STAT511HW4

Ben Straub

October 3, 2015

(4) Replicating "lm" by Hand.

Coefficients for Beta given by R

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-17.579	6.758	-2.601	0.012
speed	3.932	0.416	9.464	0.000

(4a) Estimate

$$\hat{\beta} = (X \cdot X')^{-}1 \cdot X' \cdot Y$$

```
# Creating X and I vectors from cars$speed data
X = cars$speed
X <- matrix(X,nrow = 50,ncol = 1)
I <- matrix(1,nrow = 50,ncol = 1)
X = cbind(I, X)
# Transpose of X
Xt = t(X)
# Multiplying the Transpose of X and X
XtX = Xt%*%X
# Taking the Inverse of the Transpose of X and X
invXtX <- solve(XtX)
# Creating a Y vector from the Cars data for distance
Y = cars$dist
Y <- matrix(Y,nrow = 50,ncol = 1)
# Multiplying the transpose of X and Y
XtY <- Xt%*%Y
# Multiplying to get final coefficients of Beta
coeff_Betas <- invXtX%*%XtY
# Pretty table of the Coeffecients
kable(coeff_Betas, digits=4, caption = "Beta0 and Beta1")
```

Table 2: Beta0 and Beta1

-17.5791
3.9324

(4b) The Residuals

$$\hat{\epsilon} = (I - H)Y$$

Residuals given by R

Table 3: Residual

3.8495
11.8495
-5.9478
12.0522
2.1198
-7.8126

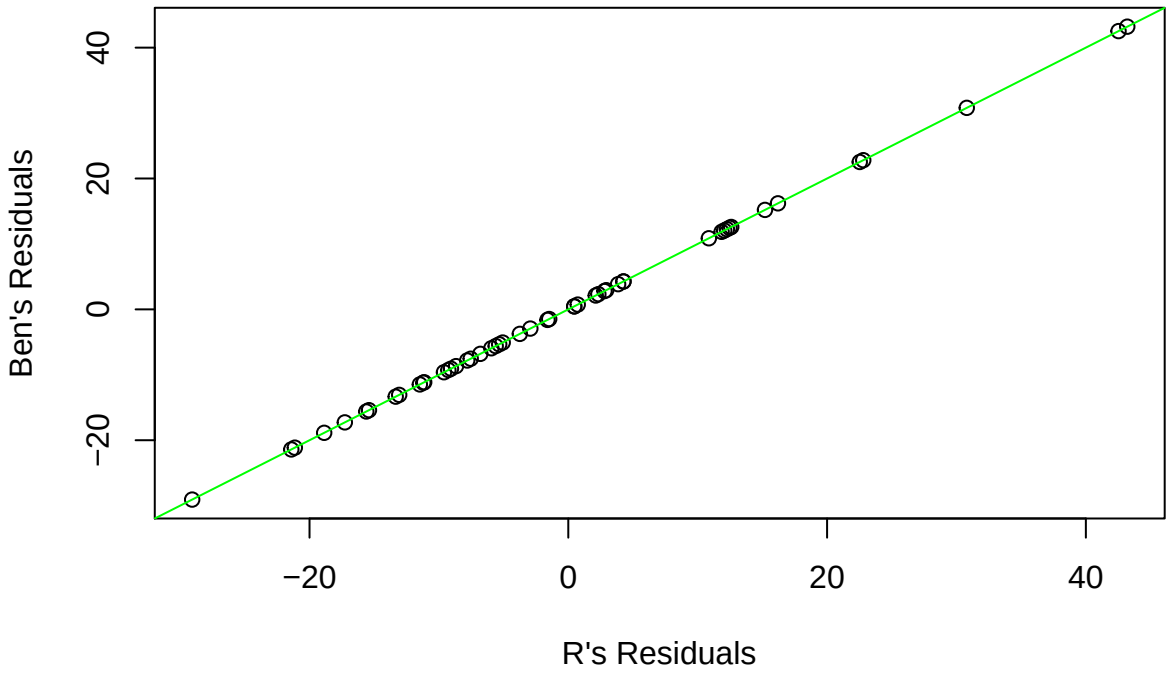
Calculation by hand

```
# Hat Matrix
Hat <- X%*%(invXtX)%*%Xt
# Identity Matrix
I <- diag(50)
# Estimated Errors
res_Err <- (I - Hat)%*%Y
kable(head(res_Err), digits=4, caption = "Residuals by hand")
```

Table 4: Residuals by hand

3.8495
11.8495
-5.9478
12.0522
2.1198
-7.8126

```
plot(res, res_Err, xlab="R's Residuals", ylab="Ben's Residuals")
abline(a=0,b=1, col="green")
```



(4c) The Studentized Residuals

Studentized Residuals given by R

```
par(mfrow=c(1,2))
kable(head(rstudent(fit)))
```

0.2634500
0.8160784
-0.3978115
0.8103526
0.1407033
-0.5171605

```
# kable(tail(rstudent(fit)))
```

Calculating by Hand

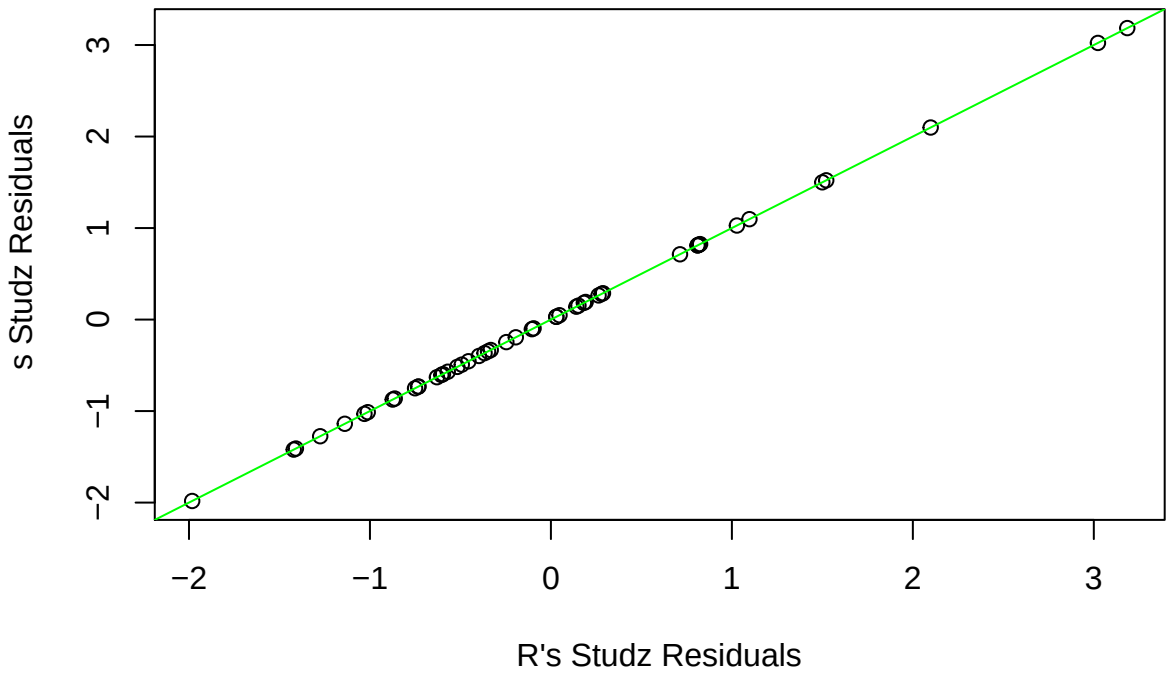
$$r = r \cdot ((n-p-1)/(n-p-r^2))^1/2$$

```
# set up for finding rank of matrix
y <- qr(I-Hat)
#rank of matrix
nminusp <- y$rank
#transpose of the Estimated Errors.
res_Errt <- t(res_Err)
```

```r
# Computing sigma sqaured
one_over_nminusp <- 1/nminusp
sigmasqrd <- (res_Errt%*%res_Err)*one_over_nminusp
# Computing sigma
sigma <- sqrt(sigmasqrd)
# Extracting Diagonal of the Hat Matrix
Hat_Diag <- diag(Hat)
# Computing Standardized Residuals
bottom_part <- sqrt(1- Hat_Diag)
sigma <- as.numeric(sigma)
bottom_part <- sigma*bottom_part
# Final computation for Standardized Residuals
r_ii <- res_Err/bottom_part
# Using Standardized to compute Studendized Residuals
res_Stud <- r_ii*(sqrt((47/(48-(r_ii)^2))))
# Pretty Table
kable(head(res_Stud))
```

0.2634500
0.8160784
-0.3978115
0.8103526
0.1407033
-0.5171605

```r
plot(rstudent(fit), res_Stud, xlab="R's Studz Residuals", ylab="Ben
     s Studz Residuals")
abline(a=0,b=1, col="green")
```



(4d) Sigma Hat Sqaured

R's sigma^2 hat

236.5317

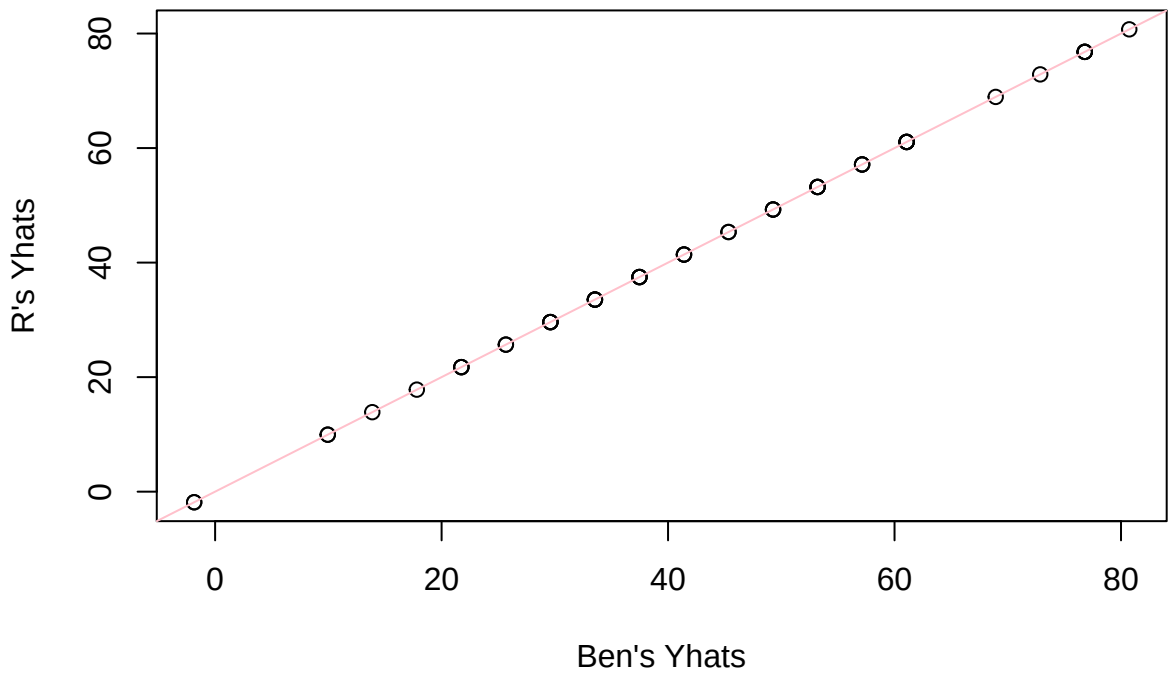
Calculation by Hand

```
# set up for finding rank of matrix
y <- qr(I-Hat)
#rank of matrix
nminusp <- y$rank
#transpose of the Estimated Errors.
res_Errt <- t(res_Err)
one_over_nminusp <- 1/nminusp
sigmasqrd <- (res_Errt%*%res_Err)*one_over_nminusp
sigmasqrd
```

```
##          [,1]
## [1,] 236.5317
```

(4e) Yhat

```
YHat <- Y - res_Err
yhat=fit$fitted
plot(YHat, yhat, xlab="Ben's Yhats", ylab="R's Yhats")
abline(a=0, b=1, col="pink")
```



(4f) se_k k = 1, 2. . . .p

```
invXtX_kk <- diag(invXtX)
se_Beta <- sqrt(sigmasqrd*invXtX_kk)
kable(summary(fit)$coeff[3:4], digits=3, caption="Standard Error from fit Model")
```

Table 8: Standard Error from fit Model

6.758
0.416

(4g) p-values for testing H0: Beta=0 vs H1: Beta not equal to 0 for each k

```
## Original Model's Information
# summary(fit)$coefficients[,4]
# coeff_Betas[1]
# coeff_Betas[2]
## Calculate by Hand
test_stat <- (coeff_Betas[1]/se_Beta[1])
p_value_B_o <- 2*(1-pt(abs(test_stat), 48))
p_value_B_o
```

```
## [1] 0.01231882
```

```
test_stat <- (coeff_Betas[2]/se_Beta[2])
p_value_B_1 <- 2*(1-pt(abs(test_stat), 48))
p_value_B_1
```

```
## [1] 1.489919e-12
```

```
# (coeff_Betas[1] - test_stat*(1-0.05/2))*se_Beta[1]
```

(4h) R^2 for regression

R's R^2

```
summary(fit)$r.squared
```

```
## [1] 0.6510794
```

Calculation by Hand

```
# Sum of Squared Errors
ss_Err <- sum((res_Err)^2)

# Sum of Y - Ymean squared
ss_Ymean <- sum((Y - mean(Y))^2)
R_squared <- (1 - (ss_Err/ss_Ymean))
R_squared
```

```
## [1] 0.6510794
```

(5) P-Value for a Constast

```
setwd("/Users/benStraub/Desktop/STAT511")
Munich = read.csv("rent99.raw", sep="")
fit=lm(rentsqm~I(1/area)+factor(location),data=Munich)
# Sigma Hat Squared form original model
s2.hat=(summary(fit)$sigma)^2
# Coefficients of Model
Beta_4 <- fit$coefficients[4]
Beta_3 <- fit$coefficients[3]
Super_Beta <- Beta_4 - Beta_3
# ???Unsu
X=model.matrix(fit)
Xt = t(X)
invXXt = solve(Xt%*%X)
diag_general <- diag(invXXt)
diag_4 <- as.numeric(diag_general[4])
diag_3 <- as.numeric(diag_general[3])
# Make sure to subtract the covariance
test_Statistic <- as.numeric((Super_Beta)/sqrt(s2.hat*(diag_4 + diag_3 - 2*invXXt[4,3])))
pt(test_Statistic, 3078)
```

```
[1] 0.9999962
```

```
# super_Beta + sigma*qt(.975, 3078)
# super_Beta - sigma*qt(.975, 3078)
#kable(summary(fit)$coeff, digits=3)
```

I choose my H-0: Beta-4 - Beta-3 = 0. It should be normally distributed with the means of Beta-4 and Beta-3 subtracted and the Variances added together with the covariance subtracted off. I called it my Super_Beta in my code.

I got 4.483316 as my Test Statistic and did a p-test? Anything that is greater than 0.975 is going to be statistically significant. Our tests give us 0.9999962 so we can reject our Null and can conclude that B-4 and B-3 are not the same and have some worth to be included in our model.

(6b) Prediction of Stopping Distance

```
fit=lm(dist/speed~speed, data=cars)
coef=summary(fit)$coefficients[2,1]
err=summary(fit)$coefficients[2,2]
coef + c(-1,1)*err*qt(0.975, 48)
```

```
[1] 0.03642026 0.14164974
```

```
#  [1] 0.03642026 0.14164974
```

I got for the 95% CI for Beta-1 to be 0.03642026 and 0.14164974

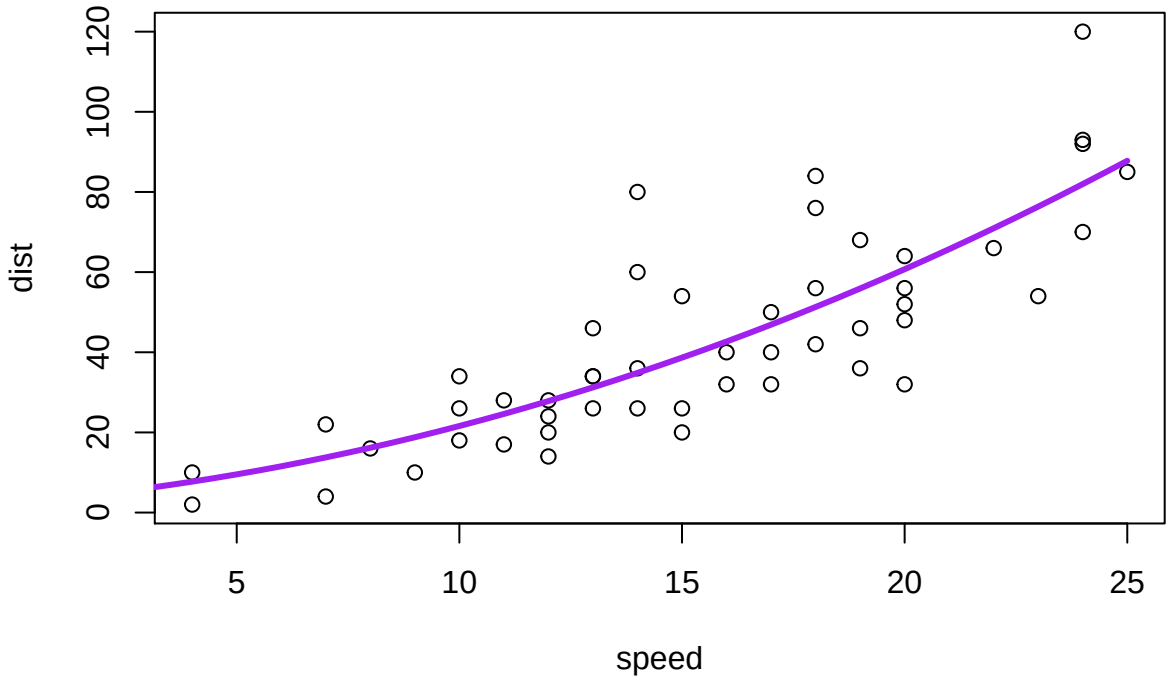
(6c)

I had a lot of difficulty with this problem and could not complete it. Below is the code I was trying to use to construct the CI. I ended up using a package called visreg just to mock something up for the CI, but that is it. Sorry :(

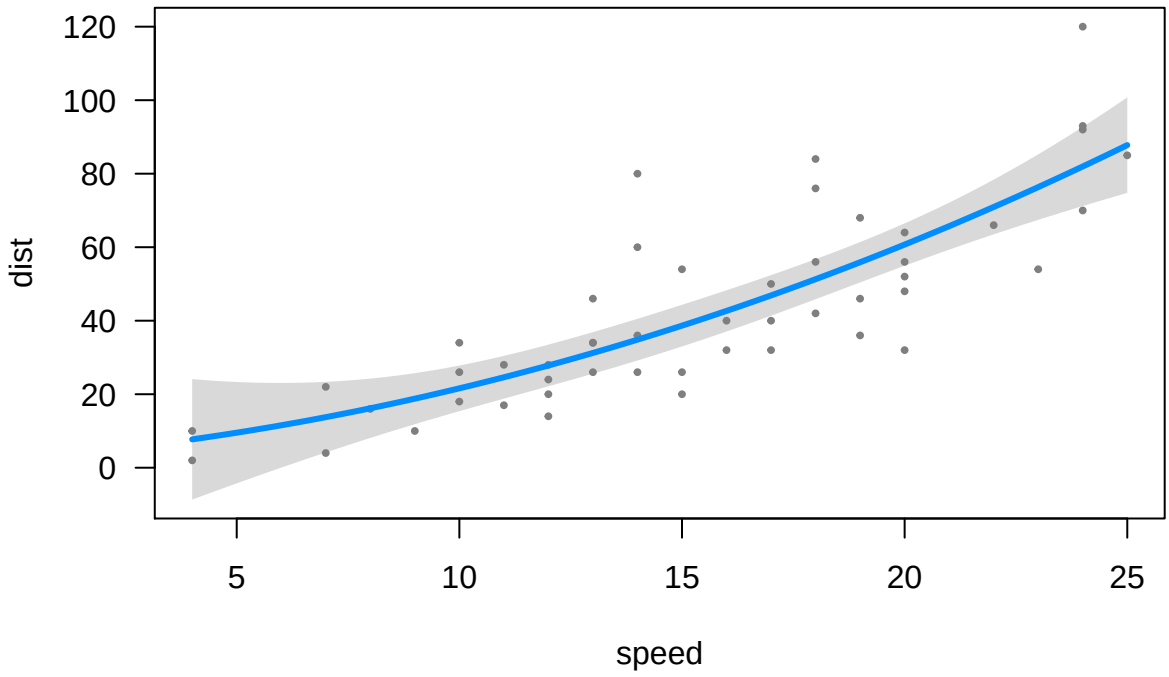
```
fit=lm(dist~speed+I(speed^2),data=cars)

attach(cars)
y<-dist
x_design<-cbind(rep(1,nrow(cars)),cars$speed,cars$speed^2)
x_0<-c(1,30,30^2)
x_0<-(cbind(rep(1,351),seq(0, 35, 0.1),seq(0, 35, 0.1)^2))

n<-nrow(x_design)
p<-ncol(x_design)
hatmat<-x_design%*%solve(t(x_design)%*%x_design)%*%t(x_design)
res<-(diag(1,nrow(x_design))-hatmat)%*%y
sigma_hat<-(t(res)%*%res)/(n-p)
beta_hat<-solve(t(x_design)%*%x_design)%*%t(x_design)%*%y
y_hat_0<-x_0%*%beta_hat

var_beta_hat<-as.numeric(sigma_hat)*(x_0)%*%solve(t(x_design)%*%x_design)%*%t(x_0)
se_betahat<-sqrt(diag(var_beta_hat))
```

```
#### Calculate Confidence Interval for the mean
high95<-y_hat_0+qt(0.975,n-p)*se_betahat
low95<-y_hat_0+qt(0.025,n-p)*se_betahat

plot(speed, dist)
xvals=data.frame(speed=seq(0, 25 ,by=0.5))
yvals=predict(fit,xvals)
## plot regression line
points(xvals$speed,yvals,type="l",col="purple",lwd=3)
# the confidence bands
require(visreg)
```

```
Loading required package: visreg
```




```
visreg(fit)
```

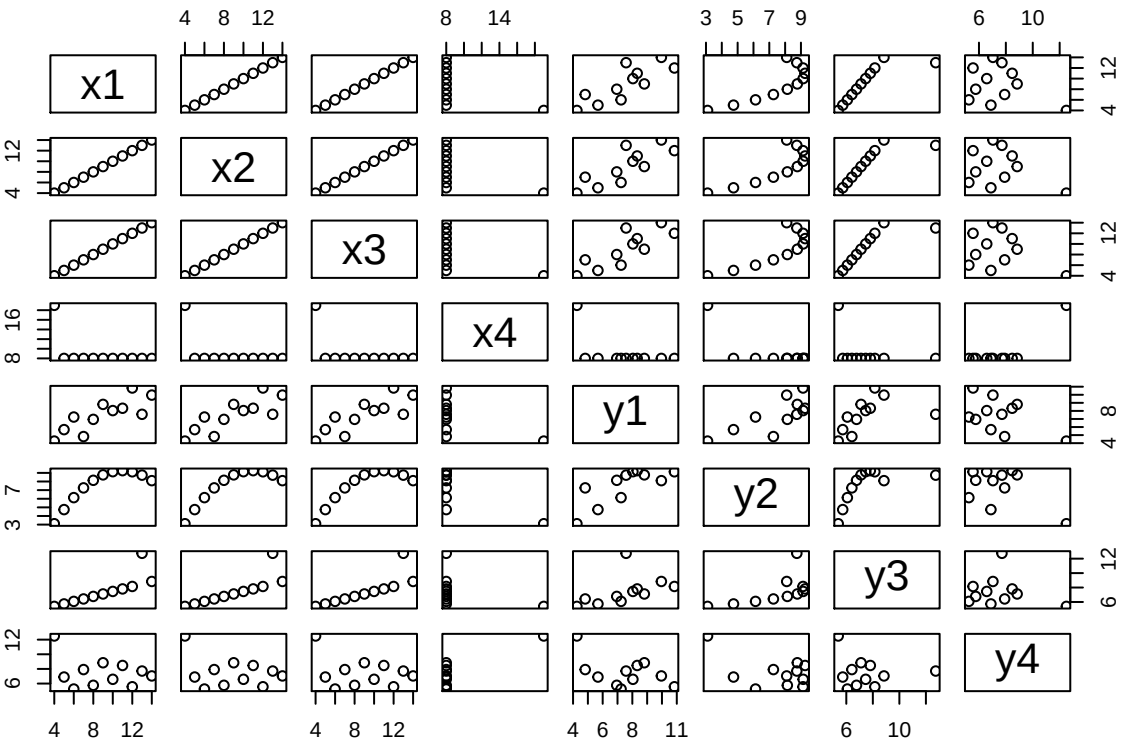



```
xnew <- seq(0,400)
int <- confint(fit)

#   lines(x, fitted[, "upr"], lty = "dotted")
```

No conclusion on why I trust the CI.

(7a)

Analysis of Anscombe



Report of the regression parameters for each pair of predictor and response variables

Table 9: x1&y1

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.0	1.125	2.667	0.026
x1	0.5	0.118	4.241	0.002

```
[1] 0.6665425
```

Table 10: x2&y2

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.001	1.125	2.667	0.026
x2	0.500	0.118	4.239	0.002

```
[1] 0.666242
```

Table 11: x3&y3

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.002	1.124	2.670	0.026
x3	0.500	0.118	4.239	0.002

```
[1] 0.666324
```

Table 12: x4&y4

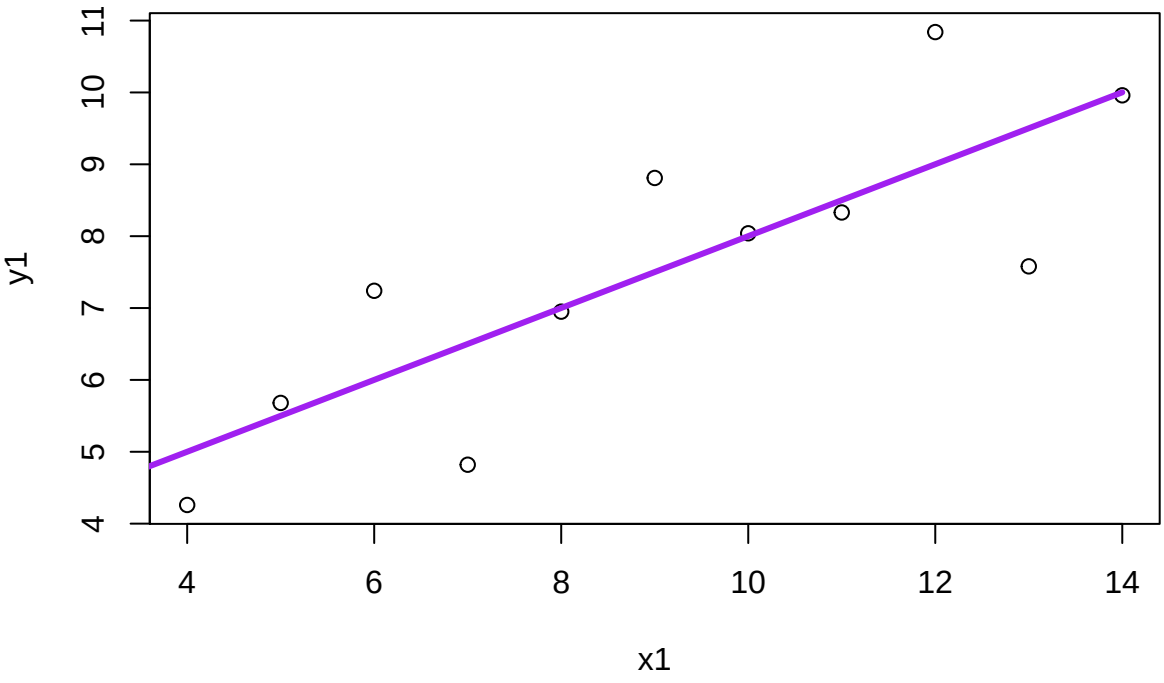
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.002	1.124	2.671	0.026
x4	0.500	0.118	4.243	0.002

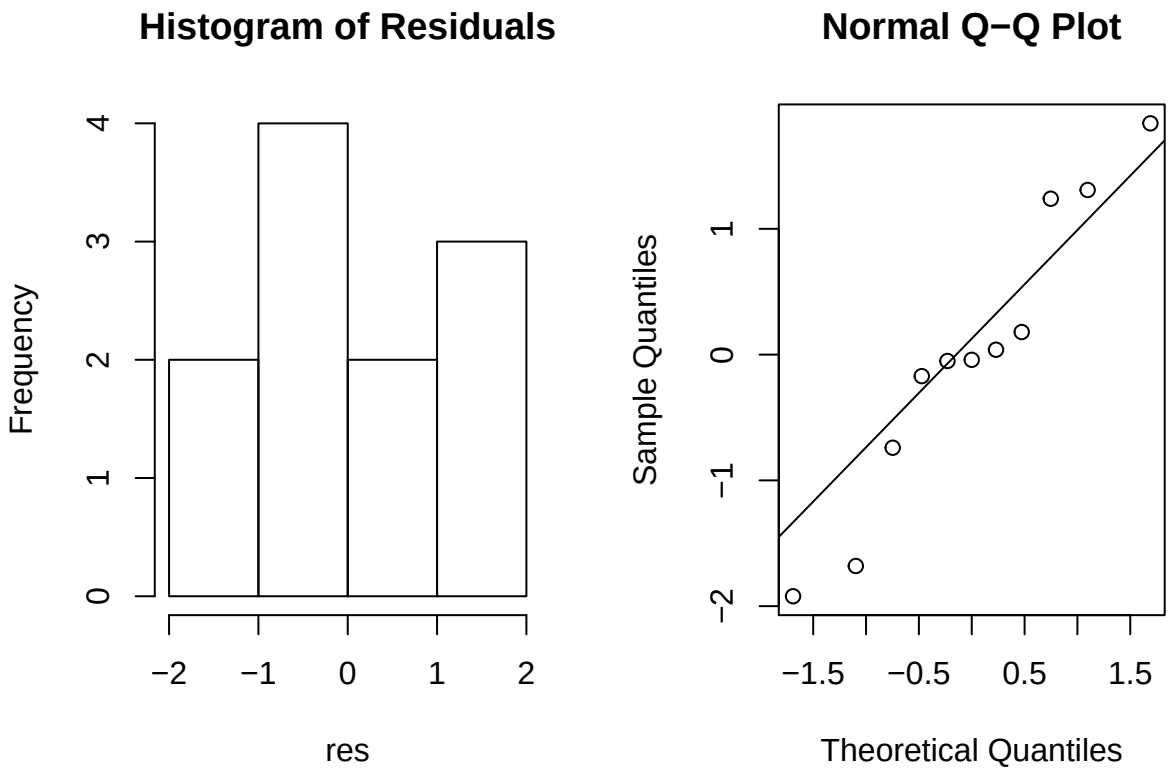
```
[1] 0.6667073
```

OBSERVATION: **Each model has the same Intercept and Estimate!! The numbers below each table is the corresponding R^2 to each model. The R^2 are almost identical, which tells us how close the data is to the fitted regression line, but some once you graph it does not make any sense going from a graph to the statistics.**

(7b)

Analysis of {x1, y1}

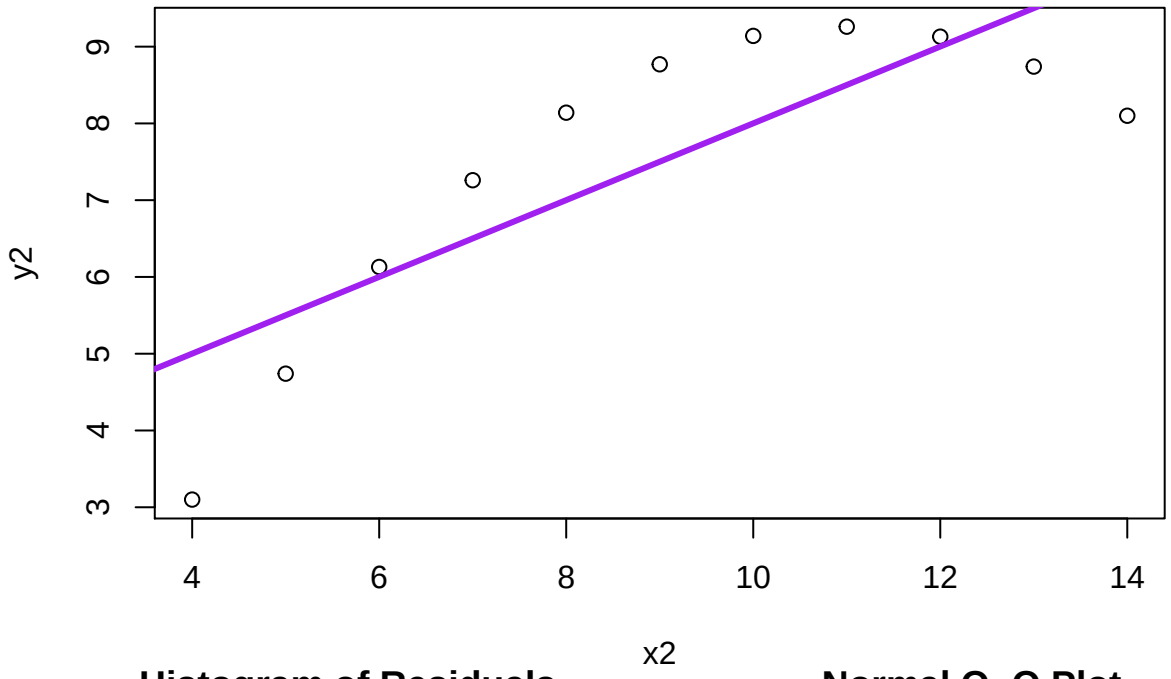


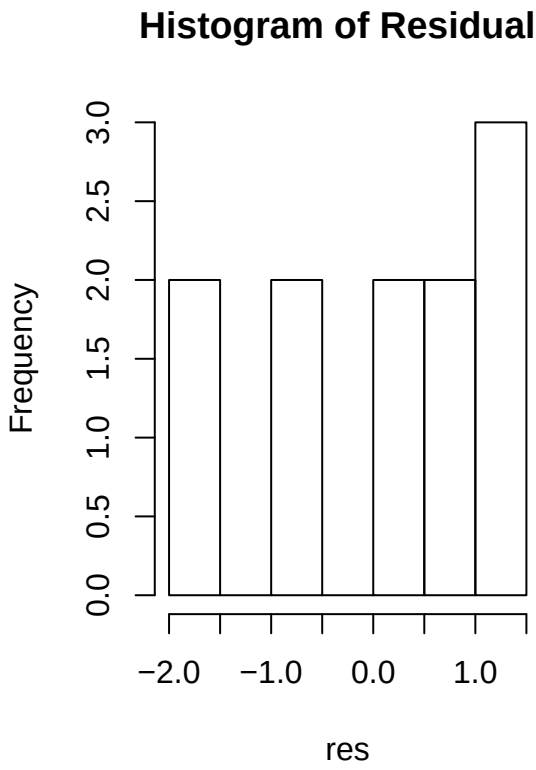


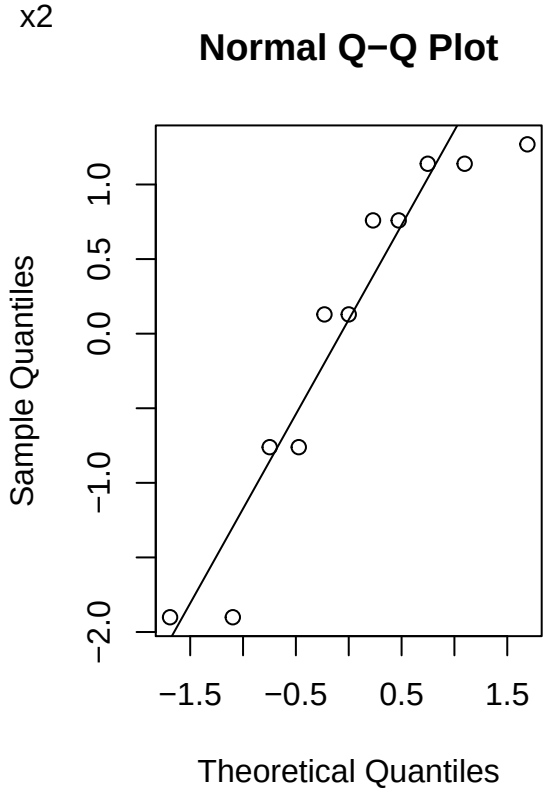
The relationship between x1 and y1 appears to be a tradtional linear relationship. However, looking at the Residuals of the Historgram they do not take on a normal distribution and the QQ-plot looks weak. It is difficult to make any generalizations due to the small sample size. However, one could make decent predictions with this model and data set.

Analysis of {x2,y2}

```
The following objects are masked from anscombe (pos = 3):

    x1, x2, x3, x4, y1, y2, y3, y4
```

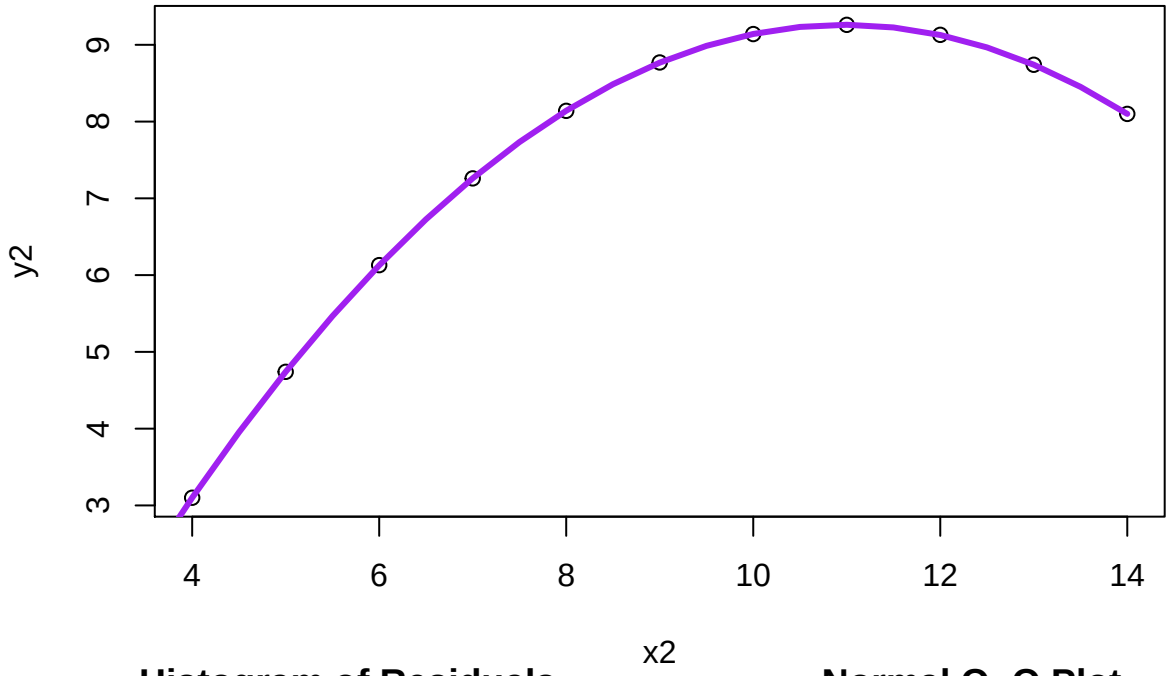

y2
3
4
5
6
7
8
9
4
6
8
10
12
14
x2

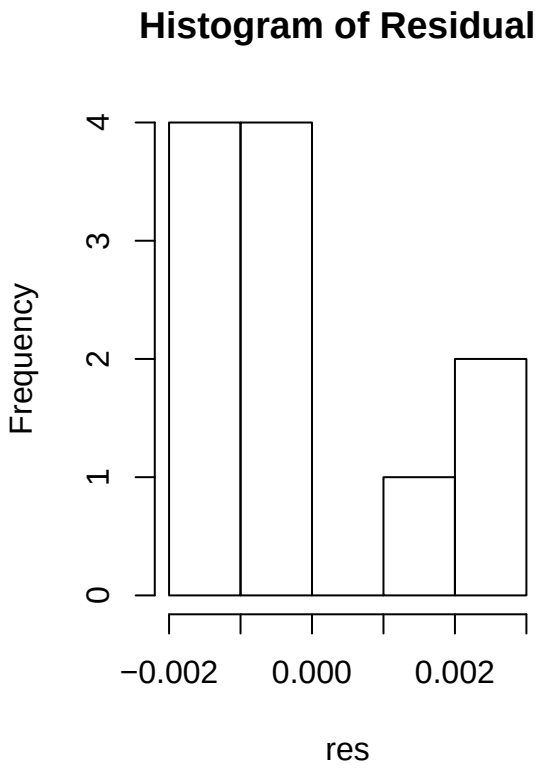

Histogram of Residuals
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
−2.0
−1.0
0.0
1.0
res

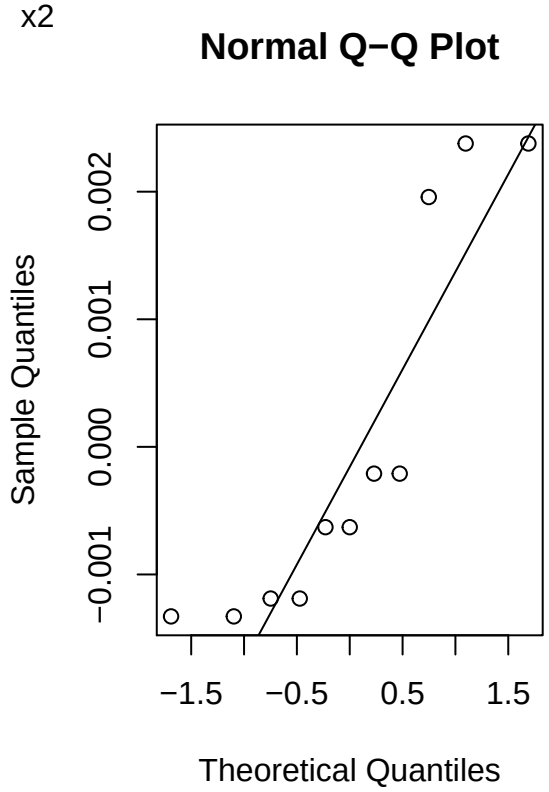

Normal Q−Q Plot
Sample Quantiles
−2.0
−1.0
0.0
0.5
1.0
−1.5
−0.5
0.5
1.5
Theoretical Quantiles

The residuals are not normal, but the qq-plot does seem to obey its theoretical quantiles. However, just by visual check of our regression line onto the plot of the data we can throw out our current linear model as making any sort of meaningful prediction would be inaccurate.

Transformation of {x2, y2} to a polynomial

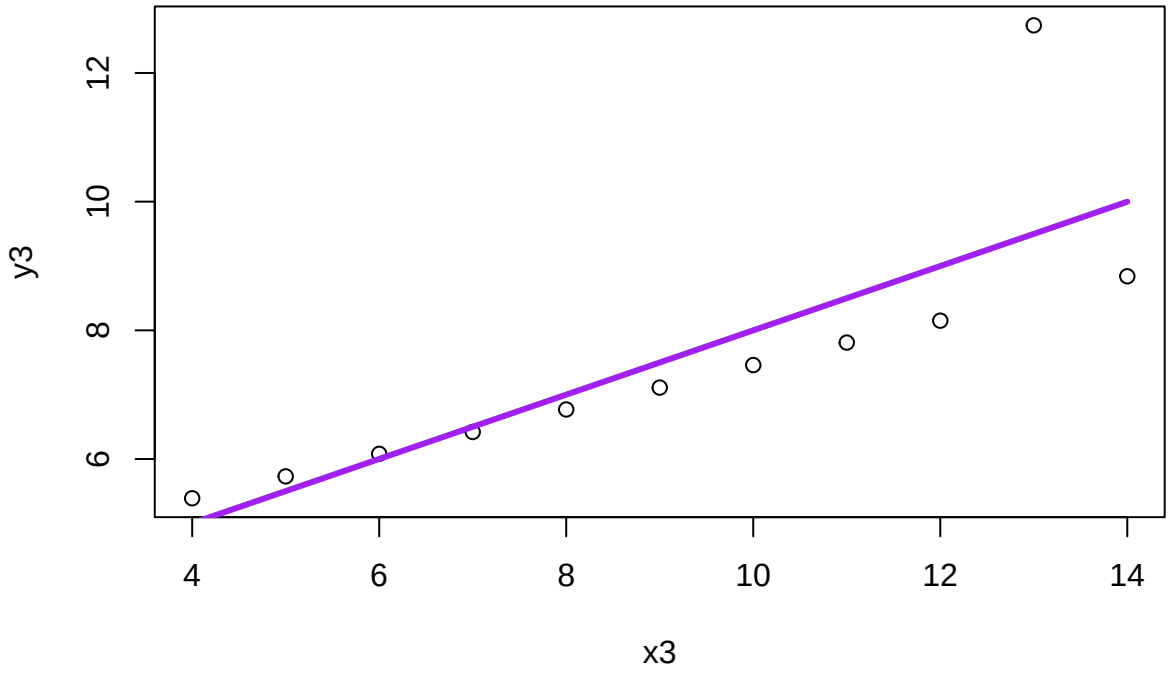




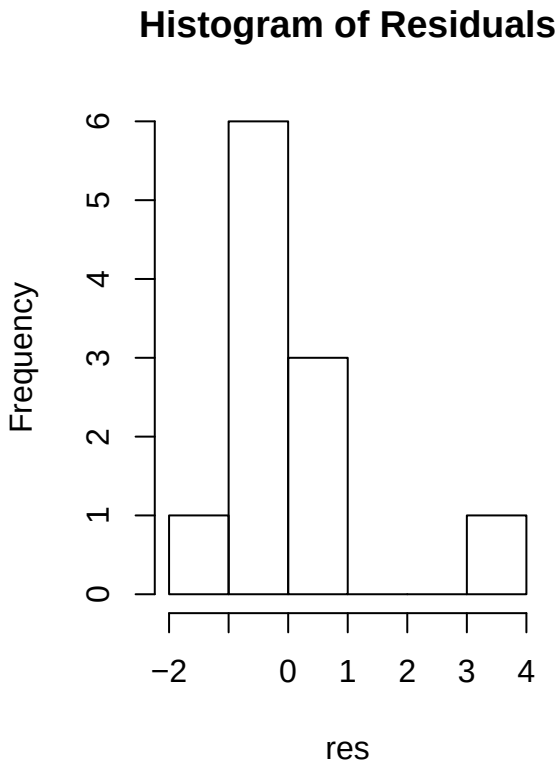


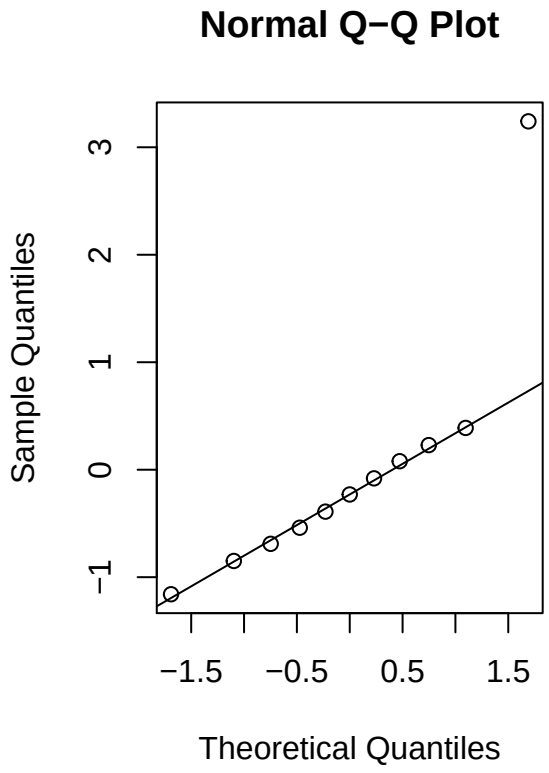
The polynomial fit has a much nicer regression line, which could lead to better prediction, but the residuals and qq-plot still indicate that the residuals are not normal. Also, the histogram demonstrates a bimodal distribution.

Analysis of {x3,y3}




```
[1] 1203.539
```

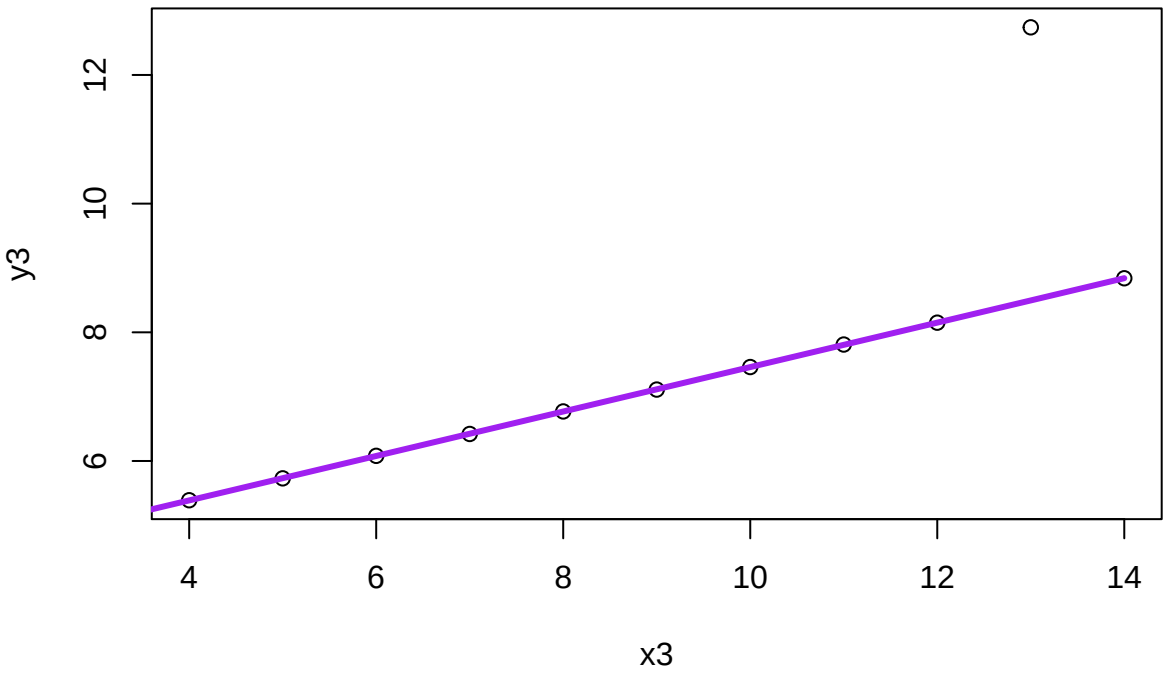


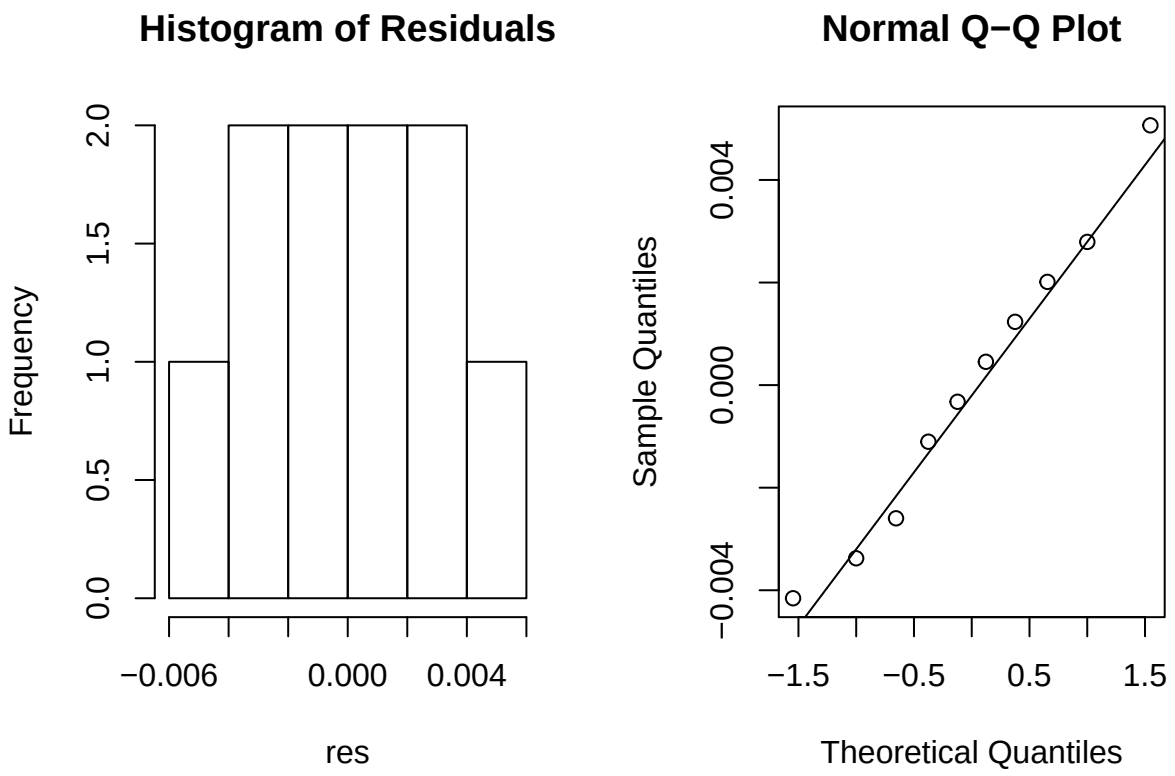


The set {x3, y3} has a linear relationship, but an outlier that wrecks havoc on our regression line!!! The Histogram of the Residuals looks like a normal distribution, but the outlier is present. The QQ-Plot obeys the theoretical quantiles completely. Yay! But the outlier!! I will transfrom the model to exclude the outlier.

Analysis of {x3,y3} without Outliers

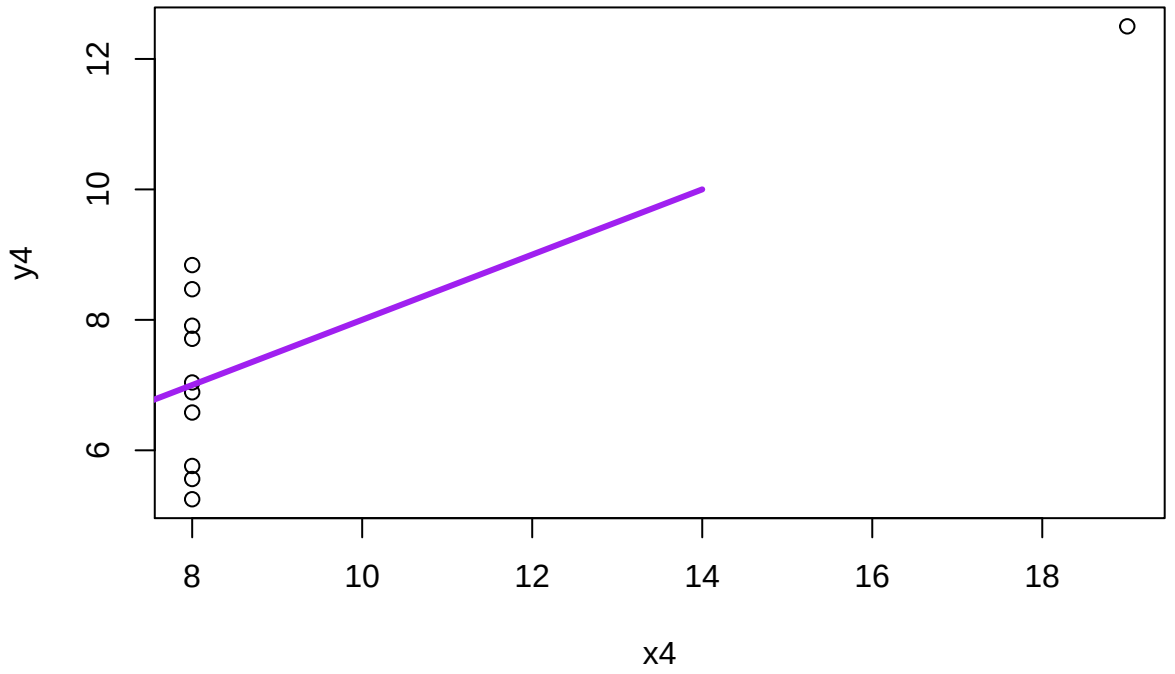
3
3

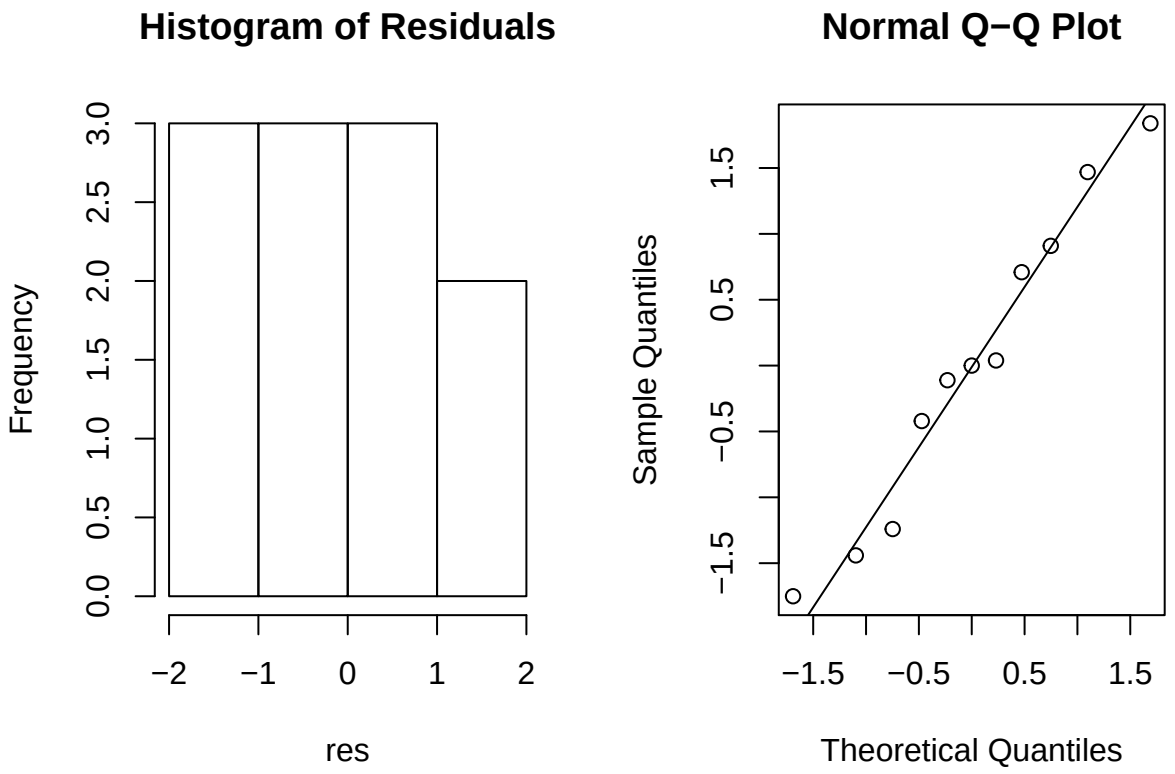




The new regression line gives us a nice fit for the data and the residulats of our histogram look much nicer as well, but our QQ-plot looks a little less robust than our previous model. However, our predictions will be slightly better with this current model.

Analysis of {x4,y4}

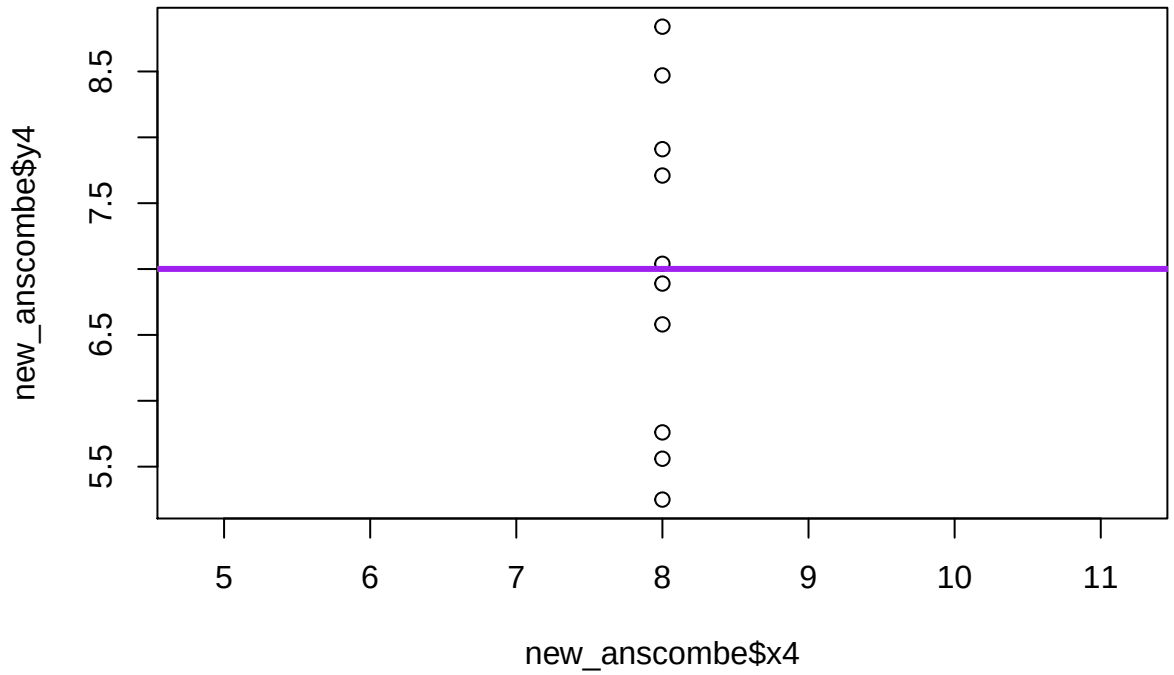


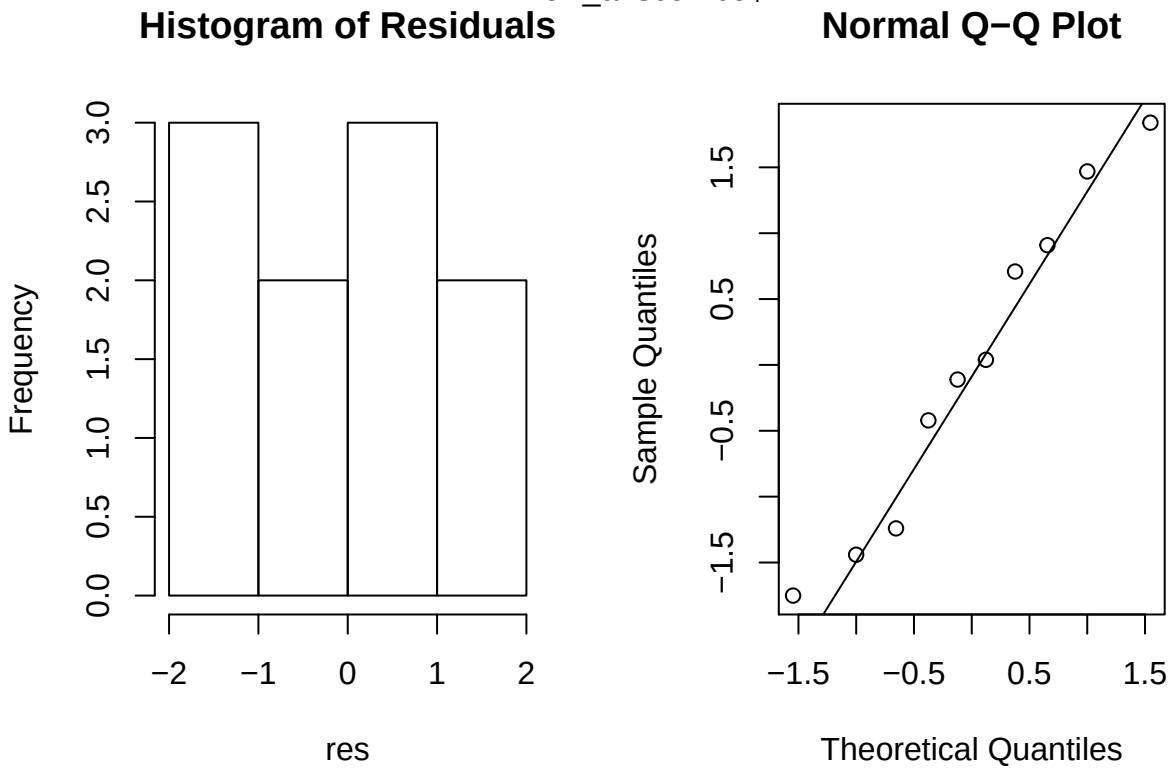


Difficult to make any meaningful analysis of this data set. It technically has an influential point, but all of the other data is centered around the point 8.

Analysis of {x4,y4} without influential point

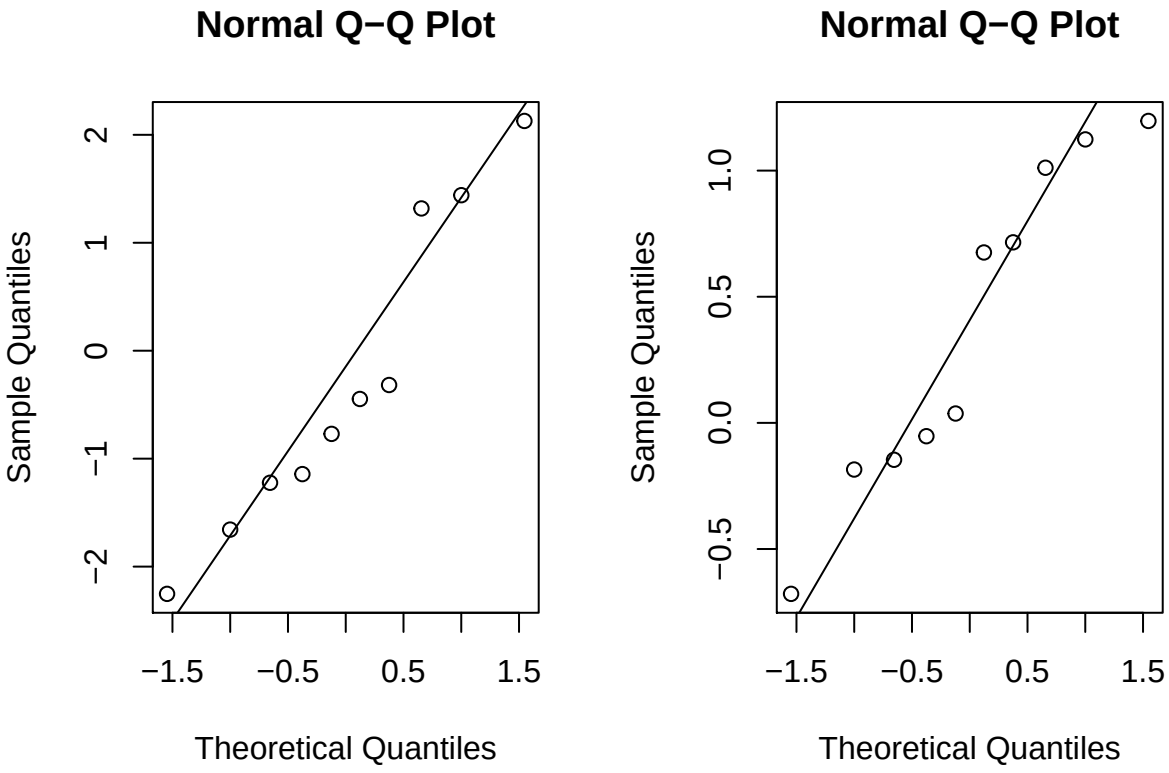
```
Warning in predict.lm(fit, xvals): prediction from a rank-deficient fit may
be misleading
```

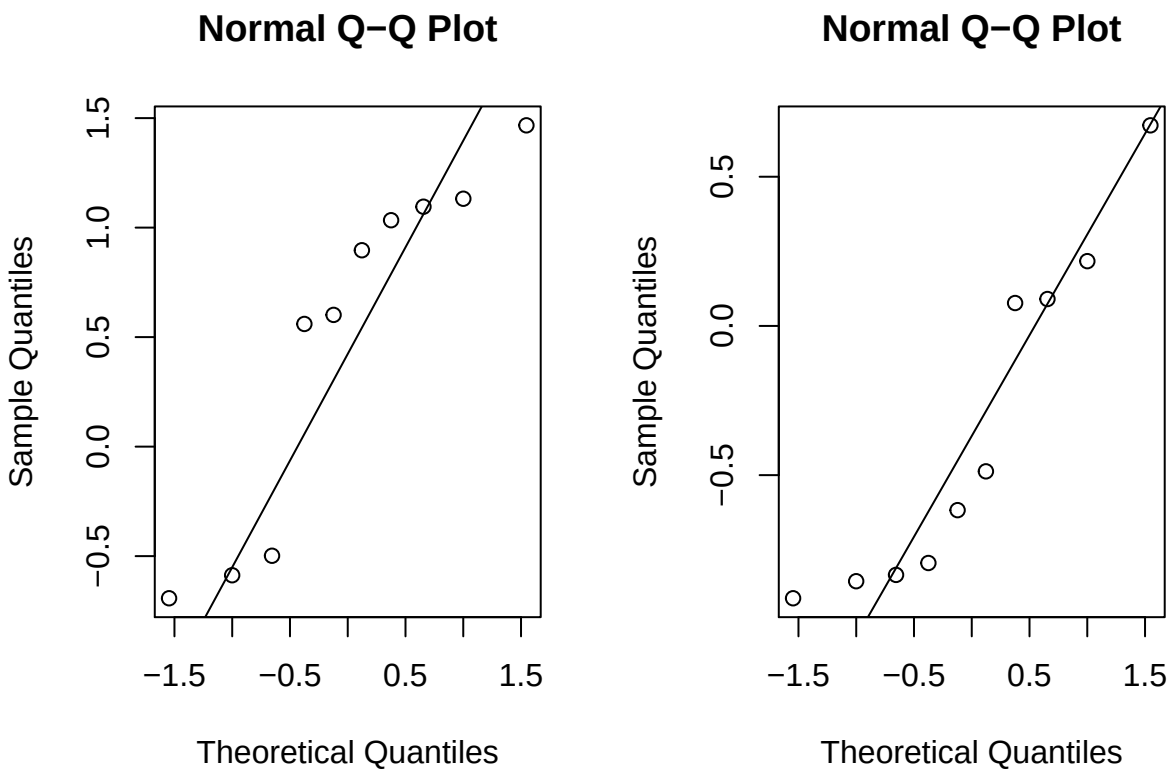

new_anscombe$y4
8.5
7.5
6.5
5.5
5
6
7
8
9
10
11
new_anscombe$x4


Histogram of Residuals
Frequency
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2
res
Normal Q−Q Plot
Sample Quantiles
1.5
0.5
−0.5
−1.5
−1.5
−0.5
0.5
1.5
Theoretical Quantiles

We have taken out the influential point to try and get a better understanding of the relationship between the data. However, our new regression line is just a horizontal straight line, which gives no power of prediction. I found the histogram of the residuals to not be normally distributed, but the the qq-plot looks like a good fit for the data. What?

QQ-Plots for Small Sample Sizes





I generated 4 normal random distributions with sample size equal to 10 to take a closer look at the behavior of small sample sizes. A few of the plots showed some wild behavior while several other showed a tight grip on the theoretical quanitles. It leads me to believe that small sample sizes and liner models have a much difficult relationship.

Conclusion **The data sets are small and differ in some slight manner, excpet for the 4th one, which has all but one data point at 8. When we generate the summary of each model we receive almost identical statistics for the regressionl line, but when we graph it they look so different!!**

(7c)

```
fit=lm(y1~x1, data=anscombe)
predict_1 <- predict(fit, newdata=data.frame(x1=13))
predict_1
```

```
       1
9.501273
```

```
fit=lm(y2~poly(x2,2), data=anscombe)
predict_2 <- predict(fit, newdata=data.frame(x2=13))
predict_2
```

```
      1
8.740629
```

Not completed