UC Berkeley
Stat 135 - Lab4
Beckah Ming-Wai Chan

Introduction

The main source of water for Northern California comes from the Sierra Nevada mountains. To help monitor this water supply, the Forest Service of the United States Department of Agriculture (USDA) operates a gamma transmission snow gauge in the Central Sierra Nevada near Soda Springs, California. The gauge is used to determine a depth profile of snow density.

The snow gauge does not disturb the snow in the measurement process, which means the same snow-pack can be measured over and over again. When rain falls on snow, the snow absorbs the water up to a certain point, after which flooding occurs. The denser the snow-pack, the less water the snow can absorb. Analysis of the snow-pack profile may help with monitoring the water supply and flood management.

The gauge doe snot directly measure snow density. The density reading is converted from a measurement of gamma ray emissions. To adjust the conversion method, a calibration run is made each year at the beginning of the winter season. In this lab, we will develop a procedure to calibrate the snow gauge.

The Data

Density	Gain									
0.686	17.6	17.3	16.9	16.2	17.1	18.5	18.7	17.4	18.6	16.8
0.604	24.8	25.9	26.3	24.8	24.8	27.6	28.5	30.5	28.4	27.7
0.508	39.4	37.6	38.1	37.7	36.3	38.7	39.4	38.8	39.2	40.3
0.412	60.0	58.3	59.6	59.1	56.3	55.0	52.9	54.1	56.9	56.0
0.318	87.0	92.7	90.5	85.8	87.5	88.3	91.6	88.2	88.6	84.7
0.223	128	130	131	129	127	129	132	133	134	133
0.148	199	204	199	207	200	200	205	202	199	199
0.080	298	298	297	288	296	293	301	299	298	293
0.001	423	421	422	428	436	427	426	428	427	429

The data are from a calibration run of the USDA Forest Service's snow gauge located in the Central Sierra Nevada mountain range near Soda Springs, California. The run consists of placing polyethylene blocks of known densities between the two poles of the snow gauge and taking readings on the blocks. The polyethylene blocks are used to simulate snow.

For each block of polyethylene, 30 measurements were taken. Only the middle 10, in the order taken, are reported here. The measurements recorded by the gauge are an amplified version of the gamma photon count made by the detector. We call the gauge measurements the "gain".

Investigations

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.

Part I

For part I, we will use the data to fit gain to density.

```
dat <- read.csv("snow_gauge.csv", sep = "")
dat
```

```
##    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
## 7    0.686  18.7
## 8    0.686  17.4
## 9    0.686  18.6
## 10   0.686  16.8
```

Then, we find detailed information on the model's performance and coefficients.

```
#predictor; x; independent
gain <- dat$gain
#dependent; y
density <- dat$density

#Fit gain to density
l_r <- lm(density ~ gain)
#Review the results
summary(l_r)
```

```
##
## Call:
## lm(formula = density ~ 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 ***
## 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
```

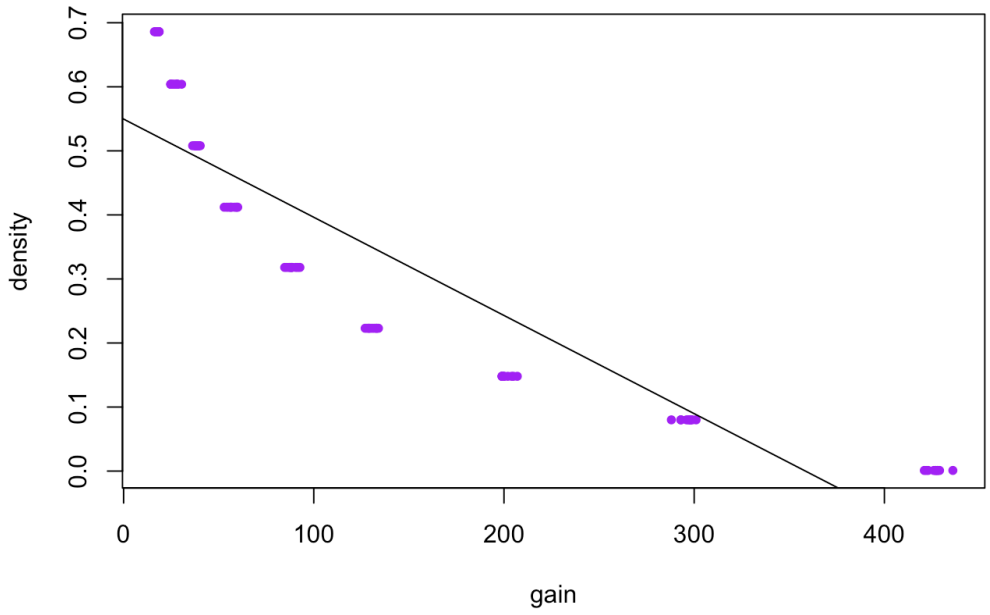
According to the detailed information above, we find the values of the intercept and the slope for gain, which is our predictor. These values plot a line between all the points of the data. The value of the intercept is 0.5497239, and the value of the slope is -0.0015334. Therefore, we have:

$$Density = intercept + Gain * slope$$

So in this case, if gain is 17.6, intercept is 0.5497239 and slope is -0.0015334, the model predicts on average that its density is around:

$$Density = 0.5497239 + 17.6 * -0.0015334$$
$$Density = 0.5227361$$

Next, we plot the data and add a least squares line we built with our linear model to our scatter plot.



We also find the correlation between density and gain, which is -0.9031597.

```
#correlation
cor(gain, density)
```

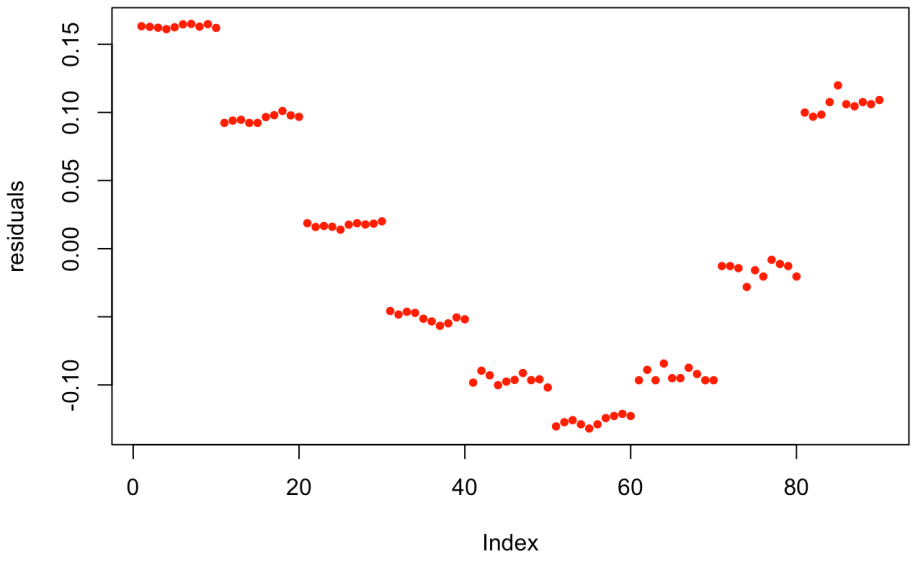
```
## [1] -0.9031597
```

A good way to test the quality of the fit of the model is to look at the residuals.

```
Residual standard error: 0.09769 on 88 degrees of freedom
Multiple R-squared:  0.8157,    Adjusted R-squared:  0.8136
```

We plot the residuals. Ideally, when you plot the residuals, they should look random. Otherwise means that maybe there is a hidden pattern that the linear model is not considering.

```
#Plot the residuals
plot(l_r$residuals, pch = 20, col = "red", ylab="residuals")
```



In the above plot, notice that there is a pattern, like a curve on the residuals. This is not random at all.

Therefore, we use the data to fit a transformation of gain to density. We use a quadratic term, and get a new detailed information on the new model's performance and coefficients.

```
#Fit a transformation of gain to density
#Try with a quadratic term

l_r2 <- lm(density ~ gain + I(gain^2))
summary(l_r2)
```

```
##
## Call:
## lm(formula = density ~ gain + I(gain^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07086 -0.04343 -0.02213  0.03776  0.08513
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.708e-01  1.110e-02   60.45   <2e-16 ***
## gain        -3.844e-03  1.549e-04  -24.82   <2e-16 ***
## I(gain^2)    5.499e-06  3.557e-07   15.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05076 on 87 degrees of freedom
## Multiple R-squared:  0.9508, Adjusted R-squared:  0.9497
## F-statistic: 840.7 on 2 and 87 DF,  p-value: < 2.2e-16
```

Notice that the model improved significantly.

gain to density:

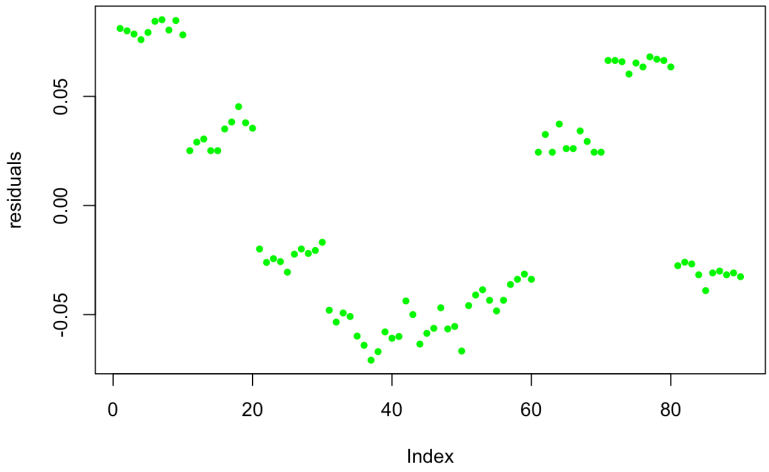
```
## Residual standard error: 0.09769 on 88 degrees of freedom
## Multiple R-squared:  0.8157, Adjusted R-squared:  0.8136
## F-statistic: 389.5 on 1 and 88 DF,  p-value: < 2.2e-16
```

a transformation of gain to density:

```
## Residual standard error: 0.05076 on 87 degrees of freedom
## Multiple R-squared:  0.9508, Adjusted R-squared:  0.9497
## F-statistic: 840.7 on 2 and 87 DF,  p-value: < 2.2e-16
```

Now, we plot the residuals of the new model. We do not see any clear patterns on our residuals, which is good.

```
#Plot the residuals of the new model
plot(l_r2$residuals, pch = 20, col = "green", ylab="residuals")
```



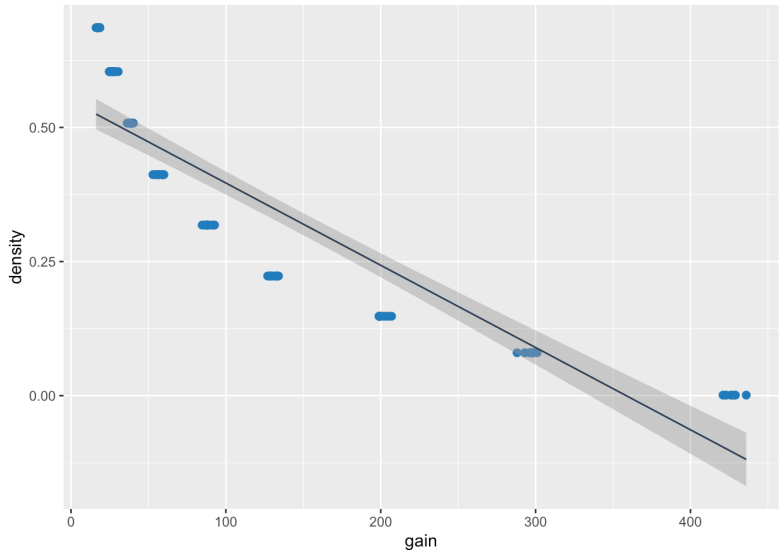
Part II

For part II, we will develop a procedure for adding bands around our least squares line that can be used to make interval estimates for snow-pack density from gain.

First, we plot the two variables as a scatterplot and draw a linear regression line through the points. Then, we add the gray bands around the line that represent the standard error of the regression line.

```
#Add the confidence bands around my linear regression line
ggplot(dat, aes(x=gain, y=density)) +
  geom_point(color="#2980B9", size = 2) +
  geom_smooth(method=lm, color="#2C3E50", size = 0.5)
```

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



Then, we find the confidence intervals and the prediction intervals.

```
#Confidence(Mean) intervals
predict(l_r, interval = "confidence")
```

```
##             fit          lwr          upr
## 1    0.52273598  0.49460919  0.55086276
## 2    0.52319600  0.49503742  0.55135458
## 3    0.52380936  0.49560832  0.55201041
## 4    0.52488275  0.49660722  0.55315827
## 5    0.52350268  0.49532287  0.55168249
## 6    0.52135591  0.49332428  0.54938754
## 7    0.52104923  0.49303870  0.54905976
## 8    0.52304266  0.49489468  0.55119064
## 9    0.52120257  0.49318149  0.54922364
## 10   0.52396270  0.49575103  0.55217438
```

```
#Prediction intervals
predict(l_r, interval = "prediction")
```

```
## Warning in predict.lm(l_r, interval =
```

```
##             fit          lwr          upr
## 1    0.52273598  0.32657158 0.71890038
## 2    0.52319600  0.32702704 0.71936496
## 3    0.52380936  0.32763430 0.71998442
## 4    0.52488275  0.32869696 0.72106853
## 5    0.52350268  0.32733067 0.71967469
## 6    0.52135591  0.32520513 0.71750669
## 7    0.52104923  0.32490146 0.71719699
## 8    0.52304266  0.32687522 0.71921010
## 9    0.52120257  0.32505330 0.71735184
## 10   0.52396270  0.32778611 0.72013929
```

Notice the fitted values in the prediction table are the same as the fitted values in the confidence table, but the intervals are wider in the prediction table. This is due to the additional term in the standard error of production. It should be noted prediction and confidence intervals are similar in that they are both predicting a response, however, they differ in what is being represented and interpreted.

We want to check how well our procedure works, so we omit the set of measurements for the density 0.508. Then, we apply our calibration procedure to the remaining data and provide an interval estimate for density.

Then, we get a new detailed information on the new model's performance and coefficients.

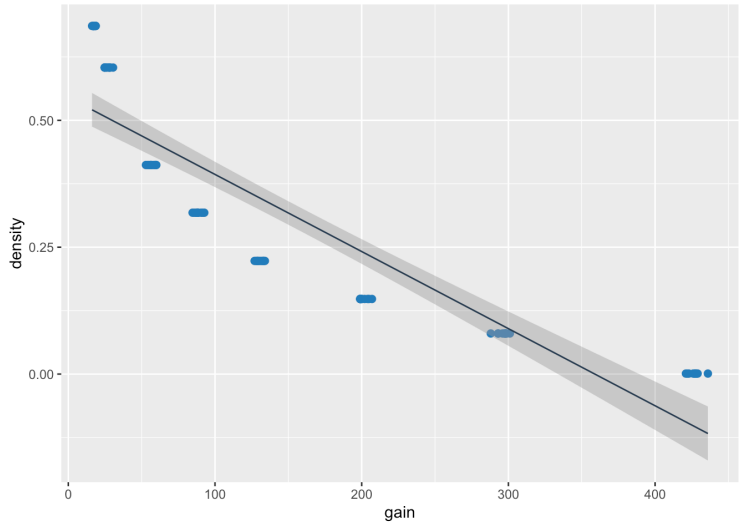
```
new_gain <- new_dat$gain
new_density <- new_dat$density

#Fit gain to density
new_fit <- lm(new_density ~ new_gain)
#Review the results
summary(new_fit)
```

```
##
## Call:
## lm(formula = new_density ~ new_gain)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.12941 -0.09340 -0.03499  0.10097  0.16903
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.454e-01  1.766e-02   30.89   <2e-16 ***
## new_gain    -1.519e-03  8.571e-05  -17.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1035 on 78 degrees of freedom
## Multiple R-squared:  0.8012, Adjusted R-squared:  0.7986
## F-statistic: 314.3 on 1 and 78 DF,  p-value: < 2.2e-16
```

We then plot the two variables as a scatterplot and draw a linear regression line through the points using our new data. We also add the gray bands around the line that represent the standard error of the regression line.

```
#Add the confidence bands around my linear regression line
ggplot(new_dat, aes(x=gain, y=density)) +
  geom_point(color="#2980B9", size = 2) +
  geom_smooth(method=lm, color="#2C3E50", size = 0.5)
```

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



Then, we find the confidence intervals and the prediction intervals based on our new data.

```
#Confidence(Mean) intervals
predict(new_fit, interval = "confidence")
```

```
##            fit         lwr         upr
## 1   0.51863989  0.48569517  0.55158461
## 2   0.51909573  0.48611441  0.55207705
## 3   0.51970352  0.48667333  0.55273371
## 4   0.52076715  0.48765129  0.55388302
## 5   0.51939963  0.48639388  0.55240537
## 6   0.51727236  0.48443722  0.55010750
## 7   0.51696847  0.48415763  0.54977931
## 8   0.51894378  0.48597467  0.55191290
## 9   0.51712042  0.48429743  0.54994340
## 10  0.51985547  0.48681305  0.55289788
```

```
#Prediction intervals
predict(new_fit, interval = "prediction")
```

```
## Warning in predict.lm(new_fit, interva
es
```

```
##            fit         lwr         upr
## 1   0.51863989  0.30990472 0.72737506
## 2   0.51909573  0.31035479 0.72783668
## 3   0.51970352  0.31095485 0.72845219
## 4   0.52076715  0.31200491 0.72952940
## 5   0.51939963  0.31065482 0.72814443
## 6   0.51727236  0.30855446 0.72599026
## 7   0.51696847  0.30825439 0.72568255
## 8   0.51894378  0.31020477 0.72768280
## 9   0.51712042  0.30840443 0.72583640
## 10  0.51985547  0.31110486 0.72860608
```

After omitting the set of measurements for the density 0.508, we noticed that we actually receive the same pattern based on our new dataset, which the fitted values in the prediction table are the same as the fitted values in the confidence table, but the intervals are wider in the prediction table.

For part II, we examined confidence and prediction intervals of predicted values from our linear regression models. We discovered that the fitted values in the prediction table are actually the same as the fitted values in the confidence table. More, since the prediction intervals must take into account the variability of the estimators for mean and standard deviation, the intervals will be wider than the ones in the confidence table.

Part III

Finally, for part III, the last part of this lab, we consider fitting a polynomial in density to the data and compare this model to the simple log-liner model we did previously.

When fitting polynomials, we use something like this:

$$y = a * q + b * q^2 + c * q^3$$

Then, we get a new detailed information on the new model's performance and coefficients.

```
#Fit a polynomial in density to the data
set.seed(20)

fit_poly <- lm(density ~ gain + I(gain^2) + I(gain^3))
summary(fit_poly)
```

```
##
## Call:
## lm(formula = density ~ gain + I(gain^2) + I(gain^3))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.050424 -0.020894  0.002485  0.018948  0.040488
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.681e-01  7.336e-03  104.70   <2e-16 ***
## gain        -7.009e-03  1.870e-04  -37.49   <2e-16 ***
## I(gain^2)    2.484e-05  1.071e-06   23.20   <2e-16 ***
## I(gain^3)   -2.964e-08  1.621e-09  -18.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0231 on 86 degrees of freedom
## Multiple R-squared:  0.9899, Adjusted R-squared:  0.9896
## F-statistic:  2818 on 3 and 86 DF,  p-value: < 2.2e-16
```

We then obtain the confidence intervals of the parameters of our model.

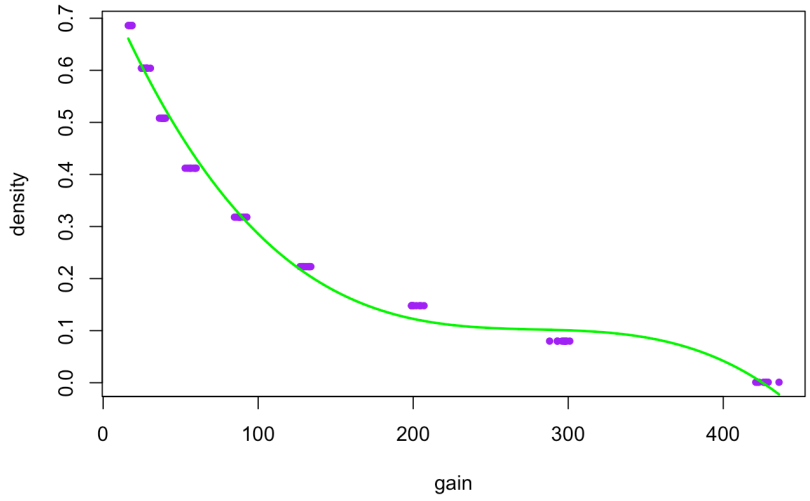
```
#Confidence intervals for the fit_ploy model paramaters
confint(fit_poly, level=0.95)
```

```
##                     2.5 %        97.5 %
## (Intercept)  7.535127e-01  7.826790e-01
## gain        -7.381093e-03 -6.637795e-03
## I(gain^2)    2.271283e-05  2.696904e-05
## I(gain^3)   -3.286173e-08 -2.641491e-08
```

Overall the model seems a good fit as the R squared of 0.9 indicates. The coefficients of the first and third order terms are statistically significant as we expected.

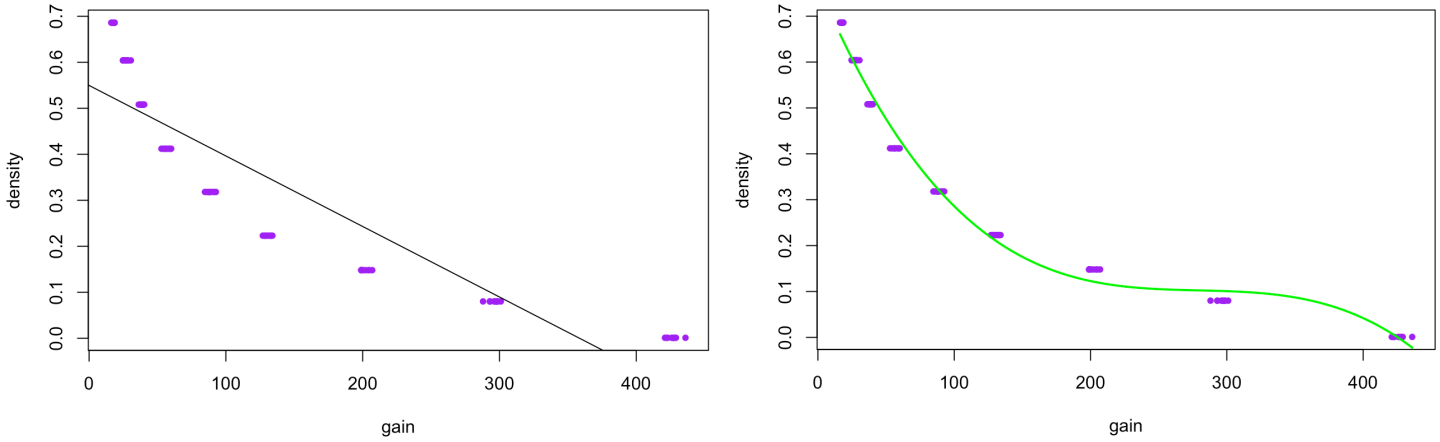
Now, we want to get the fitted values and the confidence intervals in order to add a polynomial line to our scatter plot.

```
x <- with(dat, seq(min(gain), max(gain), length.out = 2000))
y <- predict(fit_poly, newdata = data.frame(gain=x))
plot(density ~ gain, col = "purple", pch=20)
lines(x, y, col="green", lwd=2)
```



Conclusion

A linear regression model is a basic and commonly used type of predictive analysis which usually works on continuous data. On the other hand, a polynomial regression model provides the best approximation of the relationship between the dependent and independent variable.



In our case, the two variables (gain and density) in our dataset are correlated but the relationship does not look linear. Hence, a polynomial regression is a better fit for our data. If we try to use a simple linear regression in the above graph then the linear regression line will not fit very well. It is very difficult to fit a linear regression line in the above graph with a low value of error. As a result, we conclude that the polynomial regression is a better model for our case so that we can achieve a minimum error or minimum cost function.