MATH436/536Test1

Doug Nychka, Nathan Oliver

2024-01-02

Problem 1 A Nonlinear Model for A4 Prices

From the example covered in class we fit a nonlinear function to the Audi data in an attempt to capture the sharp rise in price for very low mileages.

The proposed model was:

$$y_i = \beta_1 + \beta_2 x_i + \beta_3 e^{-x_i/\gamma} + \epsilon_i$$

where ϵ_i is the random error, x_i and y_i are the mileage and price for the i^{th} car, and the parameters are $\beta_1, \beta_2, \beta_3, \gamma$.

These parameters can be estimated by OLS using the function **fitNonlinearA4** that is in the .R file with the same name. You should source this file to run the function. You can also look at this file see how the function is coded. **fitNonlinearA4** returns in a list the components: `beta`, `gamma`, `fitted.values` and `residuals`. Note that since the gamma parameter enters in a nonlinear way just using the **lm** function will not work to find the `gamma` parameter.

```
# note you only need to source this code once during your R session.
source("fitNonlinearA4.R")
```

Fitting the Audi A4 data with the extra γ parameter.

```
source("fitNonlinearA4.R")
# load the data and divide mileage and price by 1000 to make the sizes # easier to work with
load("AudiA4.rda")
 mileage<- AudiA4$mileage/1000
 price<- AudiA4$price/1000
 out<- findNonlinearA4(mileage,price)
 # look at parameter estimates
 out$beta
```

```
##     beta1     beta2     beta3
## 28.122806 -0.154187 23.269749
```

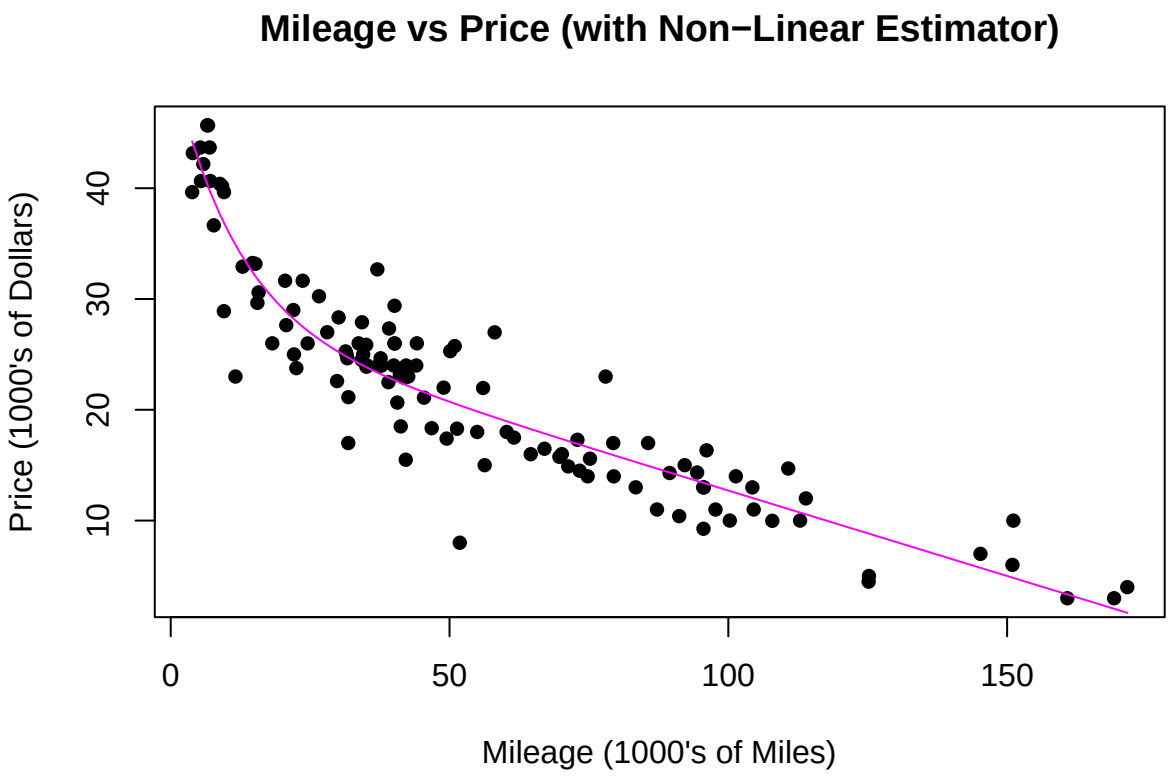
```
 out$gamma
```

```
## [1] 11.59322
```

1 (a)

Make a scatterplot of price against mileage and add the curve based on the estimated parameters. Evaluate the curve on a grid on about 200 points from 0 to 170. (Note we have divided mileage by 1000)

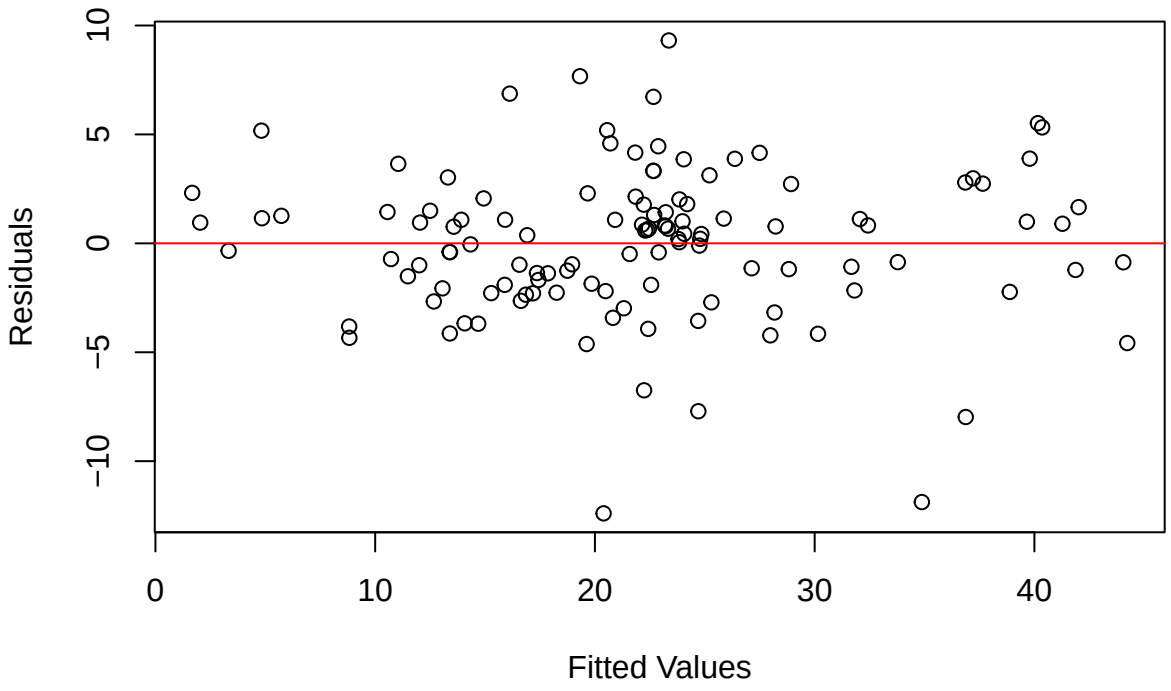
```
fun1 <- function(x){
  fun1 = 28.122806 - 0.154187*x + 23.269749*exp(-x/11.59322)
}

plot(mileage, price, xlab = "Mileage (1000's of Miles)", ylab = "Price (1000's of Dollars)",
     main = "Mileage vs Price (with Non-Linear Estimator)", pch = 16)
curve(fun1, add = TRUE, col = "magenta")
```



1(b)

Create a plot of the residuals and comment on the adequacy of this model's fit to the data.

```
NLMod1 <- lm(price ~ mileage + exp(-mileage/10.64407))
plot(out$fitted.values, out$residuals, xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



Looking at the plot of residuals vs the fitted line, I would say this non-linear model is a decentley adequate fit to the data. Although the data itself has a lot of variability, our model captures the trend very well. The only problem with this model is that it allows for negative prices. We should add another non-linear element that is more reasonable for cars with extremely low value.

1(c)

Under what circumstances will the function **findNonlinearA4** return NAs for the parameter estimates? Explain why I included this feature.

```
x_test <- seq(1,5,1)
y_test <- seq(1,10,2)
out_test <- findNonlinearA4(x_test, y_test)
out_test$beta
```

```
## [1] NA NA NA
```

```
out_test$gamma
```

```
## [1] NA
```

As shown above, when fitNonlinearA4 is fed a set of points that is perfectly linear, we recieve a NA for parameter estimates. This means that fitNonlinearA4 assumes the trend is nonlinear, and cannot properly estimate betas and a gamma for a linear trend. This feature was probably included with this fail safe as to not mistakenly model a linear trend with a non-linear assumption.

Problem 2

Problem 2(a)

Use at least 500 bootstrap replicates and report the mean and standard deviations of the gamma parameter for your bootstrap values. Do a simple test of hypothesis (aka a T-Test) to test if there is evidence that the population mean for this bootstrap sample is different from the "true" parameter value assumed in the bootstrap simulations.

```
N = 119
nBoot <- 500
betaBoot<- matrix(NA, nBoot, 4)
pivot<- rep( NA, nBoot)

beta_gamma_True<- c(28.122806, -0.154187, 23.269749, 11.59322)

nl_model<- findNonlinearA4(mileage,price)

sigmaHat<- sqrt( sum(nl_model$residuals^2)/ (N-4))
beta_gamma_Hat <- c(nl_model$beta[1], nl_model$beta[2], nl_model$beta[3], nl_model$gamma)


EHat<- nl_model$residuals

set.seed(233)

for( k in 1:nBoot){
  ES<- sample(EHat, N, replace=TRUE)
# synthetic bootstrap sample
  YS<- beta_gamma_Hat[1] + beta_gamma_Hat[2]*mileage + beta_gamma_Hat[3]*exp(-mileage/beta_gamma_Hat[4])
  fitS<- findNonlinearA4(mileage,YS)
  new_row <- rbind(fitS$beta[1], fitS$beta[2], fitS$beta[3], fitS$gamma)
  betaTmp<- new_row
  betaBoot[k,]<- betaTmp
  }
```

```
stats(betaBoot)
```

```
##                     [,1]         [,2]       [,3]       [,4]
## N              500.000000 500.00000000 500.000000 500.000000
## mean            27.941653  -0.15240382  23.899072  11.900746
## Std.Dev.         1.354809   0.01372088   2.908188   2.850155
## min             22.164467  -0.18949471  16.300101   5.423729
## Q1              27.145177  -0.16225001  22.066461   9.694915
## median          28.086067  -0.15314520  23.546863  11.593220
## Q3              28.834862  -0.14401480  25.619630  13.491525
## max             30.743462  -0.10022248  38.169120  23.457627
## missing values   0.000000   0.00000000   0.000000   0.000000
```

```
t.test(betaBoot[,4], mu = 11.59322, alternative = "two.sided")
```

```
##
##  One Sample t-test
##
## data:  betaBoot[, 4]
## t = 2.4127, df = 499, p-value = 0.0162
```

```
## alternative hypothesis: true mean is not equal to 11.59322
## 95 percent confidence interval:
##  11.65032 12.15118
## sample estimates:
## mean of x
##  11.90075
```

After performing bootstrap on this non-linear model and performing a t-test for the mean value of gamma, we can say with 95 percent certainty that the true mean value of gamma is not 11.59322.

Problem 2(b)

Based on percentiles of the bootstrap samples construct a 95% confidence interval for the gamma parameter. Is your interval symmetric about the bootstrap mean?

```
quantile( betaBoot[,4], c(.025, .975))
```

```
##      2.5%     97.5%
##  7.322034 18.486441
```

```
check = 18.486441 - 11.900746
check2 = 11.900746 - 7.322034
check
```

```
## [1] 6.585695
```

```
check2
```

```
## [1] 4.578712
```

No, the interval is not symmetric about the bootstrap mean.

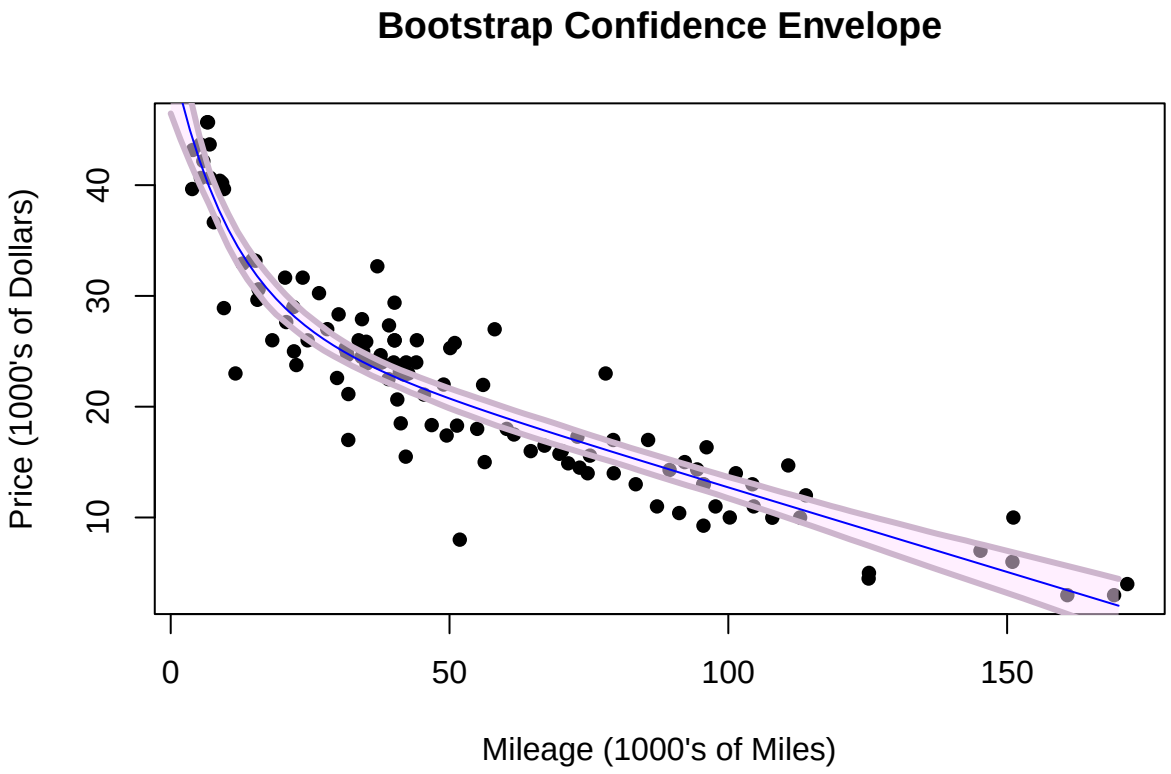
Problem 2(c)

Based on your bootstrap samples compute the predicted values for the curve on a grid of 100 points.

```
mileageGrid<- seq( 0, 170, length.out=100)

pred_price = mean(betaBoot[,1]) + mean(betaBoot[,2])*mileageGrid + mean(betaBoot[,3])*exp(-mileageGrid/


pred_mat <- matrix(NA, 100, 500)
for(i in 1:100){
  for(j in 1:500){
    pred_price = betaBoot[j,1] + betaBoot[j,2]*mileageGrid[i] + mean(betaBoot[j,3])*exp(-mileageGrid[i],
    pred_mat[i, j] <- pred_price
  }
}
```

```
library(scales)
meancurve <- c()
lowcurve <- c()
highcurve <- c()

for(k in 1:100){
  xtes <- quantile(pred_mat[k,], c(.025, .975))
  meancurve <- append(meancurve, mean(pred_mat[k,]))
  lowcurve <- append(lowcurve, xtes[1])
  highcurve <- append(highcurve, xtes[2])
```

```
}

plot(mileage, price, xlab = "Mileage (1000's of Miles)", ylab = "Price (1000's of Dollars)",
     main = "Bootstrap Confidence Envelope", pch = 16)
envelopePlot(mileageGrid, highcurve, y2 = lowcurve, col=alpha( "thistle1", .5))
lines(mileageGrid, meancurve, col = "blue")
```



Hint: If you have 500 bootstrap replicates this will result in a 100X500 matrix of predicted values.

Find the mean and 2.5% an 97.5% quantiles of the predictions for each of the grid points. There are enough closely space grid points that these can be plotted as curves. Add these three curves (2.5%, mean, 97.5%) to your scatterplot of the data from Problem 1.

BTW To create a spiffy envelope plot with the quantile curves see the fields function **envelopePlot**

Problem 2(d) GRAD/EXTRA CREDIT

The 2.5% an 97.5% quantile curves from 2(c) form an envelope around some of the data.

- Should this envelope contain roughly 95% of the data points?

No. This is a 95 percent CI for the true price given a specific mileage, not for 95 percent CI for all possible prices given a specific mileage.

- The sample size in this case is $n = 119$ cars. How would expect the envelope to change as the sample size gets very large?

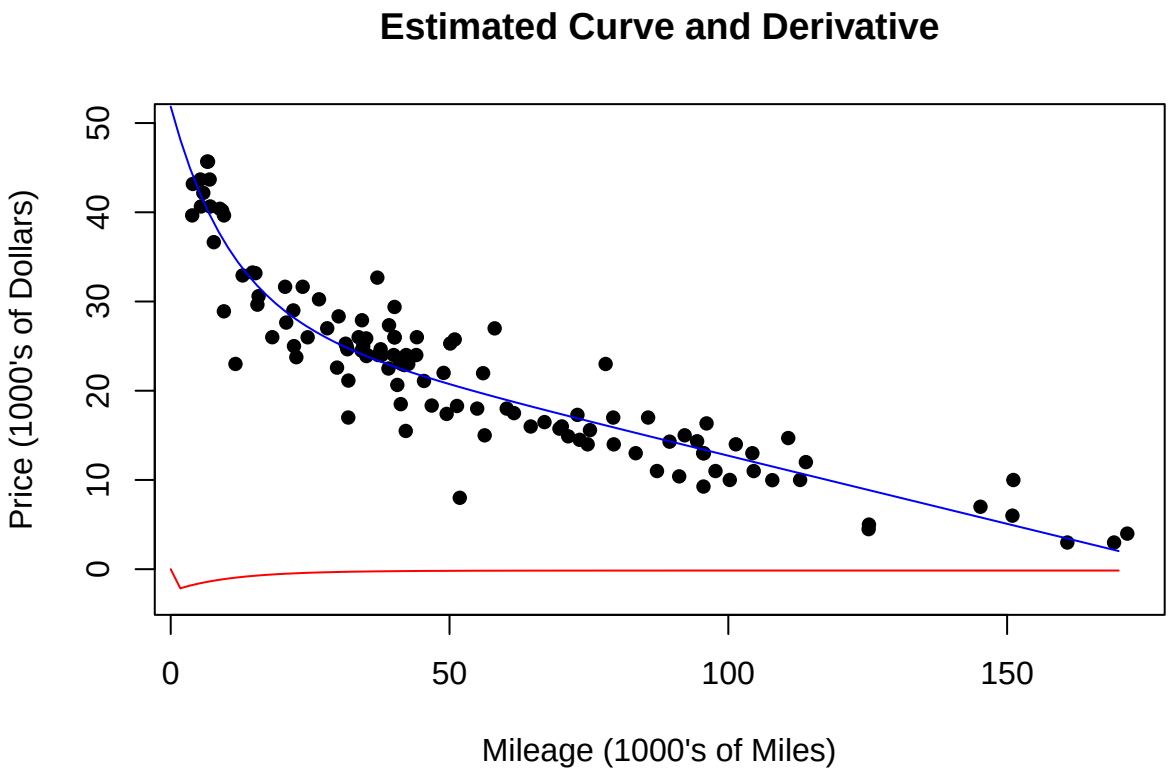
If we increase the value of n, I would expect the width of the envelope to decrease. The larger our sample, the more accurate our bootstrap simulation will be. When we have a bootstrap simulation that is more accurate, we can expect a smaller CI.

Problem 2(e) GRAD/EXTRA CREDIT

An business analyst wants to estimate the rate of decrease in the value as a function of mileage. Make a plot of the derivative of your fitted curve, add the derivative of a linear function applied to mileage and also the quadratic fit.

Note: To do this use calculus applied to the fitted curves based on the estimated parameters.

```
d_curve <- c(0)

for(k in 1:99){
  vdel <- meancurve[k+1] - meancurve[k]
  hdel <- mileageGrid[k+1] - mileageGrid[k]
  d_curve <- append(d_curve, (vdel/hdel))
}

plot(mileage, price, xlab = "Mileage (1000's of Miles)", ylab = "Price (1000's of Dollars)",
     main = "Estimated Curve and Derivative", pch = 16, ylim = c(-3,50))
lines(mileageGrid, meancurve, col = "blue")
lines(mileageGrid, d_curve, col = "red")
```



Problem 3

Refer to the **nls** function that is demonstrated in the script **Lecture08NLSExample.R** from lecture 08.

3(a)

Use **nls** to fit the nonlinear model from Problem 1. You can use starting values of 28, -.15, 23, for beta1, beta2 and beta3 and 10 for gamma.

```
nls_formula <- price ~ beta1 + beta2*mileage + beta3*exp(-mileage/gam)
b1g <- 28
b2g <- -0.15
b3g <- 23
gamg <- 10
nls_fit <- nls(formula = nls_formula, start = list(beta1 = b1g, beta2 = b2g, beta3 = b3g, gam = gamg))
```

3(b)

- Compare the standard error reported by **nls** to your bootstrap results.

```
summary(nls_fit)
```

```
##
## Formula: price ~ beta1 + beta2 * mileage + beta3 * exp(-mileage/gam)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## beta1 28.10134    1.34010  20.970  < 2e-16 ***
## beta2 -0.15399    0.01429 -10.780  < 2e-16 ***
## beta3 23.23290    2.86986   8.095 6.64e-13 ***
## gam   11.65193    2.93370   3.972 0.000125 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.474 on 115 degrees of freedom
##
## Number of iterations to convergence: 10
## Achieved convergence tolerance: 7.715e-06
```

```
stats(betaBoot)
```

```
##                      [,1]         [,2]       [,3]       [,4]
## N              500.000000 500.00000000 500.000000 500.000000
## mean            27.941653  -0.15240382  23.899072  11.900746
## Std.Dev.         1.354809   0.01372088   2.908188   2.850155
## min             22.164467  -0.18949471  16.300101   5.423729
## Q1              27.145177  -0.16225001  22.066461   9.694915
## median          28.086067  -0.15314520  23.546863  11.593220
## Q3              28.834862  -0.14401480  25.619630  13.491525
## max             30.743462  -0.10022248  38.169120  23.457627
## missing values   0.000000   0.00000000   0.000000   0.000000
```

They're all pretty similar! The SE for b1 and b3 is slightly smaller in the nls estimation compared to the bootstrap. By contrast, the b2 and gamma SE is slightly smaller for the bootstrap than the nls estimation. It appears to be a rather negligable tradeoff.

- Compare the the confidence interval reported by **nls** to the one found by the bootstrap.

```
confint(nls_fit, level = 0.95)
```

```
## Waiting for profiling to be done...
```

```
##            2.5%     97.5%
## beta1 22.969490 30.533930
## beta2 -0.181513 -0.108787
## beta3 18.113525 32.892839
## gam    6.499121 23.470796
```

```
quantile( betaBoot[,1], c(.025, .975))
```

```
##     2.5%    97.5%
## 25.07326 30.17109
```

```
quantile( betaBoot[,2], c(.025, .975))
```

```
##       2.5%      97.5%
## -0.1763120 -0.1226188
```

```
quantile( betaBoot[,3], c(.025, .975))
```

```
##     2.5%    97.5%
## 18.87382 29.99359
```

```
quantile( betaBoot[,4], c(.025, .975))
```

```
##      2.5%     97.5%
##  7.322034 18.486441
```

Across the board, the boostrap method has smaller CI's for all parameters. We like this, and shows that proven power of bootsrap.

3(c) GRAD/ EXTRA CREDIT

Determine a set of starting values where the **nls** function fails to converge. In this case explain how the use of the **try** function in R will let you include this case in your Rmd rendering.

4 Using Monte Carlo to investigate weight least squares.

Referring to AdvStatsLecture04WLSExample.Rmd and HW03 we used Monte Carlo to determine how different schemes worked with a weighted least squares linear model. In particular we could show that WLS was better than OLS or just looking at the first 20 observations.

Suppose your supervisor pitches yet another estimator of the line's slope arguing it is simple to explain and probably as good some fancy OLS.

- Find the of the 20 smallest x values and the corresponding 20 Y values. Call these $\bar{x}_1$ and $\bar{Y}_1$. Do the same for the 20 largest values of X and again find the means $\bar{x}_2$ and $\bar{Y}_2$
- Estimate the slope using

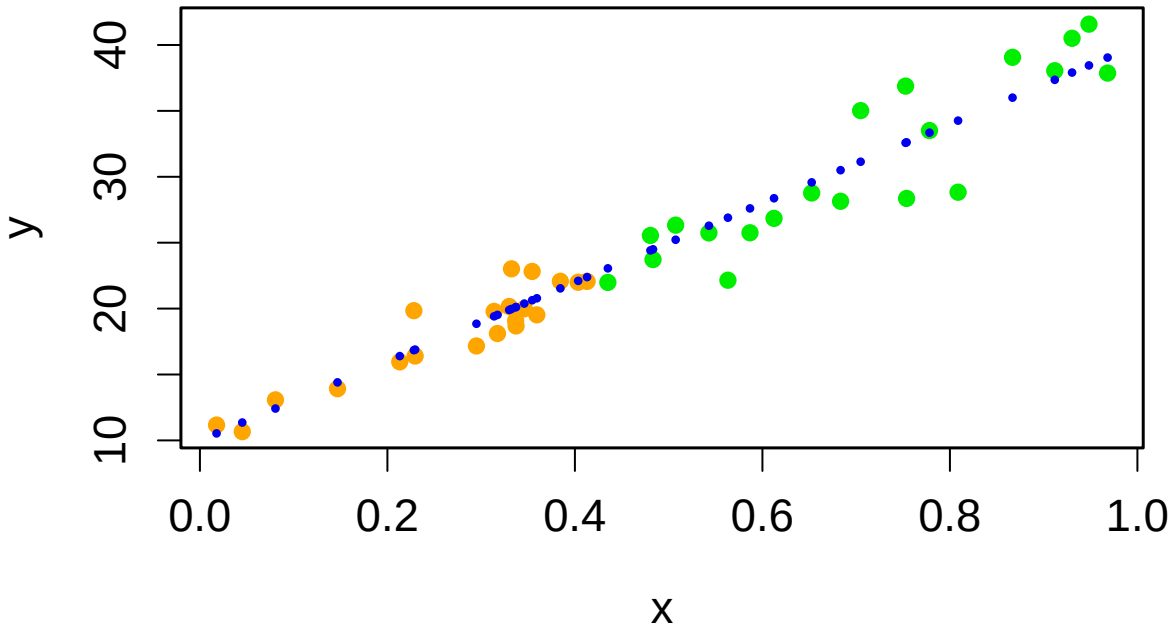
$$\hat{\beta}_2 = \frac{(\bar{Y}_2 - \bar{Y}_1)}{(\bar{x}_2 - \bar{x}_1)}$$

  (The usual slope formula for a line from two points.)

Use the Monte Carlo method from Lecture04 to see how good this estimator compared to the WLS one. To make this easy simulate both the simple and WLS in your Monte Carlo.

*** Please use the same x values from the example ***

```
n<- 40
set.seed(224)
x<- sort(runif(n))
X<- cbind( 1, x)
beta<- c( 10,30)
grp<- c(rep( 1, n/2), rep(2, n/2))
SD<- c( 1, 3)
errorSD<- SD[grp]
e<- rnorm(n)*errorSD
y<- X%*%beta + e
fields.style()
plot( x, y, col=grp)
points( x, X%*%beta, col=3, cex=.5)
```



```r
M<- 1000
betaNew <- rep(NA, M)
beta2WLS<- rep(NA, M)
set.seed(444)

for( k in 1:M){
  E<- rnorm(n)*errorSD
  YFake<- X%*%beta + E
#Newmeth
  x_1 <- x[1:20]
  x_2 <- x[21:40]
  y_1 <- YFake[1:20]
  y_2 <- YFake[21:40]

  est_slope = (mean(y_2) - mean(y_1))/(mean(x_2) - mean(x_1))
  betaNew[k] <- est_slope

# WLS
  fitWLS<- lm( YFake~ X-1, weights= 1/errorSD^2 )
  beta2WLS[k]<- fitWLS$coefficients[2]
}
```

4(a)

Is the expected value of this estimator different from the "true" slope of 30? Use a hypothesis test to determine this.

```r
t.test(betaNew, mu = 30, alternative = "two.sided")
```

```
##
##  One Sample t-test
##
## data:  betaNew
## t = -0.13004, df = 999, p-value = 0.8966
## alternative hypothesis: true mean is not equal to 30
## 95 percent confidence interval:
##  29.89114 30.09533
## sample estimates:
## mean of x
##  29.99323
```

After performing a hypothesis test, we do not have enough evidence to reject the null hypothesis, that the slope is different from 30, at the 95 percent level.

4(b)

How large is the standard error of this estimate compared to the usual WLS one?

```r
stats(cbind(beta2WLS, betaNew))
```

```
##                   beta2WLS     betaNew
## N              1000.000000 1000.000000
## mean             30.026364   29.993234
## Std.Dev.          1.179145    1.645212
## min              25.995929   24.502323
## Q1               29.193849   28.876165
```

```
## median           30.039617   30.006500
## Q3               30.844090   31.103766
## max              33.922874   35.210502
## missing values    0.000000    0.000000
```

The standard error of this estimate is noticeably larger than that of WLS. This is mostly due to the fact that this new method only really has two points per computation, where the WLS method uses all 40 points. Less data increases variance substantially.

4(c) GRAD/EXTRA

Explain why we know the simple estimate would not be better. Does its performance depend on the fact the errors have different variances?

The main takeaway here is a lack of data. By comparing two means instead of all 40 points, the error is computed differently. We also lose our conventional use of CLT. Typically, we always want more data if we want to make better approximations. The two variances just aren't all that comparable.

Problem 5 EXTRA CREDIT (5 points)

When was the idea of the bootstrap in statistics first published and who is responsible for its invention?

The idea of bootstrap was "officially" coined and introduced in the book "Bootstrap Methods: Another Look at the Jackknife" by Bradley Effron in 1979. This published work was *greatly* reliant on the Jackknife resampling method which was developed by Maurice Queoullie.