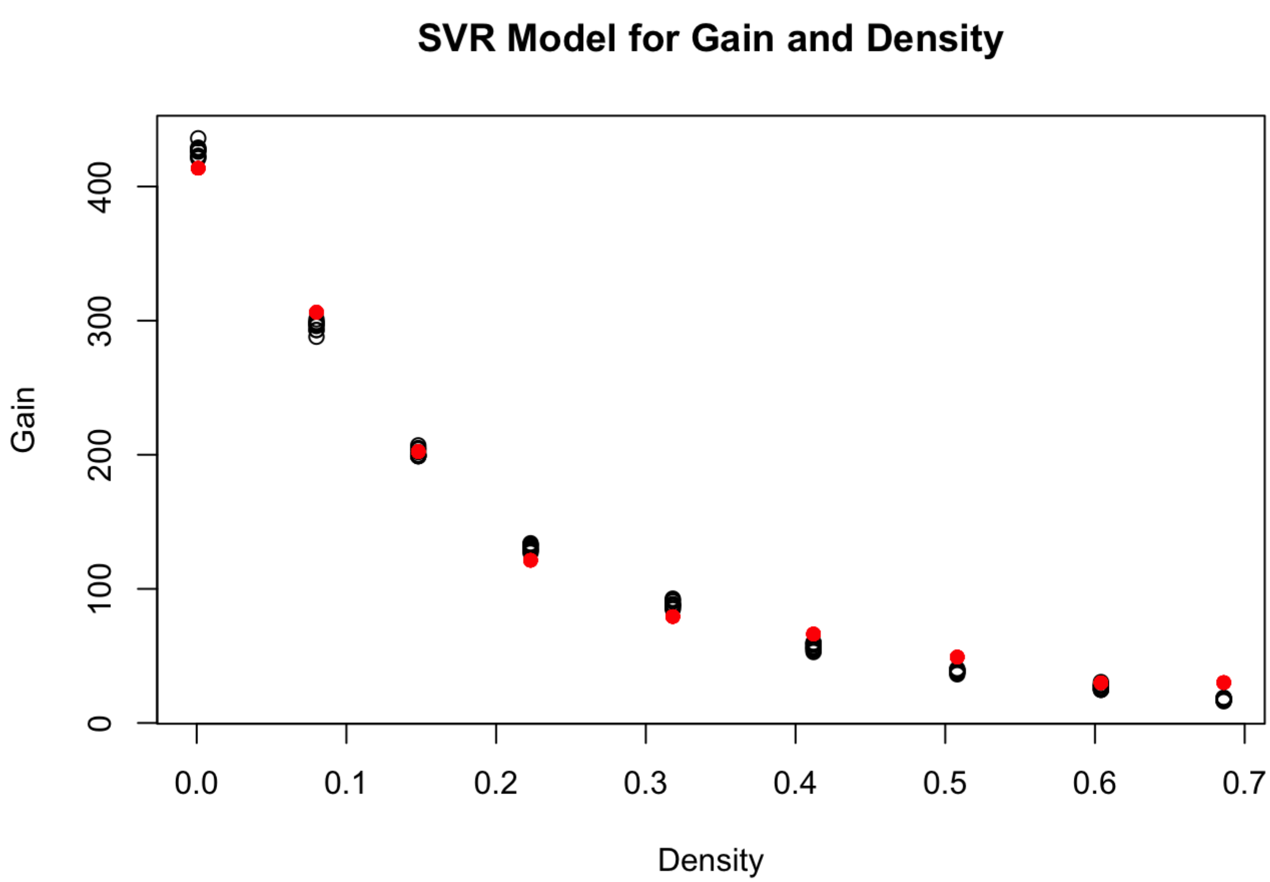
```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
#Calculate RMSE
RMSE=rmse(predY,data$gain)
```

```
## Fit SVR model and visualize using scatter plot

#Install Package
#install.packages("e1071")

#Load Library
library(e1071)

#Scatter Plot
plot(data, xlab='Density', ylab='Gain', main='SVR Model for Gain and Density')

#Regression with SVM
modelsvm = svm(data$gain~data$density,data)

#Predict using SVM regression
predYsvm = predict(modelsvm, data)

#Overlay SVM Predictions on Scatter Plot
points(data$density, predYsvm, col = "red", pch=16)
```



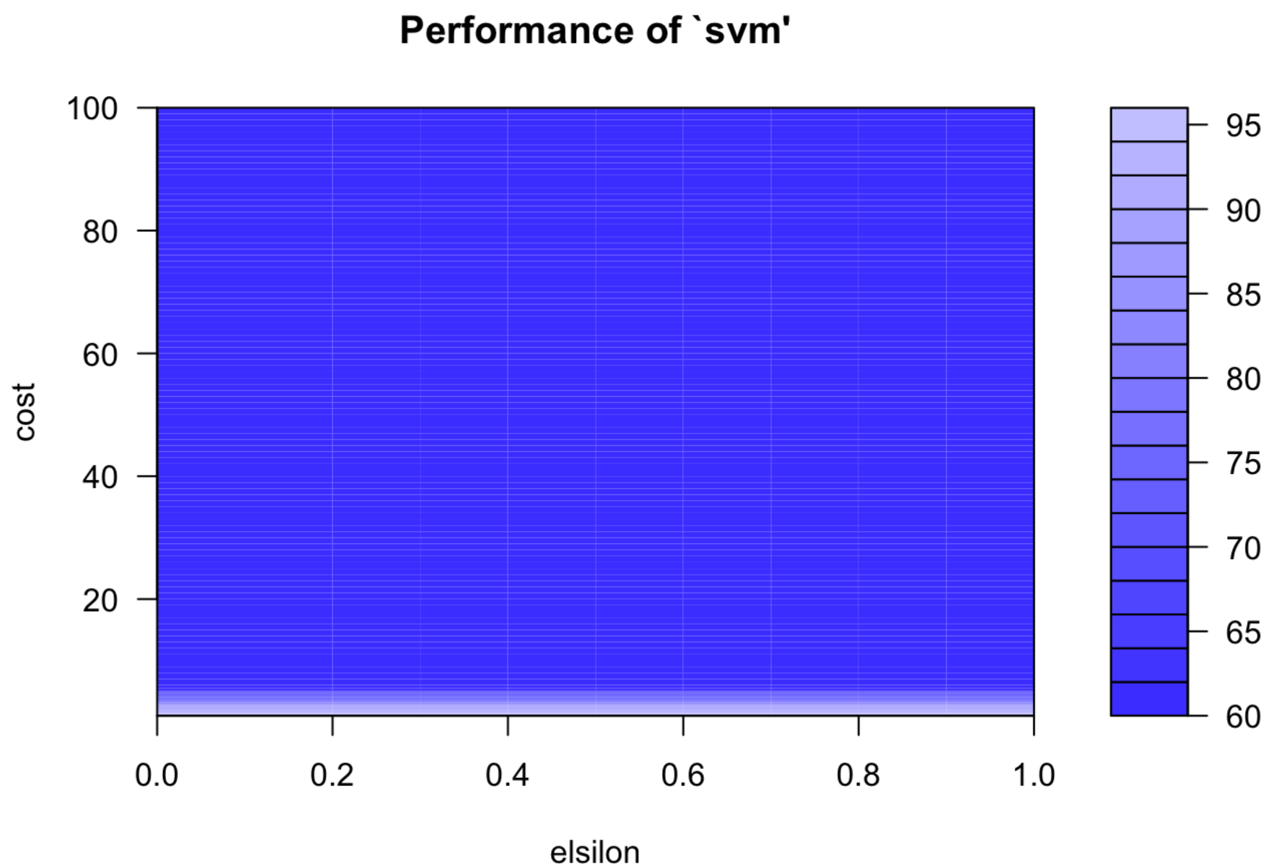
```
##Calculate parameters of the SVR model

#Find value of W
W = t(modelsvm$coefs) %*% modelsvm$SV

#Find value of b
b = modelsvm$rho
```

```
#Calculate RMSE
RMSEsvm=rmse(predYsvm,data$gain)
```

```
## Tuning SVR model by varying values of maximum allowable error and cost parameter

#Tune the SVM model
OptModelsvm=tune(svm, data$gain~data$density, data=data,ranges=list(elsilon=seq(0,1,0.1
), cost=1:100))

#Print optimum value of parameters
print(OptModelsvm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  elsilon cost
##        0    6
##
## - best performance: 61.81085
```

```
#Plot the perfrormance of SVM Regression model
plot(OptModelsvm)
```




```
## Select the best model out of 1100 trained models and compute RMSE

#Find out the best model
BstModel=OptModelsvm$best.model
BstModel
```

```
##
## Call:
## best.tune(method = svm, train.x = data$gain ~ data$density, data = data,
##     ranges = list(elsilon = seq(0, 1, 0.1), cost = 1:100))
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  6
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5
```

```
#Predict Y using best model
PredYBst=predict(BstModel,data)
PredYBst
```

```
##         1         2         3         4         5         6         7         8
##  29.56252  29.56252  29.56252  29.56252  29.56252  29.56252  29.56252  29.56252
##         9        10        11        12        13        14        15        16
##  29.56252  29.56252  25.85030  25.85030  25.85030  25.85030  25.85030  25.85030
##        17        18        19        20        21        22        23        24
##  25.85030  25.85030  25.85030  25.85030  43.82097  43.82097  43.82097  43.82097
##        25        26        27        28        29        30        31        32
##  43.82097  43.82097  43.82097  43.82097  43.82097  43.82097  66.24165  66.24165
##        33        34        35        36        37        38        39        40
##  66.24165  66.24165  66.24165  66.24165  66.24165  66.24165  66.24165  66.24165
##        41        42        43        44        45        46        47        48
##  84.14322  84.14322  84.14322  84.14322  84.14322  84.14322  84.14322  84.14322
##        49        50        51        52        53        54        55        56
##  84.14322  84.14322 120.61368 120.61368 120.61368 120.61368 120.61368 120.61368
##        57        58        59        60        61        62        63        64
## 120.61368 120.61368 120.61368 120.61368 195.28231 195.28231 195.28231 195.28231
##        65        66        67        68        69        70        71        72
## 195.28231 195.28231 195.28231 195.28231 195.28231 195.28231 301.36250 301.36250
##        73        74        75        76        77        78        79        80
## 301.36250 301.36250 301.36250 301.36250 301.36250 301.36250 301.36250 301.36250
##        81        82        83        84        85        86        87        88
## 422.64674 422.64674 422.64674 422.64674 422.64674 422.64674 422.64674 422.64674
##        89        90
## 422.64674 422.64674
```

```
#Calculate RMSE of the best model
RMSEBst=rmse(PredYBst,data$gain)
RMSEBst
```

```
## [1] 7.611187
```

```
##Calculate parameters of the Best SVR model

#Find value of W
W = t(BstModel$coefs) %*% BstModel$SV

#Find value of b
b = BstModel$rho
```

```
## Plotting SVR Model and Tuned Model in same plot

plot(data, pch=16, xlab='Density', ylab='Gain', main='SVR Model vs Tuned SVR Model')
points(data$density, predYsvm, col = "blue", pch=3)
points(data$density, PredYBst, col = "red", pch=4)
points(data$density, predYsvm, col = "blue", pch=3, type="l")
points(data$density, PredYBst, col = "red", pch=4, type="l")
```