Statistics Group Project Using R

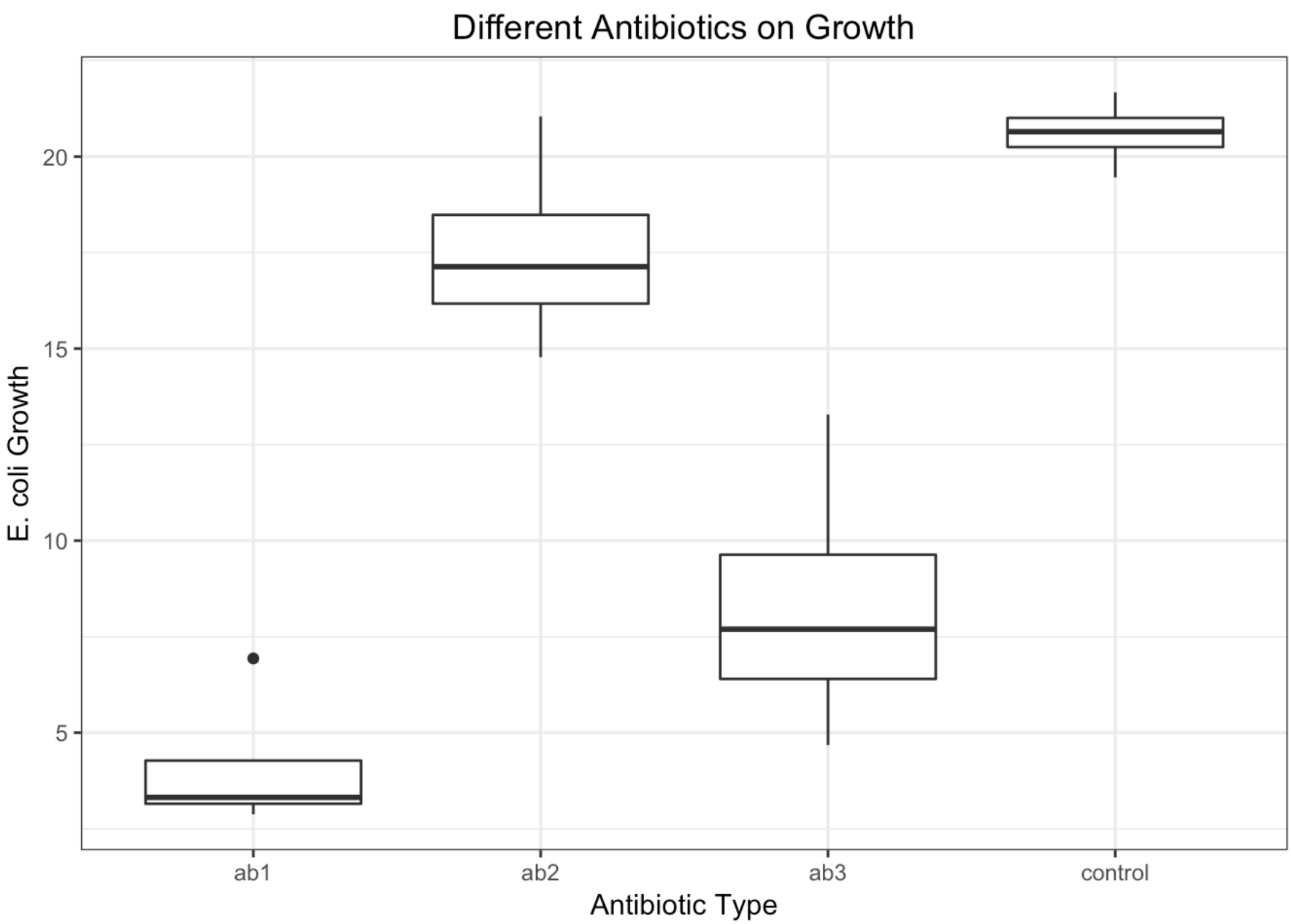
Libby Fortin (efortin@nd.edu (mailto:efortin@nd.edu)), Keith O'Connor (koconn23@nd.edu (mailto:koconn23@nd.edu)), Chelsea Weibel (cweibel@nd.edu (mailto:cweibel@nd.edu))

11/27/2017

Question 1: Evaluating the effect of three different new antibiotics on growth of E. coli in lab cultures

Let's visualize the data using ggplot. Since the data is categorical we use a boxplot:

```
library(ggplot2)

antibiotics <- read.csv("antibiotics.csv")

plot1 <- ggplot(antibiotics, aes(x = trt, y = growth)) +
      geom_boxplot() +
      theme_bw() +
      xlab("Antibiotic Type") + ylab("E. coli Growth") +
      theme(plot.title = element_text(hjust = 0.5)) + # Centers the title
      ggtitle("Different Antibiotics on Growth")

plot1
```



Next, we need to look at antibiotic treatments on the growth of E. coli using an ANOVA-design linear model and likelihood ratio test.

To do this we first need to assign a value to the effect of each antibiotic treatment

- The easiest way to do this is with a matrix that assigns zeros to the control treatment, 1s to one of the antibiotic treatments, and zeros to the remaining antibiotic treatments
- The matrix is compiled from three vectors, ab1 has antibiotic 1 having an effect, ab2 has antibiotic 2 having an effect, and ab3 has antibiotic 3 having an effect

```
ab1 <- c(0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0)
ab2 <- c(0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0)
ab3 <- c(0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1)
xvalues <- matrix(c(ab1,ab2,ab3),nrow=length(ab1),ncol=3)
xvalues
```

```
##       [,1] [,2] [,3]
##  [1,]    0    0    0
##  [2,]    0    0    0
##  [3,]    0    0    0
##  [4,]    0    0    0
##  [5,]    1    0    0
##  [6,]    1    0    0
##  [7,]    1    0    0
##  [8,]    1    0    0
##  [9,]    0    1    0
## [10,]    0    1    0
## [11,]    0    1    0
## [12,]    0    1    0
## [13,]    0    0    1
## [14,]    0    0    1
## [15,]    0    0    1
## [16,]    0    0    1
```

It's easiest to keep our results of the negative log likelihood models and the likelihood ratio test organized if we create a matrix that holds all the results

- Below we create a matrix where column 1 holds the results of the null model, column 2 holds the results of the linear model, and column 3 holds the results of the likelihood ratio test

```
results <- matrix(0,1,3)
colnames(results) <- c("null", "linear", "chisq")
```

Now we are ready to define our negative log likelihood functions, one for the null model and one for the linear model

- The null model hypothesizes that there is no relationship between antibiotics and growth
- The linear model hypothesizes that there is a linear relationship between antibiotics and growth

```
#Null model function
nllnull <- function(p,y){
  B0=p[1]
  sig=exp(p[2])
  expected=B0
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}

#Linear Model function
nlllinear <- function(p,x,y){
  B0=p[1]
  B1=p[2]
  B2=p[3]
  B3=p[4]
  sig=exp(p[5])
  expected=B0+B1*x[,1]+B2*x[,2]+B3*x[,3]
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}
```

We are ready to run the null and linear negative log likelihood models for the three different antibiotic treatments and store them in our results table

For the initial guesses in the optim function I found the means of each antibiotic treatment and the control to make them closer to the actual

```
# I found the mean of each antibiotic and the controls to make the intial guesses clo
ser
mean(antibiotics$growth[antibiotics$trt=='control'])
```

```
## [1] 20.60598
```

```
mean(antibiotics$growth[antibiotics$trt=='ab1'])
```

```
## [1] 4.109743
```

```
mean(antibiotics$growth[antibiotics$trt=='ab2'])
```

```
## [1] 17.52061
```

```
mean(antibiotics$growth[antibiotics$trt=='ab3'])
```

```
## [1] 8.336512
```

Now we run the models

```
# Antibiotic null model
initialGuess <- c(20,1)
fit <- optim(par=initialGuess,fn=nllnull,y=antibiotics$growth)
results[1,1] <- fit$value

# Antibiotic linear model results
initialGuess <- c(20,4,17,8,1)
fit <- optim(par=initialGuess,fn=nlllinear,x=xvalues,y=antibiotics$growth)
results[1,2] <- fit$value
```

Finally we calculate the likelihood ratio test and store the results in our results table

```
#likelihood ratio test results
A <- 2*(results[1,1]-results[1,2])
results[1,3] <- pchisq(q=A, df=1, lower.tail=FALSE)
results
```

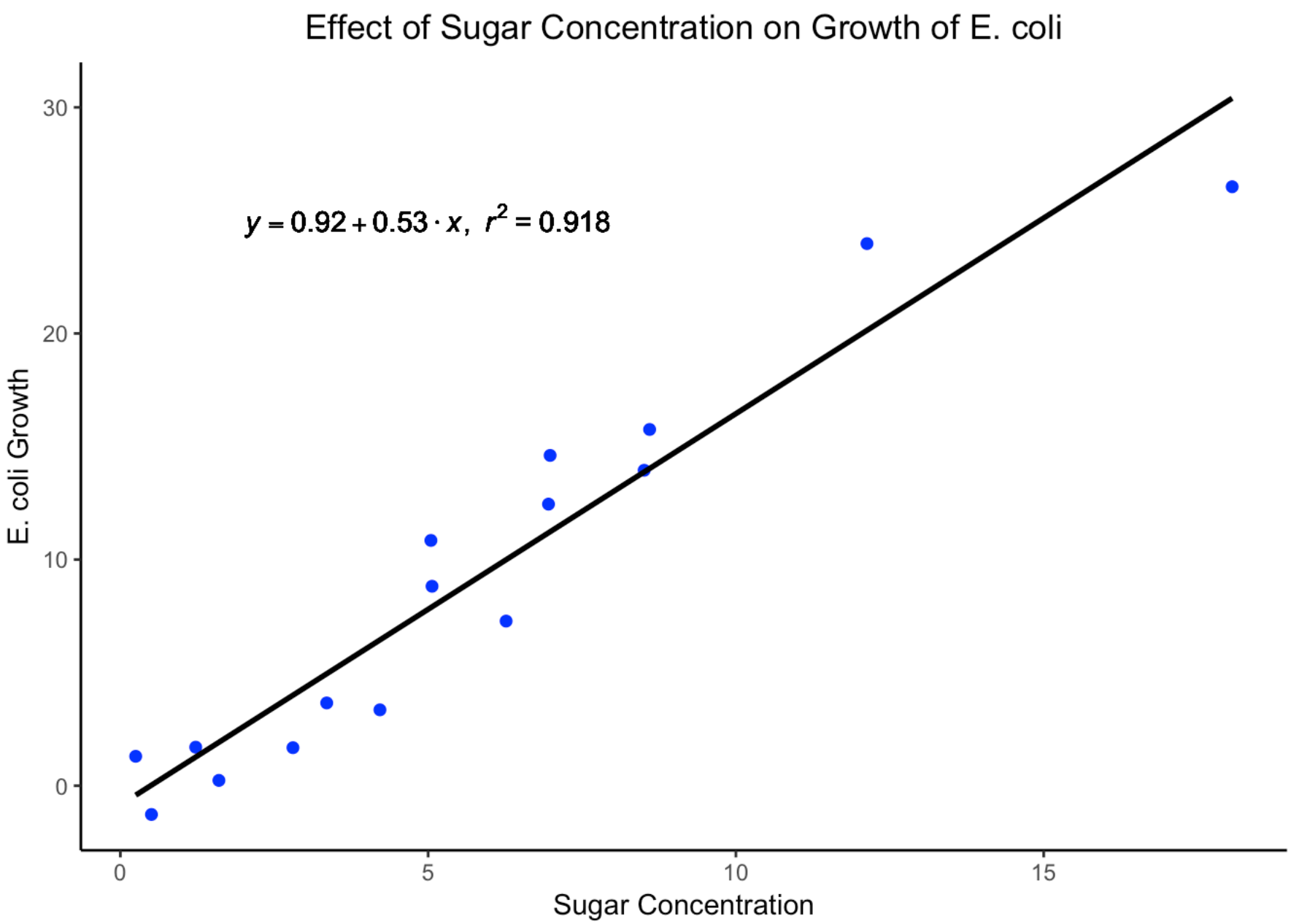
```
##          null   linear        chisq
## [1,] 53.87828 46.24091 9.295187e-05
```

Antibiotic treatment has a significant effect on growth because the p-value is less than 0.05 so we can reject the null model.

Question 2: Evaluating the effect of sugar concentration on growth of E. coli in lab cultures

Let's visualize the data using ggplot. Since the data is continuous we use a linear plot. We can also add the linear equation and R^2 value to the graph to tell us more about the data.

```
#### Linear plot of sugar effects on growth with line equation and R2 ####

sugardata <- read.csv(file = "sugar.csv", header=TRUE) # Importing the data

# Graphing the data
plot2 <- ggplot(data = sugardata, aes(x = sugar, y = growth))+
  geom_point(color = "blue1", shape = 16, size = 2) +
  theme_classic() +
  xlab("Sugar Concentration") + ylab("E. coli Growth") +
  ggtitle("Effect of Sugar Concentration on Growth of E. coli") +
  theme(plot.title = element_text(hjust = 0.5)) + #Centers the title
  stat_smooth(method = "lm", se=FALSE, color="black") + #Adds a trend line, se=FALSE
gets rid of cloud around trendline.
  geom_text(aes(x = 5, y = 25, label = lm_eqn1(lm(sugar ~ growth, sugardata))), parse
= TRUE) # Adds the values for the R2 value and equation of the trendline to the coord
inates x and y


# Adding the equation and R2 value to the graph
m= lm(sugar ~ growth, sugardata)

lm_eqn1 = function(m) {       #Function to calculate and add R2 value and equation of
the trendline to plot

  l <- list(a = format(coef(m)[1], digits = 2),
            b = format(abs(coef(m)[2]), digits = 2),
            r2 = format(summary(m)$r.squared, digits = 3));

  if (coef(m)[2] >= 0)  {
    eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(r)^2~"="~r2,l)
  } else {
    eq <- substitute(italic(y) == a - b %.% italic(x)*","~~italic(r)^2~"="~r2,l)
  }

  as.character(as.expression(eq));
}

plot2
```



Next, we need to test for an effect of sugar concentration on growth of E. coli using a regression-design linear model and likelihood ratio test.

Our null hypothesis is that the growth of the E. coli is the same regardless of the sugar concentration.

```
# Null model
nllnull = function(p,y){
  B0=p[1]
  sig=exp(p[2])
  expected=B0
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}

initialGuess = c(5,1) # Initial guess
null = optim(par=initialGuess,fn=nllnull,y=sugardata$growth) # Optimizing the model
```

Our alternative hypothesis is that the growth of the E. coli is dependent on the sugar concentration.

```
# Linear Model
nllike_linear=function(p,x,y){
  B0=p[1]
  B1=p[2]
  sigma=exp(p[3])

  expected=B0+B1*x

  nll_linear=-sum(dnorm(x=y,mean=expected,sd=sigma,log=TRUE))
  return(nll_linear)
}

initialGuess=c(5,1,1) # Initial guess
fit_linear=optim(par=initialGuess,fn=nllike_linear,x=sugardata$sugar,y=sugardata$grow
th) # Optimizing the model
```

Next, we conducted a likelihood ratio test to determine which model fit the data better.

```
# Likelihood ratio test
q = 2*(null$value-fit_linear$value) # Subtracting the linear model value from the nul
l model value and multiplying by 2
lrt = pchisq(q=q, df=1, lower.tail=FALSE) # Running the likelihood ratio test
print(lrt)
```

```
## [1] 2.638879e-10
```

Becuase the p-value is less than 0.05, this means that we can reject the null hypothesis and conclude that there is a significant effect of sugar concentration on E. coli growth

Question 3: Statistical Power Analysis

Regression-designed Experiment

First, we randomly selected 24 x-values between 0-50 and calculated the corresponding y-values based on a linear equation. We ran 10 monte carlos of this data. We then added noise to the data with 8 sigma values between 1 and 24.

```
size = 24 # Creates a random generation of 24 x-values between 0 and 50

num_sims <- 10 # Simulate the data 10 times
n_per_sim <- 24 # Each simulation will have 24 values
set.seed(24) # Seed is necessary for reproducibility
sigma_num <- 8 # The amount of sigma values for this analysis
sigma <- c(1,2,4,6,8,12,16,24) # Setting the sigma values for this analysis

outputx<-as.data.frame(matrix(0, ncol=10, nrow=24)) # Empty matrix to compile the 10
reiterations of the 24 randomly generated x variables
outputy<-as.data.frame(matrix(0, ncol=10, nrow=24)) # Empty matrix to compile the 10
reiterations of the 24 y variables
youtput <-list() # Empty file to compile the output matrices for the y values with no
ise into a list
for(sigma_number in 1:sigma_num){
  youtput[[sigma_number]]=matrix(NA,24,num_sims)
}

###### For loop to randomly generate x-values and then corresponding y-values based o
n the provided slope and intercept

for (sim_number in 1:num_sims){
  x = sample(x = 0:50, size = 24) # 24 randomly generates x-values between 0 and 50
  outputx[,sim_number] <- x # Puts the randomly generated x-values with 10 iterations
into the empty matrix we already created
  m <- 0.4 # Given slope value
  b <- 10 # Given y-intercept
  y = m*x + b # Calculates y-values from the 24 randomly generated x-values using giv
en slope and intercept
  outputy[,sim_number] <- y # Puts the y-values with 10 iterations into empty matrix
created before the for-loop
}

###### For loop to add noise to the y-values based on the sigma values given

  for (sigma_number in 1:sigma_num){
    for (j in 1:10){
  youtput[[sigma_number]][,j] = outputy[,j] + rnorm(nrow(outputy), mean=0, sd=sigma[s
igma_number])
    }
  }
```

Next, we ran our data through a null model.

```
#Null Model function

null.value.mat <- matrix(NA,8,10) # Empty matrix to store the value outputs from each
model run
null.par.mat <- array(NA,dim = c(2,8,10)) # Empty array to store the paramter outputs
from each model run

nllnull <- function(p,y){ # Null model
  B0=p[1]
  sig=exp(p[2])
  expected=B0
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}

# For loop to run each of the eight sigma values and also each of the ten monte carlo
s through the null model

for(sigma_number in 1:8){ # Running through each sigma value (each sheet in the list)
    for(num_sims in 1:10){ # Running through each monte carlo on each sheet
        initialGuess = c(2,10) # Initial guess
        null = optim(par=initialGuess,fn=nllnull,
                     y=as.matrix(youtput[[sigma_number]][,num_sims])) # Optimizing th
e model

        null.value.mat[sigma_number,num_sims] <- null$value # This stores the value o
utput from each monte carlo into a matrix
        null.par.mat[,sigma_number,num_sims] <- null$par # This stores the parameter
outputs from each monte carlo into an array
    }
}
```

Then, we ran our data through a linear model.

```
#Linear Model function

linear.value.mat <- matrix(NA,8,10) # Empty matrix to store the value outputs from ea
ch model run
linear.par.mat <- array(NA,dim = c(3,8,10)) # Empty array to store the paramter outpu
ts from each model run

nlllinear <- function(p,x,y){ # Linear Model
  B0=p[1]
  B1=p[2]
  sig=exp(p[3])
  expected=B0+B1*x
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}

# For loop to run each of the eight sigma values and also each of the ten monte carlo
s through the linear model

for(sigma_number in 1:8){ # Running through each sigma value (each sheet in the list)
    for(num_sims in 1:10){ # Running through each monte carlo on each sheet
        initialGuess = c(0,0.3,2) # Initial guess
        linear = optim(par=initialGuess,fn=nlllinear, x=as.matrix(outputx[,num_sims])
,
                       y=as.matrix(youtput[[sigma_number]][,num_sims])) # Optimizing th
e model

        linear.value.mat[sigma_number,num_sims] <- linear$value # This stores the val
ue output from each monte carlo into a matrix
        linear.par.mat[,sigma_number,num_sims] <- linear$par # This stores the parame
ter outputs from each monte carlo into an array
    }
}
```

We then performed a likelihood ratio test to determine if there was a significant difference in the fit between our null and linear models

```
####### Likelihood ratio test

lrt.value <- matrix(NA,8,10) # Empty matrix to store the p-values

for(sigma_number in 1:8){ # Start of for loop 1
    for(num_sims in 1:10){ # Start of for loop 2

      q = 2*(null.value.mat[sigma_number,num_sims]-linear.value.mat[sigma_number,num_
sims]) # Calculating the q-value for each cell - subtracting the linear value from th
e null value and multiplying by 2
      lrt = pchisq(q=q, df=1, lower.tail=FALSE) # Conducting the likelihood ratio tes
t
      lrt.value[sigma_number,num_sims] <- lrt # Storing the outputs from the likeliho
od ratio test in the matrix that was created earlier
    }
  }

# Average p-value across monte carlos
for (sigma_number in 1:8){
  print(mean(lrt.value[sigma_number,]))
}
```

```
## [1] 1.559851e-18
## [1] 2.321065e-12
## [1] 2.278654e-07
## [1] 0.001911517
## [1] 0.02297799
## [1] 0.06862814
## [1] 0.1239243
## [1] 0.1537127
```

The p-values are significnat (less than 0.05) until we get to the sixth sigma value (sigma=12).

ANOVA-designed Experiment

2-level ANOVA

First we set our two experimental trials to be 12 replicates at an x-value of 13 and 12 replicates at an x-value of 38. We then calculated our y-values from a linear equation and added some noise to the data with 8 sigma values between 1 and 24.

```
## 2-level ANOVA ##

x2 = c(13,13,13,13,13,13,13,13,13,13,13,13,38,38,38,38,38,38,38,38,38,38,38,38) # Cre
ating a vector of x-values
x_values2 = matrix(rep(x2,10),nrow=24,ncol=10) # Creating a matrix of x-values for th
e ten monte carlos

y2 <-  .4 * x_values2 + 10 # Calculating the y-values based on the x-values

youtput2 <-list() # Creating an empty list that will store the y-values once we add s
ome noise

# This for loop inserts empty data frames into the above list that will store the y-v
alues once we add some noise
for(sigma_number in 1:sigma_num){
  youtput2[[sigma_number]]=matrix(NA,24,num_sims)
}

# This for loop adds the noise to the y-values based on the eight pre-assigned sigma
values
for (sigma_number in 1:8){
  for (num_sims in 1:10){
  youtput2[[sigma_number]][,num_sims] = y2[,num_sims] + rnorm(nrow(y2), mean=0, sd=si
gma[sigma_number])
    }
}
```

We then categorized our x-values to be 0 and 1. x=13 became 0 (as the control) and x=38 became 1 (as the treatment).

```
for (i in 1:nrow(x_values2)){ # Runs the for loop through each row of the matrix
  for(j in 1:ncol(x_values2)){ # Runs the for loop through each column of the matrix
  if (x_values2[i,j] == '13'){ # If the x values equal 13 then...
    x_values2[i,j] <- 0 ### ... they becomes 0
  }
  else{ # Otherwise, if the x values are not equal to 13 then...
    x_values2[i,j] <- 1 ### ... they become 1
  }
 }
}
```

We created output matrices and arrays to store the results from our likelihood ratio test

```
A2.null.value.mat <- matrix(NA,8,10) # A matrix to store the value outputs from the n
ull model
A2.null.par.mat <- array(NA,dim = c(2,8,10)) # An array to store the parameter output
s from the null model
A2.linear.value.mat <- matrix(NA,8,10) # A matrix to store the value outputs from the
linear model
A2.linear.par.mat <- array(NA,dim = c(3,8,10)) # An array to store the parameter outp
uts from the linear model
```

Then we created custom functions - a null and a linear - that we used for our analysis.

```
# Null function
nllnull <- function(p,y){
  B0=p[1]
  sig=exp(p[2])
  expected=B0
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}

# Linear function
nlllinear <- function(p,x,y){
  B0=p[1]
  B1=p[2]
  sig=exp(p[3])
  expected=B0+B1*x
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}
```

Next, we ran the data through the models.

```
### Running the null model
for(sigma_number in 1:8){ # To run the model through each sheet of y-values (each sig
ma value)
    for(num_sims in 1:10){ # To run the model through each monte carlo (each column)
        initialGuess = c(1,10) # Initial guess
        null = optim(par=initialGuess,fn=nllnull,
                     y=as.matrix(youtput2[[sigma_number]][,num_sims])) # Optimizing t
he model

        A2.null.value.mat[sigma_number,num_sims] <- null$value # Storing the value ou
tputs into a matrix
        A2.null.par.mat[,sigma_number,num_sims] <- null$par # Storing the parameter o
utputs into an array
    }
}

### Running the linear model
for(sigma_number in 1:8){ # To run the model through each sheet of y-values (each sig
ma value)
    for(num_sims in 1:10){ # To run the model through each monte carlo (each column)
        initialGuess = c(0,0,2) # Initial guess
        linear = optim(par=initialGuess,fn=nlllinear, x=as.matrix(x_values2[,num_sims
]),
                     y=as.matrix(youtput2[[sigma_number]][,num_sims])) # Optimizing t
he model

        A2.linear.value.mat[sigma_number,num_sims] <- linear$value # Storing the valu
e outputs into a matrix
        A2.linear.par.mat[,sigma_number,num_sims] <- linear$par # Storing the paramet
er outputs into an array
    }
}
```

We then conducted a likelihood ratio test between the null and linear models to determine if there was a significant difference between the two treatments.

```
lrt.value.A2 <- matrix(NA,8,10) # Empty matrix to store the p-values

for(sigma_number in 1:8){ # Start of for loop 1
    for(num_sims in 1:10){ # Start of for loop 2

      q = 2*(A2.null.value.mat[sigma_number,num_sims]-A2.linear.value.mat[sigma_numbe
r,num_sims]) # Calculating the q-value for each cell - subtracting the linear value f
rom the null value and multiplying by 2
      lrt = pchisq(q=q, df=1, lower.tail=FALSE) # Conducting the likelihood ratio tes
t
      lrt.value.A2[sigma_number,num_sims] <- lrt # Storing the outputs from the likel
ihood ratio test in the matrix that was created earlier
    }
  }

# Average p-value across monte carlos
for (sigma_number in 1:8){
  print(mean(lrt.value.A2[sigma_number,]))
}
```

```
## [1] 2.584271e-17
## [1] 4.244588e-11
## [1] 3.388153e-06
## [1] 0.0007646754
## [1] 0.01846029
## [1] 0.201707
## [1] 0.276474
## [1] 0.5858669
```

The p-values are significnat (less than 0.05) until we get to the sixth sigma value (sigma=12).

4-level ANOVA

First we set our four experimental trials to be 10, 20, 30, and 40 - we set 10 as our control variable. Each trial was replicated six times for each monte carlo. We then calculated our y-values from a linear equation and added some noise to the data with 8 sigma values between 1 and 24.

```
## 4-level ANOVA ##
x4 = c(10,10,10,10,10,10,20,20,20,20,20,20,30,30,30,30,30,30,40,40,40,40,40,40) # Vec
tor of x-values
x_values4 = matrix(rep(x4,10),nrow=24,ncol=10) # Matrix of x-values for each monte ca
rlo


y4 <- .4 * x_values4 + 10 # Calculating the y-values that correspond with each x-valu
e based on the linear equation

youtput4 <-list() # An empty list to store the y-output values once we have added som
e noise to the data

# The following for loop creates empty data sets that will go into each sheet in the
above list (youtput4) where the y-output data will be stored once we add noise based
on the sigma values
for(sigma_number in 1:sigma_num){
  youtput4[[sigma_number]]=matrix(NA,24,num_sims)
}

# This for loop adds noise to the y-values based on the eight sigma values
for (sigma_number in 1:8){ # Runs the loop through each sheet in the list
  for (num_sims in 1:10){ # Runs the loop through each column in each sheet
    youtput4[[sigma_number]][,num_sims] = y4[,num_sims] + rnorm(nrow(y4), mean=0, sd=
sigma[sigma_number])
  }
}
```

We then categorized our x-values to be 0 or 1. x=10 is always 0 becuase it is our control. We then created three vectors that allow each of the trial values (x=20, x=30, x=40) to be 1.

```
z1=c(0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0) # This vector will test the 2nd
x-value (x=20)
z2=c(0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0) # This vector wil test the 3rd
x-value (x=30)
z3=c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1) # This vector will test the 4th
x-value (x=40)

xoutput4=matrix(c(z1,z2,z3),nrow=length(z1),ncol=3) # This puts the above vectors int
o a matrix that will represent out x-values
```

We created output matrices and arrays to store the results from our likelihood ratio test

```
A4.null.value.mat <- matrix(NA,8,10) # Matrix to store the value output from the null model
A4.null.par.mat <- array(NA,dim = c(2,8,10)) # Array to store the parameter outputs from the null model
A4.linear.value.mat <- matrix(NA,8,10) # Matrix to store the value output from the linear model
A4.linear.par.mat <- array(NA,dim = c(5,8,10)) # Array to store the parameter outputs from the linear model
```

Then we created custom functions - a null and a linear - that we used for our analysis.

```
# Null function
nllnull <- function(p,y){
  B0=p[1]
  sig=exp(p[2])
  expected=B0
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}

# Linear function
nlllinear <- function(p,xoutput4,y){
  B0=p[1]
  B1=p[2]
  B2=p[3]
  B3=p[4]
  sig=exp(p[5])
  expected=B0+B1*xoutput4[,1]+B2*xoutput4[,2]+B3*xoutput4[,3]
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}
```

Next, we ran the models.

```
### Running the null model
for(sigma_number in 1:8){ # Looping through each sheet in the list (each sigma value)
    for(num_sims in 1:10){ # Looping through each column in each sheet (each monte ca
rlo)
        initialGuess = c(1,10) # Initial Guess
        null = optim(par=initialGuess,fn=nllnull,
                     y=as.matrix(youtput4[[sigma_number]][,num_sims])) # Optimizing t
he model

        A4.null.value.mat[sigma_number,num_sims] <- null$value # Storing the value ou
tputs into a matrix
        A4.null.par.mat[,sigma_number,num_sims] <- null$par # Storing the parameter o
utputs into an array
    }
}

### Running the linear model
for(sigma_number in 1:8){ # Looping through each sheet in the list (each sigma value)
    for(num_sims in 1:10){ # Looping through each column in each sheet (each monte ca
rlo)
        initialGuess = c(15,4,8,12,1) # Initial Guess
        linear = optim(par=initialGuess,fn=nlllinear, x=as.matrix(xoutput4),
                     y=as.matrix(youtput4[[sigma_number]][,num_sims])) # Optimizing t
he model

        A4.linear.value.mat[sigma_number,num_sims] <- linear$value # Storing the valu
e outputs into a matrix
        A4.linear.par.mat[,sigma_number,num_sims] <- linear$par # Storing the paramet
er outputs into an array
     }
    }
```

We then conducted a likelihood ratio test between the null and linear models to determine if there was a significant difference between the control and test treatments.

```
lrt.value.A4 <- matrix(NA,8,10) # Empty matrix to store the p-values

for(sigma_number in 1:8){ # Start of for loop 1
    for(num_sims in 1:10){ # Start of for loop 2

      q = 2*(A4.null.value.mat[sigma_number,num_sims]-A4.linear.value.mat[sigma_numbe
r,num_sims]) # Calculating the q-value for each cell - subtracting the linear value f
rom the null value and multiplying by 2
      lrt = pchisq(q=q, df=3, lower.tail=FALSE) # Conducting the likelihood ratio tes
t
      lrt.value.A4[sigma_number,num_sims] <- lrt # Storing the outputs from the likel
ihood ratio test in the matrix that was created earlier
    }
  }

# Average p-value across monte carlos
for (sigma_number in 1:8){
  print(mean(lrt.value.A4[sigma_number,]))
}
```

```
## [1] 1.035018e-16
## [1] 4.526209e-08
## [1] 0.0001068817
## [1] 0.008581468
## [1] 0.09100051
## [1] 0.2109194
## [1] 0.4157486
## [1] 0.2611385
```

The p-values are significnat (less than 0.05) until we get to the fifth sigma value (sigma=8).

8-level ANOVA

First we set our eight experimental trials to be 5, 10, 15, 20, 25, 30, 35, 40 - x=5 was set as the control. Each trial was replicated three times for each monte carlo replicate. We then calculated our y-values from a linear equation and added some noise to the data with 8 sigma values between 1 and 24.

```
## 8-level ANOVA ##
x8 = c(5,5,5,10,10,10,15,15,15,20,20,20,25,25,25,30,30,30,35,35,35,40,40,40) # Vector of x-values
x_values8 = matrix(rep(x8,10),nrow=24,ncol=10) # Creating a matrix of x-values for each of the monte carlo tests

y8 <- .4 * x_values8 + 10 # Calculating the y-values from the x-values using a linear equation

youtput8 <-list() # Creating an empty list to store the y-values once we add some noise using sigma

# This for loop creates empty data frames that wil go into each sheet of the list that will hold the y-values with noise once they are generated below
for(sigma_number in 1:sigma_num){
  youtput8[[sigma_number]]=matrix(NA,24,num_sims)
}

# This for loop adds noise to the y-values
for (sigma_number in 1:8){
  for (num_sims in 1:10){
  youtput8[[sigma_number]][,num_sims] = y8[,num_sims] + rnorm(nrow(y8), mean=0, sd=sigma[sigma_number])
    }
}
```

We then categorized our x-values to be 0 or 1. x=5 is always 0 becuase it is our control. We then created seven vectors that allow each of the trial values (x=10, x=15, x=20, x=25, x=30, x=35, x=40) to be 1.

```
v1=c(0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) # This vector will test the 2nd x-value (x=10)
v2=c(0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) # This vector will test the 3rd x-value (x=15)
v3=c(0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0) # This vector will test the 4th x-value (x=20)
v4=c(0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0) # This vector will test the 5th x-value (x=25)
v5=c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0) # This vector will test the 6th x-value (x=30)
v6=c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0) # This vector will test the 7th x-value (x=35)
v7=c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1) # This vector will test the 8th x-value (x=40)

xoutput8=matrix(c(v1,v2,v3,v4,v5,v6,v7),nrow=length(v1),ncol=7) # This puts the above vectors into a matrix that will represent out x-values
```

We created output matrices and arrays to store the results from our likelihood ratio test

```
A8.null.value.mat <- matrix(NA,8,10) # Matrix to store the value outputs from the nul
l model
A8.null.par.mat <- array(NA,dim = c(2,8,10)) # Array to store the parameter outputs f
rom the null model
A8.linear.value.mat <- matrix(NA,8,10) # Matrix to store the value outputs from the l
inear model
A8.linear.par.mat <- array(NA,dim = c(9,8,10)) # Array to store the parameter outputs
from the linear model
```

Then we created custom functions - a null and a linear - that we used for our analysis.

```
# Null function
nllnull <- function(p,y){
  B0=p[1]
  sig=exp(p[2])
  expected=B0
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}

# Linear function
nlllinear <- function(p,xoutput8,y){
  B0=p[1]
  B1=p[2]
  B2=p[3]
  B3=p[4]
  B4=p[5]
  B5=p[6]
  B6=p[7]
  B7=p[8]
  sig=exp(p[9])
  expected=B0+B1*xoutput8[,1]+B2*xoutput8[,2]+B3*xoutput8[,3]+B4*xoutput8[,4]+B5*xout
put8[,5]+B6*xoutput8[,6]+B7*xoutput8[,7]
  nll=-sum(dnorm(x=y, mean=expected, sd=sig,log=TRUE))
  return(nll)
}
```

Next, we ran the models.

```
### Running the null model
for(sigma_number in 1:8){ # Loops through each sheet in the list (each sigma value)
    for(num_sims in 1:10){ # Loops through each column in each sheet (each monte carl
o)
        initialGuess = c(1,10) # Initial guess
        null = optim(par=initialGuess,fn=nllnull,
                     y=as.matrix(youtput8[[sigma_number]][,num_sims])) # Optimizing t
he model

        A8.null.value.mat[sigma_number,num_sims] <- null$value # Storing the value ou
tputs into the matrix
        A8.null.par.mat[,sigma_number,num_sims] <- null$par # Storing the parameter o
utputs into the array
    }
}

### Running the linear model
for(sigma_number in 1:8){ # Loops through each sheet in the list (each sigma value)
    for(num_sims in 1:10){ # Loops through each column in each sheet (each monte carl
o)
        initialGuess = c(15,4,6,8,10,12,14,16,1) # Initial guess
        linear = optim(par=initialGuess,fn=nlllinear, x=as.matrix(xoutput8),
                     y=as.matrix(youtput8[[sigma_number]][,num_sims])) # Optimizing t
he model

        A8.linear.value.mat[sigma_number,num_sims] <- linear$value # Storing the valu
e outputs into the matrix
        A8.linear.par.mat[,sigma_number,num_sims] <- linear$par # Storing the paramet
er outputs into the array
    }
}
```

We then conducted a likelihood ratio test between the linear and null models to determine if there was a significant difference between the control treatment and the test treatments.

```
lrt.value.A8 <- matrix(NA,8,10) # Empty matrix to store the p-values

for(sigma_number in 1:8){ # Start of for loop 1
    for(num_sims in 1:10){ # Start of for loop 2

      q = 2*(A8.null.value.mat[sigma_number,num_sims]-A8.linear.value.mat[sigma_numbe
r,num_sims]) # Calculating the q-value for each cell - subtracting the linear value f
rom the null value and multiplying by 2
      lrt = pchisq(q=q, df=7, lower.tail=FALSE) # Conducting the likelihood ratio tes
t
      lrt.value.A8[sigma_number,num_sims] <- lrt # Storing the outputs from the likel
ihood ratio test in the matrix that was created earlier
    }
  }

# Average p-value across monte carlos
for (sigma_number in 1:8){
  print(mean(lrt.value.A8[sigma_number,]))
}
```

```
## [1] 3.698037e-09
## [1] 2.149233e-05
## [1] 0.006329164
## [1] 0.05208028
## [1] 0.2677172
## [1] 0.2930433
## [1] 0.3645108
## [1] 0.7447357
```

The p-values are significnat until we get to the fourth sigma value (sigma=6).

Assessing the ANOVA-vs regression-design performance

As a reminder, here are the average p-values across the monte carlos for each sigma value for each of our 4 tests.

Regression-design

```
for (sigma_number in 1:8){
  print(mean(lrt.value[sigma_number,]))
}
```

```
## [1] 1.559851e-18
## [1] 2.321065e-12
## [1] 2.278654e-07
## [1] 0.001911517
## [1] 0.02297799
## [1] 0.06862814
## [1] 0.1239243
## [1] 0.1537127
```

2-level ANOVA

```
for (sigma_number in 1:8){
  print(mean(lrt.value.A2[sigma_number,]))
}
```

```
## [1] 2.584271e-17
## [1] 4.244588e-11
## [1] 3.388153e-06
## [1] 0.0007646754
## [1] 0.01846029
## [1] 0.201707
## [1] 0.276474
## [1] 0.5858669
```

4-level ANOVA

```
for (sigma_number in 1:8){
  print(mean(lrt.value.A4[sigma_number,]))
}
```

```
## [1] 1.035018e-16
## [1] 4.526209e-08
## [1] 0.0001068817
## [1] 0.008581468
## [1] 0.09100051
## [1] 0.2109194
## [1] 0.4157486
## [1] 0.2611385
```

8-level ANOVA

```
for (sigma_number in 1:8){
  print(mean(lrt.value.A8[sigma_number,]))
}
```

```
## [1] 3.698037e-09
## [1] 2.149233e-05
## [1] 0.006329164
## [1] 0.05208028
## [1] 0.2677172
## [1] 0.2930433
## [1] 0.3645108
## [1] 0.7447357
```

Overall, the regression-design performs better than the 2-level, 4-level and 8-level ANOVA-designs; we can conclude this by looking at the corresponding p-values bewteen the four different designs.

Additionally, as the number of levels in the ANOVA increased, the statistical power decreased (i.e. the 2-level performed better than the 4-level, which performed better than the 8-level). A likely explanation for this is the difference in the sample size across the different designs. Because the number of observations for each trial was held constant at 24, this meant that as the number of levels in the ANOVA-design increased, the sample size decreased for each treatment.