1. For each of the following models, state whether it is a linear model, an intrinsically linear model, or a non linear model. In the case of an intrinsically linear model, indicate how it can be transformed to a linear model by describing a suitable transformation. In each case, ϵ_i denotes a random error.

   (a) Intrinsically linear.

   $$y_i \;=\; \exp(\beta_0 + \beta_1 x_i)\epsilon_i \quad \Rightarrow \quad \log(y_i) \;=\; \beta_0 + \beta_1 x_i + \log(\epsilon_i)$$

   (b) Linear.

   $$y_i \;=\; \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i$$

   (c) Intrinsically linear.

   $$y_i \;=\; [1 + exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i)]^{-1} \quad \Rightarrow \quad \log\left(\frac{1}{y_i} - 1\right) \;=\; \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i$$

   (d) Non linear.

   $$y_i \;=\; exp(\beta_0 + \log(\beta_1 x_{1i})) + \epsilon_i$$

   (e) Intrinsically linear.

   $$\begin{aligned} y_i \;&=\; \beta_0 x_{1i}^{\beta_1} x_{2i}^{\beta_2} + \epsilon_i^{\beta_3} \\ \Rightarrow \quad & \log(y_i) = \log(\beta_0) + \beta_1 \log(x_{1i}) + \beta_2 \log(x_{2i}) + \eta_i \text{ , where } \eta_i = \beta_3 \log(\epsilon_i) \end{aligned}$$

   (f) Non linear.

   $$y_i \;=\; \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}^{\beta_3} + \epsilon_i$$

2. Use least squares analysis to fit the following model.

   $$Y_i \;= \beta_0 x_{1i}^{\beta_1} x_{2i}^{\beta_2} + \epsilon_i \quad \text{where} \quad \epsilon_i \;\sim\; NID(0, \sigma^2) \text{ , } i = 1, 2, \cdots, 18.$$

   (a) To obtain starting values, ignore the additive random error in the nonlinear model shown above and use least squares to fit the model

   $$\log(Y_i) = \log(\beta_0) + \beta_1 \log(x_{1i}) + \beta_2 \log(x_{2i}) + \text{error} \text{ .}$$

   Evaluate starting values for β_0, β_1, and β_2.

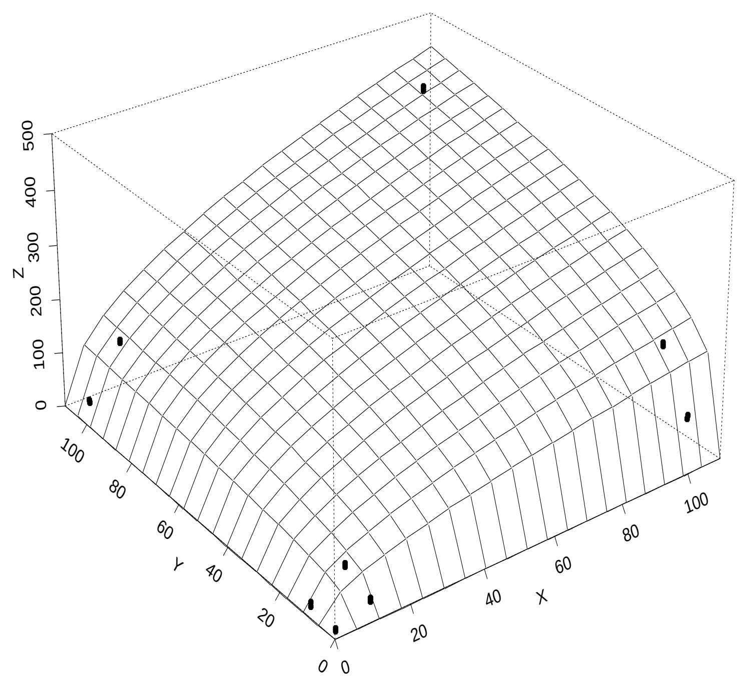
```
       pnonlin <- read.table("pnonlin.dat",col.names=c("Y","X1","X2"))
       lm.out <- lm(log(Y) ~ log(X1)+log(X2), data=pnonlin)
       start.val <- coef(lm.out)             # Note that Intercept = log(B0)
       start.val[1] <- exp(start.val[1])     # Therefore, B0 = exp(Intercept)
       names(start.val) <- c("b0","b1","b2") # specify the name of each
starting
                                             # value ( to use this in nls())
       > start.val
              b0        b1        b2
        9.591047 0.5148508 0.2984546
```

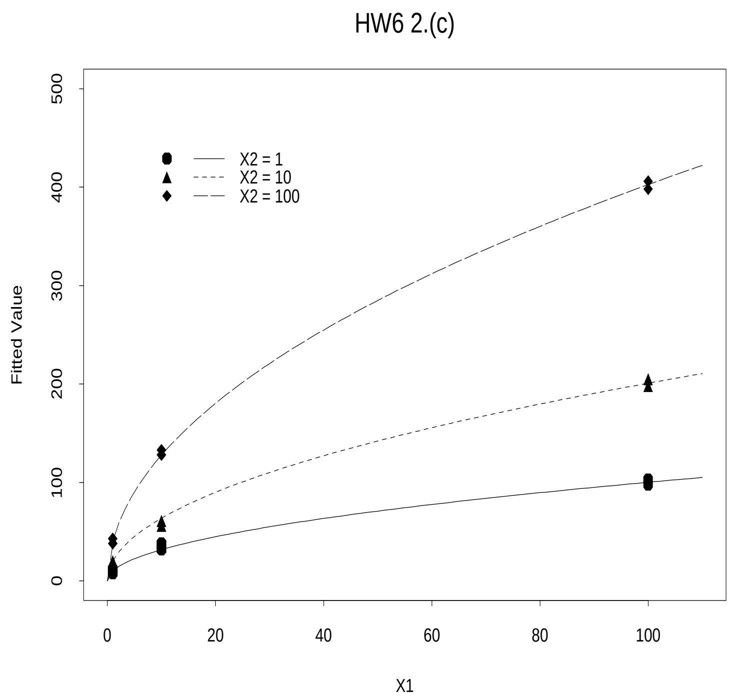
   (b) Using the starting values from part (a), find the least squares estimates of the parameters in the non-linear model.

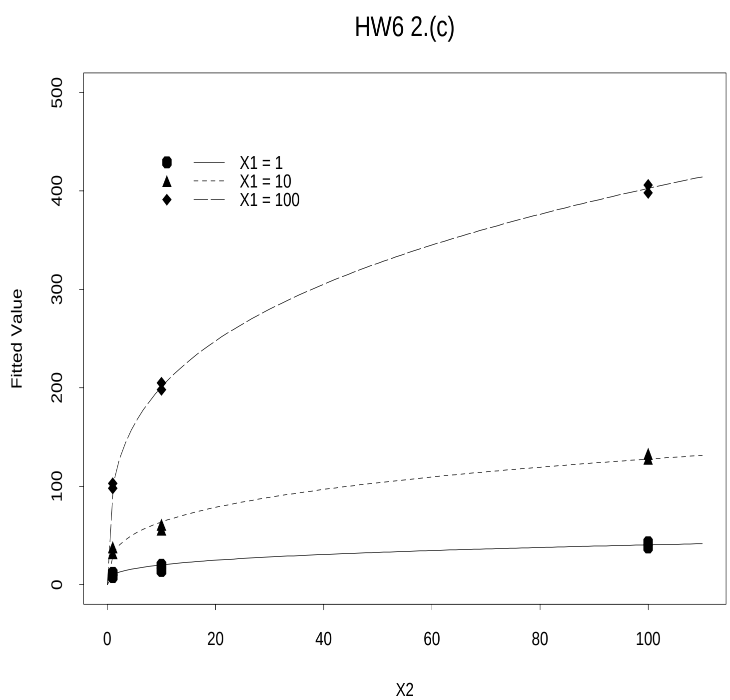
```
hw9.2.nls <- nls(formula = Y ~ b0*(X1^b1)*(X2^b2),
                   data = pnonlin,
                  start = start.val,
                  trace = T)
> summary(hw9.2.nls)

Formula: Y ~ b0 * (X1^b1) * (X2^b2)

Parameters:
       Value Std. Error t value
b0 10.079600 0.39817000 25.3147
b1  0.498710 0.00780672 63.8821
b2  0.301989 0.00485308 62.2262

Residual standard error: 4.19081 on 15 degrees of freedom
```

$$\Rightarrow \quad \beta_0 = 10.1,\ \beta_1 = 0.499, \text{ and } \beta_2 = 0.302$$

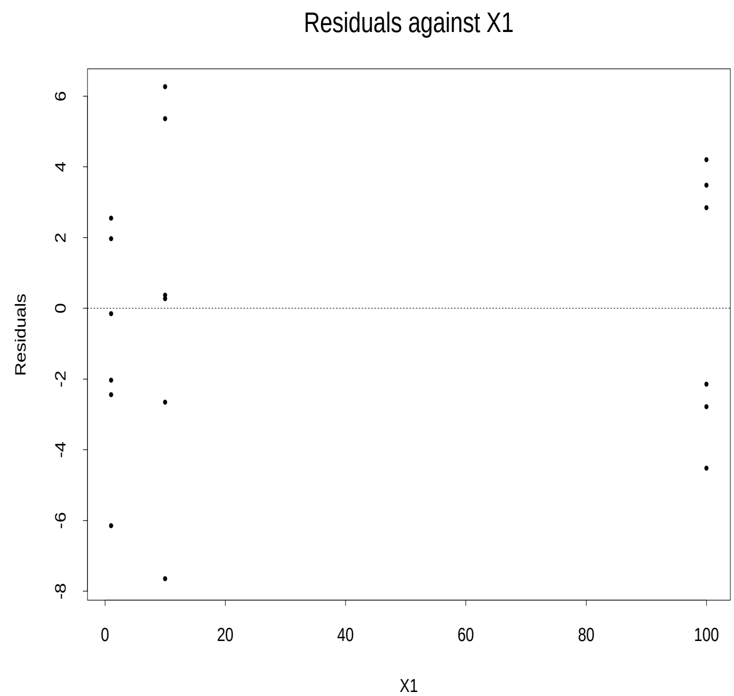
(c) Make a plot showing the observed data and the estimated surface for the mean yield. Does this model appear to be adequate?
Yes. The following plots include a 3-dimensional plot of the estimated surface and profile plots that more clearly show how close the observed responses are to the estimated surface.

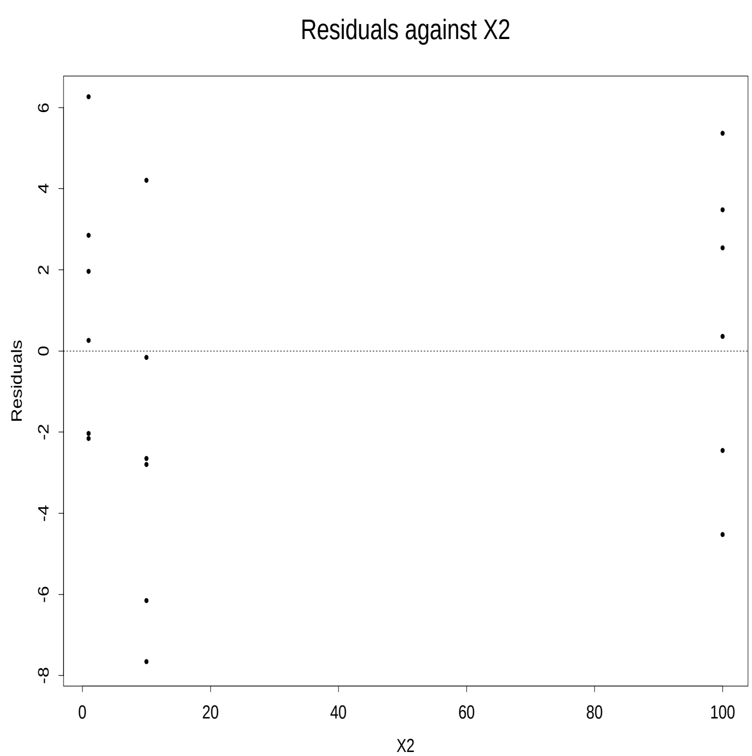



HW6 2.(c)
X2 = 1
X2 = 10
X2 = 100
Fitted Value
X1
0
100
200
300
400
500
0
20
40
60
80
100


HW6 2.(c)
X1 = 1
X1 = 10
X1 = 100
Fitted Value
X2
0
100
200
300
400
500
0
20
40
60
80
100

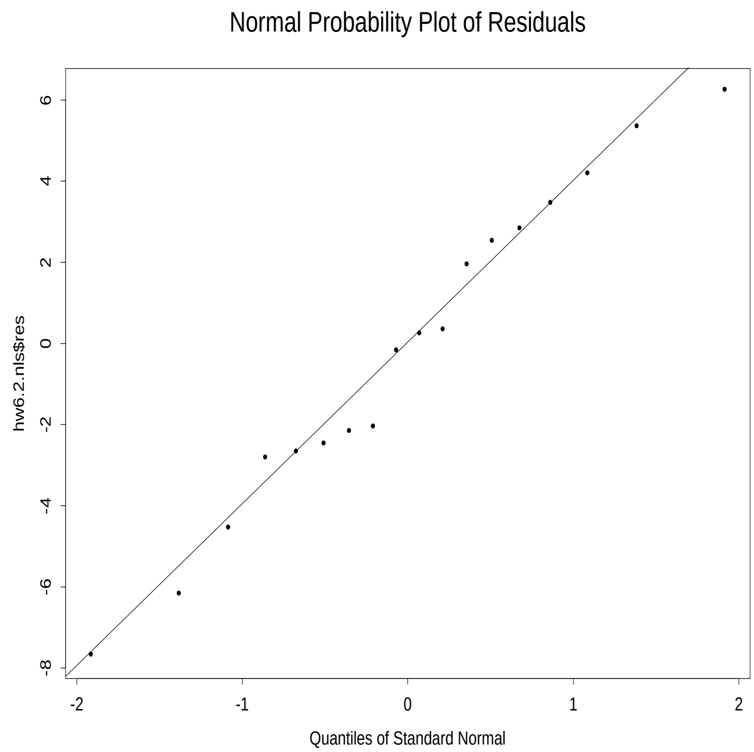
(d) Plot the residuals against the values of x_1 and x_2, on seperate plots. What do these plots indicate?
The plots do not reveal any serious departures from the assumption of homogeneous error variances. The plot of the residuals against x_2 suggests that there may be some effect of x_2, in the lower range of values, that is not fully described by the model.





(e) Create a normal probability plot for the residuals. What does this plot suggest?

This plot suggests that an assumption of normally distributed errors may be a reasonable approximation. The largest positive residual is somewhat closer to zero than expected.



(f) Assume that the proposed non linear model is appropriate and use the large sample distributional properties of the least squares estimators to do the following.

(i) Construct 95% confidence intervals for β_0, β_1, and β_2.

```
> # Approximate profile likelihood confidence intervals.
> rbind(b0 = confint.nls(hw9.2.nls,parm="b0"),
+       b1 = confint.nls(hw9.2.nls,parm="b1"),
+       b2 = confint.nls(hw9.2.nls,parm="b2"))  # Lecture Note 11.15
        [,1]       [,2]
b0 9.2548676 10.9461031
b1 0.4824410  0.5156617
b2 0.2917425  0.3124756

> # Compute confidence intervals for parameters using
> # standard large sample normal theory

> est.b(hw9.2.nls) # Lecture Note 11.16
            b      stderr       low       high
b0 10.0795655 0.398169756 9.2308868 10.9282442
b1  0.4987098 0.007806722 0.4820702  0.5153495
b2  0.3019886 0.004853080 0.2916445  0.3123327
```

These two methods provide very similar confidence intervals in this case.

(ii) Construct a 95% confidence interval for the mean yield when $x_1 = 50$ and $x_2 = 60$.

```
# first partial derviatives of the mean function needed to
# apply the delta method

yield.pds <- deriv3(~ b0*x1^b1*x2^b2,
```

```
                c("b0","b1","b2"), function(x1,x2, b0, b1, b2) NULL)

        #  Compute the estimates of the times and the large sample
        #  covariance matrix and standard errors

        est.yield<- function(x1, x2, obj,level=.95)
        {      b <- coef(obj)
             tmp <- yield.pds(x1, x2, b["b0"], b["b1"], b["b2"])
           y.hat <- as.vector(tmp)
               G <- matrix(attr(tmp, "gradient"),nrow=1)
               v <- (G %*% vcov(obj) %*% t(G))
               n <- length(obj$fitted.values) - length(obj$parameters)
             low <- y.hat - qt(1 - (1 - level)/2, n)*sqrt(v)
            high <- y.hat + qt(1 - (1 - level)/2, n)*sqrt(v)
          result <- c(y.hat = y.hat, std.err = sqrt(v), lower.bound=low,
upper.bound=high)
          return(G.hat=G, V.hat= vcov(obj), S2 = v, result=result)
        }

        > est.yield(x1=50,x2=60, hw6.2.nls)
        $G.hat:    <-- First partial derivatives of the estimated surface
at (X1,X2)=(50,60).
                [,1]     [,2]     [,3]
        [1,] 24.2256 955.2514 999.7713

        $V.hat:    <-- covariance matrix for the parameter estimates.
                       b0             b1             b2
        b0  0.1585391548 -2.675067e-03 -9.377983e-04
        b1 -0.0026750667  6.094491e-05  1.503385e-13
        b2 -0.0009377983  1.503385e-13  2.355238e-05

        $S2:       <-- variance for the estimated mean velocity.
                 [,1]
        [1,] 2.960183

        $result:
            y.hat  std.err lower.bound upper.bound  <-- Approximate 95%
Confidence Interval.
         244.1835 1.720518    240.5163    247.8507
```

(iii) Test the null hypothesis $H_0: \ \beta_1 = \beta_2$ versus $H_A: \ \beta_1 \neq \beta_2$. Indicate how your test was performed. State your conclusion.

For any estimable function $C\underset{\sim}{b}$,

$$C\underset{\sim}{b} \dot{\sim} \mathrm{N}(C\underset{\sim}{b},\ CV_{\underset{\sim}{b}}C^T)\,.$$

Let $C = [0 \ \ 1 \ -1]$, then

$$b_1 - b_2 \dot{\sim} \mathrm{N}(\beta_1 - \beta_2,\ CV_{\underset{\sim}{b}}C^T)\,.$$

Therefore, under the null hypothesis,

$$\frac{b_1 - b_2}{\sqrt{C\hat{V}_{\underset{\sim}{b}}C^T}} \dot{\sim} t_{n-p}$$

Reject H_0 if $|\,t\,| > t_{n-p,\alpha/2}$.
This can be computed with Splus as follows:

```
              C <- matrix(c(0,1,-1),nrow=1)
              b <- coef(hw9.2.nls)
           V.Cb <-  C %*% vcov(hw9.2.nls) %*% t(C)
         t.stat <- (C %*% b) / sqrt(V.Cb)
             df <- length(hw9.2.nls$fitted) - length(hw9.2.nls$par)
          p.val <- 2*(1- pt(abs(t.stat),df))
         result <- c(b1.minus.b2= C %*% b, std.err=sqrt(V.Cb),
t.stat=t.stat,df=df,p.val=p.val)

        > result
             b1-b2     std.err   t.stat df        p.val
         0.1967212 0.009192241 21.40079 15 1.180611e-12
```

Reject the null hypothesis $H_0 : \beta_1 = \beta_2$. β_1 is significantly different from β_2.

A second method uses an approximate F-test. From part (b), the sum of squared residuals for fitting the model

$$y_i \;=\; \beta_0 x_{1i}^{\beta_1} x_{2i}^{\beta_2} + \epsilon_i$$

is $RSS_1 = 263.4433$. Fit the more restricted model

$$y_i \;=\; \alpha_0 \left(x_{1i} x_{2i}\right)^{\alpha_1} + \epsilon_i \,.$$

The residual sum of squares for this model is $RSS_2 = 9331.624$. Compute an approximate F-test for comparing the fit of nested models.

$$F \;=\; \frac{(RSS_2 - RSS_1)\,/\,1}{RSS_1/(n-3)} \;=\; 516.3263 \quad \text{on (1,n-3) d.f.}$$

Then p-value < 0.0001. β_1 is significantly different from β_2.

```
        # Fit a restricted model, Y ~ a0 * (X1*X2)^a1, to calculate RSS.2

        X1X2 <- pnonlin$X1 * pnonlin$X2
        lm.out <- lm(log(pnonlin$Y) ~ log(X1X2))
        start.val <- coef(lm.out)             # Note that Intercept = log(a0)
        start.val[1] <- exp(start.val[1])     # Therefore, a0 =
exp(Intercept)
        names(start.val) <- c("a0","a1")      # specify the name of each
starting
                                              # value ( to use this in nls())
        # Calculate F-statistics.

         RSS.1 <- deviance(hw6.2.nls)
         RSS.2 <- deviance(nls(formula = Y ~ a0*(X1*X2)^a1,start =
start.val,data=pnonlin))
            df <- length(hw6.2.nls$fit) - length(hw6.2.nls$par)
        F.stat <- (RSS.2-RSS.1)/(RSS.1/df)
         p.val <- 1-pf(F.stat,1,df)

        > c(RSS.1=RSS.1, RSS.2=RSS.2,NDF=1,DDF=
df,"F-value"=F.stat,"p-value"=p.val)
            RSS.1    RSS.2 NDF DDF  F-value     p-value
         263.4433 9331.624   1  15 516.3263 4.92717e-13
```

3. In an enzyme kinetics study, the velocity of a reaction (y) is expected to be related to the concentration (x) of a substance as follows:

$$y_i = \frac{\beta_0 x_i}{\beta_1 + x_i} + \epsilon_i$$

(a) To obtain starting values, ignore that the additive error and note that

$$y_i^{-1} \approx \frac{1}{\beta_0} + \frac{\beta_1}{\beta_0} x_i^{-1}.$$

Use least squares regression analysis to obtain estimates of β_0^{-1} and β_1/β_0, and list your starting values for β_0 and β_1.

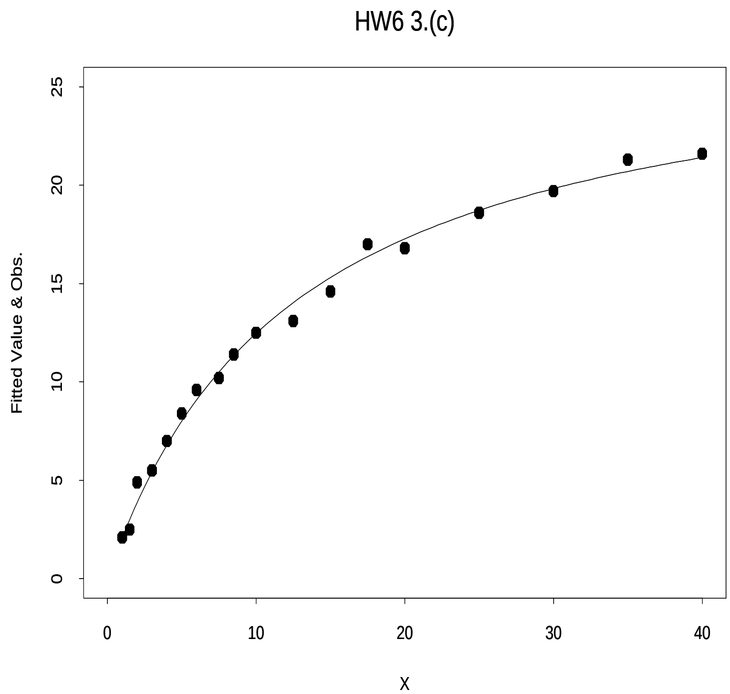
```
       kinetics <- read.table("kinetics.dat",col.names=c("Y","X"))
       YY <- 1/kinetics$Y
       XX <- 1/kinetics$X
       lm.out <- lm(YY ~ XX)
       #lm.out <- lm(1/Y ~ 1/X, data=kinetics)   # This is not right;
this fits
                                                # 1/Y = a0 + a1*(X %in% 1)
                                                # i.e. 1/Y = a0 + a1*X

       start.val <- coef(lm.out)                # Note that Intercept = 1/b0
       start.val[1] <- 1/start.val[1]           # Therefore, b0 =
1/Intercept
       start.val[2] <- start.val[1]*start.val[2] # b1 = b0 * (b1/b0)
       names(start.val) <- c("b0","b1")         # specify the name of
each starting
                                                # value ( to use this in
nls())
       start.val
              b0       b1
        29.62201 13.44881      <--- starting values
```

(b) Using the starting values from part (a), find the least squares estimates of β_0 and β_1 in the orginal non-linear model.

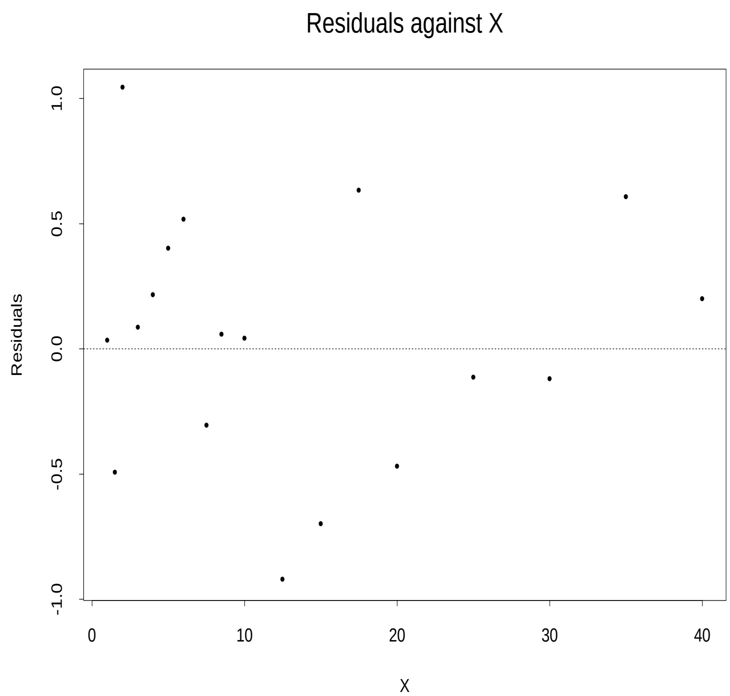
```
       hw9.3.nls <- nls(formula = Y ~ b0*X/(b1+X),
                          data = kinetics,
                         start = start.val,
                         trace = T)

       > summary(hw9.3.nls)

       Formula: Y ~ (b0 * X)/(b1 + X)

       Parameters:
            Value Std. Error t value
       b0 28.1379   0.728023 38.6497
       b1 12.5753   0.763112 16.4790

       Residual standard error: 0.518548 on 16 degrees of freedom
```

$$\Rightarrow \quad \beta_0 = 28.1 \quad \text{and} \quad \beta_1 = 12.6$$

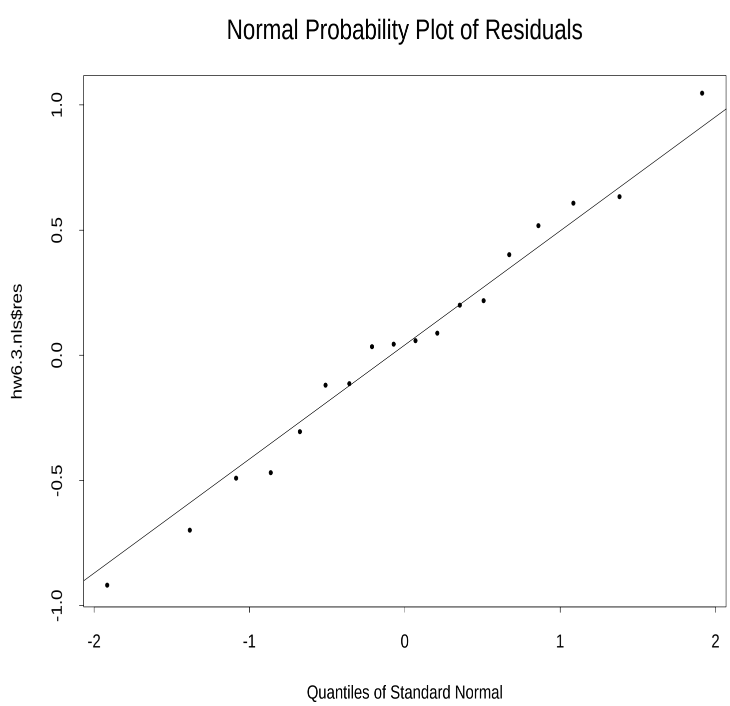
(c) Make a plot showing the estimated velocity and the observed data. Does the model appear to be adequate?
Yes.



(d) Plot the residuals against the concentration(x). What does this plot suggest?
There is no obvious trend in these residuals to suggest that the model is not adequate. There is no evidence to refute the assumption of homogeneous error variances across values of x_1.



(e) Create a normal probability plot for the residuals. What does this plot suggest?
An assumption of normally distributed random errors seems to be reasonable.



(f) Construct 95% confidence intervals for β_0 and β_1.

```
> rbind(b0 = confint.nls(hw6.3.nls,parm="b0"),
+       b1 = confint.nls(hw6.3.nls,parm="b1")) # Lecture Note 11.15
       [,1]     [,2]
b0 26.64508 29.79735
b1 11.03554 14.35170

> est.b(hw6.3.nls)  # Lecture Note 11.16
          b    stderr      low     high
b0 28.13786 0.7280235 26.59452 29.68120
b1 12.57534 0.7631121 10.95762 14.19307
```

(g) Estimate the concentration (x) at which the expected velocity of the reaction is 25. Report a 95% confidence interval for the concentration.

$$y_i^{-1} \;=\; \frac{1}{\beta_0} + \frac{\beta_1}{\beta_0} x_i^{-1}. \quad \Rightarrow \quad x_i \;=\; \frac{\beta_1}{\beta_0\left(\frac{1}{y_i} - \frac{1}{\beta_0}\right)} \;=\; \frac{\beta_1}{\frac{\beta_0}{y_i} - 1}$$

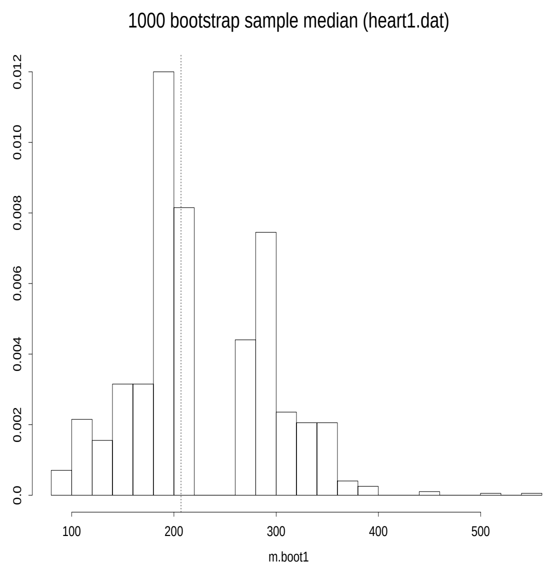
```
# first partial derviatives of the mean function needed to
# apply the delta method

x.pds <- deriv(~ b1/(b0/y -1),
                c("b0","b1"), function(y, b0, b1) NULL)

#  Compute the estimates of the times and the large sample
#  covariance matrix and standard errors

est.x <- function(y, obj,level=.95)
{      b <- coef(obj)
     tmp <- x.pds(y, b["b0"], b["b1"])
   x.hat <- as.vector(tmp)
       G <- matrix(attr(tmp, "gradient"),nrow=1)
       v <- (G %*% vcov(obj) %*% t(G))
       n <- length(obj$fitted.values) - length(obj$parameters)
     low <- x.hat - qt(1 - (1 - level)/2, n)*sqrt(v)
    high <- x.hat + qt(1 - (1 - level)/2, n)*sqrt(v)
  result <- c(x.hat = x.hat, std.err = sqrt(v), lower.bound=low,
upper.bound=high)
  return(G.hat=G, V.hat= vcov(obj), S2 = v, result=result)
}

> est.x(y=25, hw6.3.nls)
$G.hat:
   [,1]     [,2]
[1,] -31.94452 7.969379

$V.hat:
          b0        b1
b0 0.5299500 0.5202788
b1 0.5202788 0.5822457

$S2:
         [,1]
[1,] 312.8647

$result:
```

```
      x.hat  std.err lower.bound upper.bound
   100.2102 17.68798    62.71335     137.707
```

4. The bootstrap provides a reasonably good estimator for the standard error of the sample median, as well as a confidence interval for the median, without any knowledge of the c.d.f. $F(x)$. Basically, the bootstrap replaces the derivation of asymptotic theoretical results with simulation.

   (a) Compute the sample median of the survival times. Use bootstrap methods to obtain a standard error for the sample median and 95% percentile confidence limits for the population median. Also compute bias corrected accelerated 95% confidence limits. (Use 1,000 bootstrap samples.)

```
heart1 <- c(scan("heart1.dat")) # read the data

nboot <- 1000   # size of bootstrap samples

> median(heart1)  # sample median
[1] 207

m.boot1 <- bootstrap(heart1, nboot=nboot, theta=median)$thetastar
```

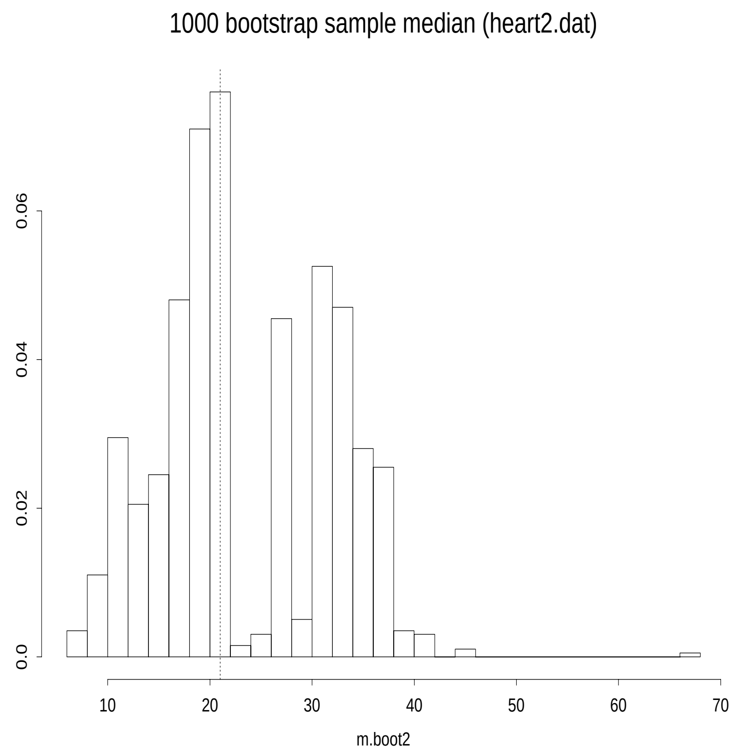



```
> sqrt(var(m.boot1)) # standard error for the sample median
[1] 68.78402

# construct 95% percentile confidence limits for the population median
m.boot1 <-sort(m.boot1)
alpha <- .05
kl <- floor((nboot+1)*alpha/2)
ku <- nboot+1 - kl
mlower <- round(m.boot1[kl],digits=4)
mupper <- round(m.boot1[ku],digits=4)

> c(kl,ku)
[1]  25 976
> c("Lower Limit"=mlower, "Upper Limit"=mupper)
 Lower Limit Upper Limit
        109         342
```

```
bca.interval <- bcanon(x=heart1,nboot=nboot,theta=median,alpha=c(.025,.975))

> bca.interval$confpoints
     alpha bca point
[1,] 0.025       100
[2,] 0.975       340
```

(b) Repeat part(a) for the survival times for a similar set of 34 patients that did not receive heart transplants.

```
heart2 <- c(scan("heart2.dat")) # read the data

> median(heart2)  # sample median
[1] 21

m.boot2 <- bootstrap(heart2, nboot=nboot, theta=median)$thetastar
```




```
> sqrt(var(m.boot2)) # standard error for the sample median
[1] 8.396851

# construct 95% percentile confidence limits for the population median
m.boot2 <-sort(m.boot2)
mlower <- round(m.boot2[kl],digits=4)
mupper <- round(m.boot2[ku],digits=4)
> c("Lower Limit"=mlower, "Upper Limit"=mupper)
 Lower Limit Upper Limit
          9          37

# Compute the bias corrected accelerated intervals
bca.interval <- bcanon(x=heart2,nboot=nboot,theta=median,alpha=c(.025,.975))
```

```
> bca.interval$acc
[1] NA
> bca.interval$confpoints
     alpha bca point
[1,] 0.025        NA
[2,] 0.975        NA
```

The BCA confidence interval method does not work in part (b) of problem 1. This happens because the accelation constant($\hat{a}$) cannot be computed. This is a feature of the data (and the sample median). Note that the two middle values in this data set are both 21. When the program uses a jackknife method (leave one case at a time out of the data set and compute the statistic (sample median) with the remaining n-1 cases) it always gets a value of 21 for the resulting sample median. Hence, the sum of squares in the denominator of the formula for the accelaration constant is zero (so is the numerator) and the algorithm refuses to divide by zero and records the value of the accelaration constant as "NA" (to indicate a missing value). The BCA confidence intervals do not seem to be a very good idea for creating a confidence interval for a median.

(c) Use bootstrap methods to test the null hypothesis that the median survival time for patients that received heart transplants is equal to the median survival time for similar patients that do not receive heart transplants. Describe your procedure and state your conclusion.

(1) take independnt bootstrap samples from the heart1 and heart2 data files and calculate sample medians. Call these bootstrapped medians $\theta_{1,b}$ and $\theta_{2,b}$.

(2) calculate $\theta_{\text{diff,b}} = \theta_{1,b} - \theta_{2,b}$.

(3) repeat steps (1) – (2) B times. (e.g. $B = 1000$)

(4) order the bootstrapped values from smallest to largest.

$$\theta_{\text{diff},(1)}, \theta_{\text{diff},(2)}, \cdots, \theta_{\text{diff},(\text{B})},$$

(5) An approximate $(1-\alpha)\%$ confidence interval for θ_{diff} (the difference between median survival times) is

$$\left[\theta_{\text{diff},(K_L)}, \theta_{\text{diff},(K_U)}\right]$$

```
diff.boot <- m.boot1 - m.boot2
diff.boot <- sort(diff.boot)
diff.lower <- round(diff.boot[kl],digits=4)
diff.upper <- round(diff.boot[ku],digits=4)

> c("Lower Limit"=diff.lower, "Upper Limit"=diff.upper)
 Lower Limit Upper Limit
        91.5         305
```

The approximate 95% confidence interval does not contain 0. The median survival time for patients that received heart transplants is significantly greater than the median survival time for patients that did not receive heart transplants.

5. Consider the model you fit to the chemical process data in Problem 2.

$$\begin{aligned} y_i &= \beta_0 x_{1i}^{\beta_1} x_{2i}^{\beta_2} + \epsilon_i \\ &\text{where } \epsilon_i \sim NID(0, \sigma^2)\ ,\ i = 1, 2, \cdots, 18. \end{aligned}$$

(a) Use bootstrap methods to solve Part (f) of that problem, i.e. construct confidence intervals for the parameters.

```
pnonlin <- read.table("HW9/pnonlin.dat",col.names=c("Y","X1","X2"))
# Use the residuals from the Splus object created
# with the S-plus code for problem 2
s2 <- sum(resid(hw9.2.nls)^2)/(18-3)
centered.res <- scale(resid(hw9.2.nls),center=T,scale=F)
rescaled.res <- centered.res * sqrt(s2/var(c(centered.res)))

# Now, sum of centered.res and rescaled.res is zero.
round( c(sum(resid(hw9.2.nls)), sum(centered.res), sum(rescaled.res)), 6 )

# variance of rescaled.res is equal to the variance estimate of the model.
c(s2, var(centered.res), var(rescaled.res) )

# define a function to calculate b0,b1,b2 and y.hat for each replicate.
hw9.5.boot.f1 <- function( id, x1=100, x2=50 ){
    newdata <-
data.frame(Y=pnonlin$Y[id],X1=pnonlin$X1[id],X2=pnonlin$X2[id])
    nls.out <- nls(formula = Y ~ b0*(X1^b1)*(X2^b2),
```

```
                   data = newdata, start = coef(hw9.2.nls))
    beta0 <- coef(nls.out)[1]
    beta1 <- coef(nls.out)[2]
    beta2 <- coef(nls.out)[3]
    y.hat <- beta0*(x1^beta1)*(x2^beta2)
    return(c(beta0=beta0,beta1=beta1,beta2=beta2,y.hat=y.hat))
}
hw9.5.boot.f2 <- function( resids, x1=100, x2=50 ){
    assign("new.Y", fitted(hw9.2.nls)+resids, frame=1)
    nls.out <- nls(formula = new.Y ~ b0*(X1^b1)*(X2^b2),
                   data = pnonlin, start = coef(hw9.2.nls))
    beta0 <- coef(nls.out)[1]
    beta1 <- coef(nls.out)[2]
    beta2 <- coef(nls.out)[3]
    y.hat <- beta0*(x1^beta1)*(x2^beta2)
    return(c(beta0=beta0,beta1=beta1,beta2=beta2,y.hat=y.hat))
}
# bootstrapping
# (a)-i Resample cases with replacement
hw9.5.boot1 <- bootstrap(data=1:nrow(pnonlin), B=1000,
statistic=hw9.5.boot.f1 )
hw9.5.boot2 <- bootstrap(data=centered.res,B=1000,statistic=hw9.5.boot.f2)
hw9.5.boot3 <- bootstrap(data=rescaled.res,B=1000,statistic=hw9.5.boot.f2)

names(hw9.5.boot1)
dim(hw9.5.boot1$replicates)  # 1000 x 4 matrix
# Each row of hw9.5.boot$replicates contains 1000 bootstrap samples
# for b0,b1,b2 and y.hat respectively.
get.bootconfint <- function( bootobj, alpha=.05 ){
    nboot <- bootobj$B
    boot.b0 <- sort(bootobj$replicates[,1])   # B bootstrap samples for b0
    boot.b1 <- sort(bootobj$replicates[,2])   # B bootstrap samples for b1
    boot.b2 <- sort(bootobj$replicates[,3])   # B bootstrap samples for b2
    boot.yhat <- sort(bootobj$replicates[,4]) # B bootstrap samples for y.hat
    kl <- floor((nboot+1)*alpha/2)
    ku <- nboot+1 - kl
    result <- rbind( c("Lower Limit"=boot.b0[kl],"Upper Limit"=boot.b0[ku]),
                     c("Lower Limit"=boot.b1[kl],"Upper Limit"=boot.b1[ku]),
                     c("Lower Limit"=boot.b2[kl],"Upper Limit"=boot.b2[ku]),
                     c("Lower Limit"=boot.yhat[kl],"Upper
Limit"=boot.yhat[ku]))
    row.names(result)<- c("b0","b1","b2","y.hat")
    result
}

> get.bootconfint( bootobj=hw9.5.boot1 ) #(i)
      Lower.Limit Upper.Limit
   b0   8.8612263  10.7739221
   b1   0.4857380   0.5263606
   b2   0.2934535   0.3196113
y.hat 322.2808683 339.5373815
> get.bootconfint( bootobj=hw9.5.boot2 ) #(ii)
      Lower.Limit Upper.Limit
   b0   9.3812374  10.7308708
```

```
   b1   0.4855504   0.5128529
   b2   0.2935353   0.3104085
y.hat 322.8196396 330.1081198
> get.bootconfint( bootobj=hw9.5.boot3 ) #(iii)
      Lower.Limit Upper.Limit
   b0   9.3250777  10.8731550
   b1   0.4838211   0.5139061
   b2   0.2930917   0.3110945
y.hat 322.3038933 330.3749005
```

(b) Construct a 95% confidence interval for the expected yield of the process when $x_1 = 100$ and $x_2 = 50$.

(c) Construct an approximate 95% prediction interval for the realization of the yield when $x_1 = 100$ and $x_2 = 50$.

Large sample normal confidence interval and prediction interval for for the realization of the yield when $x_1 = 100$ and $x_2 = 50$ are computed as follows.

```
confint.5b <- est.yield(x1=100,x2=50, obj=hw9.2.nls)[[4]]
yhat <- confint.5b["y.hat"]
pred.err <- sqrt(confint.5b["std.err"]^2+s2) # add mse to the std error
predint.5c <- c( yhat, pred.err,
                 yhat - qt(1-.05/2, n)* pred.err,
                 yhat + qt(1-.05/2, n)* pred.err )
result <- rbind( confint.5b, predint.5c )
dimnames(result) <- list( c("5(b):conf.int", "5(c):pred.int"),
                   c("y.hat","std.err","lower.bound","upper.bound") )
> result
                y.hat  std.err lower.bound upper.bound
5(b):conf.int 326.536 2.092765    322.0754    330.9966
5(c):pred.int 326.536 4.684288    316.6947    336.3773
```

Bootstrap prediction intervals using centered and centered and rescaled residuals are given below.

```
n <- nrow(pnonlin)
nboot <- length(length(hw9.5.boot2$replicates[,4]))
pred.boot2 <- hw9.5.boot2$replicates[,4] +
                centered.res[sample(n, size=nboot, replace=T)]
pred.boot3 <- hw9.5.boot3$replicates[,4] +
                rescaled.res[sample(n, size=nboot, replace=T)]


> rbind( quantile( pred.boot2, c(.0, .025, .5, .975, 1) ),
+        quantile( pred.boot3, c(.0, .025, .5, .975, 1) ) )

         0.0%     2.5%    50.0%    97.5%   100.0%
[1,] 322.3229 325.1030 328.7789 332.2324 334.0550
[2,] 317.9855 320.3341 324.7133 328.3916 330.0262
```

Note that it would be better to studentize the residuals before they are rescaled in the manner shown above, but it may not change the results very much.

Compare the bootstrap intervals to the large sample normal intervals.

```
                              Comparisons (95% confidence intervals)
                              --------------------------------------
```

	Large Sample Normal approx.		Bootstrap	
	Lower	Upper	Lower	Upper
b0	9.2	10.9	9.4	10.8
b1	0.482	0.515	0.485	0.514
b2	0.292	0.312	0.293	0.310
yhat	322.1	330.4	322.8	330.1

As in the analysis of the stormer data given in the lectures, and in Venables and Ripley, the bootstrap confidence intervals tend to be shorter than confidence intervals based on the large sample normal approximation to the distribution of the parameter estimates. This occurs because residuals exhibit less variation than the actual random errors.