Week 6: Logistic Regression

John Navarro

November 8, 2016

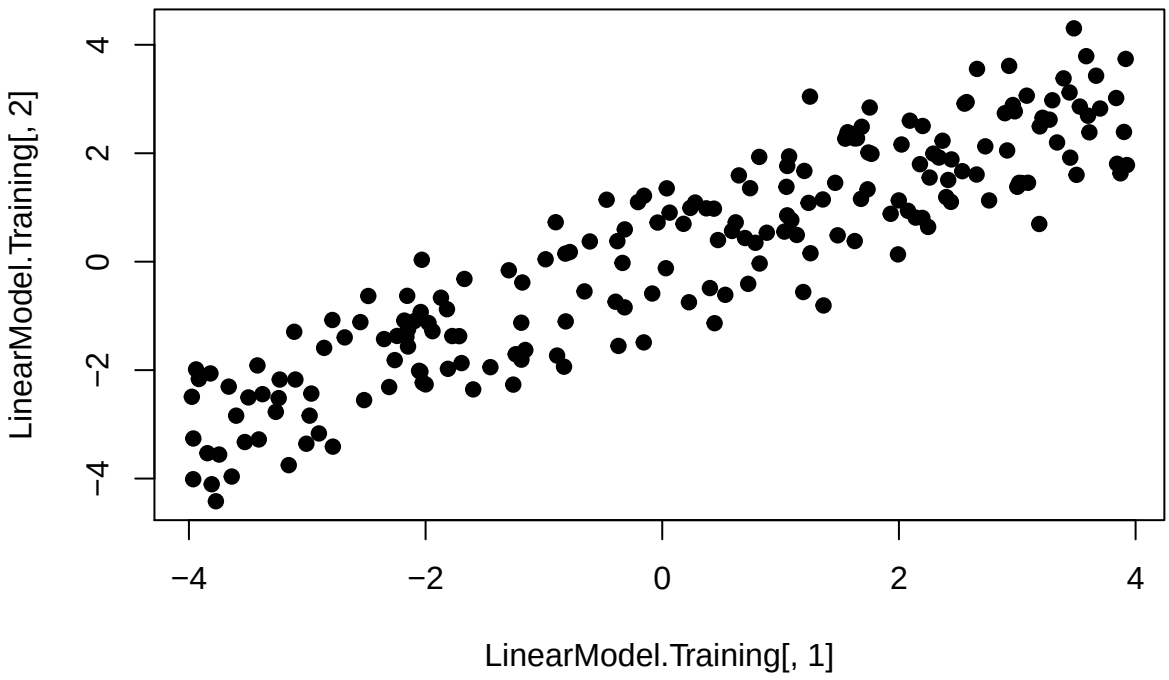
1. Separate mixed samples of linear model data using logistic regression

1.1. Analyze the training sample

```
##read file into LinearModel.Training, set nSample.Training, print head, plot the training sample
datapath <- "C:/Users/JohntheGreat/Documents/MSCA/StatisticalAnalysis/Week6/Assignments"
LinearModel.Training<-read.csv(file=paste(datapath,'ResidualAnalysisProjectData_1_Train.csv',sep="/"), h
nSample.Training<-length(LinearModel.Training[,1])
head(LinearModel.Training)
```

```
##        Input     Output Model.Switch
## 1  3.6664327  3.4295281            1
## 2 -2.5194424 -2.5528623            0
## 3  0.6475581  1.5908455            1
## 4  2.4439621  1.8855023            1
## 5  1.9921334  0.1343168            0
## 6  1.7534556  2.8432003            1
```

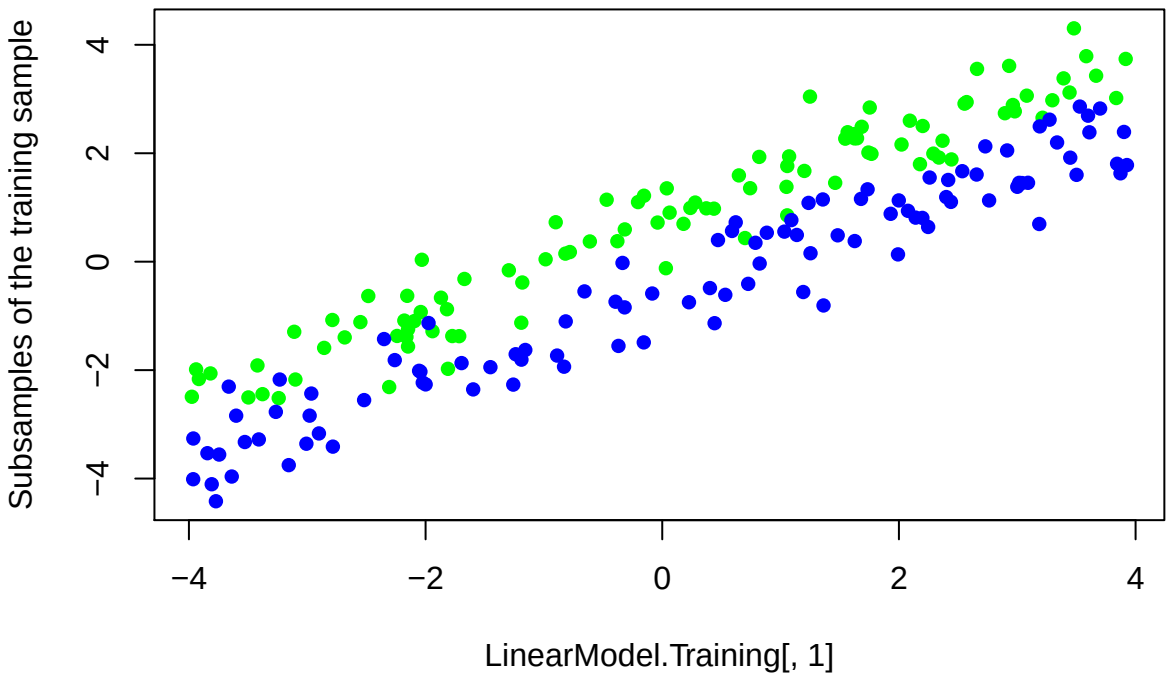
```
plot(LinearModel.Training[,1],LinearModel.Training[,2], type="p",pch=19)
```




```
##Separate the models by using the 3rd column and plot the subsamples
LinearModel.Training.1<-cbind(LinearModel.Training[,1],rep(NA,nSample.Training))
LinearModel.Training.2<-cbind(LinearModel.Training[,1],rep(NA,nSample.Training))
LinearModel.Training.1[LinearModel.Training[,3]*(1:nSample.Training),2]<-
  LinearModel.Training[LinearModel.Training[,3]*(1:nSample.Training),2]
LinearModel.Training.2[(1-LinearModel.Training[,3])*(1:nSample.Training),2]<-
  LinearModel.Training[(1-LinearModel.Training[,3])*(1:nSample.Training),2]
head(cbind(LinearModel.Training,
           Training1=LinearModel.Training.1[,2],
           Training2=LinearModel.Training.2[,2]))
```

```
##        Input     Output Model.Switch Training1  Training2
## 1  3.6664327  3.4295281            1  3.429528         NA
## 2 -2.5194424 -2.5528623            0        NA -2.5528623
## 3  0.6475581  1.5908455            1  1.590846         NA
## 4  2.4439621  1.8855023            1  1.885502         NA
## 5  1.9921334  0.1343168            0        NA  0.1343168
## 6  1.7534556  2.8432003            1  2.843200         NA
```

```
# Plot the subsamples
matplot(LinearModel.Training[,1],cbind(LinearModel.Training.1[,2],LinearModel.Training.2[,2]),
        pch=16,col=c("green","blue"),ylab="Subsamples of the training sample")
```




```
##Estimate linear model for training sample, look at the output
EstimatedLinearModel.Training <- lm(LinearModel.Training$Output ~ LinearModel.Training$Input)
summary(EstimatedLinearModel.Training)$coefficients
```

```
##                               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)                 0.03864599 0.05860770  0.6594012 5.104044e-01
## LinearModel.Training$Input 0.77157337 0.02490368 30.9823033 7.238330e-78
```

```
summary(EstimatedLinearModel.Training)$r.squared
```

```
## [1] 0.8290012
```

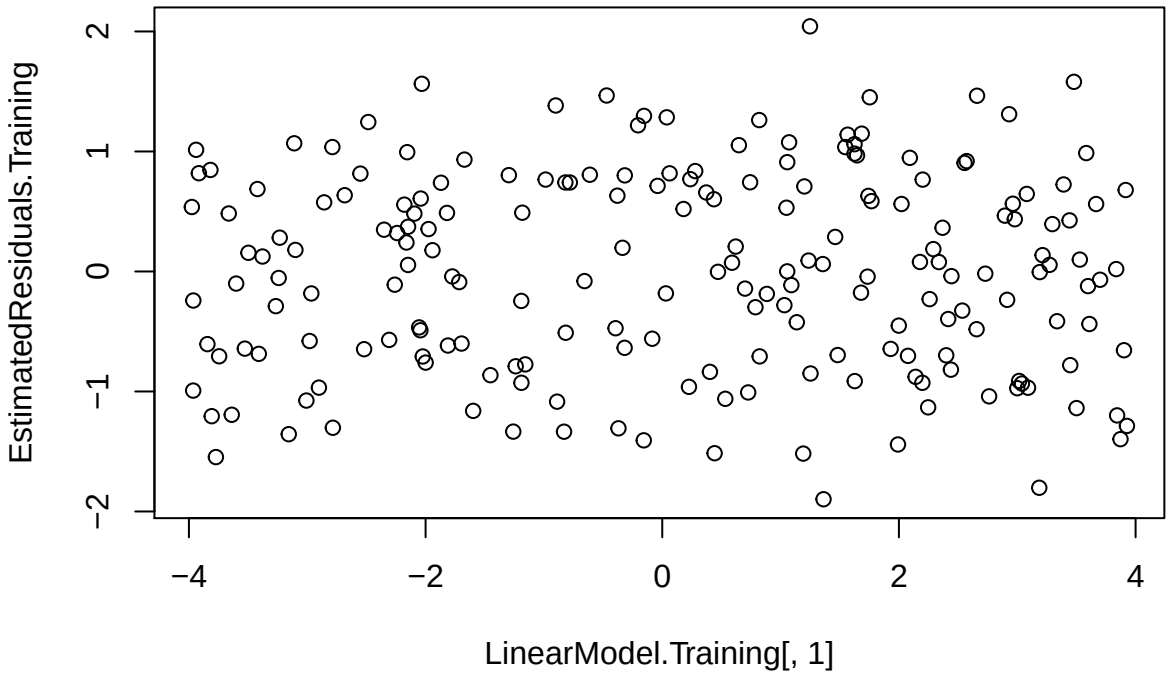
```
summary(EstimatedLinearModel.Training)$sigma
```

```
## [1] 0.8267488
```

Interpret the results in the output. Compare the results for the training sample and the main sample from previous week: coefficients, R2, ????. The samples have similar intercepts, the main sample has a higher slope than the training sample. Both of these coefficients have Std errors that are similar. However for the training sample both coefficients are statistically different from zero, but in the main sample only the slope is significantly different from zero.

Both models have similar R-squared values, close to 0.83. The main sample has a higher Residual standard error as well as more degrees of freedom. 1.154 on 998 df vs 0.827 on 198 df.

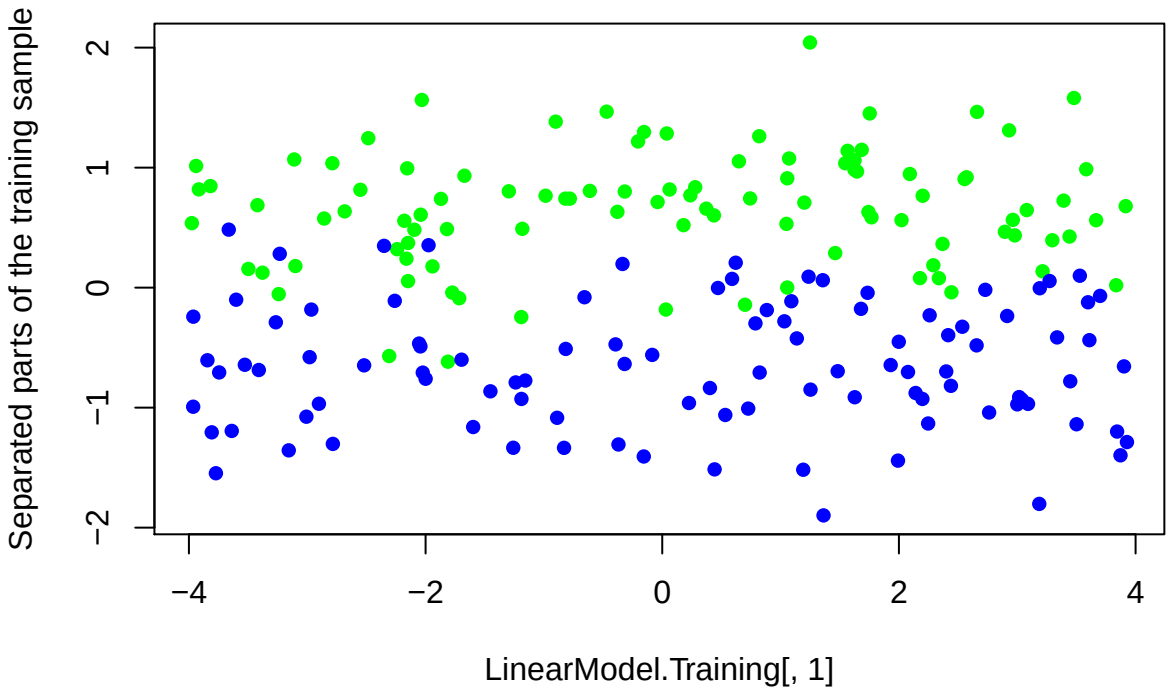
```
##Plot the residuals
EstimatedResiduals.Training<-EstimatedLinearModel.Training$residuals
plot(LinearModel.Training[,1],EstimatedResiduals.Training)
```




```
##Separate the residuals by model and plot
EstimatedResiduals.Training.1<-EstimatedResiduals.Training
EstimatedResiduals.Training.2<-EstimatedResiduals.Training
EstimatedResiduals.Training.1[(LinearModel.Training[,3]==0)*(1:nSample.Training)]<-NA
EstimatedResiduals.Training.2[(LinearModel.Training[,3]==1)*(1:nSample.Training)]<-NA
## Print first 10 rows
head(cbind(AllResiduals=EstimatedResiduals.Training,
      Training1Residuals=EstimatedResiduals.Training.1,
      Training2Residuals=EstimatedResiduals.Training.2,
      TrainingClass=LinearModel.Training[,3]))
```

```
##   AllResiduals Training1Residuals Training2Residuals TrainingClass
## 1   0.56196024         0.56196024                 NA             1
## 2  -0.64757362                 NA         -0.6475736             0
## 3   1.05256097         1.05256097                 NA             1
## 4  -0.03883971        -0.03883971                 NA             1
## 5  -1.44140627                 NA         -1.4414063             0
## 6   1.45163470         1.45163470                 NA             1
```

```
##Plot the residuals, separated by model
# Plot the residuals corresponding to different models
matplot(LinearModel.Training[,1],cbind(EstimatedResiduals.Training.1, EstimatedResiduals.Training.2),pc
```



What do you think about best way to separate the samples of residuals? Right now, it is hard to tell, since they overlap at this point. The pattern seems to be pretty symmetric. But I would guess logistic regression, since that is the topic of the week!

1.2. Logistic regression

```
Logistic.Model.Data<-data.frame(Logistic.Output=LinearModel.Training[,3], Logistic.Input=EstimatedResidu
LinearModel.Training.Logistic<-glm(Logistic.Output~Logistic.Input,data=Logistic.Model.Data, family=binor
summary(LinearModel.Training.Logistic)
```

```
##
## Call:
## glm(formula = Logistic.Output ~ Logistic.Input, family = binomial(link = logit),
##     data = Logistic.Model.Data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.19898  -0.21506  -0.01018   0.24713   2.63249
##
## Coefficients:
```

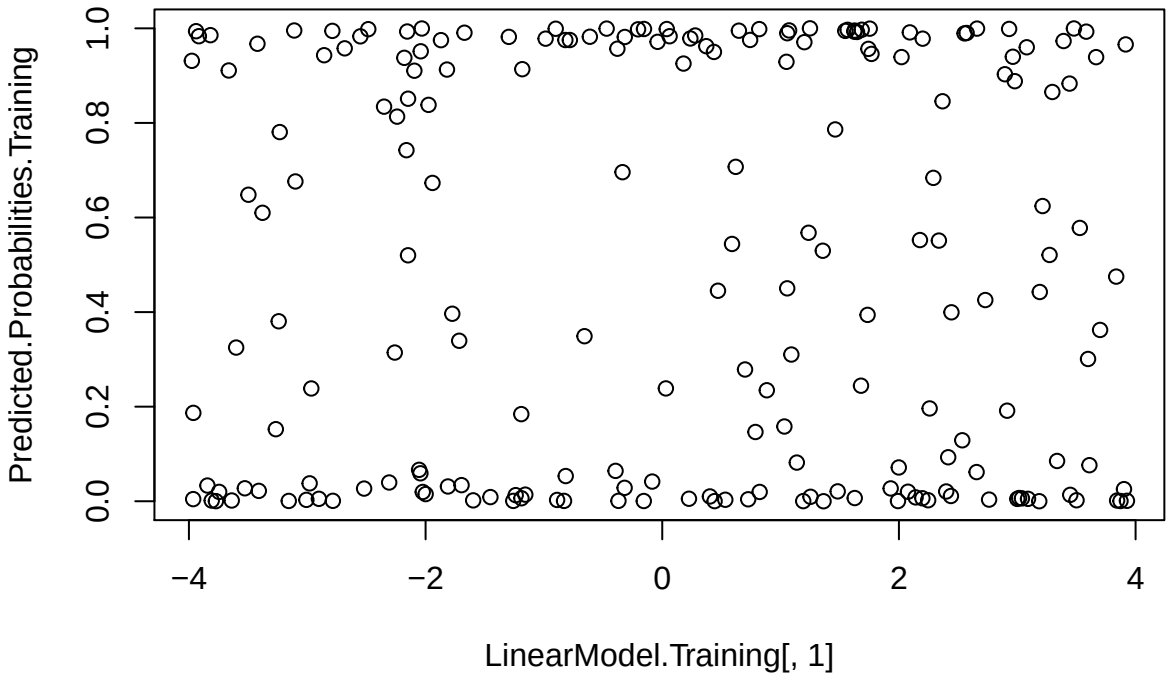
```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.2044     0.2680  -0.762    0.446
## Logistic.Input   5.2318     0.7928   6.599 4.14e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.239  on 199  degrees of freedom
## Residual deviance:  92.557  on 198  degrees of freedom
## AIC: 96.557
##
## Number of Fisher Scoring iterations: 7
```

```
names(LinearModel.Training.Logistic)
```

```
##  [1] "coefficients"      "residuals"         "fitted.values"
##  [4] "effects"           "R"                 "rank"
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"
## [13] "iter"              "weights"           "prior.weights"
## [16] "df.residual"       "df.null"           "y"
## [19] "converged"         "boundary"          "model"
## [22] "call"              "formula"           "terms"
## [25] "data"              "offset"            "control"
## [28] "method"            "contrasts"         "xlevels"
```

Interpret the summary of the model: what is the meaning and significance of coefficients. The Intercept is B0, and the "slope" is B1 These are the two values that can be used in the function Y = B0 + B1X. B0 gives a fitted probability when X=0. B1 is a multiplier on the log odds of success. A 1 unit change in X will increase log odds by eˆB1

```
Predicted.Probabilities.Training<-predict(LinearModel.Training.Logistic,type="response")
plot(LinearModel.Training[,1],Predicted.Probabilities.Training)
```

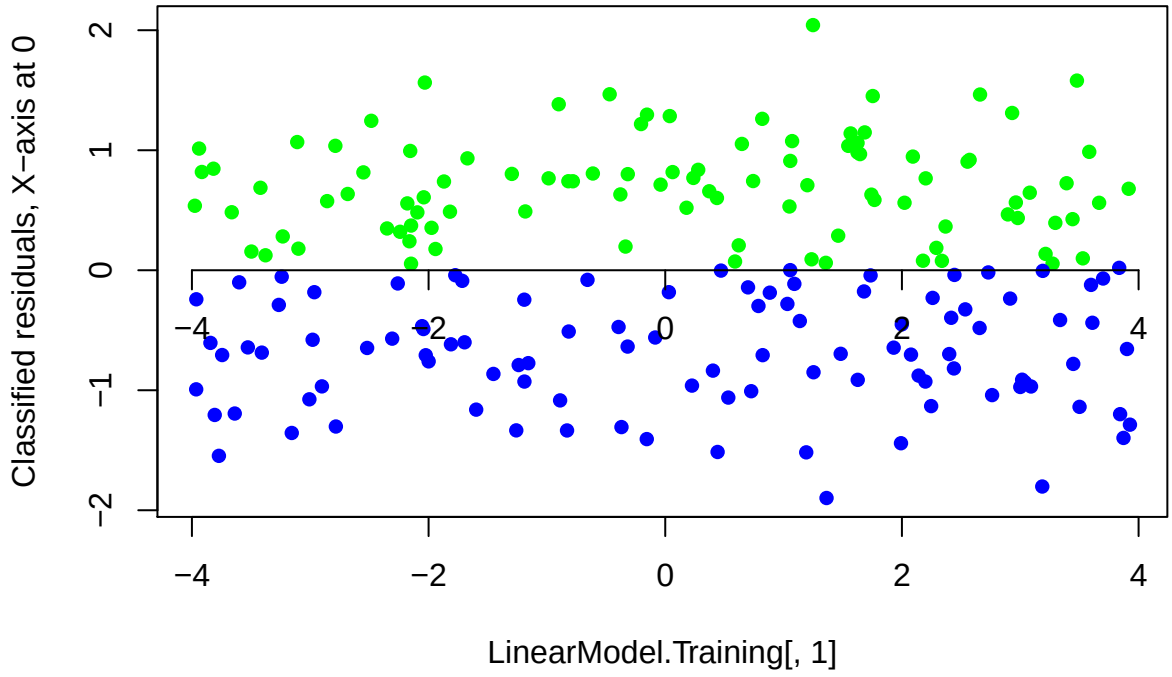


How can we use this graph? What does it tell us? We can use this graph to confirm that logistic regression is predicting the residual probabilities properly. This graph is telling us the predicted probabilities from x using the B0 and B1s given from the logistic regression. The predict function is giving us the probabilities using the formula p = $(e^{(B0+B1X))/(1+e}(B0+B1))$

```
##Create the unscrambling sequence
Unscrambling.Sequence.Training.Logistic<-
  (predict(LinearModel.Training.Logistic,type="response")>.5)*1
##Create classified residuals
ClassifiedResiduals.Training.1<-EstimatedResiduals.Training
ClassifiedResiduals.Training.2<-EstimatedResiduals.Training
ClassifiedResiduals.Training.1[(Unscrambling.Sequence.Training.Logistic==0)*(1:nSample.Training)]<-NA
ClassifiedResiduals.Training.2[(Unscrambling.Sequence.Training.Logistic==1)*(1:nSample.Training)]<-NA
head(cbind(AllTraining=EstimatedResiduals.Training,
           Training1=ClassifiedResiduals.Training.1,
           Training2=ClassifiedResiduals.Training.2))
```

```
##   AllTraining Training1   Training2
## 1  0.56196024 0.5619602          NA
## 2 -0.64757362        NA -0.64757362
## 3  1.05256097 1.0525610          NA
## 4 -0.03883971        NA -0.03883971
## 5 -1.44140627        NA -1.44140627
## 6  1.45163470 1.4516347          NA
```

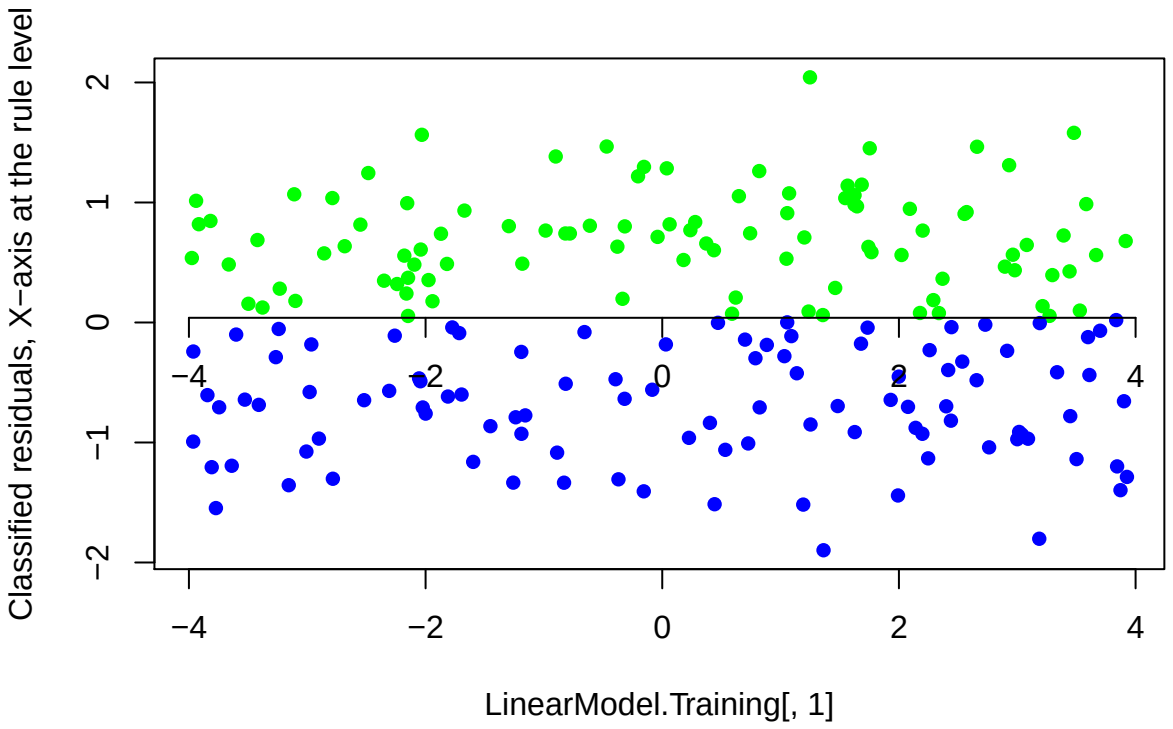
```
##Plot both classes of the residuals
matplot(LinearModel.Training[,1],cbind(ClassifiedResiduals.Training.1,
                                       ClassifiedResiduals.Training.2),
        pch=16,col=c("green","blue"),ylab="Classified residuals, X-axis at 0")
axis(1,pos=0)
```



Recall what classification rule we used in the previous assignment with these data? We used the difference in parabola slope. *What is the classification rule estimated by logistic regression?* We used p=0.5 which will give us X = (-B0/B1)

```
##Calculate classification boundary for the models using estimated coefficients of logistic regression.
##We can use the two coefficients B0 and B1 to find the classification boundary.
Classification.Rule.Logistic <- -summary(LinearModel.Training.Logistic)$coefficients[1]/summary(LinearM


##Plot the data using the Classification.Rule.Logistic
matplot(LinearModel.Training[,1],cbind(ClassifiedResiduals.Training.1,
                                       ClassifiedResiduals.Training.2),
        pch=16,col=c("green","blue"),ylab="Classified residuals, X-axis at the rule level")
axis(1,pos=Classification.Rule.Logistic)
```



##1.3. Separate subsamples in the main sample using the classifier trained on the training sample#

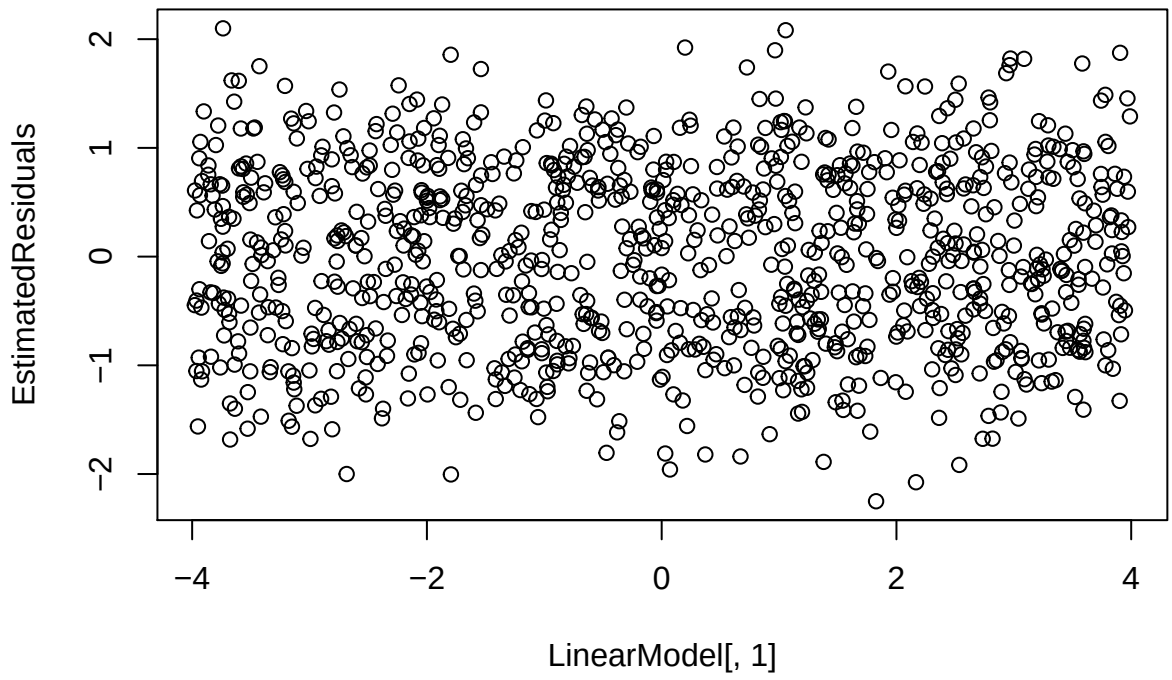
```
datapath <- "C:/Users/JohntheGreat/Documents/MSCA/StatisticalAnalysis/Week6/Assignments"
LinearModel<-read.csv(file=paste(datapath,'ResidualAnalysisProjectData_1.csv',sep="/"),header=TRUE,sep=
nSample<-length(LinearModel[,1])
head(LinearModel)
```

```
##        Input    Output
## 1  3.6664327  2.747905
## 2 -2.5194424 -3.242035
## 3  0.6475581  1.559734
## 4  2.4439621  1.292082
## 5  1.9921334  1.958417
## 6  1.7534556  2.049381
```

```
##Estimate Linear Model
EstimatedLinearModel<-lm(LinearModel[,2]~LinearModel[,1])
EstimatedLinearModel$coefficients
```

```
##      (Intercept) LinearModel[, 1]
##       0.03160231       0.79627673
```

```
EstimatedResiduals<-EstimatedLinearModel$residuals
plot(LinearModel[,1],EstimatedResiduals)
```




```
Unscrambling.Sequence.Logistic<-(predict(LinearModel.Training.Logistic,
                                         newdata=data.frame(Logistic.Output=EstimatedResiduals,
                                                            Logistic.Input=EstimatedResiduals),
                                         type="response")>.5)*1
Probability<-sum(Unscrambling.Sequence.Logistic)/length(Unscrambling.Sequence.Logistic)
Probability
```

```
## [1] 0.489
```

```
binom.test(sum(Unscrambling.Sequence.Logistic),nSample,p = 0.5)
```

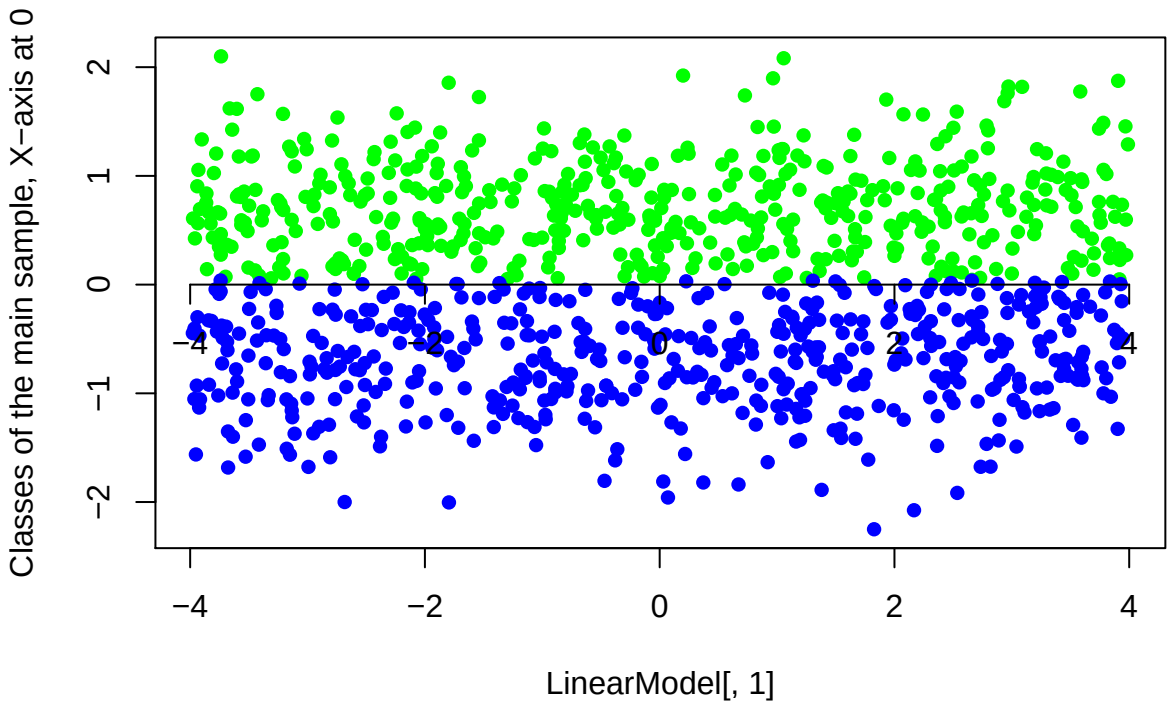
```
##
##  Exact binomial test
##
## data:  sum(Unscrambling.Sequence.Logistic) and nSample
## number of successes = 489, number of trials = 1000, p-value =
## 0.5067
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.4575891 0.5204758
## sample estimates:
## probability of success
##                  0.489
```

What do you conclude based on the binomial test? It appears that you cannot conclude that the probability is different from 0.5. since the p-value = 0.5067

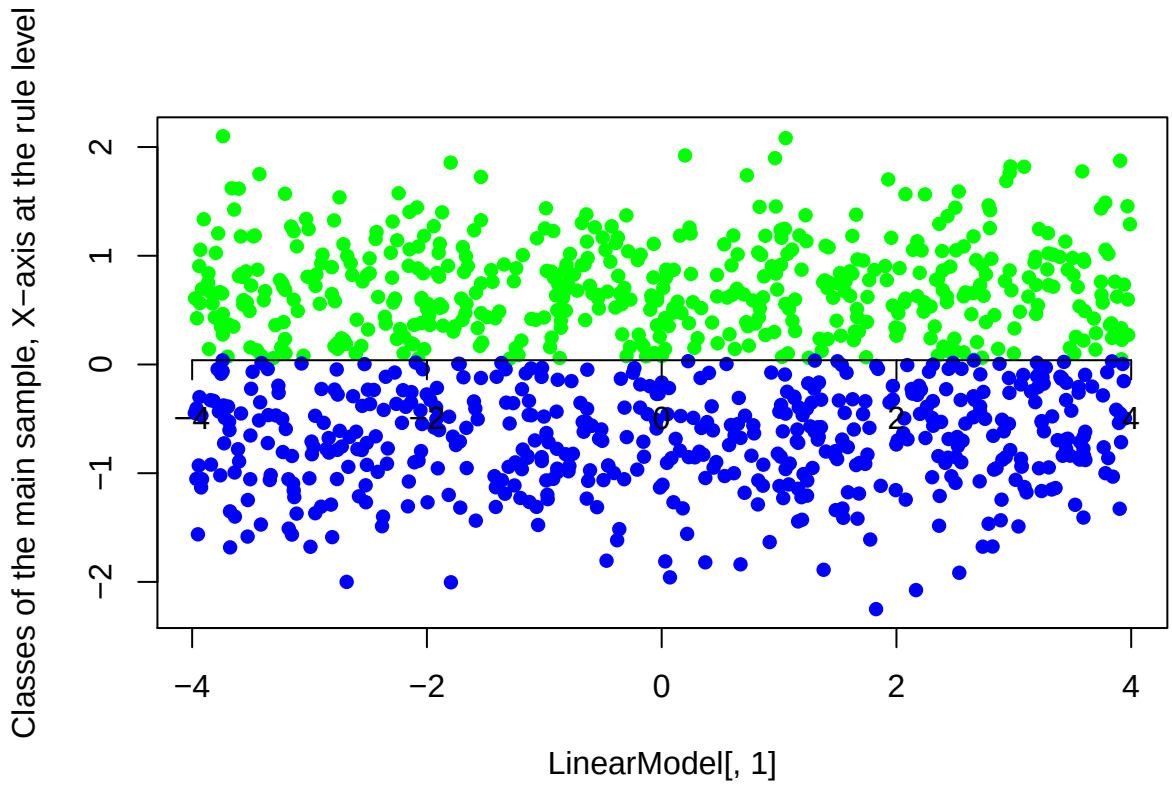
```
##Create classified residuals
ClassifiedResiduals.1<-EstimatedResiduals
ClassifiedResiduals.2<-EstimatedResiduals
ClassifiedResiduals.1[(Unscrambling.Sequence.Logistic==0)*(1:nSample)]<-NA
ClassifiedResiduals.2[(Unscrambling.Sequence.Logistic==1)*(1:nSample)]<-NA
##Print first 10 rows to check
cbind(EstimatedResiduals,ClassifiedResiduals.1,ClassifiedResiduals.2)[1:10,]
```

```
##    EstimatedResiduals ClassifiedResiduals.1 ClassifiedResiduals.2
## 1         -0.20319222                    NA            -0.2031922
## 2         -1.26746423                    NA            -1.2674642
## 3          1.01249601             1.01249601                    NA
## 4         -0.68559015                    NA            -0.6855901
## 5          0.34052527             0.34052527                    NA
## 6          0.62154299             0.62154299                    NA
## 7          0.06188062             0.06188062                    NA
## 8         -0.40786555                    NA            -0.4078656
## 9         -0.33822412                    NA            -0.3382241
## 10         0.04381600             0.04381600                    NA
```

```
# Plot both classes of the residuals
matplot(LinearModel[,1],cbind(ClassifiedResiduals.1,
                              ClassifiedResiduals.2),
        pch=16,col=c("green","blue"),ylab="Classes of the main sample, X-axis at 0")
axis(1,pos=0)
```



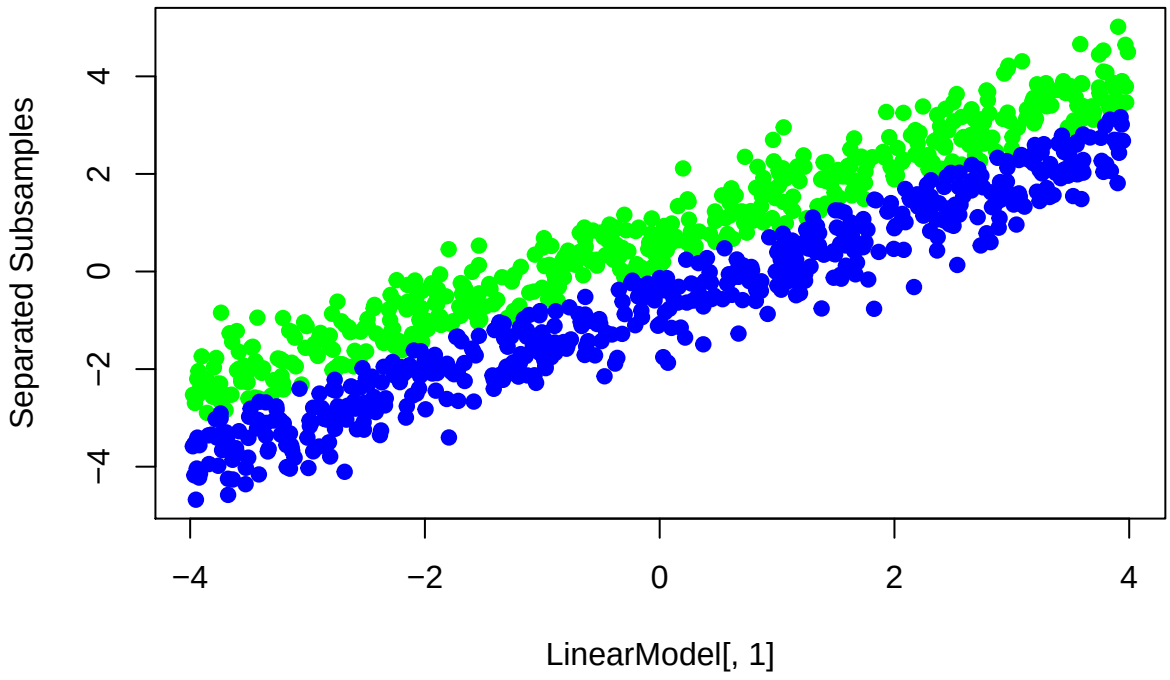
```
##Plot with division using classification boundary
matplot(LinearModel[,1],cbind(ClassifiedResiduals.1,ClassifiedResiduals.2),
        pch=16,col=c("green","blue"),ylab="Classes of the main sample, X-axis at the rule level")
axis(1,pos=Classification.Rule.Logistic)
```




```
# Create recovered models
LinearModel1.Recovered<-LinearModel
LinearModel2.Recovered<-LinearModel
LinearModel1.Recovered[(1-Unscrambling.Sequence.Logistic)*(1:nSample),2]<-NA
LinearModel2.Recovered[Unscrambling.Sequence.Logistic*(1:nSample),2]<-NA
# Print the first 1 rows of scrambled and unscrambled samples
cbind(LinearModel,LinearModel1.Recovered,LinearModel2.Recovered)[1:10,]
```

```
##         Input     Output      Input   Output      Input     Output
## 1   3.6664327  2.7479052  3.6664327       NA  3.6664327  2.7479052
## 2  -2.5194424 -3.2420353 -2.5194424       NA -2.5194424 -3.2420353
## 3   0.6475581  1.5597337  0.6475581 1.559734  0.6475581         NA
## 4   2.4439621  1.2920823  2.4439621       NA  2.4439621  1.2920823
## 5   1.9921334  1.9584170  1.9921334 1.958417  1.9921334         NA
## 6   1.7534556  2.0493812  1.7534556 2.049381  1.7534556         NA
## 7   2.7300053  2.2673226  2.7300053 2.267323  2.7300053         NA
## 8   1.2366129  0.6084228  1.2366129       NA  1.2366129  0.6084228
## 9   1.7351840  1.0750648  1.7351840       NA  1.7351840  1.0750648
## 10  2.6600869  2.1935836  2.6600869 2.193584  2.6600869         NA
```

```r
# Plot the unscrambled subsamples
matplot(LinearModel[,1],cbind(LinearModel1.Recovered[,2],LinearModel2.Recovered[,2]), type="p",col=c("g
```




```r
##Estimate linear models for the subsamples
LinearModel1.Recovered.lm<-lm(LinearModel1.Recovered[,2]~LinearModel1.Recovered[,1])
LinearModel2.Recovered.lm<-lm(LinearModel2.Recovered[,2]~LinearModel2.Recovered[,1])
summary(LinearModel1.Recovered.lm)
```

```
##
## Call:
## lm(formula = LinearModel1.Recovered[, 2] ~ LinearModel1.Recovered[,
##     1])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70777 -0.33381 -0.04006  0.27395  1.39762
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.759904   0.019251   39.47   <2e-16 ***
## LinearModel1.Recovered[, 1] 0.803360   0.008292   96.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4256 on 487 degrees of freedom
```

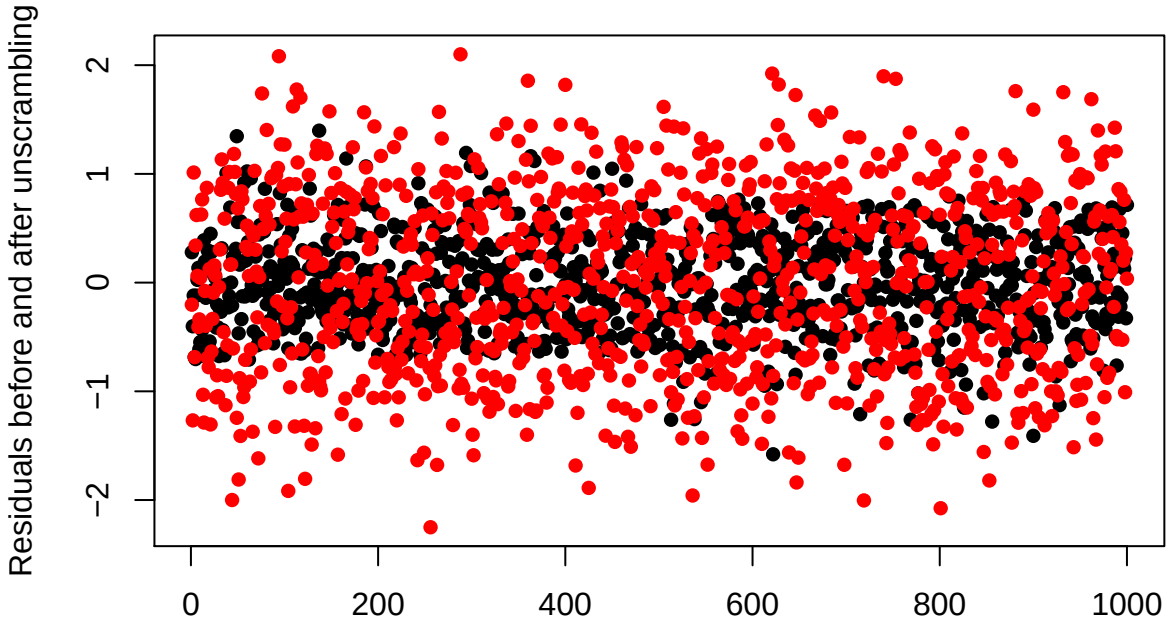
```
##   (511 observations deleted due to missingness)
## Multiple R-squared:  0.9507, Adjusted R-squared:  0.9506
## F-statistic:  9386 on 1 and 487 DF,  p-value: < 2.2e-16
```

```
summary(LinearModel2.Recovered.lm)
```

```
##
## Call:
## lm(formula = LinearModel2.Recovered[, 2] ~ LinearModel2.Recovered[,
##     1])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.57919 -0.29823  0.02554  0.36264  0.79212
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -0.666713   0.020242  -32.94   <2e-16 ***
## LinearModel2.Recovered[, 1]  0.811196   0.008643   93.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.457 on 509 degrees of freedom
##   (489 observations deleted due to missingness)
## Multiple R-squared:  0.9454, Adjusted R-squared:  0.9453
## F-statistic:  8809 on 1 and 509 DF,  p-value: < 2.2e-16
```

Compare the summaries of the mix with the summary of the single linear model fit. The 3 models all have similar slopes, basically 0.80. But they all have different intercepts. LM1.R has a postive intercept at 0.759 and LM2.R has a negative intercept at -0.66, while EstLinModel has its intercept in between the other two at 0.03. It seems that we have 3 close to parallel lines with the EstLinMod between the two separated models. The residual standard error in the separated models is about half of the SE in the single linear model. Also, the 2 separated models show a higher R squared than the single model.

```
# Plot residuals
Residuals.Comparison<-cbind(Unscrambled.residuals=c(summary(LinearModel1.Recovered.lm)$residuals,summary
matplot(Residuals.Comparison,type="p",pch=16,ylab="Residuals before and after unscrambling")
```




```
##Estimate standard deviations
apply(Residuals.Comparison,2,sd)
```

```
##  Unscrambled.residuals Single.Model.residuals
##              0.4412653              0.8384730
```

Conclusion

How the sample was mixed? With what probability? It seems like the sample was mixed 50/50 from the binomial test. *Was the probability significantly different from 0.5?* Not according to the binomial test, which gave us a 0.489 probability of success. *What were the parameters of mixed models?* LinearModel1.Recovered Int = 0.759 Slope = 0.803 LinearModel2.Recovered Int = -0.66 Slope = 0.811 *How much we reduced variance of residuals by separating the models?* We reduced the variance of the residuals by about half.

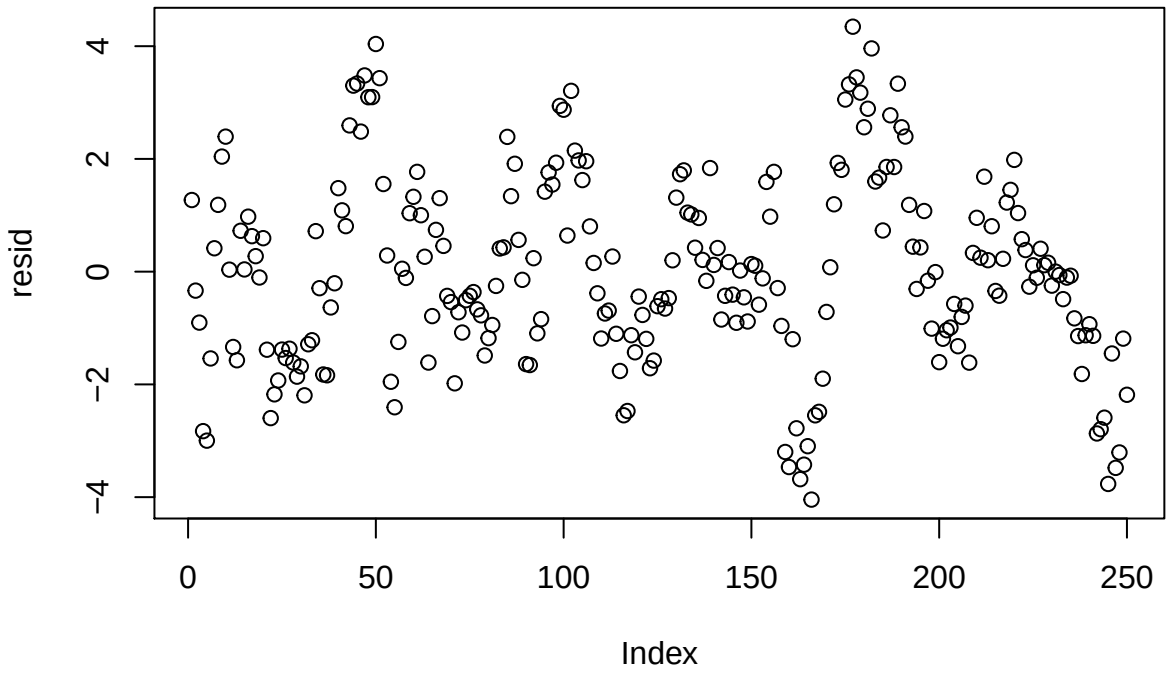
2. Check Assumptions of Linear Model.

```
assignmentData<-read.csv(file=paste(datapath,"Week6AssignmentData.csv",sep="/"),header = TRUE,sep=",")
head(assignmentData)
```

```
##         Input    Output
## 1 -2.23464884 -3.498104
```

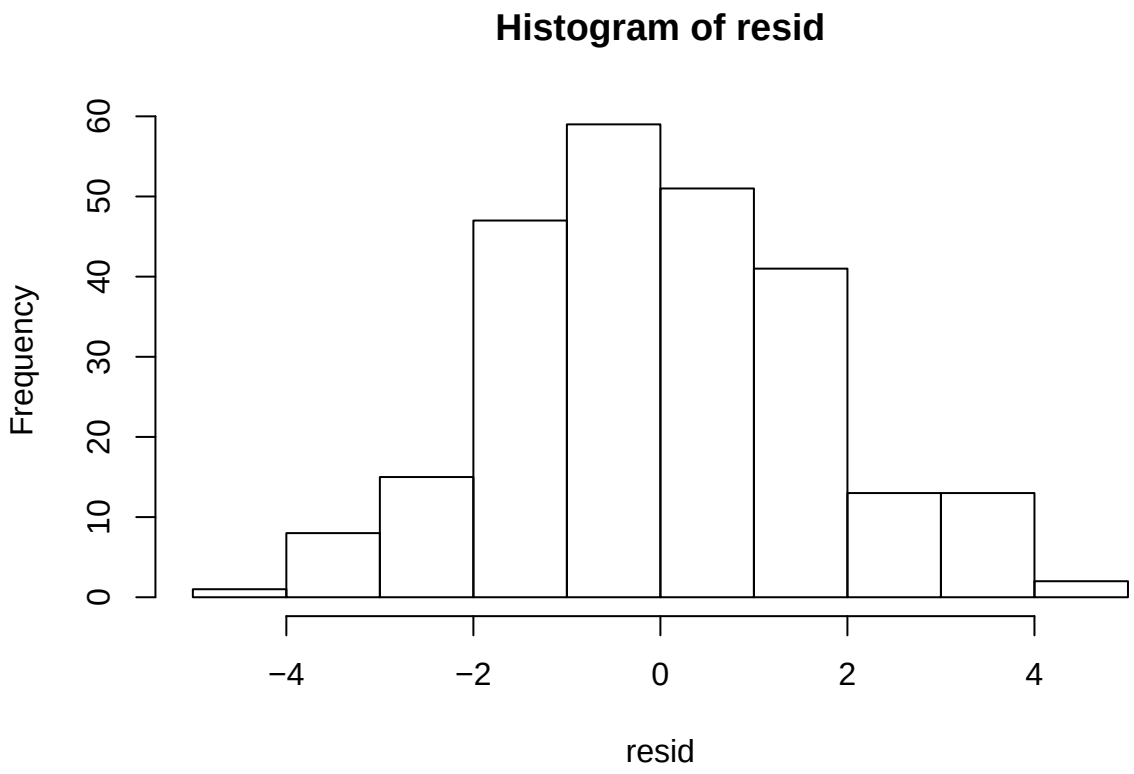
```
## 2  0.78746123  0.771090
## 3  1.99680464  2.554582
## 4 -0.15115047 -3.547054
## 5  0.09096086 -3.244699
## 6  2.57175934  3.036760
```

```
lin.mod <- lm(assignmentData$Output ~ assignmentData$Input)
resid <- lin.mod$residuals
plot(resid)
```

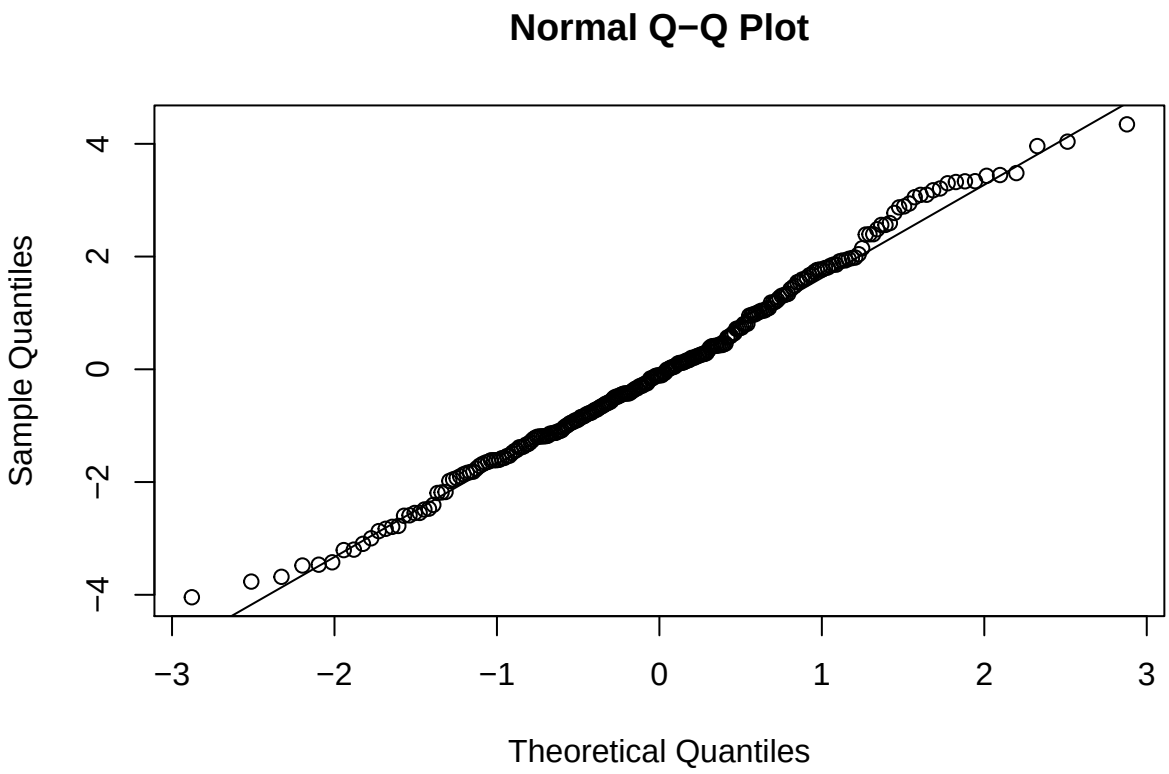


Are main assumptions of linear model satisfied?

```
##Gaussian assumption
hist(resid)
```



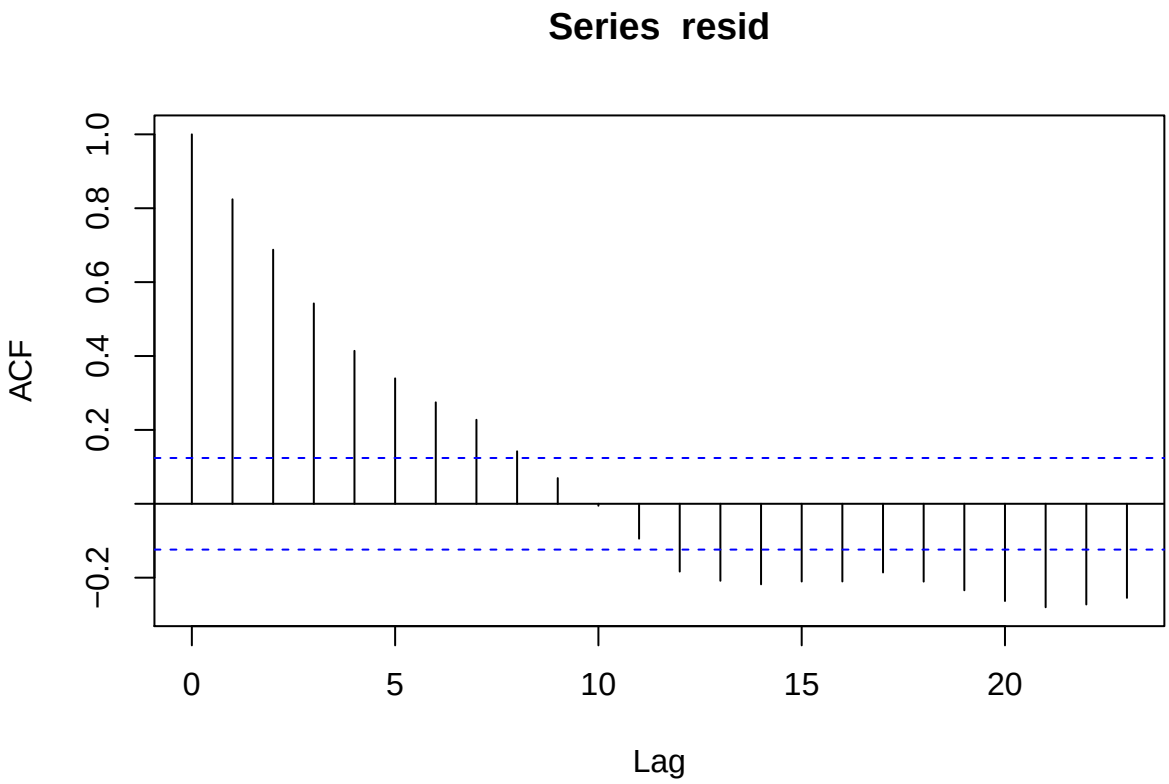

```
qqnorm(resid)
qqline(resid)
```




```
##IID assumption.
library(randtests)
turning.point.test(resid)
```

```
##
##  Turning Point Test
##
## data:  resid
## statistic = -6.2226, n = 250, p-value = 4.89e-10
## alternative hypothesis: non randomness
```

```
##Autocorrelation function
#Autocorrelation with lag 1
acf(resid)
```



* Gaussian assumption - We can check for normality by looking at a histogram of the residuals, as well as running a qq plot. The histogram looks to be close to normal, and the qq plot looks to follow the line y = x. Therefore we can say that the distribution of the residuals is normal. * IID assumption - We can check if the data is independent and identically distributed by using the turning point test. This returns a p-value of close to zero which tells us that randomness (iid) needs to be rejected. * Autocorrelation with lag 0 and 1 - Using the acf() function in r shows us that there is autocorrelation with lag 0 and 1.