Week 5 Assignment

Matthew Dunne

April 24, 2018

1 Method 1

1.1 Project data

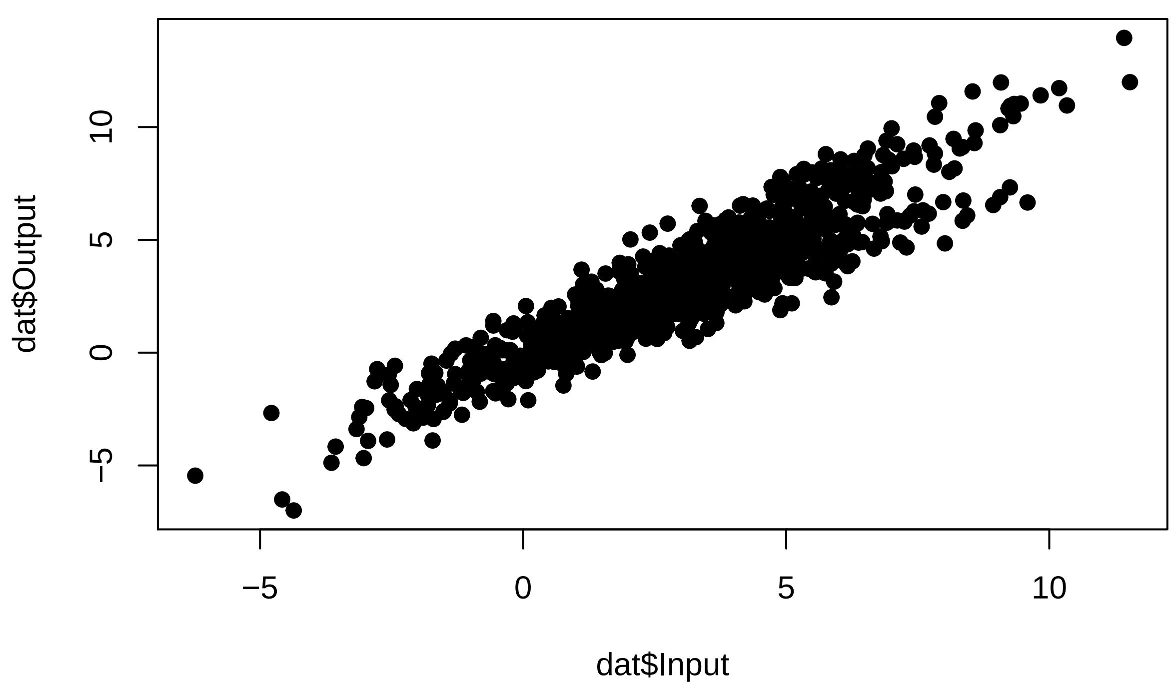
Analyze the second case data from file ResidualAnalysisProjectData_2.csv.

Download the data.

```
datapath<-"C:/Users/mjdun/Desktop/Master Classes/Q1/Statistical Analysis/Lecture 5"
dat<-read.csv(file=paste(datapath,"ResidualAnalysisProjectData_2.csv",sep="/"),header=TRUE,sep=",")
head(dat)
```

```
##      Input      Output
## 1 3.132859  4.17792255
## 2 5.561134  5.84669919
## 3 1.984543 -0.09834184
## 4 5.619160  7.84692946
## 5 6.378149  7.57941491
## 6 6.123204  5.68973137
```

```
plot(dat$Input,dat$Output, type="p",pch=19)
```



set the number of observations

```
nSample<-length(dat$Input)
```

1.2 Estimate linear model

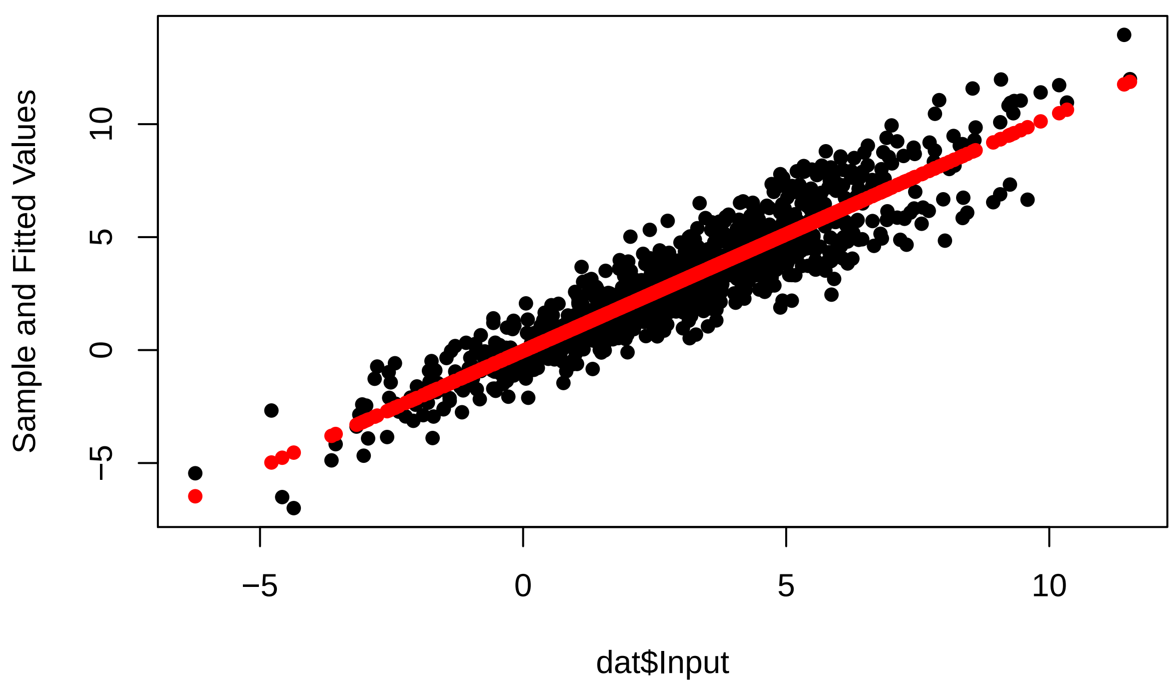
Fit linear model to the data and plot the sample and the fitted values.

```
m1<-lm(Output~Input,dat)
m1$coefficients
```

```
## (Intercept)       Input
## -0.03657717  1.03270579
```

plot input against both the prediction/fitted values (red) and original output (black) and put in same graph

```
matplot(dat$Input,cbind(dat$Output,m1$fitted.values),type="p",pch=16,ylab="Sample and Fitted Values")
```



Analyze the result of the fitting

```
summary(m1)
```

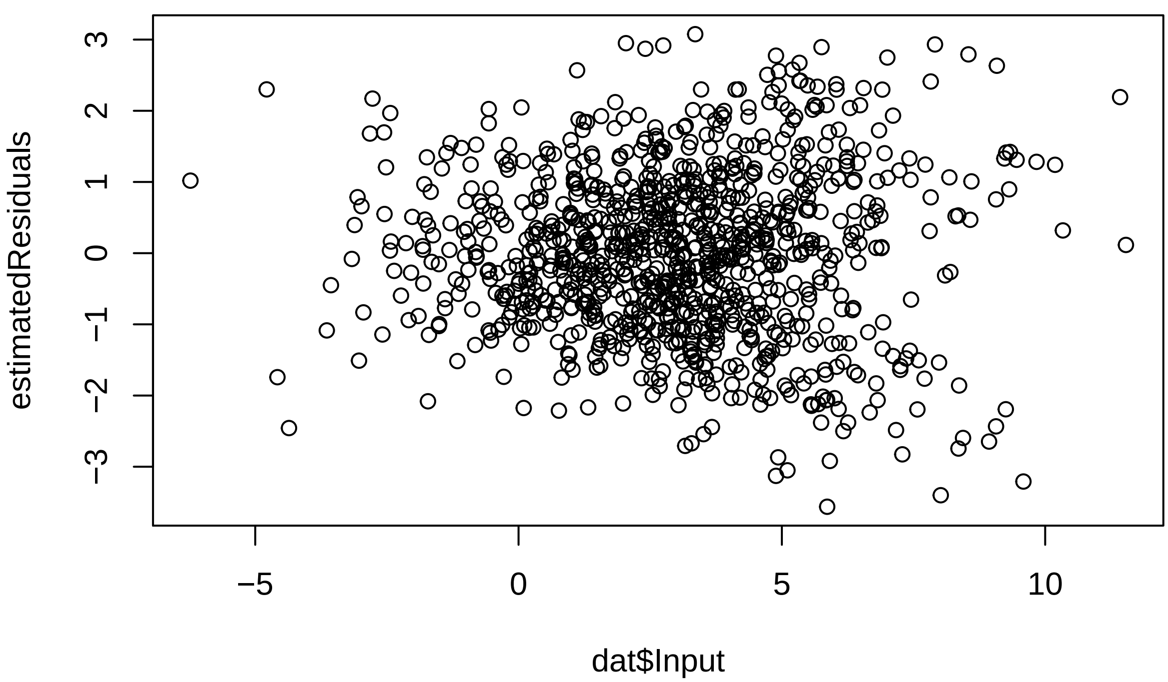
```
##
## Call:
## lm(formula = Output ~ Input, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5620 -0.7987  0.0267  0.7870  3.0759
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.03658    0.05718   -0.64    0.523
## Input        1.03271    0.01453   71.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.157 on 998 degrees of freedom
## Multiple R-squared:  0.835,  Adjusted R-squared:  0.8348
## F-statistic:  5050 on 1 and 998 DF,  p-value: < 2.2e-16
```

Interpret the summary of the model.

Analyze the residuals, plot them.

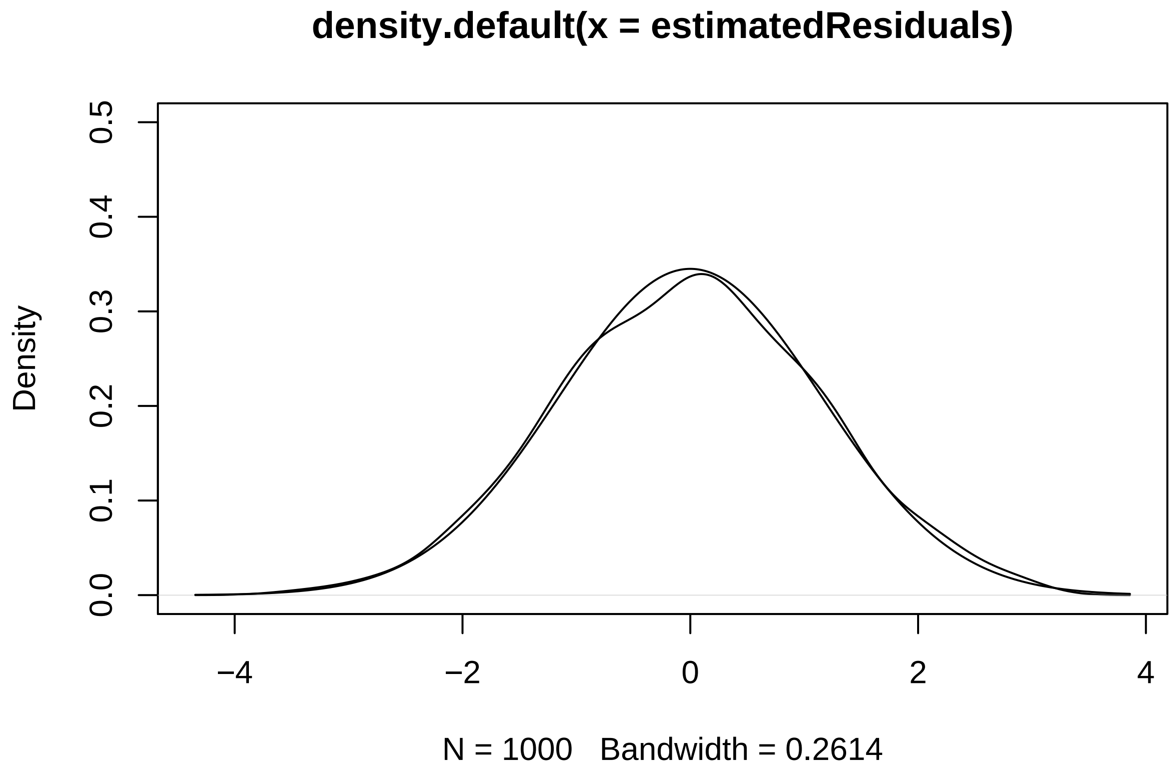
```r
#EstimatedResiduals.Case2<-Estimated.Linear.Model.Case2$residuals
#put residuals into a vector
estimatedResiduals<-m1$residuals
#plot residuals against input values
plot(dat$Input,estimatedResiduals)
```



, and the probability density function of the residuals.

```r
#compute a density estimate for the residuals of the model
Probability.Density.Residuals<-density(estimatedResiduals)
#plot the density estimate
plot(Probability.Density.Residuals,ylim=c(0,.5))
```

```
#plot the theoretical normal distribution of the residuals of this model
lines(Probability.Density.Residuals$x,dnorm(Probability.Density.Residuals$x,mean=mean(estimatedResiduals
```



What does the pattern of residuals and the pattern of the data tell you about the sample? The residuals are not normally distributed. There are probably two distinct distributions.

What kind of mixture of two models do you see in the data? Two separate lines. Somewhat intermixed but diverging past 5 on the input axis

So we need to try to separate the subsamples with different models.

1.3 Creating training sample for separation of mixed models

Create training sample with Input >= 5 and separate the points above the fitted line and below.

```
# Create NA vectors
Train.Sample<-data.frame(trainInput=dat$Input,trainOutput=rep(NA,nSample))
Train.Sample.Steeper<-data.frame(trainSteepInput=dat$Input,
                                 trainSteepOutput=rep(NA,nSample))
Train.Sample.Flatter<-data.frame(trainFlatInput=dat$Input,
                                 trainFlatOutput=rep(NA,nSample))
head(cbind(dat,
           Train.Sample,
           Train.Sample.Steeper,
           Train.Sample.Flatter))
```

```
##      Input      Output trainInput trainOutput trainSteepInput
```

```
## 1 3.132859  4.17792255   3.132859          NA        3.132859
## 2 5.561134  5.84669919   5.561134          NA        5.561134
## 3 1.984543 -0.09834184   1.984543          NA        1.984543
## 4 5.619160  7.84692946   5.619160          NA        5.619160
## 5 6.378149  7.57941491   6.378149          NA        6.378149
## 6 6.123204  5.68973137   6.123204          NA        6.123204
##   trainSteepOutput trainFlatInput trainFlatOutput
## 1               NA       3.132859              NA
## 2               NA       5.561134              NA
## 3               NA       1.984543              NA
## 4               NA       5.619160              NA
## 5               NA       6.378149              NA
## 6               NA       6.123204              NA
```

Select parts of the sample with Input greater than 5 and Output either above the estimated regression line or below it.

```
# Create selectors
#where input column from original data frame is >=5 (Boolean Mask)
Train.Sample.Selector<-dat$Input>=5
#where input >=5 and output>fitted values
Train.Sample.Steeper.Selector<-Train.Sample.Selector&
  (dat$Output>m1$fitted.values)
#where input >=5 and output<=fitted values
Train.Sample.Flatter.Selector<-Train.Sample.Selector&
  (dat$Output<=m1$fitted.values)
```

Create training samples for steep and flat slopes.

```
# Select subsamples
#put output column from where input column >=5 (else NA) to overwrite 2nd output column from Train.Samp
Train.Sample[Train.Sample.Selector,2]<-dat[Train.Sample.Selector,2]
#put output column from where input column >=5 (else NA) to overwrite 2nd output column from Train.Samp
Train.Sample.Steeper[Train.Sample.Steeper.Selector,2]<-dat[Train.Sample.Steeper.Selector,2]
#put output column from where input column >=5 (else NA) to overwrite 2nd output column from Train.Samp
Train.Sample.Flatter[Train.Sample.Flatter.Selector,2]<-dat[Train.Sample.Flatter.Selector,2]
head(Train.Sample)
```

```
##   trainInput trainOutput
## 1   3.132859          NA
## 2   5.561134    5.846699
## 3   1.984543          NA
## 4   5.619160    7.846929
## 5   6.378149    7.579415
## 6   6.123204    5.689731
```

Data frame Train.Sample satisfies condition dat$Input>=5.

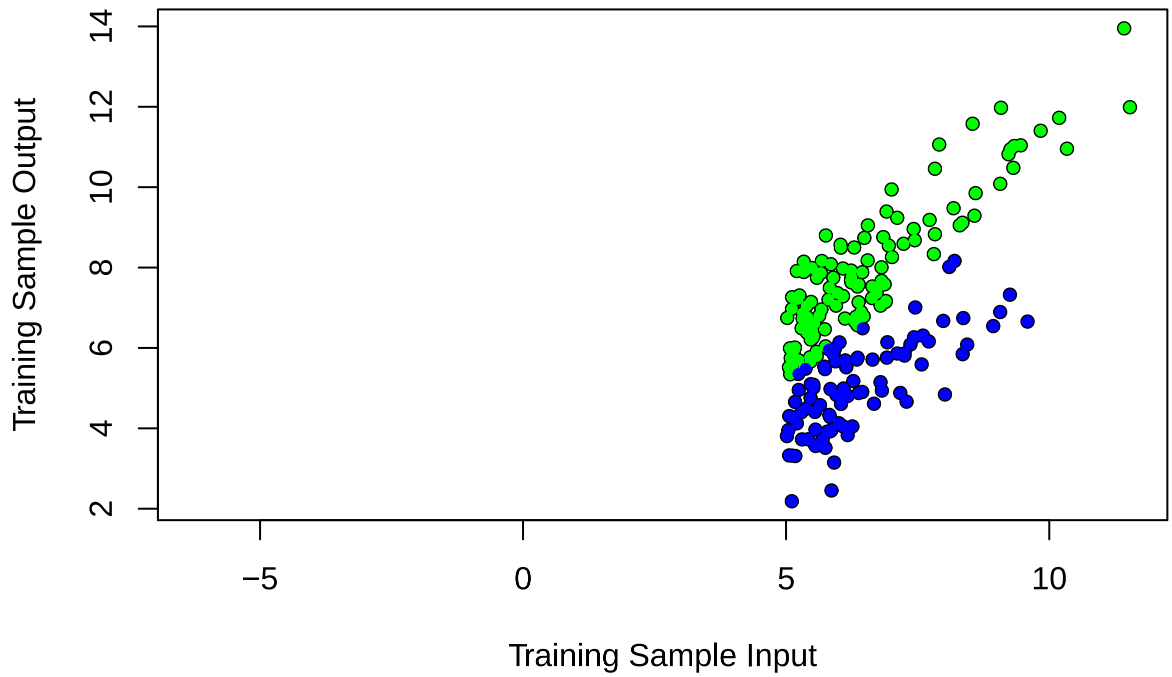
Check what are the resulting training samples.

```
#put all data sets side by side
head(cbind(dat,
           Train.Sample,
           Train.Sample.Steeper,
           Train.Sample.Flatter),10)
```

```
##        Input      Output trainInput trainOutput trainSteepInput
## 1  3.1328589  4.17792255  3.1328589          NA       3.1328589
```

```
## 2  5.5611337  5.84669919  5.5611337    5.846699       5.5611337
## 3  1.9845429 -0.09834184  1.9845429          NA       1.9845429
## 4  5.6191601  7.84692946  5.6191601    7.846929       5.6191601
## 5  6.3781486  7.57941491  6.3781486    7.579415       6.3781486
## 6  6.1232040  5.68973137  6.1232040    5.689731       6.1232040
## 7  0.7666195 -1.45675560  0.7666195          NA       0.7666195
## 8  4.3535141  4.16746077  4.3535141          NA       4.3535141
## 9  2.3627156  2.38611901  2.3627156          NA       2.3627156
## 10 6.3272368  6.77354738  6.3272368    6.773547       6.3272368
##    trainSteepOutput trainFlatInput trainFlatOutput
## 1                NA      3.1328589              NA
## 2          5.846699      5.5611337              NA
## 3                NA      1.9845429              NA
## 4          7.846929      5.6191601              NA
## 5          7.579415      6.3781486              NA
## 6                NA      6.1232040        5.689731
## 7                NA      0.7666195              NA
## 8                NA      4.3535141              NA
## 9                NA      2.3627156              NA
## 10         6.773547      6.3272368              NA
```

```r
#plot all point from Train.Sample (where input must be >=5)
plot(Train.Sample$trainInput,Train.Sample$trainOutput,pch=16,ylab="Training Sample Output",
     xlab="Training Sample Input")
#color all points that are also in the steep output green
points(Train.Sample.Steeper$trainSteepInput,Train.Sample.Steeper$trainSteepOutput,pch=20,col="green")
#color all points that are also in the flatter output blue
points(Train.Sample.Flatter$trainFlatInput,Train.Sample.Flatter$trainFlatOutput,pch=20,col="blue")
```



1.4 Fit linear models to train samples

Fit linear models to both training samples, interpret the summaries of both models.

```
Train.Sample.Steep.lm<-lm(trainSteepOutput~trainSteepInput,
                                              data=Train.Sample.Steeper)
Train.Sample.Flat.lm<-lm(trainFlatOutput~trainFlatInput,
                                              data=Train.Sample.Flatter)
```

get the relevant info for the steep model

```
summary(Train.Sample.Steep.lm)$coefficients
```

```
##                   Estimate Std. Error  t value     Pr(>|t|)
## (Intercept)     1.021798 0.33271502  3.07109 2.647868e-03
## trainSteepInput 1.048763 0.04981059 21.05503 8.979480e-42
```

```
summary(Train.Sample.Steep.lm)$sigma
```

```
## [1] 0.7800507
```

```
summary(Train.Sample.Steep.lm)$df
```

```
## [1]   2 118   2
```

```
summary(Train.Sample.Steep.lm)$r.squared
```

```
## [1] 0.7897791
```

```r
summary(Train.Sample.Steep.lm)$adj.r.squared
```

```
## [1] 0.7879975
```

```r
summary(Train.Sample.Steep.lm)$fstatistic
```

```
##    value    numdf    dendf
## 443.3142   1.0000 118.0000
```

and get the relevant info for the flatter model

```r
summary(Train.Sample.Flat.lm)$coefficients
```

```
##                  Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)    0.08316391 0.50585815 0.1644016 8.698008e-01
## trainFlatInput 0.77825624 0.07855636 9.9069789 7.031944e-16
```

```r
summary(Train.Sample.Flat.lm)$sigma
```

```
## [1] 0.8072447
```

```r
summary(Train.Sample.Flat.lm)$df
```

```
## [1]  2 86  2
```

```r
summary(Train.Sample.Flat.lm)$r.squared
```

```
## [1] 0.5329849
```

```r
summary(Train.Sample.Flat.lm)$adj.r.squared
```

```
## [1] 0.5275545
```

```r
summary(Train.Sample.Flat.lm)$fstatistic
```

```
##    value    numdf    dendf
## 98.14823  1.00000 86.00000
```

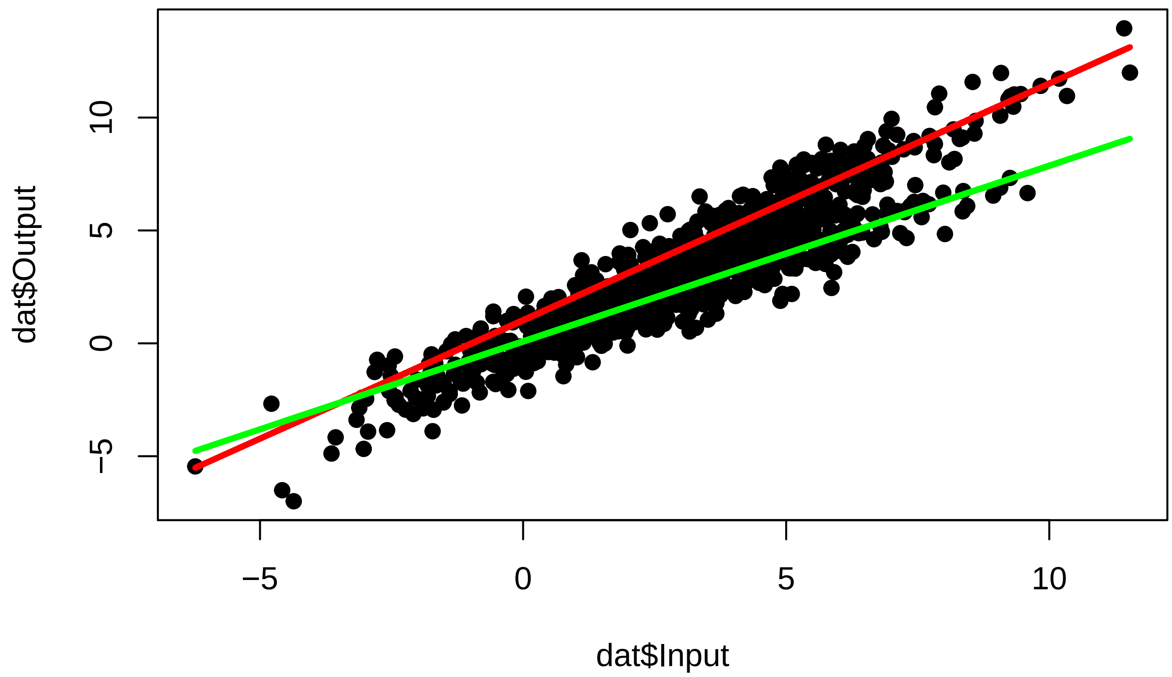
Print out the coefficients of both models for the training sample.

```r
rbind(Steeper.Coefficients=Train.Sample.Steep.lm$coefficients,
      Flatter.Coefficients=Train.Sample.Flat.lm$coefficients)
```

```
##                       (Intercept) trainSteepInput
## Steeper.Coefficients  1.02179775       1.0487634
## Flatter.Coefficients  0.08316391       0.7782562
```

Plot the entire sample with the fitted regression lines estimated from both training subsamples.

```r
#plot original data
plot(dat$Input,dat$Output, type="p",pch=19)
#add prediction line for steep model
lines(dat$Input,predict(Train.Sample.Steep.lm,
                        data.frame(trainSteepInput=dat$Input),
                        interval="prediction")[,1],col="red",lwd=3)
#add prediction line for flat model
lines(dat$Input,predict(Train.Sample.Flat.lm,data.frame(trainFlatInput=dat$Input),
                        interval="prediction")[,1],col="green",lwd=3)
```



Separate the entire sample using the estimated train linear models.

Define distances from each point to both regression lines.

```
# Define the distances from each Output point to both estimated training lines
Distances.to.Steeper<-abs(dat$Output-
                            dat$Input*Train.Sample.Steep.lm$coefficients[2]-
                            Train.Sample.Steep.lm$coefficients[1])
Distances.to.Flatter<-abs(dat$Output-
                           dat$Input*Train.Sample.Flat.lm$coefficients[2]-
                           Train.Sample.Flat.lm$coefficients[1])
```

Define separating sequence which equals TRUE if observation belongs to model with steeper slope and FALSE otherwise.

```
# Define the unscramble sequence
Unscrambling.Sequence.Steeper<-Distances.to.Steeper<Distances.to.Flatter
```

Separate the sample into steeper and flatter parts.

Create data frames. And then fill in data frames

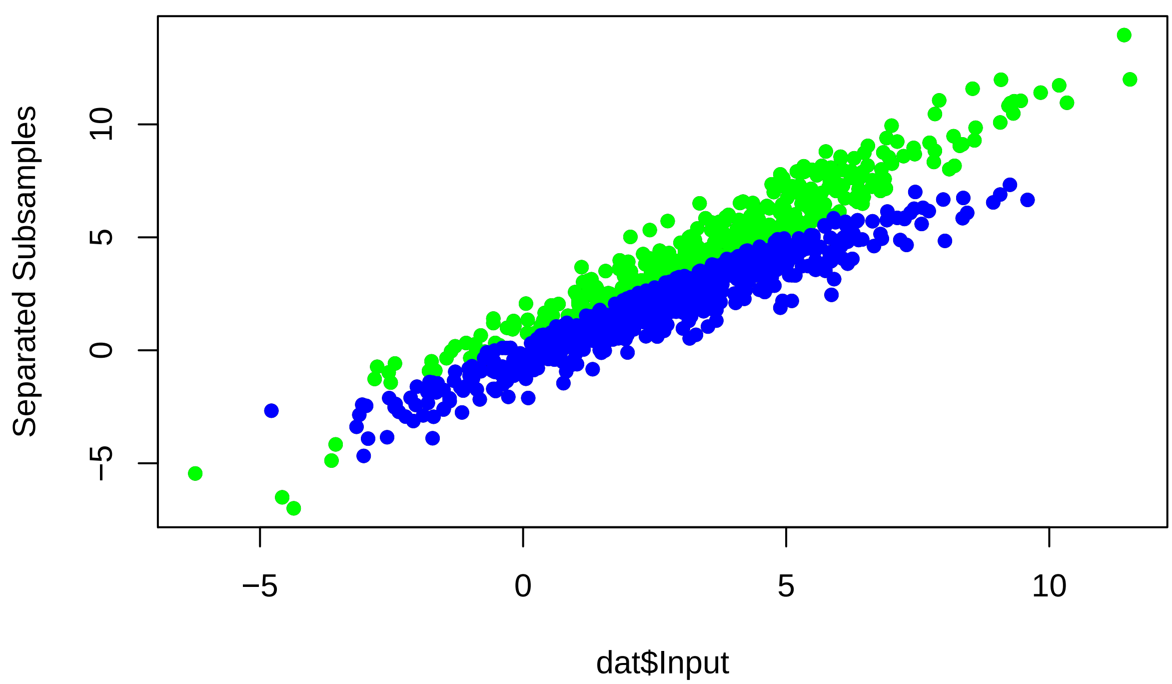
```
# Define  two subsamples with NAs in the Output columns
Subsample.Steeper<-data.frame(steeperInput=dat$Input,steeperOutput=rep(NA,nSample))
Subsample.Flatter<-data.frame(flatterInput=dat$Input,flatterOutput=rep(NA,nSample))
# Fill in the unscrambled outputs instead of NAs where necessary
Subsample.Steeper[Unscrambling.Sequence.Steeper,2]<-dat[Unscrambling.Sequence.Steeper,2]
Subsample.Flatter[!Unscrambling.Sequence.Steeper,2]<-dat[!Unscrambling.Sequence.Steeper,2]
```

```r
# Check the first rows
head(cbind(dat,Subsample.Steeper,Subsample.Flatter))
```

```
##      Input      Output steeperInput steeperOutput flatterInput
## 1 3.132859  4.17792255     3.132859      4.177923     3.132859
## 2 5.561134  5.84669919     5.561134      5.846699     5.561134
## 3 1.984543 -0.09834184     1.984543            NA     1.984543
## 4 5.619160  7.84692946     5.619160      7.846929     5.619160
## 5 6.378149  7.57941491     6.378149      7.579415     6.378149
## 6 6.123204  5.68973137     6.123204            NA     6.123204
##   flatterOutput
## 1            NA
## 2            NA
## 3   -0.09834184
## 4            NA
## 5            NA
## 6    5.68973137
```

And Plot the two samples.

```r
matplot(dat$Input,cbind(dat$Output,
                        Subsample.Steeper$steeperOutput,
                        Subsample.Flatter$flatterOutput),
        type="p",col=c("black","green","blue"),
        pch=16,ylab="Separated Subsamples")
```



Find mixing probability.

```r
# Mixing Probability Of Steeper Slope
(Mixing.Probability.Of.Steeper.Slope<-sum(Unscrambling.Sequence.Steeper)/length(Unscrambling.Sequence.St
```

```
## [1] 0.417
```

Run binomial test for the null hypothesis p=0.5 and two-sided alternative "p is not equal to 0.5". Interpret the output of binom.test

```r
binom.test(sum(Unscrambling.Sequence.Steeper),nSample,p=.5,alternative="t")
```

```
##
##  Exact binomial test
##
## data:  sum(Unscrambling.Sequence.Steeper) and nSample
## number of successes = 417, number of trials = 1000, p-value =
## 1.702e-07
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.3862231 0.4482670
## sample estimates:
## probability of success
##                  0.417
```

What do you conclude from the test results?. We reject the null hypothesis of p=0.5. It is not a "fair coin"

1.5 Fitting models to separated samples

Estimate linear models for separated subsamples.

```r
Linear.Model.Steeper.Recovered<-lm(steeperOutput~steeperInput,data=Subsample.Steeper)
Linear.Model.Flatter.Recovered<-lm(flatterOutput~flatterInput,data=Subsample.Flatter)
```

Print out coefficients for both separated models. Check the summaries (specifically the R-Squared.

```r
rbind(Steeper.Coefficients=Linear.Model.Steeper.Recovered$coefficients,
      Flatter.Coefficients=Linear.Model.Flatter.Recovered$coefficients)
```

```
##                      (Intercept) steeperInput
## Steeper.Coefficients   0.9325475    1.0517077
## Flatter.Coefficients  -0.3467106    0.8630519
```

```r
summary(Linear.Model.Steeper.Recovered)$r.sq
```

```
## [1] 0.9365043
```

```r
summary(Linear.Model.Flatter.Recovered)$r.sq
```

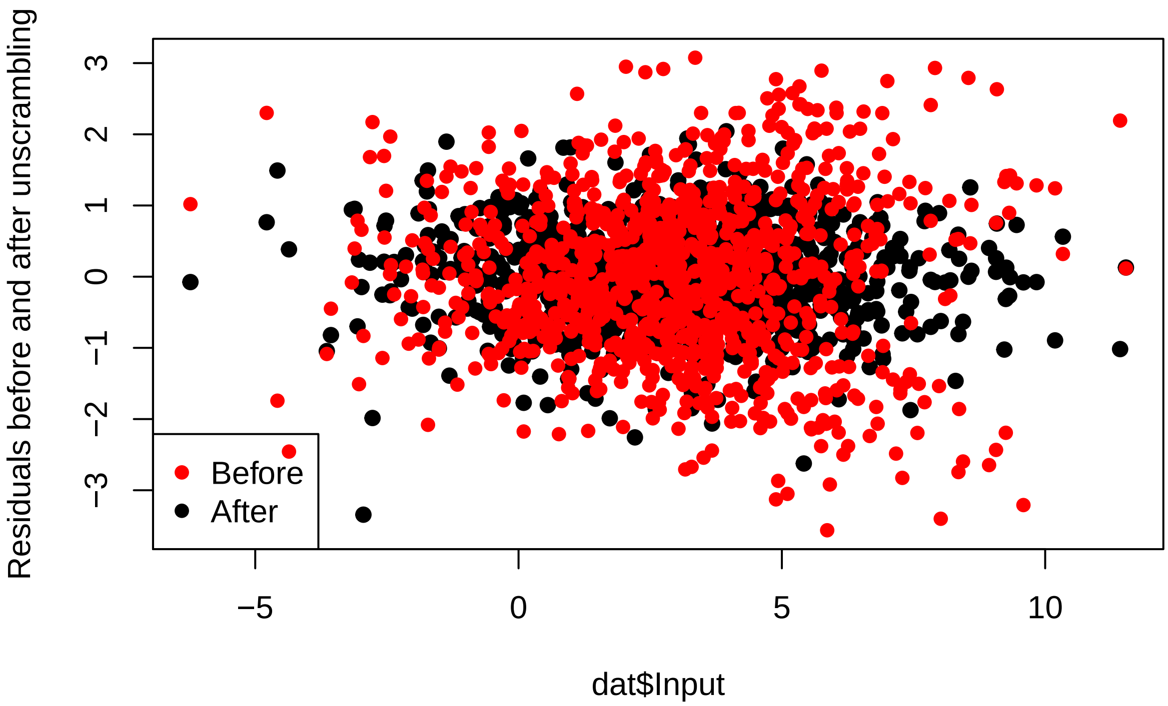
```
## [1] 0.902158
```

1.6 Analyze the Residuals

Compare the residuals of separated models with the residuals of the single model.

```r
# Plot residuals, c() adds the black, estimatedResiduals adds the red dots
matplot(dat$Input,cbind(c(summary(Linear.Model.Steeper.Recovered)$residuals,
                           summary(Linear.Model.Flatter.Recovered)$residuals),
```

```
                        estimatedResiduals),type="p",pch=c(19,16),ylab="Residuals before and after unsc
legend("bottomleft",legend=c("Before","After"),col=c("red","black"),pch=16)
```



Estimate standard deviations of the residuals.

```
# Estimate standard deviations
unmixedResiduals<-c(summary(Linear.Model.Steeper.Recovered)$residuals,
                                    summary(Linear.Model.Flatter.Recovered)$residuals)
apply(cbind(ResidualsAfter=unmixedResiduals,
            ResidualsBefore=estimatedResiduals),2,sd)
```

```
##  ResidualsAfter ResidualsBefore
##       0.6863947       1.1564568
```

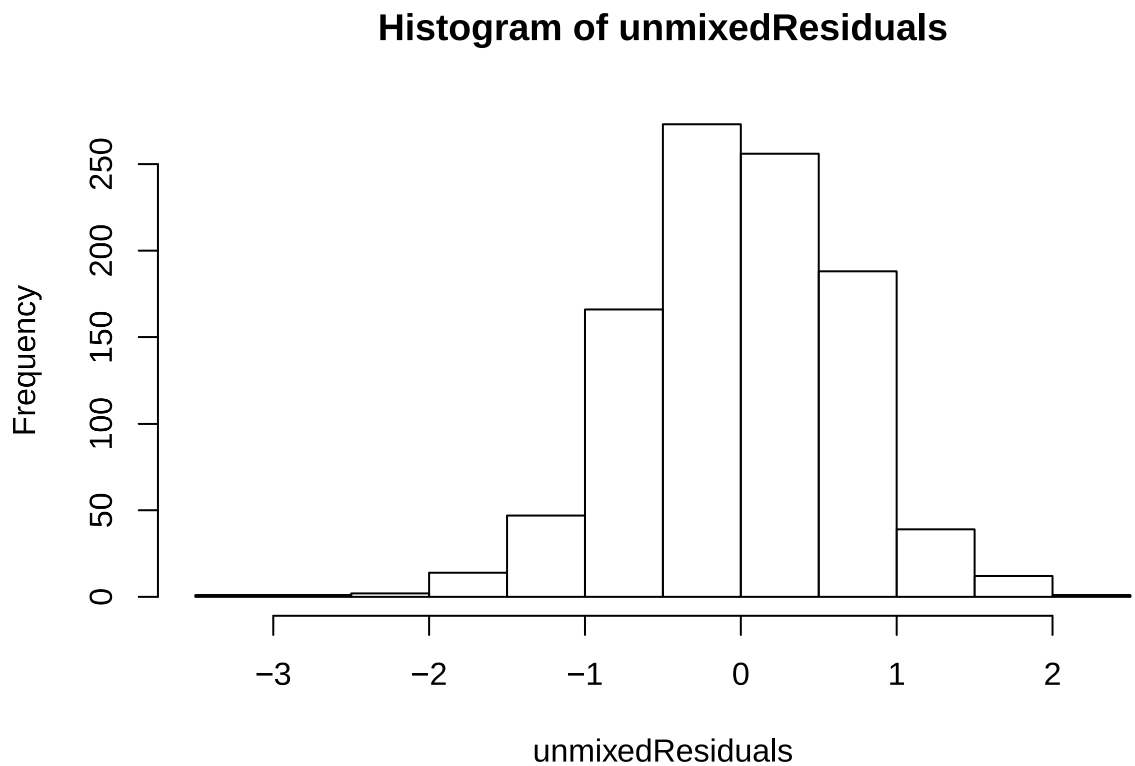
Check assumptions about the residuals

```
suppressWarnings(library(fitdistrplus))
```

```
## Loading required package: MASS
```

```
## Loading required package: survival
```

```
hist(unmixedResiduals)
```



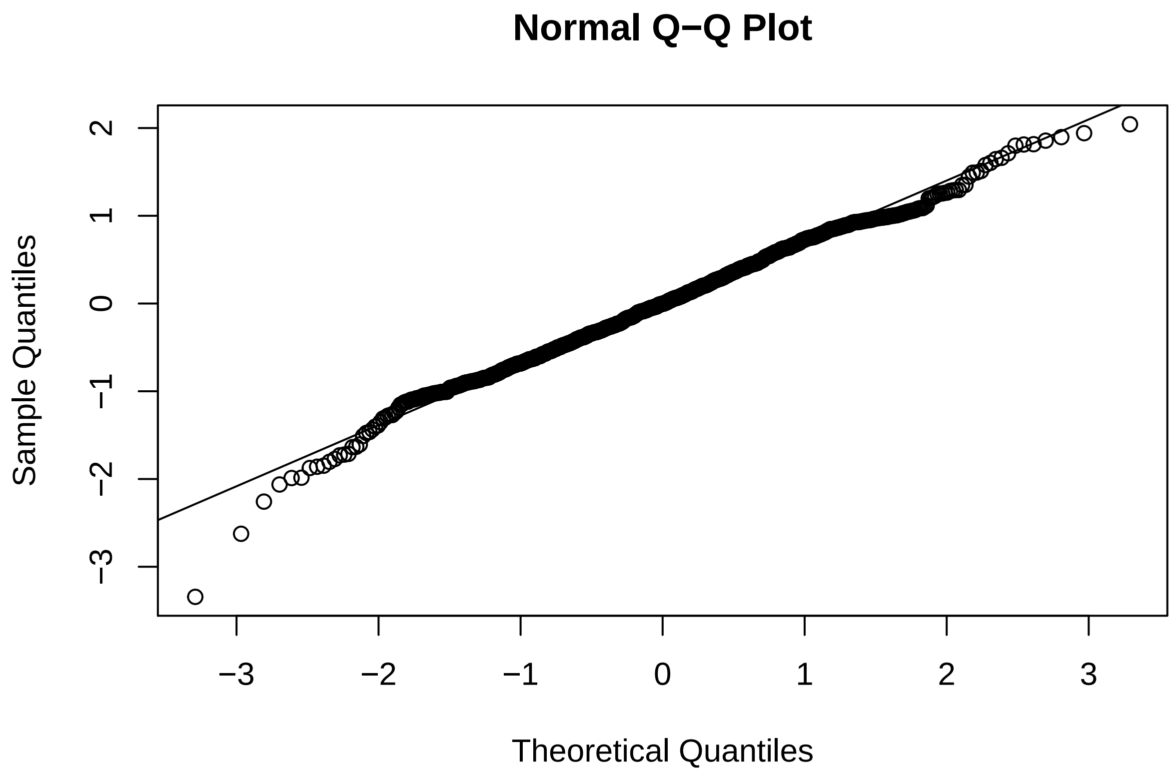

```r
(residualsParam<-fitdistr(unmixedResiduals,"normal"))
```

```
##        mean             sd
##    2.399123e-17   6.860514e-01
##   (2.169485e-02) (1.534058e-02)
```

```r
ks.test(unmixedResiduals,"pnorm",residualsParam$estimate[1],residualsParam$estimate[2])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  unmixedResiduals
## D = 0.023344, p-value = 0.6471
## alternative hypothesis: two-sided
```

```r
qqnorm(unmixedResiduals)
qqline(unmixedResiduals)
```



Finally, print out the slopes and intercepts of both models.

```
# Slopes
c(Steeper.SLope=Linear.Model.Steeper.Recovered$coefficients[2],Flatter.Slope=Linear.Model.Flatter.Recove
```

```
## Steeper.SLope.steeperInput Flatter.Slope.flatterInput
##                  1.0517077                  0.8630519
```

```
# Intercepts
c(Steeper.Intercept=Linear.Model.Steeper.Recovered$coefficients[1],Flatter.Intercept=Linear.Model.Flatte
```

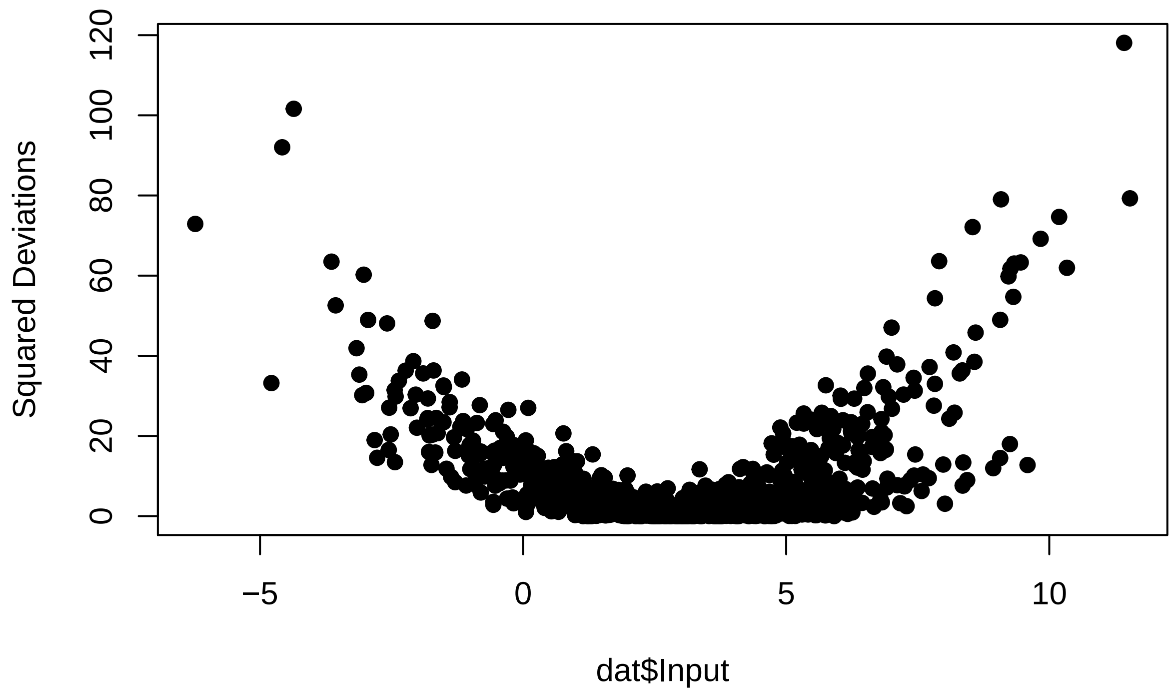
```
## Steeper.Intercept.(Intercept) Flatter.Intercept.(Intercept)
##                     0.9325475                    -0.3467106
```

2 Alternative Method Based on Volatility Clustering

If the sample is <y1,..,yn> then estimate of variance is built by averaging terms (yi - ybar)^2

Make a plot of squared deviations zi=(yi - ybar)^2.

```
plot(dat$Input,(dat$Output-mean(dat$Output))^2, type="p",pch=19,
     ylab="Squared Deviations")
```



Data points on this plot seem to cluster in two or more parabolic shapes.

Use the following interactive application to understand how change in slope of simple linear model affects the shape of the plot of zi=(yi - ybar)^2. (link that doesn't work)

An alternative approach to unmixing the models can be based on separating two parabolas on the data plot.

Explain how increased slope affects variance of the output and the pattern of variables zi. What are the differences between the shapes of parabolas corresponding to a steeper slope versus flatter slope?

Separate the models using this approach.

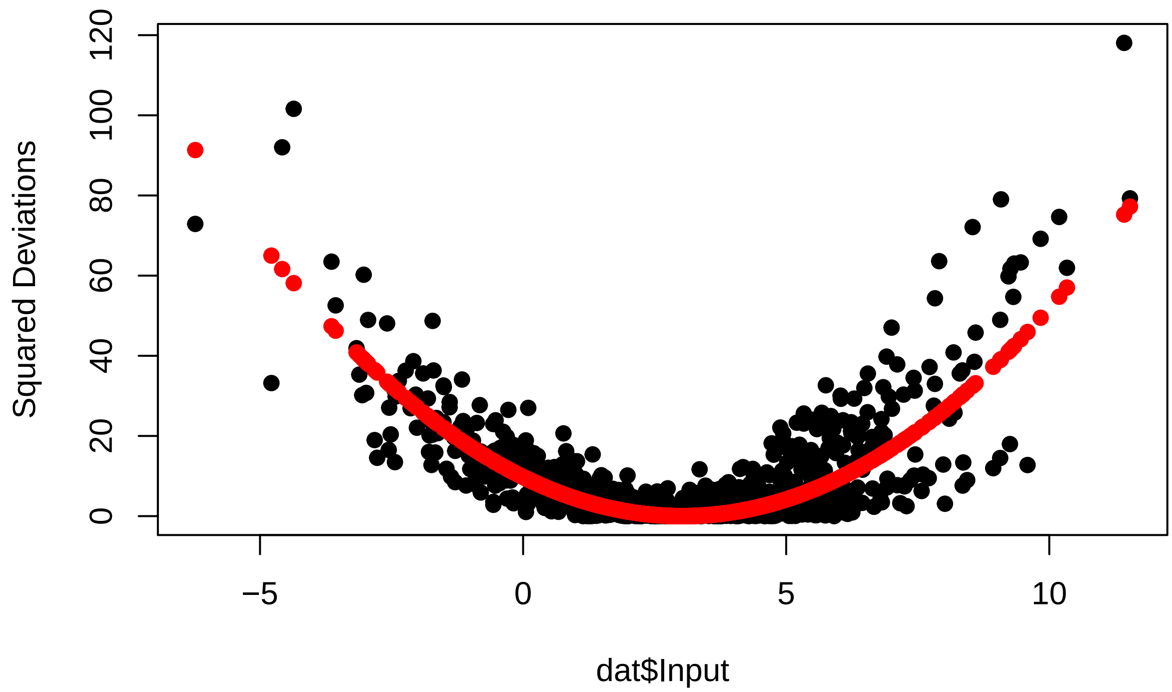
Find parabola corresponding to fitted model m1.

Hint. Find ybar using model expression yi=Beta0+Beta1xi+Ei.

Then substitute Beta0hat, Beta1hat estimated by linear model and form (yi - ybar)^2

Separate clusters using clustering parabola defined by the fitted model.

```
clusteringParabola<-(m1$fitted.values-mean(m1$fitted.values))^2
plot(dat$Input,(dat$Output-mean(dat$Output))^2, type="p",pch=19,ylab="Squared Deviations")
points(dat$Input,clusteringParabola,pch=19,col="red")
```



Define the separating sequence Unscrambling.Sequence.Steeper.var, such that it is equal to TRUE for steeper slope subsample and FALSE for flatter slope subsample.

```
Unscrambling.Sequence.Steeper.var<-(dat$Output-mean(dat$Output))^2>clusteringParabola
head(Unscrambling.Sequence.Steeper.var,10)
```

```
##     1     2     3     4     5     6     7     8     9    10
##  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
```

So where the actual value (looking at variance) is above the "fitted value" (looking at variance) it is part of the steeper sequence.

Separate the sample into steeper and flatter part. Create data frames. Define two subsamples with NAs in the Output columns

```
Subsample.Steeper.var<-
  data.frame(steeperInput.var=dat$Input,steeperOutput.var=rep(NA,nSample))
Subsample.Flatter.var<-
  data.frame(flatterInput.var=dat$Input,flatterOutput.var=rep(NA,nSample))
```

Fill in the unscrambled outputs instead of NAs where necessary

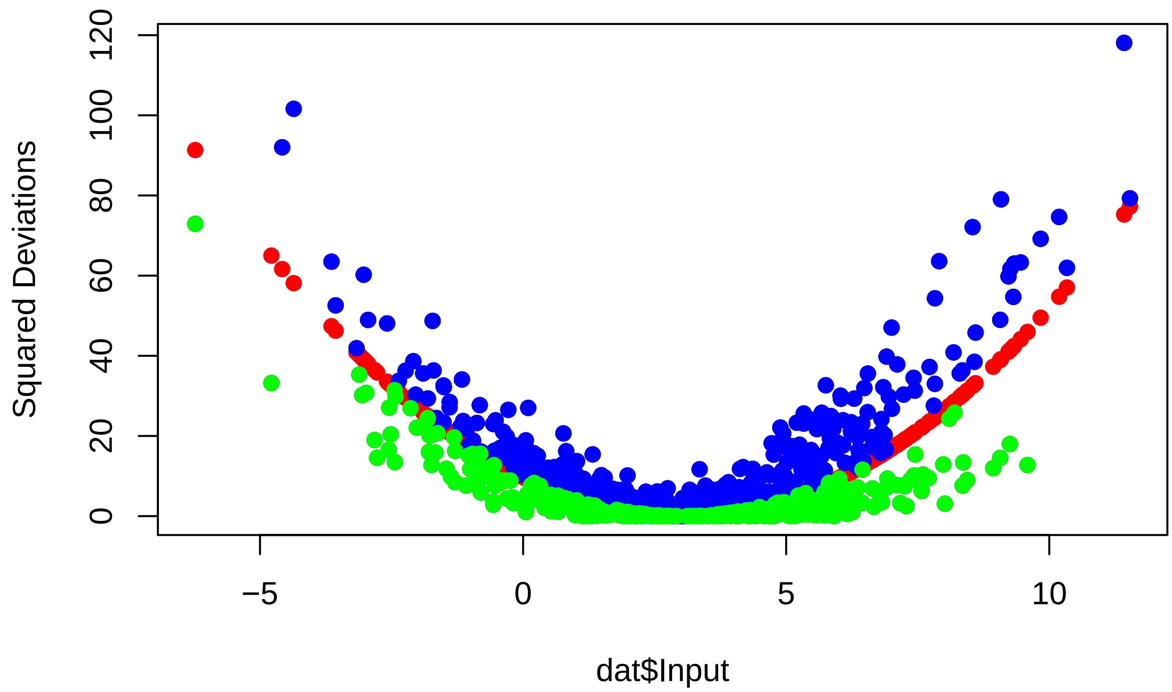
```
Subsample.Steeper.var[Unscrambling.Sequence.Steeper.var,2]<-
  dat[Unscrambling.Sequence.Steeper.var,2]
Subsample.Flatter.var[!Unscrambling.Sequence.Steeper.var,2]<-
  dat[!Unscrambling.Sequence.Steeper.var,2]
# Check the first 10 rows
head(cbind(dat,Subsample.Steeper.var,Subsample.Flatter.var),10)
```

```
##        Input      Output steeperInput.var steeperOutput.var
## 1  3.1328589  4.17792255        3.1328589        4.17792255
## 2  5.5611337  5.84669919        5.5611337        5.84669919
## 3  1.9845429 -0.09834184        1.9845429       -0.09834184
## 4  5.6191601  7.84692946        5.6191601        7.84692946
## 5  6.3781486  7.57941491        6.3781486        7.57941491
## 6  6.1232040  5.68973137        6.1232040                NA
## 7  0.7666195 -1.45675560        0.7666195       -1.45675560
## 8  4.3535141  4.16746077        4.3535141                NA
## 9  2.3627156  2.38611901        2.3627156        2.38611901
## 10 6.3272368  6.77354738        6.3272368        6.77354738
##    flatterInput.var flatterOutput.var
## 1         3.1328589                NA
## 2         5.5611337                NA
## 3         1.9845429                NA
## 4         5.6191601                NA
## 5         6.3781486                NA
## 6         6.1232040          5.689731
## 7         0.7666195                NA
## 8         4.3535141          4.167461
## 9         2.3627156                NA
## 10        6.3272368                NA
```

Notice it is still the original input, output values, just unscrambled differently from before

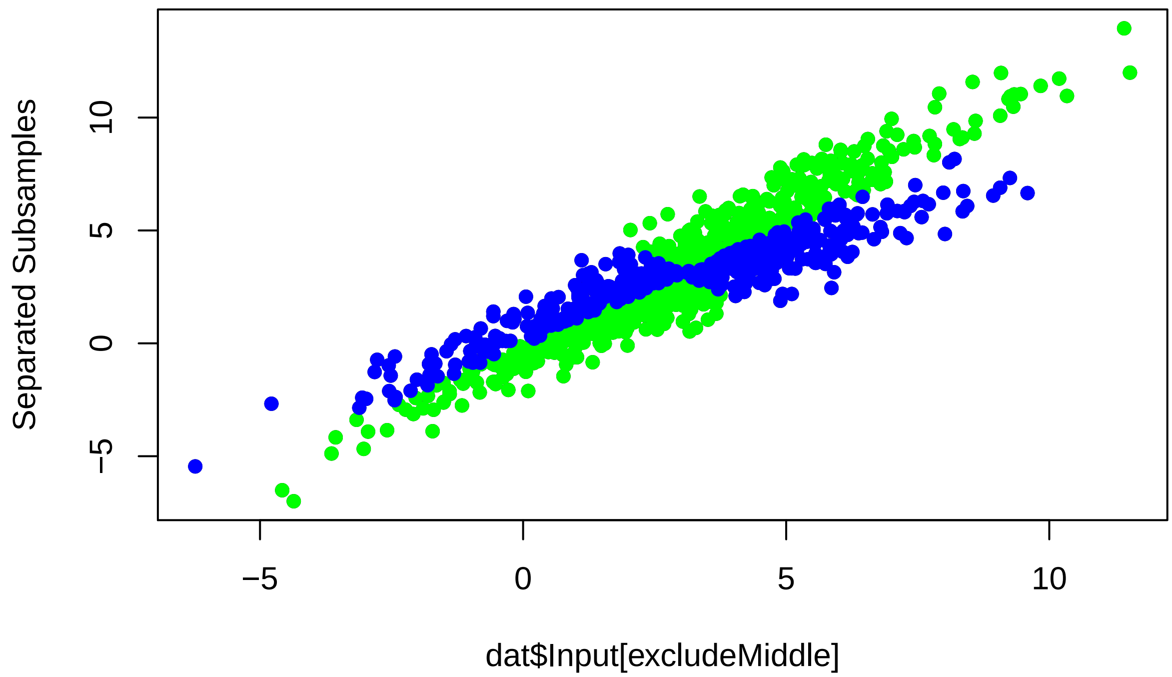
Plot clusters of the variance data and the separating parabola

```
plot(dat$Input,
     (dat$Output-mean(dat$Output))^2,
     type="p",pch=19,ylab="Squared Deviations")
points(dat$Input,clusteringParabola,pch=19,col="red")
points(dat$Input[Unscrambling.Sequence.Steeper.var],
       (dat$Output[Unscrambling.Sequence.Steeper.var]-
          mean(dat$Output))^2,
       pch=19,col="blue")
points(dat$Input[!Unscrambling.Sequence.Steeper.var],
       (dat$Output[!Unscrambling.Sequence.Steeper.var]-
          mean(dat$Output))^2,
       pch=19,col="green")
```



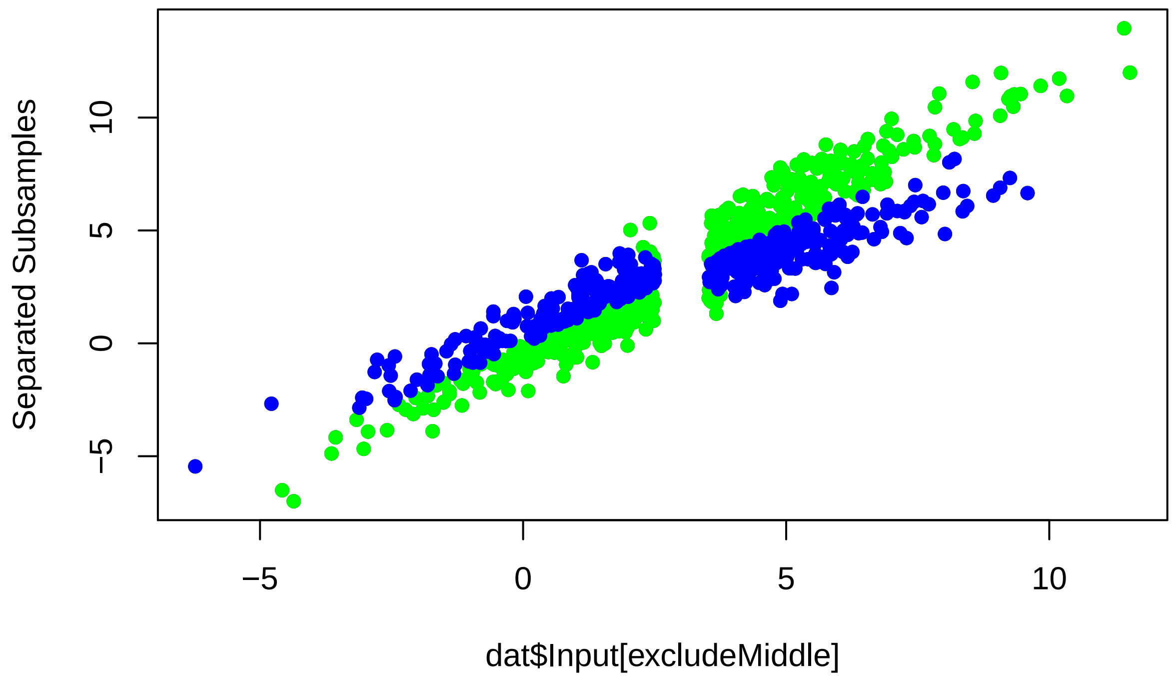
Plot the unscrambled subsamples, include the original entire sample as a check.

```
excludeMiddle<-(dat$Input<=mean(dat$Input)-0)|
                (dat$Input>=mean(dat$Input)+0)
matplot(dat$Input[excludeMiddle],cbind(dat$Output[excludeMiddle],
                                       Subsample.Steeper.var$steeperOutput.var[excludeMiddle],
                                       Subsample.Flatter.var$flatterOutput.var[excludeMiddle]),
        type="p",col=c("black","green","blue"),
        pch=16,ylab="Separated Subsamples")
```



Note that observations corresponding to the minimum of the variance data are difficult to separate. Consider omitting some observations around that point. For example, make omitted interval equal to LeftBound=-0.5, RightBound=0.5.

```
#this is the line from above function you change
excludeMiddle<-(dat$Input<=mean(dat$Input)-.5)|
                (dat$Input>=mean(dat$Input)+.5)
matplot(dat$Input[excludeMiddle],cbind(dat$Output[excludeMiddle],
                                       Subsample.Steeper.var$steeperOutput.var[excludeMiddle],
                                       Subsample.Flatter.var$flatterOutput.var[excludeMiddle]),
        type="p",col=c("black","green","blue"),
        pch=16,ylab="Separated Subsamples")
```

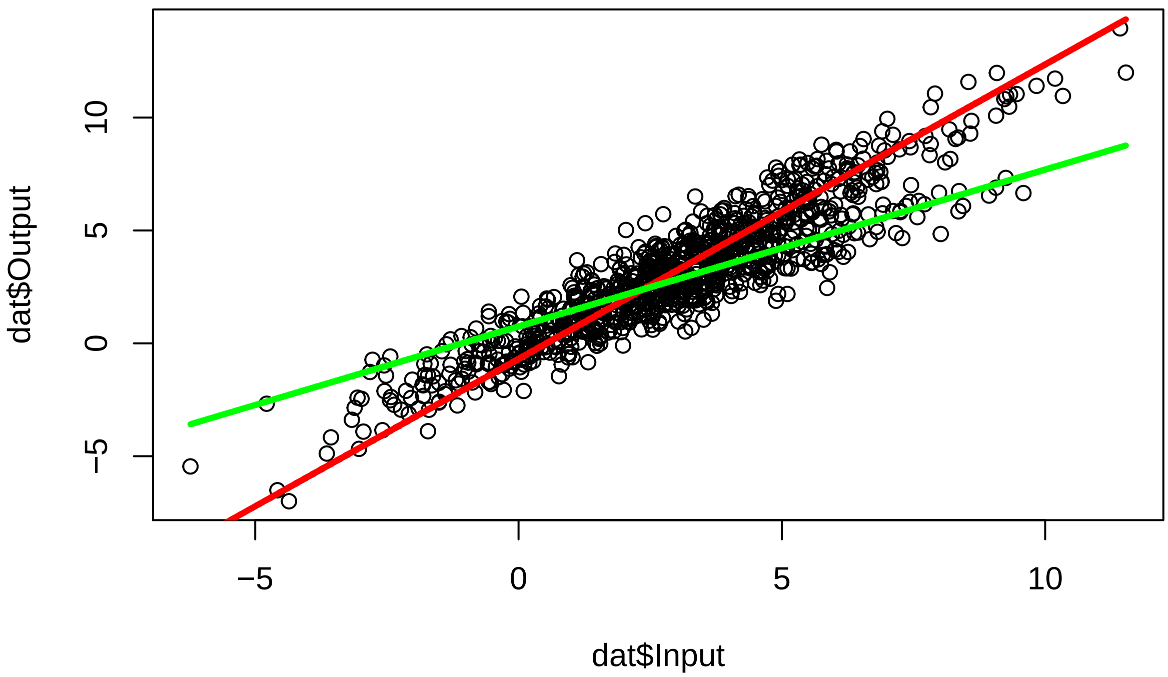


Fit linear models to the separated samples.

```
dat.Steep.var<-lm(Subsample.Steeper.var$steeperOutput.var[excludeMiddle]~dat$Input[excludeMiddle])
dat.Flat.var<-lm(Subsample.Flatter.var$flatterOutput.var[excludeMiddle]~dat$Input[excludeMiddle])
```

Plot the data and the estimated regression lines

```
plot(dat$Input,dat$Output, type="p",pch=1)
lines(dat$Input[excludeMiddle],predict(dat.Steep.var,
                              data.frame(trainSteepInput=dat$Input[excludeMiddle]),
                              interval="prediction")[,1],col="red",lwd=3)
lines(dat$Input[excludeMiddle],predict(dat.Flat.var,data.frame(trainFlatInput=dat$Input[excludeMiddle])
                              interval="prediction")[,1],col="green",lwd=3)
```



Print estimated parameters and summaries of both models

```
rbind(Steeper.Coefficients.var=dat.Steep.var$coefficients,
      Flatter.Coefficients.var=dat.Flat.var$coefficients)
```

```
##                          (Intercept) dat$Input[excludeMiddle]
## Steeper.Coefficients.var  -0.6944529                1.3045575
## Flatter.Coefficients.var   0.7454309                0.6950256
```

```
summary(dat.Steep.var)
```

```
##
## Call:
## lm(formula = Subsample.Steeper.var$steeperOutput.var[excludeMiddle] ~
##     dat$Input[excludeMiddle])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.78292 -0.45940 -0.04679  0.48595  3.05298
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -0.69445    0.06083  -11.42   <2e-16 ***
## dat$Input[excludeMiddle]  1.30456    0.01443   90.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8488 on 441 degrees of freedom
```

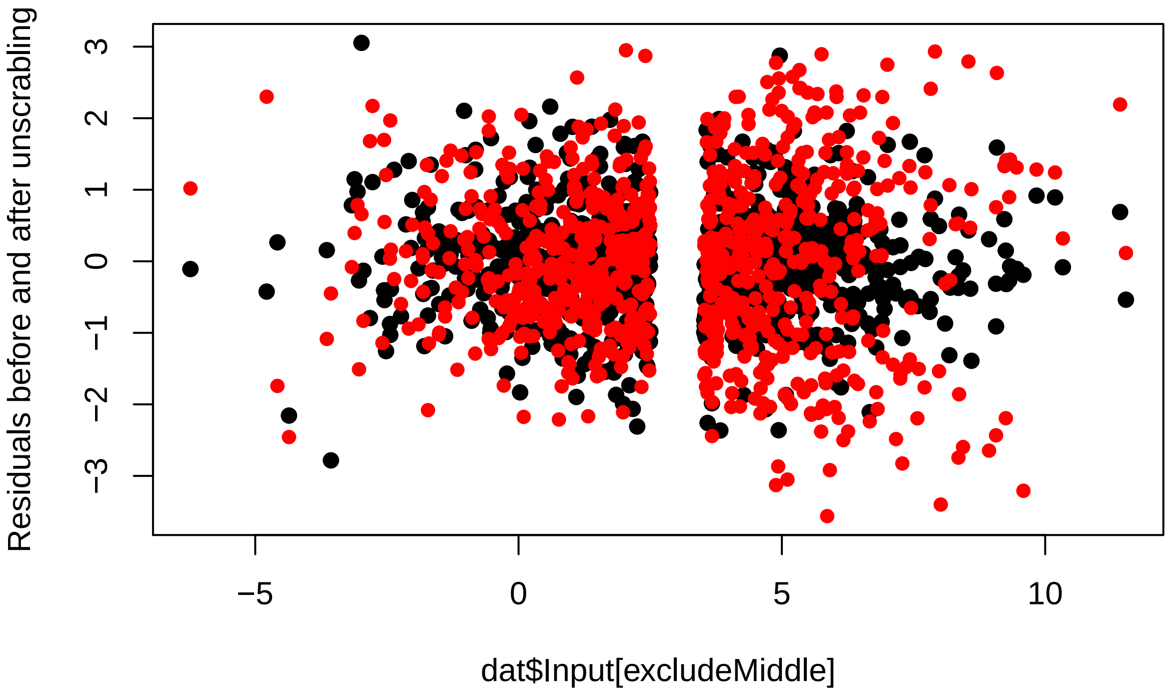
```
##   (382 observations deleted due to missingness)
## Multiple R-squared:  0.9488, Adjusted R-squared:  0.9487
## F-statistic:  8176 on 1 and 441 DF,  p-value: < 2.2e-16
```

```r
summary(dat.Flat.var)
```

```
##
## Call:
## lm(formula = Subsample.Flatter.var$flatterOutput.var[excludeMiddle] ~
##     dat$Input[excludeMiddle])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3651 -0.4251  0.0361  0.5039  2.1618
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.74543    0.05434   13.72   <2e-16 ***
## dat$Input[excludeMiddle]  0.69503    0.01369   50.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7309 on 380 degrees of freedom
##   (443 observations deleted due to missingness)
## Multiple R-squared:  0.8714, Adjusted R-squared:  0.8711
## F-statistic:  2576 on 1 and 380 DF,  p-value: < 2.2e-16
```

Plot residuals from the combined model and the models for separated samples

```r
matplot(dat$Input[excludeMiddle],
        cbind(c(summary(dat.Steep.var)$residuals,
                summary(dat.Flat.var)$residuals),
              estimatedResiduals[excludeMiddle]),
        type="p",pch=c(19,16),ylab="Residuals before and after unscrabling")
```


Residuals before and after unscrabling
dat$Input[excludeMiddle]
−5
0
5
10
−3
−2
−1
0
1
2
3