# Chapter 7(Eighth Edition): 7.1, 7.2, 7.4, 7.13, 7.24, 7.25

*Jeremy Ling & Emmanuel Mejia*

*May 1, 2018*

```
library(car)
library("gplots")
```

## 7.1

**Consider the experiment described in Problem 6.1. Analyze this experiment assuming that each replicate represents a block of a single production shift.**

```
# defining coded
coded=function(x) #a function to code variable x
{
  ifelse(x=="+", 1, -1)
}

# creating data table
factorA = rep(c("-","+","-","+","-","+","-","+"), times = 3)
factorB = rep(c("-","-","+","+","-","-","+","+"), times = 3)
factorC = rep(c("-","-","-","-","+","+","+","+"), times = 3)
Rep = rep(c("I", "II", "III"), each = 8)
yield = c(22,32,35,55,44,40,60,39,31,43,34,47,45,37,50,41,25,29,50,46,38,36,54,47)

#dataframe
cutting.speed.long = data.frame(factorA, factorB, factorC, Rep, yield)

cutting.aov = aov(yield~Rep+factorA*factorB*factorC, cutting.speed.long)
cutting.aov.og = aov(yield~factorA*factorB*factorC, cutting.speed.long)
summary(cutting.aov); summary(cutting.aov.og)
```

```
##                         Df Sum Sq Mean Sq F value   Pr(>F)
## Rep                      2    0.6     0.3   0.008 0.991571
## factorA                  1    0.7     0.7   0.019 0.891320
## factorB                  1  770.7   770.7  22.381 0.000322 ***
## factorC                  1  280.2   280.2   8.136 0.012789 *
## factorA:factorB          1   16.7    16.7   0.484 0.497998
## factorA:factorC          1  468.2   468.2  13.596 0.002438 **
## factorB:factorC          1   48.2    48.2   1.399 0.256623
## factorA:factorB:factorC  1   28.2    28.2   0.818 0.381072
## Residuals               14  482.1    34.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##                         Df Sum Sq Mean Sq F value   Pr(>F)
## factorA                  1    0.7     0.7   0.022 0.883680
## factorB                  1  770.7   770.7  25.547 0.000117 ***
## factorC                  1  280.2   280.2   9.287 0.007679 **
## factorA:factorB          1   16.7    16.7   0.552 0.468078
## factorA:factorC          1  468.2   468.2  15.519 0.001172 **
```

```
## factorB:factorC          1   48.2    48.2   1.597 0.224475
## factorA:factorB:factorC  1   28.2    28.2   0.934 0.348282
## Residuals               16  482.7    30.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cutting.lm = lm(yield~Rep+factorA*factorB*factorC, cutting.speed.long)
cutting.lm.og = lm(yield~factorA*factorB*factorC, cutting.speed.long)
summary(cutting.lm); summary(cutting.lm.og)
```

```
##
## Call:
## lm(formula = yield ~ Rep + factorA * factorB * factorC, data = cutting.speed.long)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.833 -3.542 -1.146  3.083 10.542
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   26.042      3.788   6.875 7.62e-06 ***
## RepII                          0.125      2.934   0.043  0.96662
## RepIII                        -0.250      2.934  -0.085  0.93330
## factorA+                       8.667      4.791   1.809  0.09200 .
## factorB+                      13.667      4.791   2.852  0.01279 *
## factorC+                      16.333      4.791   3.409  0.00424 **
## factorA+:factorB+              1.000      6.776   0.148  0.88478
## factorA+:factorC+            -13.333      6.776  -1.968  0.06923 .
## factorB+:factorC+             -1.333      6.776  -0.197  0.84683
## factorA+:factorB+:factorC+    -8.667      9.583  -0.904  0.38107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.868 on 14 degrees of freedom
## Multiple R-squared:  0.7699, Adjusted R-squared:  0.622
## F-statistic: 5.206 on 9 and 14 DF,  p-value: 0.003168
```

```
##
## Call:
## lm(formula = yield ~ factorA * factorB * factorC, data = cutting.speed.long)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.667 -3.500 -1.167  3.167 10.333
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   26.000      3.171   8.199 4.02e-07 ***
## factorA+                       8.667      4.485   1.933  0.07119 .
## factorB+                      13.667      4.485   3.048  0.00768 **
## factorC+                      16.333      4.485   3.642  0.00219 **
## factorA+:factorB+              1.000      6.342   0.158  0.87668
## factorA+:factorC+            -13.333      6.342  -2.102  0.05171 .
## factorB+:factorC+             -1.333      6.342  -0.210  0.83614
## factorA+:factorB+:factorC+    -8.667      8.969  -0.966  0.34828
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.492 on 16 degrees of freedom
## Multiple R-squared:  0.7696, Adjusted R-squared:  0.6689
## F-statistic: 7.637 on 7 and 16 DF,  p-value: 0.0003977
```

We look at the analysis of variance we see that the block effect is insignificant because the p-value is small, but effect B, C, and AC are significant. We also make a comparison to the analysis of Variance without the blocking to see if there was a difference. In this case there is no difference at the coefficients of our effects.

## 7.2

**Consider the experiment described in Problem 6.5. Analyze this experiment assuming that each one of the four replicates represents a block.**

```
# creating data table
A <- rep(c("-","+","-","+"), times = 4)
B <- rep(c("-","-","+","+"), times = 4)
Rep <- rep(c("I","II","III","IV"), each = 4)
Vibes <- c(18.2, 27.2, 15.9, 41.0, 18.9, 24.0, 14.5, 43.9, 12.9, 22.4, 15.1, 36.3, 14.4, 22.5, 14.2, 39
router.long <- data.frame(A, B, Rep, Vibes)

# defining coded
coded=function(x) #a function to code variable x
{
  ifelse(x=="+", 1, -1)
}

# coding A and B
for (j in 1:2)
  router.long[, j]=as.numeric(coded(router.long[, j]))

#######
#router.long$Block=router.long$A * router.long$B
#router.lm = lm(Vibes ~ Block + A * B, router.long)
#router.lm.og = lm(Vibes ~ A * B, router.long)
#summary(router.lm); summary(router.lm.og)

router.aov = aov(Vibes ~ Rep + A * B, router.long)
router.aov.og = aov(Vibes ~ A * B, router.long)
summary(router.aov); summary(router.aov.og)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## Rep          3   44.4    14.8   4.864    0.028 *
## A            1 1107.2  1107.2 364.211 1.37e-08 ***
## B            1  227.3   227.3  74.753 1.18e-05 ***
## A:B          1  303.6   303.6  99.876 3.60e-06 ***
## Residuals    9   27.4     3.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## A            1 1107.2  1107.2  185.25 1.17e-08 ***
## B            1  227.3   227.3   38.02 4.83e-05 ***
```

```
## A:B          1  303.6   303.6   50.80 1.20e-05 ***
## Residuals   12   71.7     6.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After we analyze our variance we see that all main effects are significant including the Blocking effect because all p-values are small. Blocking the replicates will create a better model for our experiment.

## 7.4

**Consider the data from the first replicate of Problem 6.1. Suppose that these observations could not all be run using the same bar stock. Set up a design to run these observations in two blocks of four observations each with ABC confounded. Analyze the data.**

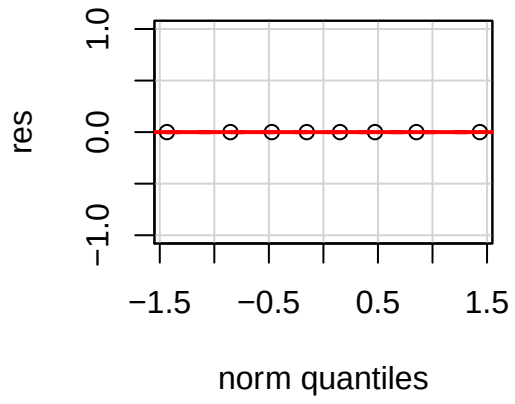
```r
# creating data table
A = rep(c("-","+","-","+","-","+","-","+"), times = 1)
B = rep(c("-","-","+","+","-","-","+","+"), times = 1)
C = rep(c("-","-","-","-","+","+","+","+"), times = 1)
Rep = rep(c("I"))
yield = c(22,32,35,55,44,40,60,39)

cutting.speed.short = data.frame(A, B, C, Rep, yield)

# defining coded
coded=function(x) #a function to code variable x
{
  ifelse(x=="+", 1, -1)
}

for (j in 1:3)
  cutting.speed.short[, j]=as.numeric(coded(cutting.speed.short[, j]))

cutting.speed.short$Block = cutting.speed.short$A * cutting.speed.short$B * cutting.speed.short$C

cutting.lm_7.4 = lm(yield ~ Block + A * B  * C, cutting.speed.short)#blocking
cutting.lm_7.4.og = lm(yield ~ A * B  * C, cutting.speed.short)#original
summary(cutting.lm_7.4); summary(cutting.lm_7.4.og)#comparing coefficients
```

```
##
## Call:
## lm(formula = yield ~ Block + A * B * C, data = cutting.speed.short)
##
## Residuals:
## ALL 8 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   40.875         NA      NA       NA
## Block         -3.375         NA      NA       NA
## A              0.625         NA      NA       NA
## B              6.375         NA      NA       NA
## C              4.875         NA      NA       NA
## A:B           -0.875         NA      NA       NA
## A:C           -6.875         NA      NA       NA
```

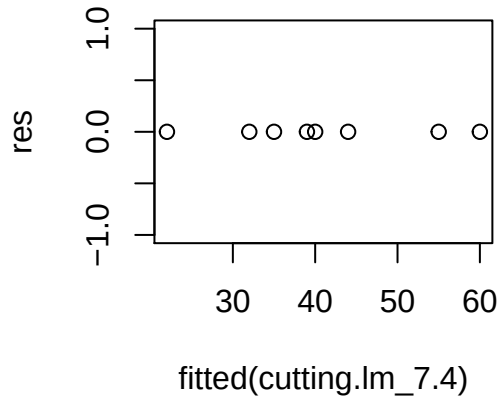
```
## B:C           -2.625         NA      NA       NA
## A:B:C             NA         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 7 and 0 DF,  p-value: NA
```

```
##
## Call:
## lm(formula = yield ~ A * B * C, data = cutting.speed.short)
##
## Residuals:
## ALL 8 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   40.875         NA      NA       NA
## A              0.625         NA      NA       NA
## B              6.375         NA      NA       NA
## C              4.875         NA      NA       NA
## A:B           -0.875         NA      NA       NA
## A:C           -6.875         NA      NA       NA
## B:C           -2.625         NA      NA       NA
## A:B:C         -3.375         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 7 and 0 DF,  p-value: NA
```

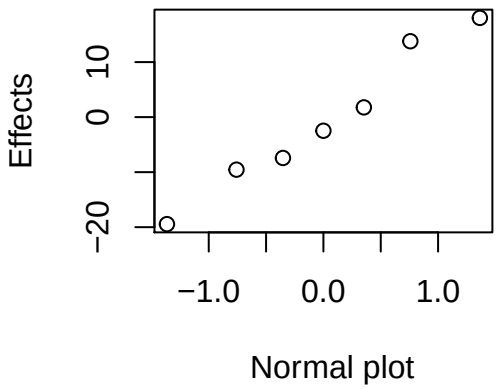
```
#residual analysis
res=cutting.speed.short$yield-fitted(cutting.lm_7.4)
qqPlot(res)
```



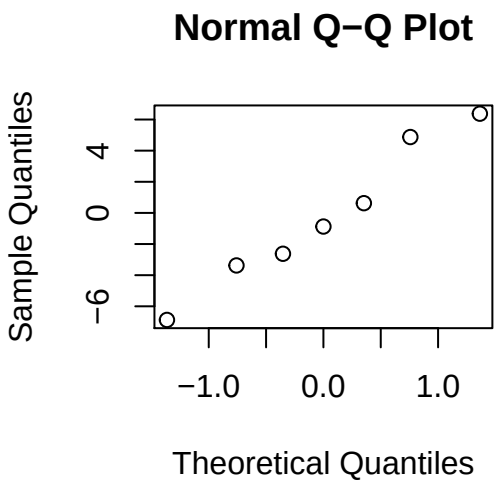

```
plot(fitted(cutting.lm_7.4), res)#bad model
```




```
#Reduced Model using probability plot
qqnorm(aov(yield ~ Block + A*B*C, cutting.speed.short), full=T,label=T)
```




```
coef=cutting.lm_7.4$coefficients[-1]
variables=names(coef)
plot=qqnorm(coef)
variables[identify(plot)]# B C AC
```

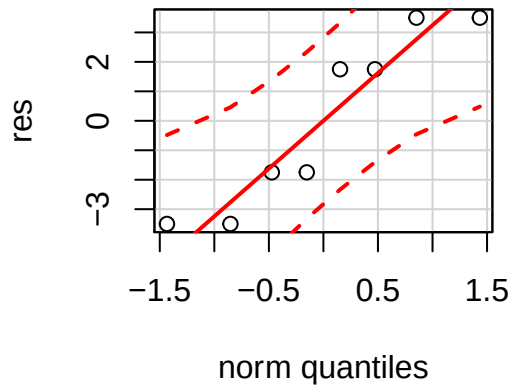



```
## character(0)
```

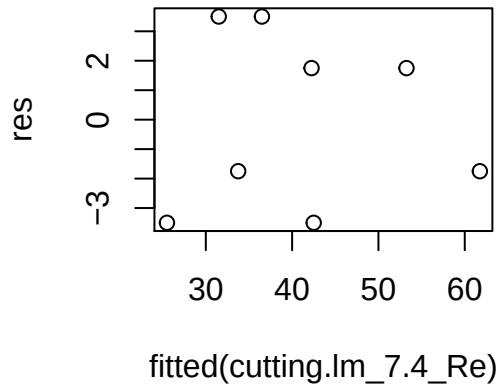
```
#New model
cutting.lm_7.4_Re = lm(yield ~ Block + A + B + C + A*C, cutting.speed.short)
summary(cutting.lm_7.4_Re)
```

```
##
## Call:
## lm(formula = yield ~ Block + A + B + C + A * C, data = cutting.speed.short)
##
## Residuals:
##     1     2     3     4     5     6     7     8
## -3.50 -1.75  3.50  1.75  1.75  3.50 -1.75 -3.50
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   40.875      1.957  20.891  0.00228 **
## Block         -3.375      1.957  -1.725  0.22668
## A              0.625      1.957   0.319  0.77967
## B              6.375      1.957   3.258  0.08268 .
## C              4.875      1.957   2.492  0.13032
## A:C           -6.875      1.957  -3.514  0.07232 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.534 on 2 degrees of freedom
## Multiple R-squared:  0.9416, Adjusted R-squared:  0.7956
## F-statistic:  6.45 on 5 and 2 DF,  p-value: 0.1397
```

```
#New Model Residual Analysis
res=cutting.speed.short$yield-fitted(cutting.lm_7.4_Re)
qqPlot(res)
```




```
plot(fitted(cutting.lm_7.4_Re), res) #better model
```




We begin by Blocing our main effects and we take a look at the coefficients of our model that includes blocking and the model that excludes blocking. Notice all coefficients are similar. To check if the model that includes the blocked data is a good model we quickly do a residual analysis and realize that our residuals are trending to a straight line, we do not have a good model, we must reduce it. Using the same method from chapter six, the probability plot, we have chosen main effects B, C, and interaction effects AC to be important effects. We apply this to a new model and check the residuals, no pattern and normality is well. The model with the Block Design data and main effects B, C, and interaction effect AC is a good model.

### 7.13

**Using the data from the $2^4$ design in Problem 6.22, construct and analyze a design in two blocks with ABCD confounded with blocks.**

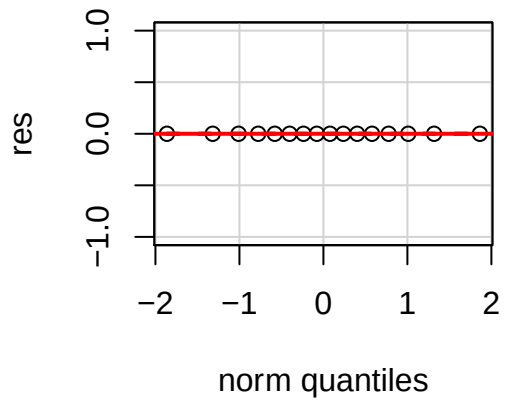
```r
Standard.Order = c(8,10,12,9,7,15,2,6,16,13,5,14,1,3,4,11)
Run.Order = c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)
A = Laser.Power = c(1,1,1,-1,-1,-1,1,1,1,-1,-1,1,-1,-1,1,-1)
B = Pulse.Freq = c(1,-1,1,-1,1,1,-1,-1,1,-1,-1,-1,-1,1,1,1)
C = Cell.Size = c(1,-1,-1,-1,1,1,-1,1,1,1,1,1,-1,-1,-1,-1)
D = Writing.Speed = c(-1,1,1,1,-1,1,-1,-1,1,1,-1,1,-1,-1,-1,1)
UEC = c(0.8,0.81,0.79,0.6,0.65,0.55,0.98,0.67,0.69,0.56,0.63,0.65,0.75,0.72,0.98,0.63)
error = data.frame(Standard.Order,Run.Order,A,B,C,D,UEC)

#Blocking#
error$Block=error$A * error$B * error$C * error$D
#Linear Model#
error.lm = lm(UEC ~ Block + A*B*C*D, error)#blocking
error.lm.og = lm(UEC ~ A*B*C*D, error)#regular
summary(error.lm); summary(error.lm.og)
```

```
##
## Call:
## lm(formula = UEC ~ Block + A * B * C * D, data = error)
##
## Residuals:
## ALL 16 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.71625         NA      NA       NA
## Block        0.00125         NA      NA       NA
## A            0.08000         NA      NA       NA
## B            0.01000         NA      NA       NA
## C           -0.06625         NA      NA       NA
## D           -0.05625         NA      NA       NA
## A:B          0.00875         NA      NA       NA
## A:C         -0.02750         NA      NA       NA
## B:C          0.01250         NA      NA       NA
## A:D         -0.00500         NA      NA       NA
## B:D         -0.00500         NA      NA       NA
## C:D          0.01875         NA      NA       NA
## A:B:C        0.01125         NA      NA       NA
## A:B:D       -0.00875         NA      NA       NA
## A:C:D        0.01000         NA      NA       NA
## B:C:D       -0.01000         NA      NA       NA
## A:B:C:D           NA         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 15 and 0 DF,  p-value: NA
```

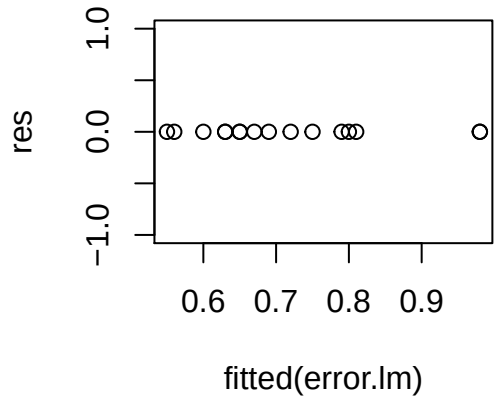
```
##
## Call:
## lm(formula = UEC ~ A * B * C * D, data = error)
##
## Residuals:
## ALL 16 residuals are 0: no residual degrees of freedom!
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.71625         NA      NA       NA
## A            0.08000         NA      NA       NA
## B            0.01000         NA      NA       NA
## C           -0.06625         NA      NA       NA
## D           -0.05625         NA      NA       NA
## A:B          0.00875         NA      NA       NA
## A:C         -0.02750         NA      NA       NA
## B:C          0.01250         NA      NA       NA
## A:D         -0.00500         NA      NA       NA
## B:D         -0.00500         NA      NA       NA
## C:D          0.01875         NA      NA       NA
## A:B:C        0.01125         NA      NA       NA
## A:B:D       -0.00875         NA      NA       NA
## A:C:D        0.01000         NA      NA       NA
## B:C:D       -0.01000         NA      NA       NA
## A:B:C:D      0.00125         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 15 and 0 DF,  p-value: NA
```

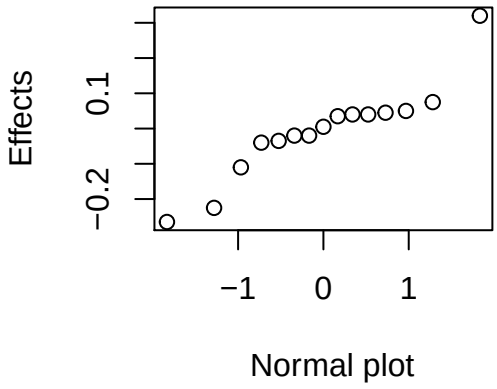
```r
#Residual Analysis
res=error$UEC-fitted(error.lm)
qqPlot(res)
```



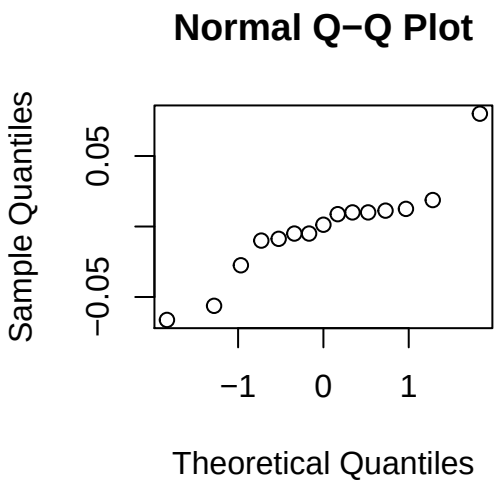

```r
plot(fitted(error.lm), res)#Bad Model
```




```
#Reduced Model Procedure
qqnorm(aov(UEC ~ Block + A*B*C*D, error), full=T,label=T)
```




```
coef=error.lm$coefficients[-1]
variables=names(coef)
plot=qqnorm(coef)
variables[identify(plot)]# A C D
```

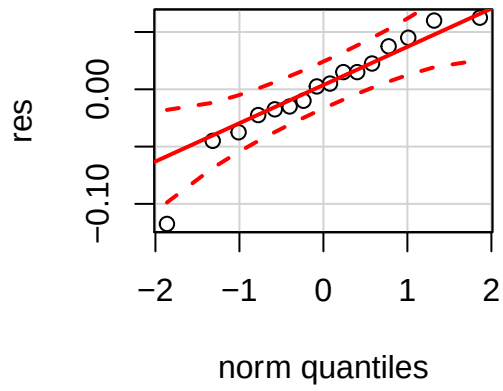



```
## character(0)
```

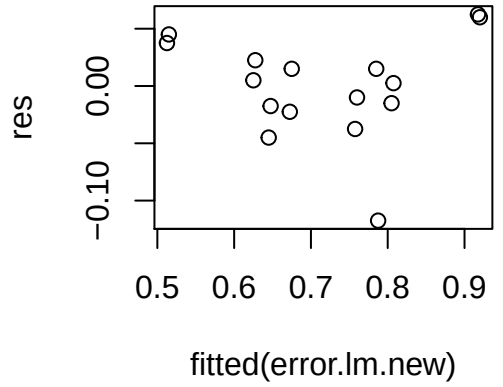
```
#New Model
error.lm.new = lm(UEC ~ Block + A + C + D, error)
summary(error.lm.new)
```

```
##
## Call:
## lm(formula = UEC ~ Block + A + C + D, data = error)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.11750 -0.01875  0.00375  0.02625  0.06250
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.71625    0.01312  54.588 9.61e-15 ***
## Block        0.00125    0.01312   0.095 0.925816
## A            0.08000    0.01312   6.097 7.77e-05 ***
## C           -0.06625    0.01312  -5.049 0.000373 ***
## D           -0.05625    0.01312  -4.287 0.001283 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05248 on 11 degrees of freedom
## Multiple R-squared:  0.8805, Adjusted R-squared:  0.8371
## F-statistic: 20.26 on 4 and 11 DF,  p-value: 4.92e-05
```

```
#Residual Analysis on New Model
res=error$UEC-fitted(error.lm.new)
qqPlot(res)
```




```
plot(fitted(error.lm.new), res)#Better Model
```




We design our data into two blocks with the confounded effects ABCD. We construct two linear models one with and the other without the Block variable. To our results we notice both models share the same coefficients. We check the residual for the Block model to see if it is a good model, unfortunately it is not, we must reduce it. We use the probability plot method to pick out which main effects are important, our results were A, C, and D. Now we apply that to a new model with the Block design and check the residuals. From our residual analysis from out new model we can state that we have a good model with main effects A, C, and D in a block design.

### 7.24

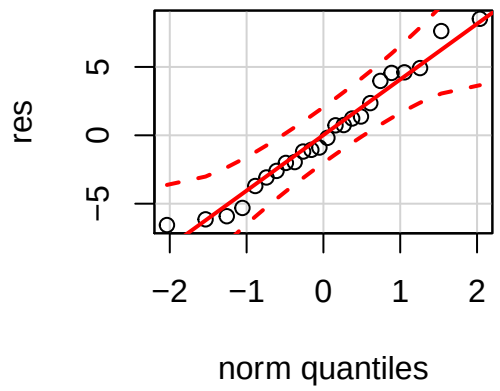
**Suppose that in Problem 6.1 we had confounded ABC in replicate I, AB in replicate II, and BC in replicate III. Calculate the factor effect estimates. Construct the analysis of variance table.**

```r
# creating data table
A = rep(c("-","+","-","+","-","+","-","+"), times = 3)
B = rep(c("-","-","+","+","-","-","+","+"), times = 3)
C = rep(c("-","-","-","-","+","+","+","+"), times = 3)
Rep = rep(c("I", "II", "III"), each = 8)
yield = c(22,32,35,55,44,40,60,39,31,43,34,47,45,37,50,41,25,29,50,46,38,36,54,47)
#dataframe
cutting.speed.long = data.frame(A, B, C, Rep, yield)

# defining coded
coded=function(x) #a function to code variable x
{
  ifelse(x=="+", 1, -1)
}
# coding A and B
#for (j in 1:3)
 # cutting.speed.long[, j]=as.numeric(coded(cutting.speed.long[, j]))

#confounding ABC in RepI
cutting.speed.longRep1=cutting.speed.long[cutting.speed.long$Rep == "I",]
A=coded(cutting.speed.longRep1$A)
B=coded(cutting.speed.longRep1$B)
C=coded(cutting.speed.longRep1$C)
cutting.speed.longRep1$Block=ifelse(A * B * C < 0, 1, 2)
cutting.speed.longRep1=cutting.speed.longRep1[order(cutting.speed.longRep1$Block),]
#confounding AB in RepII
cutting.speed.longRep2=cutting.speed.long[cutting.speed.long$Rep == "II",]
A=coded(cutting.speed.longRep2$A)
B=coded(cutting.speed.longRep2$B)
cutting.speed.longRep2$Block=ifelse(A * B > 0, 1, 2)
cutting.speed.longRep2=cutting.speed.longRep2[order(cutting.speed.longRep2$Block),]
#confounding BC in RepIII
cutting.speed.longRep3=cutting.speed.long[cutting.speed.long$Rep == "III",]
B=coded(cutting.speed.longRep3$B)
C=coded(cutting.speed.longRep3$C)
cutting.speed.longRep3$Block=ifelse(B * C > 0, 1, 2)
cutting.speed.longRep3=cutting.speed.longRep3[order(cutting.speed.longRep3$Block),]
partialConfounding=rbind(cutting.speed.longRep1, cutting.speed.longRep2, cutting.speed.longRep3)
partialConfounding$Rep=factor(partialConfounding$Rep)
partialConfounding$Blocks=factor(paste(partialConfounding$Rep,
                                       partialConfounding$Block,sep = "-"))
#ANOVA
cutting.confounded.aov = aov(yield ~ Blocks + A * B * C, partialConfounding)
summary(cutting.confounded.aov)
```

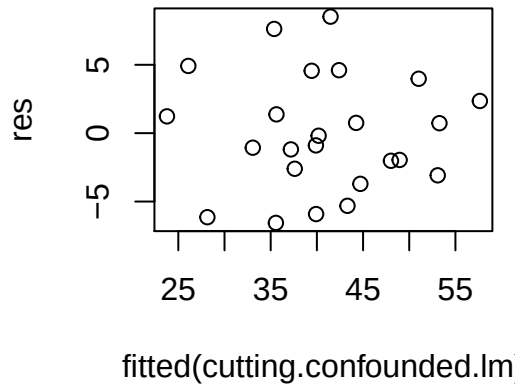
```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## Blocks       5  119.8    24.0   0.646 0.670549
## A            1    0.7     0.7   0.018 0.895798
## B            1  770.7   770.7  20.767 0.000821 ***
## C            1  280.2   280.2   7.550 0.018970 *
## A:B          1   25.0    25.0   0.674 0.429202
## A:C          1  468.2   468.2  12.616 0.004537 **
## B:C          1   22.6    22.6   0.608 0.451996
## A:B:C        1    0.1     0.1   0.002 0.968000
```

```
## Residuals   11  408.2    37.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Residual Analysis for cutting.confounded.aov
cutting.confounded.lm = lm(yield ~ Blocks + A * B * C, partialConfounding)
res=partialConfounding$yield-fitted(cutting.confounded.lm)
qqPlot(res)
```




```
plot(fitted(cutting.confounded.lm), res)#Good Model
```




After confounding certain main effect cominations to an assigned Replicate, we analyze the ANOVA our confounded data has to offer. Main effects B, C, and interaction effect AC are significant. We quickly build a model and check our residuals and see that normality is good and residuals are random. Our model is good.

## 7.25

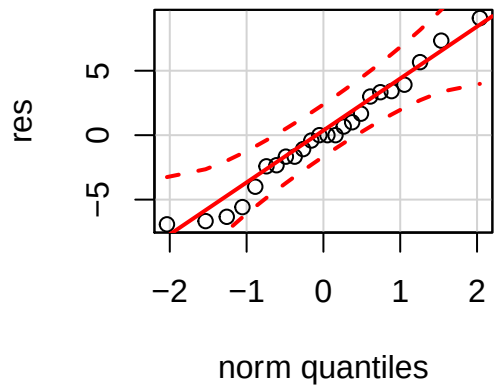
**Repeat the analysis of Problem 6.1 assuming that ABC was confounded with blocks in each replicate.**

```
# creating data table
A = rep(c("-","+","-","+","-","+","-","+"), times = 3)
B = rep(c("-","-","+","+","-","-","+","+"), times = 3)
C = rep(c("-","-","-","-","+","+","+","+"), times = 3)
Rep = rep(c(1, 2, 3), each = 8)
yield = c(22,32,35,55,44,40,60,39,31,43,34,47,45,37,50,41,25,29,50,46,38,36,54,47)
#dataframe
cutting.speed.long = data.frame(A, B, C, Rep, yield)

# defining coded
coded=function(x) #a function to code variable x
{
  ifelse(x=="+", 1, -1)
}

# coding A and B
#for (j in 1:3)
 # cutting.speed.long[, j]=as.numeric(coded(cutting.speed.long[, j]))

#confounding ABC in RepI
cutting.speed.longRep1=cutting.speed.long[cutting.speed.long$Rep == 1,]
A=coded(cutting.speed.longRep1$A)
B=coded(cutting.speed.longRep1$B)
C=coded(cutting.speed.longRep1$C)
cutting.speed.longRep1$Block=ifelse(A * B * C < 0, 1, 2)
cutting.speed.longRep1=cutting.speed.longRep1[order(cutting.speed.longRep1$Block),]
#confounding ABC in RepII
cutting.speed.longRep2=cutting.speed.long[cutting.speed.long$Rep == 2,]
A=coded(cutting.speed.longRep2$A)
B=coded(cutting.speed.longRep2$B)
C=coded(cutting.speed.longRep2$C)
cutting.speed.longRep2$Block=ifelse(A * B * C < 0, 1, 2)
cutting.speed.longRep2=cutting.speed.longRep2[order(cutting.speed.longRep2$Block),]
#confounding ABC in RepIII
cutting.speed.longRep3=cutting.speed.long[cutting.speed.long$Rep == 3,]
A=coded(cutting.speed.longRep3$A)
B=coded(cutting.speed.longRep3$B)
C=coded(cutting.speed.longRep3$C)
cutting.speed.longRep3$Block=ifelse(A * B * C < 0, 1, 2)
cutting.speed.longRep3=cutting.speed.longRep3[order(cutting.speed.longRep3$Block),]
partialConfounding.abc=rbind(cutting.speed.longRep1, cutting.speed.longRep2, cutting.speed.longRep3)
partialConfounding.abc$Rep=factor(partialConfounding.abc$Rep)
partialConfounding.abc$Blocks=factor(paste(partialConfounding.abc$Rep,
                                    partialConfounding.abc$Block,sep = "-"))
#ANOVA
cutting.confoundedabc.aov = aov(yield ~ Blocks + A * B * C, partialConfounding.abc)
summary(cutting.confoundedabc.aov)
```

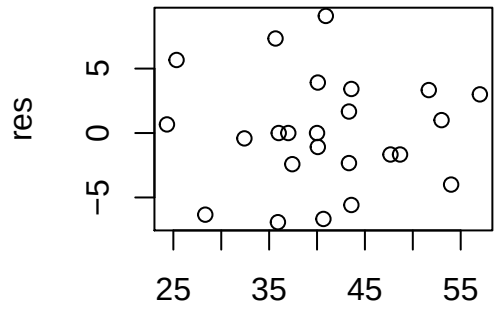
```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## Blocks       5   93.3    18.7   0.537 0.745271
```

```
## A            1    0.7     0.7   0.019 0.892200
## B            1  770.7   770.7  22.151 0.000509 ***
## C            1  280.2   280.2   8.053 0.014960 *
## A:B          1   16.7    16.7   0.479 0.502030
## A:C          1  468.2   468.2  13.456 0.003217 **
## B:C          1   48.2    48.2   1.384 0.262161
## Residuals   12  417.5    34.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Residual Analysis for cutting.confounded.aov
cutting.confoundedabc.lm = lm(yield ~ Blocks + A * B * C, partialConfounding.abc)
res=partialConfounding.abc$yield-fitted(cutting.confoundedabc.lm)
qqPlot(res)
```




```
plot(fitted(cutting.confoundedabc.lm), res)#Good Model
```



```r
# test stuff
partialConfounding.abc$A <-coded(partialConfounding.abc$A)
partialConfounding.abc$B <-coded(partialConfounding.abc$B)
partialConfounding.abc$C <-coded(partialConfounding.abc$C)

partialConfounding.abc$Block2 <- ifelse(partialConfounding.abc$A * partialConfounding.abc$B * partialCo
summary(aov(yield ~ Block + A * B * C, partialConfounding.abc))
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## Block        1   28.2    28.2   0.934 0.348282
## A            1    0.7     0.7   0.022 0.883680
## B            1  770.7   770.7  25.547 0.000117 ***
## C            1  280.2   280.2   9.287 0.007679 **
## A:B          1   16.7    16.7   0.552 0.468078
## A:C          1  468.2   468.2  15.519 0.001172 **
## B:C          1   48.2    48.2   1.597 0.224475
## Residuals   16  482.7    30.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
partialConfounding.abc$Block == partialConfounding.abc$Block2
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Partially confounding is identical to completely confounding when using both methods. We compare the results of the blocking for both partially and completely confounding. For the partially confounded data we check the anova and residuals, we have a good model.