Week 1: Homework Assignment

Patrick Kelly

Saturday, January 17, 2015

Objective

This project helps understanding the linear model as a particular case of generalized linear model.

The project is due on the day of the second class, 11:59 pm.

This project is individual.

Assignment description

Use the data from the file Week1_Homework_Project_Data.csv to estimate linear model with `lm()`. Analize `summary()` of the estimated linear model.

What can you tell about the data and the fit?

```
Linear.Model.Data <- read.csv("C:/Users/Patrick/SkyDrive/Documents/Education/UChicago/Linear_NonLinear/Week
1_Homework_Project_Data.csv",header=TRUE,sep=",")
Linear.Model.Data[1:10,]
```

```
##       Output    Input1    Input2       Input3
## 1   5.451630 1.875976 2.721764 -0.011754874
## 2   4.544792 1.713108 2.570596  0.007136568
## 3   5.345051 2.804193 4.700366  0.056975906
## 4  11.239833 3.895376 6.863106 -0.103290465
## 5  10.055252 3.694171 6.405116 -0.142105913
## 6   1.720138 1.104742 1.203397  0.241145841
## 7   9.010973 2.947958 4.968758 -0.059639593
## 8   5.860532 2.743866 4.545334  0.048181014
## 9  13.368089 4.595102 8.226541 -0.189332876
## 10 10.193232 3.639261 6.228819 -0.074765940
```

```
Linear.Model.Data.Frame<-as.data.frame(Linear.Model.Data)
```

Estimate the model using `lm()`. Analize the summary of the model: what can you tell from it?

```
Linear.Model.Data.lm<-lm(Output~Input1+Input2+Input3,data=Linear.Model.Data)
summary(Linear.Model.Data.lm)
```

```
##
## Call:
## lm(formula = Output ~ Input1 + Input2 + Input3, data = Linear.Model.Data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4384 -0.8355  0.0354  0.8174  3.7067
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.3070     0.5617  -4.108 4.68e-05 ***
## Input1        2.3842     1.0733   2.221   0.0268 *
## Input2        0.6476     0.5341   1.212   0.2259
## Input3        0.6869     0.5339   1.286   0.1989
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.237 on 496 degrees of freedom
## Multiple R-squared:   0.95,  Adjusted R-squared:  0.9497
## F-statistic:  3143 on 3 and 496 DF,  p-value: < 2.2e-16
```

To check the answers estimate `glm`.

```
Returned.from.glm<-glm(Output~Input1+Input2+Input3,family=gaussian(link="identity"),data=Linear.Model.Data)
names(Returned.from.glm)
```

```
##  [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"
```

```
Returned.from.glm$coefficients
```

```
## (Intercept)      Input1      Input2      Input3
##  -2.3070078   2.3841593   0.6475835   0.6868760
```

Analysis:

The model seems to be a good fit with a multiple R-squared of 0.95. Also the coefficients for the intercept and Input1 seem to be good fits for the data as their pvalues are both significant. Although the coefficients for Input2 and Input3 are not significant they are still relevant with pvalues ~0.2.

By using any variables returned by `lm()` or `summary(lm.fit)` calculate the following characteristics of `glm()` that you would obtain if applied `glm()` to the same data.

You use `glm()` to check your answers, but, please, do not use `glm` object or any functions applied to `glm` object to calculate your results.

Calculate variables:

1. `coefficients` (5%)
2. `residuals` (5%)
3. `fitted.values` (5%)
4. `linear.predictors` (10%)
5. `deviance` (25%)
    1. Use `deviance()` (10%)
    2. Calculate deviance manually based on the definition given in the lecture (15%)
6. Akaike Information Criterion `aic` (25%)
    1. Obtain it by using `AIC()` (10%)
    2. Calculate it manually using the definition given in the lecture. (15%)
7. `y` (5%)
8. `null.deviance` (10%)

9. `dispersion` (10%)

first, fit `lm()` and `glm()`

```
lm.fit<-lm(Output~Input1+Input2+Input3,data=Linear.Model.Data)
glm.fit<-glm(Output~Input1+Input2+Input3,family=gaussian(link="identity"),data=Linear.Model.Data)
```

Variable 1 - coefficients

```
#get coefficients from lm()
lm.fit$coefficients
```

```
## (Intercept)      Input1      Input2      Input3
##  -2.3070078   2.3841593   0.6475835   0.6868760
```

```
#check these against the coefficients glm()
glm.fit$coefficients
```

```
## (Intercept)      Input1      Input2      Input3
##  -2.3070078   2.3841593   0.6475835   0.6868760
```

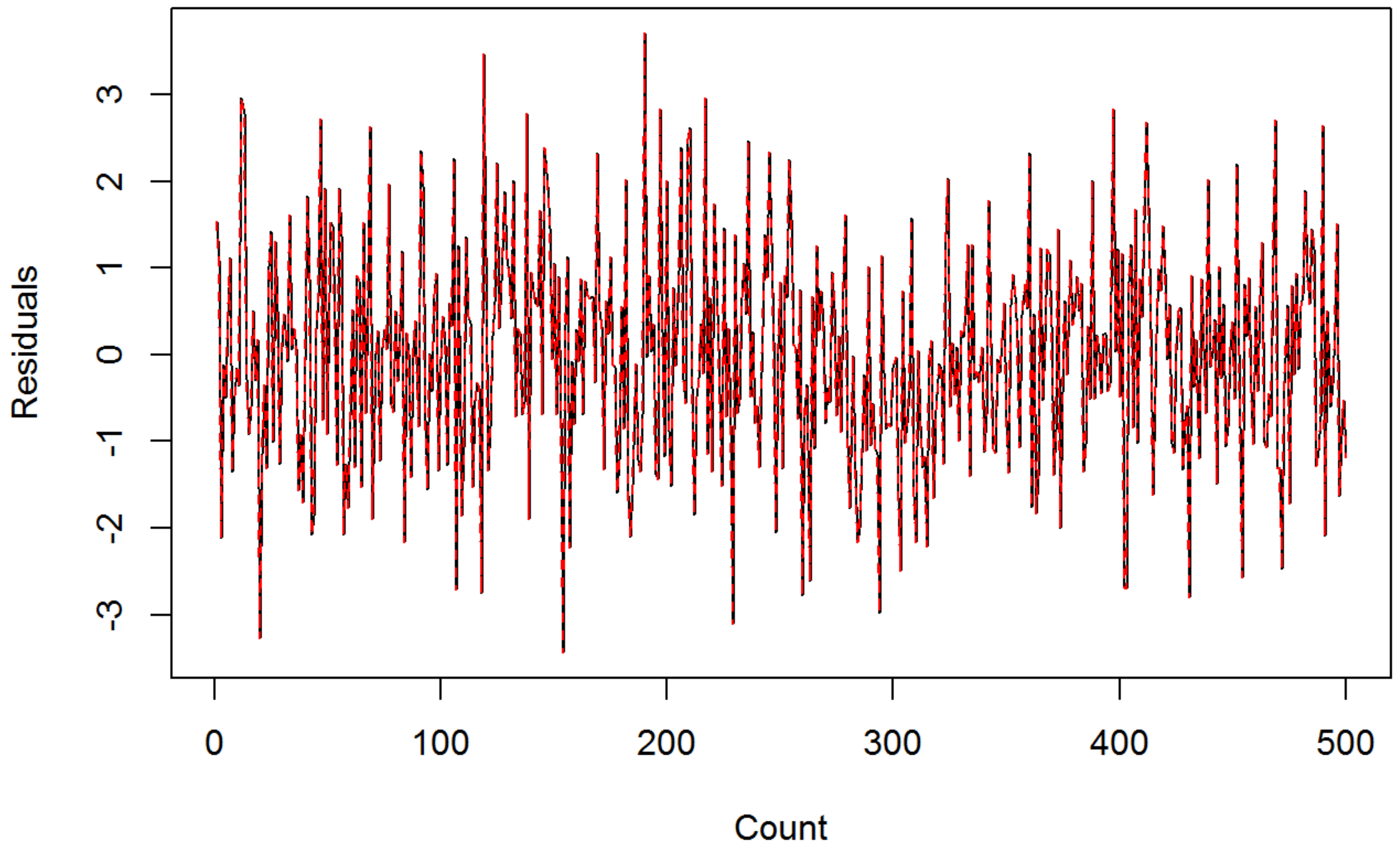
These values are equal.

Variable 2 - residuals ... from the example that was provided

Residuals from `glm()` are calculated the same way as from `lm()`.

To show equivalence you can, for example, plot the residuals from both `lm()` and `glm()` and also check the deviations from one another like it is done below.

```
matplot(1:length(Linear.Model.Data[,1]),cbind(Linear.Model.Data.lm$residuals,Returned.from.glm$residuals),t
ype="l",ylab="Residuals",xlab="Count")
```



```
sum(abs(Linear.Model.Data.lm$residuals-Returned.from.glm$residuals)>.00000000001)
```

```
## [1] 0
```

The last output shows that maximum deviation of Linear.Model.Data.lm $residuals from Returned. from. glm$residuals in absolute value is less than or equal to 0.0000000001.

Variable 3 - fitted values

```
#get fitted values from lm() --- display the first 20
lm.fit$fitted.values[1:20]
```

```
##         1         2         3         4         5         6         7 
##  3.920114  3.446893  7.461650 11.353676 10.550724  1.271810  7.898114 
##         8         9        10        11        12        13        14 
##  7.211383 13.845771 10.351894 -1.986692  4.260614 16.033563 16.719832 
##        15        16        17        18        19        20 
##  7.549951 13.410357 12.575873  8.316385  5.187473  5.175219 
```

```
#prove that these fitted values are identical to those from the glm model
#use the same methodology to compare fitted values as was used in comparing the residuals above
sum(abs(lm.fit$fitted.values - glm.fit$fitted.values) > 0.00000000001)
```

```
## [1] 0
```

These values are equal.

Variable 4 - linear predictors

```
#get linear predictors from lm()
```

```
#since the link function = identity, the linear predictors are equalivalent to the fitted values
#display the first linear predictors / fitted values
lm.fit$fitted.values[1:20]
```

```
##         1         2         3         4         5         6         7
##  3.920114  3.446893  7.461650 11.353676 10.550724  1.271810  7.898114
##         8         9        10        11        12        13        14
##  7.211383 13.845771 10.351894 -1.986692  4.260614 16.033563 16.719832
##        15        16        17        18        19        20
##  7.549951 13.410357 12.575873  8.316385  5.187473  5.175219
```

```
#prove that these fitted values are identical to the linear predictors from the glm model
#use the same methodology as above
sum(abs(lm.fit$fitted.values - glm.fit$linear.predictors) > 0.00000000001)
```

```
## [1] 0
```

These values are equal.

Variable 5 - deviance

1. Use `deviance()`

```
deviance(lm.fit)
```

```
## [1] 758.9499
```

```
#check the deviance from the glm fit
glm.fit$deviance
```

```
## [1] 758.9499
```

These values are equal.

2. Calculate deviance manually based on the definition given in the lecture

```
#manual calculation of log-likelihood
log.like.fn <- function(residuals){
  n<-length(residuals)
  sigma.sq <- sd(residuals)^2
  term1 <- -(n/2)* log(2*pi*sigma.sq)
  term2 <- -(1/(2*sigma.sq))*(sum((residuals)^2))

  ll.output <- term1 + term2

  return(ll.output)
}

#manual calculation of deviance
deviance.fn <- function(fit){

  n <- length(fit$residuals)
  sigma.sq <- var(fit$residuals)

  saturated <- -(n/2) * log(2*pi*sigma.sq)
  estimated <- log.like.fn(fit$residuals)

  dev <- 2*sigma.sq*(saturated - estimated)

  return(dev)
}

deviance.fn(lm.fit)
```

```
## [1] 758.9499
```

```
#also check that your deviance is equal to the linear model SSE
all.equal(sum(lm.fit$residuals^2), deviance.fn(lm.fit))
```

```
## [1] TRUE
```

These values are equal.

Variable 6 - Akaike Information Criterion

1. Obtain it by using `AIC()`

```
AIC(lm.fit)
```

```
## [1] 1637.602
```

```
#check the deviance from the glm fit
```

```
glm.fit$aic
```

```
## [1] 1637.602
```

These values are equal.

2. Calculate it manually using the definition given in the lecture

```
#manual calculation of log-likelihood
log.like.fn2 <- function(residuals){
  n<-length(residuals)
#   sigma.sq <- sd(residuals)^2
  sigma.sq <- mean(residuals^2)
  term1 <- -(n/2)* log(2*pi*sigma.sq)
  term2 <- -(1/(2*sigma.sq))*(sum((residuals)^2))

  ll.output <- term1 + term2

  return(ll.output)
}

#manually calculate the AIC value
aic.fn <- function(fit){

  resids <- fit$residuals
  p <- length(fit$coefficients)

  return( (-2)*(log.like.fn2(resids)) + (2*p))
}

aic.fn(lm.fit)
```

```
## [1] 1635.602
```

These values are nearly equal.

The difference between the 2 is insignificant and may be due to 2 different methods for calculating AIC. Through some basic internet searches it becomes apparent that there are several different common methodologies for calculating AIC.

Variable 7 - y

The `y` variables are simply equal to the “Output” data from our Linear.Model.Data

```
#check the first 20 values
Linear.Model.Data[1:20,1]
```

```
##  [1]  5.451630  4.544792  5.345051 11.239833 10.055252  1.720138  9.010973
##  [8]  5.860532 13.368089 10.193232 -2.350510  7.217363 18.810761 16.441207
## [15]  6.628425 12.816096 13.079441  8.023034  5.367397  1.908260
```

```
#check that our "Output" data is equal to the glm `y` variable
check.y <- cbind(glm.fit$y,Linear.Model.Data[,1])
all.equal(check.y[,1],check.y[,2])
```

```
## [1] TRUE
```

These values are equal.

Variable 8 - null.deviance

```
glm.fit$null.deviance
```

```
## [1] 15187.73
```

```
#manual calculation of null.deviance


#first fit null model
lm.nullfit<-lm(Output~1,data=Linear.Model.Data)


#then use the deviance fn that I created above to determine the null deviance
deviance.fn(lm.nullfit)
```

```
## [1] 15187.73
```

These values are equal.

Variable 9 - dispersion

```
# the dispersion from the glm model can be observed in the output from summary(glm.fit)
summary(glm.fit)
```

```
##
```

```
## Call:
## glm(formula = Output ~ Input1 + Input2 + Input3, family = gaussian(link = "identity"),
##     data = Linear.Model.Data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4384  -0.8355   0.0354   0.8174   3.7067
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.3070     0.5617  -4.108 4.68e-05 ***
## Input1        2.3842     1.0733   2.221   0.0268 *
## Input2        0.6476     0.5341   1.212   0.2259
## Input3        0.6869     0.5339   1.286   0.1989
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.530141)
##
##     Null deviance: 15187.73  on 499  degrees of freedom
## Residual deviance:   758.95  on 496  degrees of freedom
## AIC: 1637.6
##
## Number of Fisher Scoring iterations: 2
```

```
# you can see that it is equal to 1.530141

#the dispersion is equal to the (residual squared error)^2
```

```r
#below is the calculation for dispersion from the lm() fit
sum(lm.fit$residuals^2) / lm.fit$df.residual
```

```
## [1] 1.530141
```

Typesetting math: 100% equal.