**Lab 2: Synthetic Data and Evaluating Patterns in R**

By: Yvette Lindler, Leon Davis, Jennifer McLean, and Jim Graham

10th of February, 2020

## Methods

Linear regression will be evaluated in the statistics package R by creating synthetic linear trends and injecting noise into the resulting data. Linear models will be fitted to the data and then the residuals will be evaluated using histograms. Four different sets of data, or runs, will be created with different parameters for the equations that create the trends and noise (Table 1).

Table 1. Equations and parameters for the noise (error, ϵ) that will be injected into the data. The Mean and Standard deviation define the parameters for the normal distribution used to generate the noise.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Equation** | **Mean (µ)** | **Standard Deviation (σ)** |
| **1** |  | 0 | 10 |
| **2** |  | 0 | -1 |
| **3** |  | 10 | 1 |
| **4** |  | 0 | 0 |

## Results

Results show the range of the random data increases with increasing standard deviation (Figure 1) and the relationship between the Y Values and X Values for each model (Figure 2).

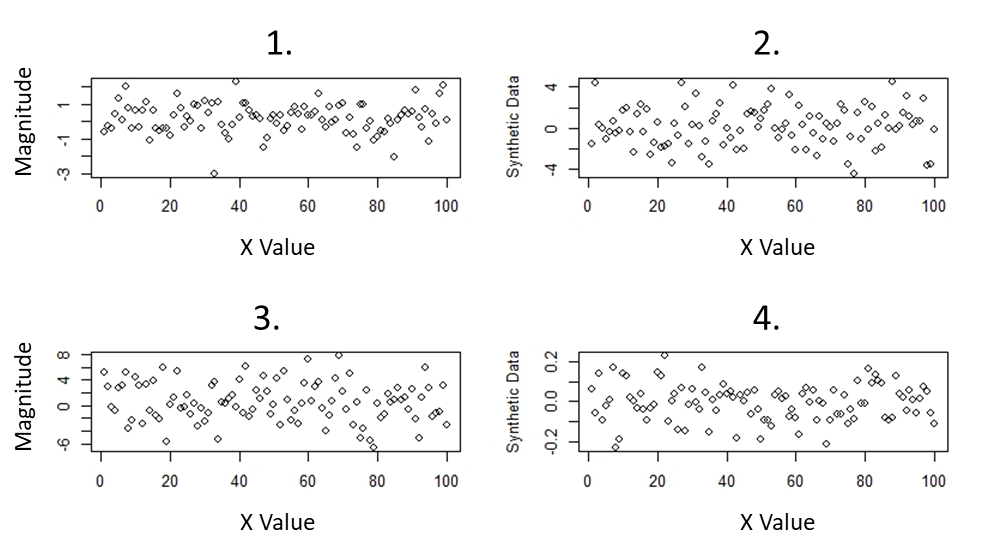
****

Figure 1. Plots of the magnitude of the noise that was added to the data for each X Value.

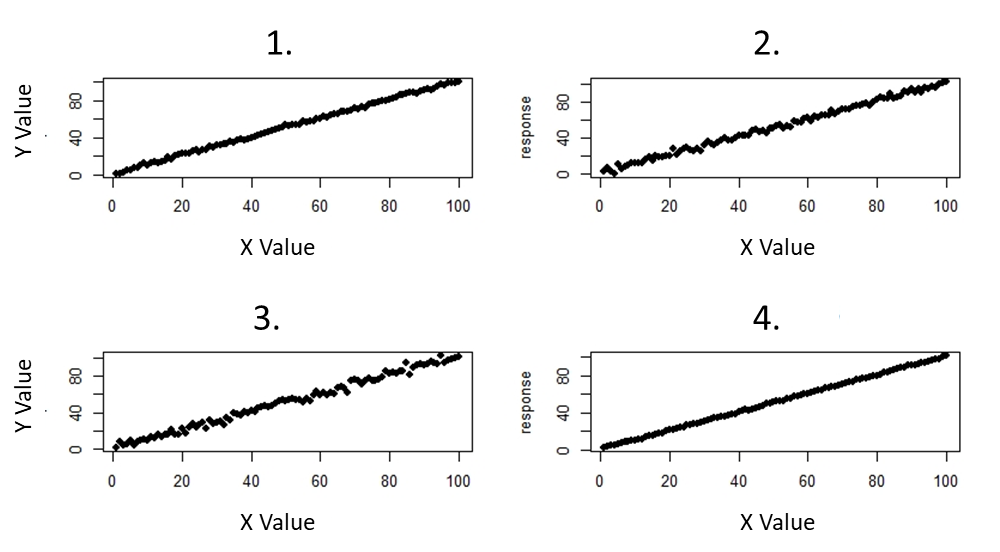


Figure 2. Trends with added noise for each run where the y value is the response and the x value is the independent variable.

Performance measures showed that the ability for R to fit a linear trend to the data was correlated with the amount of noise that was injected into the data (Table 2).

Table 2. Performance results for each model run.

|  |  |
| --- | --- |
| **Model** | **R2** |
| 1 | 99.87% |
| 2 | 99.47% |
| 3 | 99.09% |
| 4 | 99.97% |

Histograms of the residuals (Figure 3) from linear models, incorporating the altered standard deviations, reveal increased residual distance for models with higher standard deviations, and decreased residual distance for models with lower standard deviations.

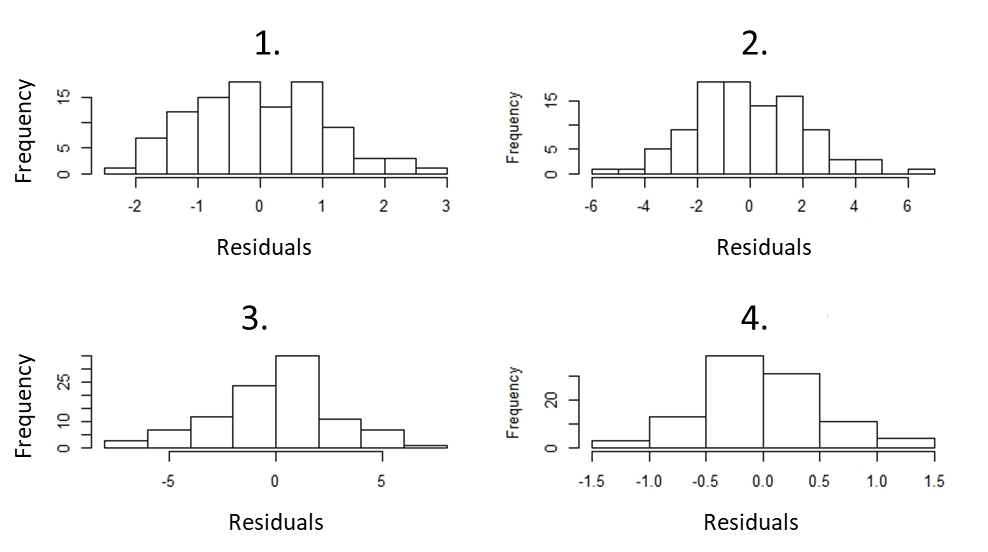


Figure 3. Histograms of the residuals for each model with varying degrees of standard deviation.

Changing standard deviations also changed the structure of each model (figure 4). No noticeable pattern was found in the models as standard deviations changed. Changing the standard deviation resulted in models with poor fit to the data. Each model determined the intercept and explanatory variable to be significant predictors of the response (p<.05).

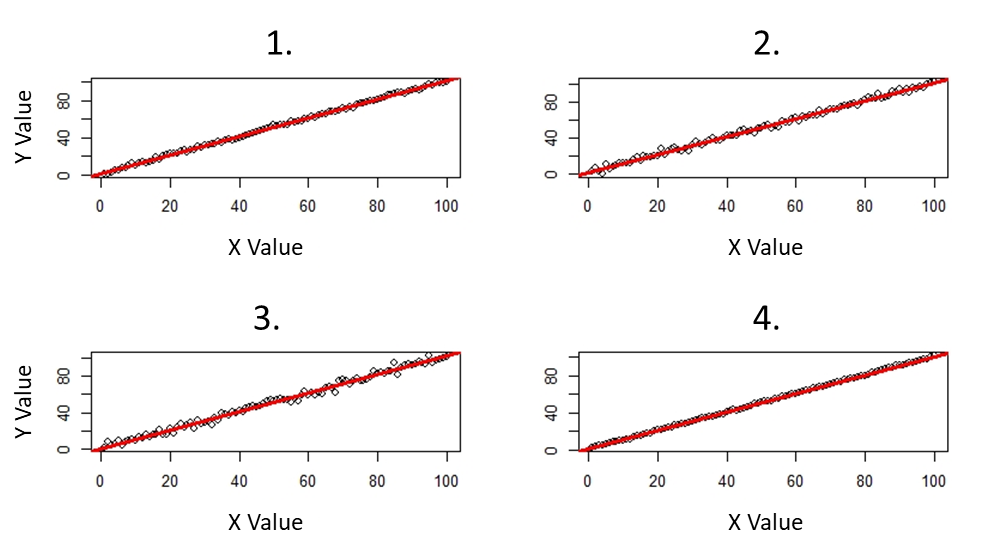


Figure 4. Models fit to their respective datasets for each run.

## Creating Two Dimensional Data

The model with two independent variables showed the same behaviors as the ones with one independent variable except 3D plots were used to visualize the results (Figure 5, Figure 6).

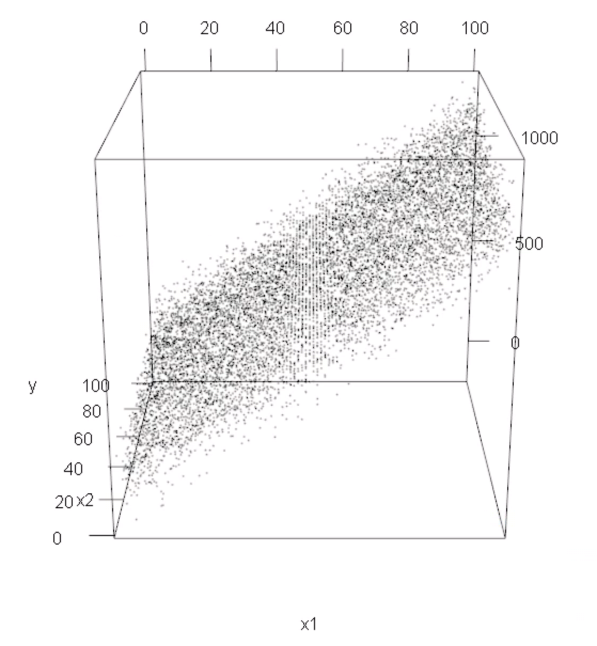
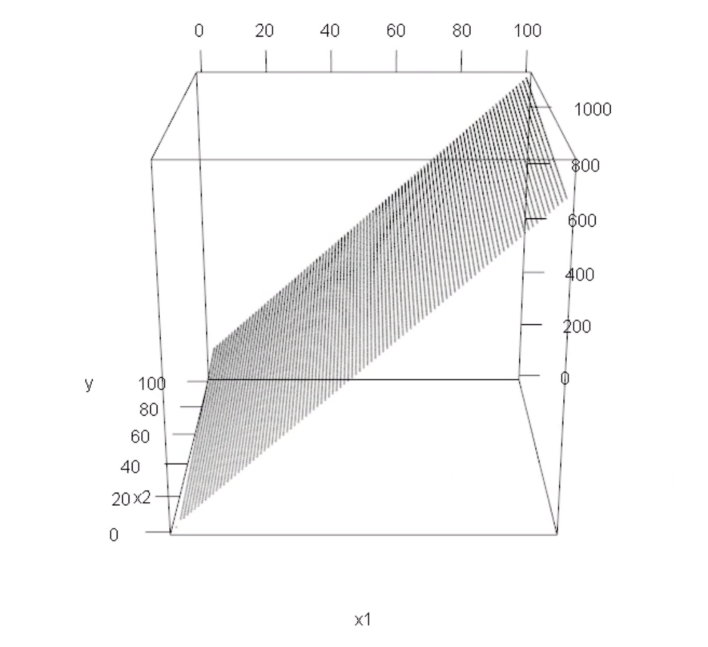


Figure 5: A three dimensional plot of a linear trend between two independent variables.

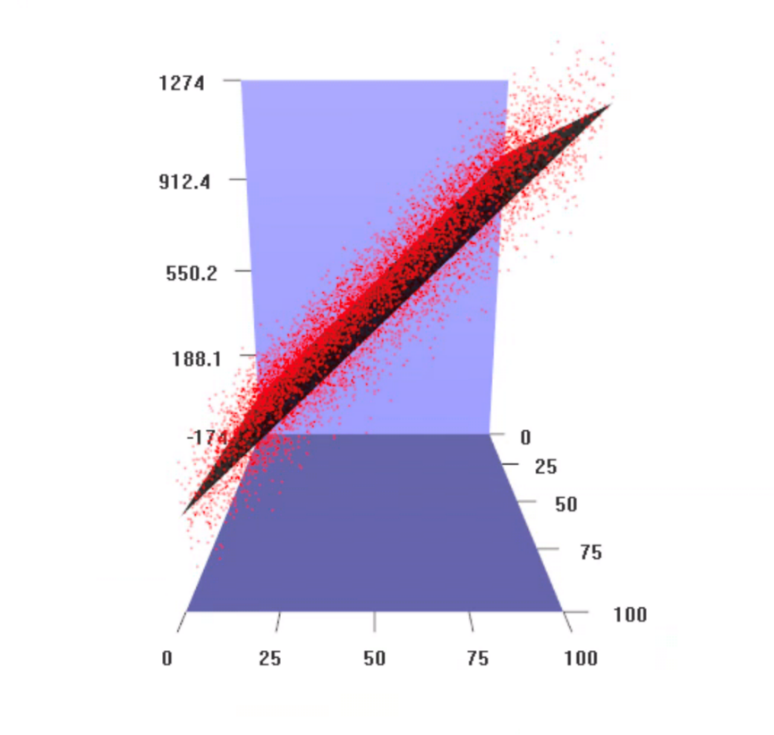


Figure 6: A three dimensional plot based on a two-dimensional matrix created by model parameters or coefficients.

## Questions

**QUESTION 1.** What effect does the mean and stdev have on the data?

**QUESTION 2.** What effect does setting B1 to 10 have? What effect does setting B1 to -1 have?

**QUESTION 3.** What effect does changing B0 have?

**QUESTION 4.** What effect does increasing and decreasing the value of the standard deviation in the random function have?

**QUESTION 5.** How well does R find the original coefficients of your polynomials?

**QUESTION 6.** How good a job did the prediction do at removing the trend in your data?

**QUESTION 7.** What effect does increasing and decreasing the values of B3 and B4?

**QUESTION 8.** What is the value of Moran's I?

**Acknowledgements**

Thank you to Professor Jim Graham for guiding us through our first lab activity using R.

**References**

RStudio Team (2019). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA

URL <http://www.rstudio.com/>

**Appendix A – R Code**

############## Changing the standard deviation

##### response

n=seq(1:100)

noise=rnorm(n,mean = 0, sd=1)

#### explanatory

x=seq(1:100)

x1=rnorm(x, mean=0, sd=2)

x2=rnorm(x, mean=0, sd=3)

x3=rnorm(x, mean=0, sd=.5)

par(mfrow=c(2,2))

plot(x, main = 'mean=0, sd=1', ylab='Synthetic Data')

plot(x1,main = 'mean=0, sd=2', ylab='Synthetic Data')

plot(x2,main = 'mean=0, sd=3', ylab='Synthetic Data')

plot(x3,main = 'mean=0, sd=.5', ylab='Synthetic Data')

####### generate some noise

n1=rnorm(x, mean=0, sd=2)

n2=rnorm(x, mean=0, sd=3)

n3=rnorm(x, mean=0, sd=.5)

####### data fit to response

b0=1

b1=1

y=b0+(b1\*x)+noise

y1=b0+(b1\*x)+ n1

y2=b0+(b1\*x)+ n2

y3=b0+(b1\*x)+ n3

plot(x,y, xlab='explanatory', ylab = 'response', main='mean=0, sd=1', pch=16)

plot(x,y1, xlab='explanatory', ylab = 'response', main='mean=0, sd=2',pch=16)

plot(x,y2, xlab='explanatory', ylab = 'response', main='mean=0, sd=3',pch=16)

plot(x,y3, xlab='explanatory', ylab = 'response', main='mean=0, sd=.5',pch=16)

######## Histograms

m1=lm(y~x)

m2=lm(y1~x)

m3=lm(y2~x)

m4=lm(y3~x)

par(mfrow=c(1,1))

hist(resid(m1),main='mean=0, sd=1',xlab = 'Residuals')

hist(resid(m2),main='mean=0, sd=2',xlab = 'Residuals')

hist(resid(m3),main='mean=0, sd=3',xlab = 'Residuals')

hist(resid(m4),main='mean=0, sd=.5',xlab = 'Residuals')

##### model strucutre

print(m1)

print(m2)

print(m3)

print(m4)

summary(m1)

summary(m2)

summary(m3)

summary(m4)

plot(y~x, xlab='explanatory', ylab = 'response', main='Y=1.10+1\*x;R-squared=99.87%')

abline(m1, col='red',lwd=3)

plot(y1~x, xlab='explanatory', ylab = 'response', main='Y=.96+1\*x;R-squared=99.47%')

abline(m2, col='red',lwd=3)

plot(y2~x, xlab='explanatory', ylab = 'response', main='Y=.81+1\*x;R-squared=99.09%')

abline(m3, col='red',lwd=3)

plot(y3~x, xlab='explanatory', ylab = 'response', main='Y=.95+1\*x;R-squared=99.97%')

abline(m4, col='red',lwd=3)

######## autocorrelation

RandomMean=0

RandomStdDev=1

RandomValues=rnorm(NumEntries, mean = RandomMean, sd = RandomStdDev);

Frequency=0.3

Magnitude=45

Y=B0+B1\*X + RandomValues + sin(X\*Frequency)\*Magnitude

lm1=lm(Y~X+RandomValues)

summary(lm1)

lm2=update(lm1,.~.-RandomValues)

summary(lm2)

lm.p=predict.lm(lm2)

lm.p=order(lm.p)

par(mfrow=c(1,1))

plot(Y,pch=16, main = 'Y=8.23+.88\*x; R-squared=38.79') ########## linear model

abline(lm2,col='orange')

TheMean=mean(Y) # find the overall mean of the array

# find the sum of squares

SumOfSquares=0

for (i in 1:NumEntries ) {

SumOfSquares=SumOfSquares+(Y[i]-TheMean)^2

}

print(SumOfSquares)

# find the sum of the weighted differences

TheWeights=0

TheSum=0

for (i in 1:NumEntries ) {

for (j in 1:NumEntries )

{

if (i!=j) # don't compute differences between the same values

{

TheWeight=1/((i-j)^2) # compute a weight based on position in the array (i-j)

TheWeights=TheWeights+TheWeight # add the weights to an overall sum of weights

# the key equation where were add the weighted differences from the mean and sum it

TheSum=TheSum+TheWeight\*(Y[i]-TheMean)\*(Y[j]-TheMean)

}

}

}

# scale the sum to be from 0 to 1

MoransI=(NumEntries/TheWeights)\*(TheSum/SumOfSquares)

print(MoransI)

####### Beta Changing

B0=1

B1=2

B2=5

B3=-20

Y=B0+B1\*x+B2\*x^2+B3\*x^3 + noise

plot(Y)