Final Exam (S-670)

FNU Anirudh

December 12, 2015

# Solution 1

## Five R's to EDA are

1. Resistance
2. Residuals
3. Re-expression
4. Revelation
5. Re-iteration

# Solution 2

## I am assuming score of 10 students to list five-number summary.

marks<- c(55,70,40,35,90,30,80,95,22)  
summary(marks)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 22.00 35.00 55.00 57.44 80.00 95.00

fivenum(marks)

## [1] 22 35 55 80 95

# Solution 3

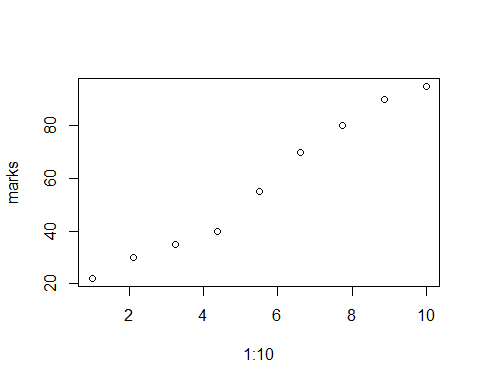
## Goals achieved by Re-expressing data

1. We can know effects of departures from normality.
2. We can know about methods that are resistant to Gaussian assumption.
3. We can know much more about methods to diagnose nonlinearity.
4. We can know about methods to detect outliers.

# Solution 4

1. To detect long tailness we use qqplot or histogram which tells us if data is normally distributed or not.
2. We can also use kurtosis and skewness function to detect long tailness.

#Example  
qqplot(1:10,marks)



library(e1071)  
kurtosis(marks)

## [1] -1.821302

skewness(marks)

## [1] 0.10435

1. We can use Power transformation using Tukey's ladder of transformation, we can plot the points with respect to tukey's proposed value of x-axis and y-axis and then calculate the slope, subtracting it from 1 to get the value of power transformation.
2. We can use the H distribution for long tails i.e. if h = 0 (Normal distribution, no tails) h>0 Long tails. To transform the data: X = A + B\* Y\_h(Z) Where, Y\_h(Z) = Zh((hZ2)/2)

# Solution 5

When g = 0, h=0 Gaussian Data , No skweness and No long tails.

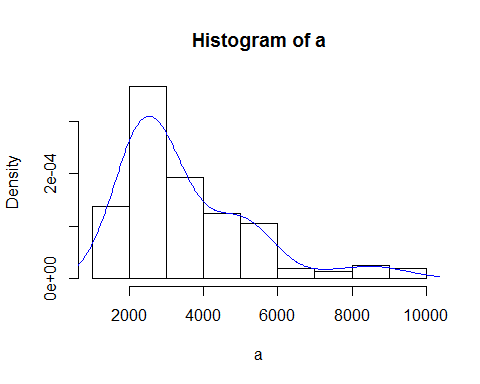
When g<0.25 , h>0 slight skewness with Long tail

When g~1, h>0 Skewed with long tails.

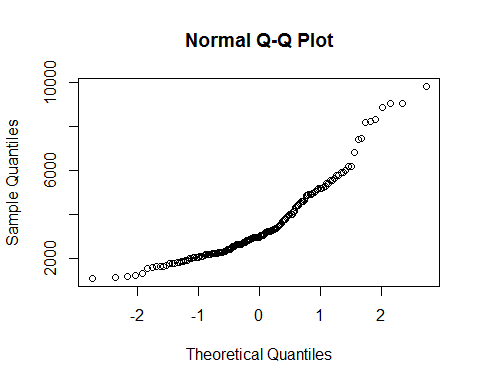
1. For, (-0.5,0.3) -> Left skewed with light tails.
2. For, (0.5,0.3) -> Right skewed with light tails.
3. For, (1,0.6) -> Heavily Right skewed with Heavy tails.

# Solution 6

a=c(1092,1137,1197,1237,1301,1523,1577, 1619,1626,1644,1672,1748,1768,1780, 1796,1816,1843,1844,1902,1919,1983, 1993,2025,2028,2032,2036,2072,2078, 2090,2137,2162,2163,2180,2185,2194,2225,2230,2233,2234,2235,2265,2270, 2274,2281,2289,2319,2322,2357,2381, 2398,2421,2421,2443,2522,2549,2552, 2581,2618,2618,2620,2624,2642,2647, 2666,2705,2721,2740,2804,2819,2823, 2860,2873,2906,2913,2926,2929,2931, 2931,2934,2939,2961,3020,3023,3044, 3047,3048,3096,3174,3190,3199,3204, 3222,3225,3278,3287,3292,3300,3339, 3361,3412,3462,3503,3530,3589,3672, 3734,3749,3783,3854,3901,3932,3995, 4001,4006,4118,4134,4320,4346,4385, 4401,4522,4565,4581,4593,4629,4855, 4868,4878,4885,4907,4962,4975,5021, 5127,5155,5160,5183,5229,5242,5379, 5383,5513,5555,5619,5755,5774,5890, 5899,5988,6161,6185,6818,7406,7419, 8175,8220,8282,8827,9027,9042,9805)  
hist(a,probability=TRUE)  
lines(density(a), col="blue")



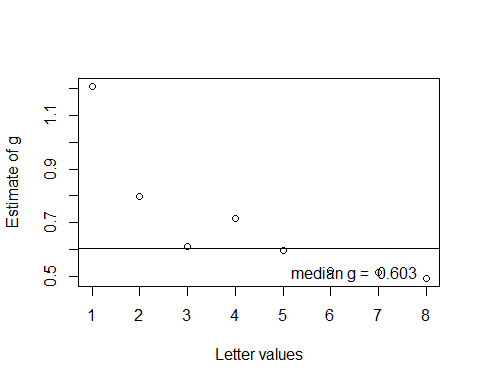
qqnorm(a)



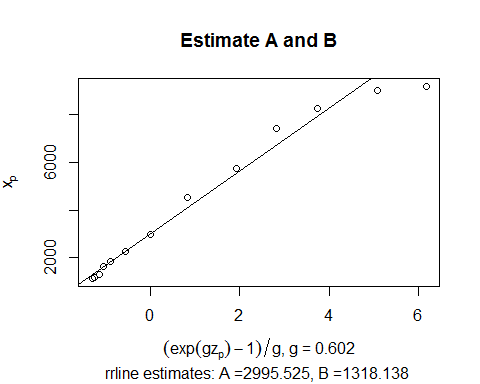
#b)  
source("lvalprogs.R")  
lvals<- lval(a)  
lvals

## Depth Lower Upper Mid Spread pseudo-s  
## M 81.0 2961.0 2961.0 2961.0 0 0.000  
## F 41.0 2265.0 4522.0 3393.5 2257 1673.117  
## E 21.0 1983.0 5383.0 3683.0 3400 1477.812  
## D 11.0 1672.0 6185.0 3928.5 4513 1470.875  
## C 6.0 1523.0 8220.0 4871.5 6697 1797.629  
## B 3.5 1217.0 8927.0 5072.0 7710 1789.798  
## A 2.0 1137.0 9042.0 5089.5 7905 1634.914  
## Z 1.5 1114.5 9423.5 5269.0 8309 1561.803  
## Y 1.0 1092.0 9805.0 5448.5 8713 1509.720

len <-length(a)  
pp <- 1/2^(1:9);   
gau <- abs(qnorm(pp))   
pp\_2 <- (lvals[,1]-1/3)/(len + 1/3)  
gau\_2 <- abs(qnorm(pp\_2))   
  
est.g <- log((lvals[,3] - lvals[1,2])/(lvals[1,2]-lvals[,2]))/gau\_2  
  
plot(1:(dim(lvals)[1]-1), est.g[-1],  
 xlab="Letter values",  
 ylab="Estimate of g")  
abline(h=median(est.g[-1]))  
text(6.5,0.51,paste("median g = ",format(round(median(est.g[-1]),3))))



source("rrline.R")  
  
# Estimate of A and B (selected p)  
est.g <- median(est.g[-1])   
p <- c(0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.975, 0.99, 0.995)  
zp <- qnorm(p)  
est.Y <- (exp(est.g\*zp)-1)/est.g  
plot(est.Y,quantile(a,p),main="Estimate A and B",ylab=expression(x[p]),  
 xlab=expression(paste((exp(gz[p])-1)/g,", g = 0.602")),  
 sub="rrline estimates: A =2995.525, B =1318.138 ")  
rr <- rrline1(est.Y,quantile(a,p))  
abline(rr$a,rr$b)



#c)  
library(boot)  
  
#estimate of A   
data<-a  
fboota <- function(d, i){  
 d=d[i]  
 fit=rrline1(est.Y,quantile(d,p))  
 a=fit$a  
 return(a)   
}  
  
boot\_corr\_1<- boot(data, fboota, R = 1000)  
boot.ci(boot\_corr\_1, type = "all",conf=0.9)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_corr\_1, conf = 0.9, type = "all")  
##   
## Intervals :   
## Level Normal Basic   
## 90% (2840, 3213 ) (2845, 3231 )   
##   
## Level Percentile BCa   
## 90% (2763, 3149 ) (2831, 3204 )   
## Calculations and Intervals on Original Scale

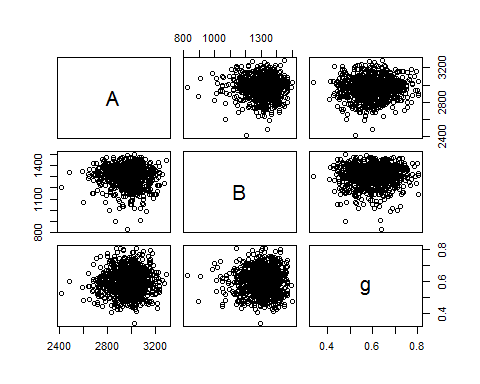
#Estimate of B  
fbootb <- function(dat, i){  
 dat=dat[i]  
 fit=rrline1(est.Y,quantile(dat,p))  
 b=fit$b  
 return(b)  
 }  
  
boot\_corr\_2 <- boot(data,fbootb, R=1000)  
boot.ci(boot.out = boot\_corr\_2, conf = 0.9, type = "all")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_corr\_2, conf = 0.9, type = "all")  
##   
## Intervals :   
## Level Normal Basic   
## 90% (1190, 1476 ) (1218, 1491 )   
##   
## Level Percentile BCa   
## 90% (1151, 1424 ) (1179, 1453 )   
## Calculations and Intervals on Original Scale

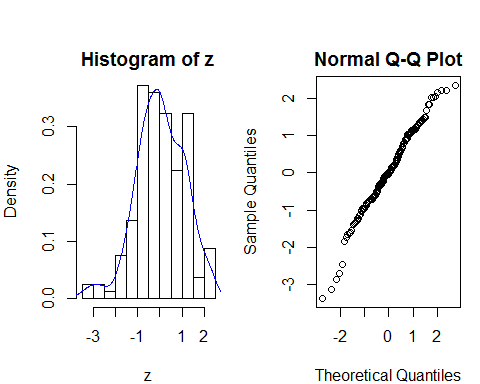
#Estimate of g  
  
fbootg <- function(dat, i){  
 n=length(data)  
 dat\_2=dat[i]  
 lvals=lval(dat\_2)  
 pp <- (lvals[,1]-1/3)/(n + 1/3)  
 gau <- qnorm(pp)  
 est\_1.g <- (-log((lvals[,3] - lvals[1,2])/(lvals[1,2]-lvals[,2]))/gau)  
 est.g <- median(est\_1.g[-1])   
 return(est.g)   
}  
  
boot\_corr\_3 <- boot(data,fbootg, R=1000)  
boot.ci(boot\_corr\_3, type = "all",conf=0.9)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_corr\_3, conf = 0.9, type = "all")  
##   
## Intervals :   
## Level Normal Basic   
## 90% ( 0.4858, 0.7298 ) ( 0.4818, 0.7301 )   
##   
## Level Percentile BCa   
## 90% ( 0.4755, 0.7238 ) ( 0.4914, 0.7376 )   
## Calculations and Intervals on Original Scale

#Pairs plot  
D=data.frame(as.vector(boot\_corr\_1$t), as.vector(boot\_corr\_2$t),as.vector(boot\_corr\_3$t))  
  
A=D[,1]  
  
B=D[,2]  
  
g=D[,3]  
  
pairs(~A+B+g,data=D)



#d)  
g<-0.602  
A<-2995.525  
B<-1318.138  
z<- 1/g\*log(((a-A)\*g)/B +1)  
par(mfrow=c(1,2),mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0))  
hist(z,prob=TRUE)  
lines(density(z),col="blue")  
qqnorm(z)



#e)  
# Pearson Goodness of fit Test  
gof.pearson=function (x,nbins) {  
 n = length(x)   
 m = floor(n/nbins)   
 k = n - m\*nbins  
 xx=sort(x)   
 index = rep(1:nbins,m)  
 if(k >0){ d=sample(1:nbins,k,replace=FALSE);   
 index=c(index,d) }   
 bincount=as.numeric(table(index))   
 binindicies = cumsum(bincount)   
 binbreaks = rev(rev(xx[binindicies])[-1])   
 binbreaks = c(-Inf,binbreaks,Inf)   
 bins=cut(x,breaks=binbreaks)  
 internalbreaks = rev(rev(xx[binindicies])[-1])   
 p = pnorm(internalbreaks,mean(x),sd(x))  
 p = c(p[1],diff(p),1-pnorm(max(internalbreaks),mean(x),sd(x)))  
 exp = n\*p  
 df = data.frame(bin=levels(bins),bincount=bincount,prob=p,expectedcount=exp)  
 chisqstat = sum((bincount - exp)^2/exp)  
 pval = 1- pchisq(chisqstat,nbins-1)  
 output = list(df=df,chisq=chisqstat,pval=pval)  
 output = list(df=df,chisq=chisqstat,pval=pval)  
}  
out=gof.pearson(z,2\*sqrt(length(z)))   
out

## $df  
## bin bincount prob expectedcount  
## 1 (-Inf,-1.73] 7 0.05144838 8.283189  
## 2 (-1.73,-1.37] 6 0.04692035 7.554177  
## 3 (-1.37,-1.12] 7 0.04464469 7.187794  
## 4 (-1.12,-0.958] 6 0.03753055 6.042418  
## 5 (-0.958,-0.774] 7 0.04814227 7.750905  
## 6 (-0.774,-0.71] 6 0.01831760 2.949134  
## 7 (-0.71,-0.614] 7 0.02888255 4.650090  
## 8 (-0.614,-0.506] 6 0.03446307 5.548554  
## 9 (-0.506,-0.314] 6 0.06489017 10.447317  
## 10 (-0.314,-0.236] 7 0.02762151 4.447062  
## 11 (-0.236,-0.106] 6 0.04724772 7.606882  
## 12 (-0.106,-0.0497] 6 0.02072680 3.337015  
## 13 (-0.0497,0.0207] 6 0.02595509 4.178769  
## 14 (0.0207,0.141] 6 0.04444573 7.155763  
## 15 (0.141,0.208] 6 0.02422508 3.900238  
## 16 (0.208,0.321] 6 0.04079059 6.567285  
## 17 (0.321,0.491] 6 0.05914215 9.521885  
## 18 (0.491,0.63] 7 0.04545790 7.318723  
## 19 (0.63,0.824] 6 0.05789436 9.320992  
## 20 (0.824,1.02] 6 0.05180435 8.340500  
## 21 (1.02,1.09] 7 0.01571016 2.529336  
## 22 (1.09,1.17] 6 0.01865020 3.002682  
## 23 (1.17,1.35] 6 0.03496808 5.629862  
## 24 (1.35,1.49] 6 0.02223843 3.580388  
## 25 (1.49, Inf] 7 0.08788223 14.149038  
##   
## $chisq  
## [1] 32.2822  
##   
## $pval  
## [1] 0.1309236

# ECDF Based Test Statistics  
library("goftest")  
ks.test(z,"pnorm") #Kolmogorov Test

##   
## One-sample Kolmogorov-Smirnov test  
##   
## data: z  
## D = 0.070026, p-value = 0.4088  
## alternative hypothesis: two-sided

ad.test(z,"pnorm") #Anderson-Darling Test

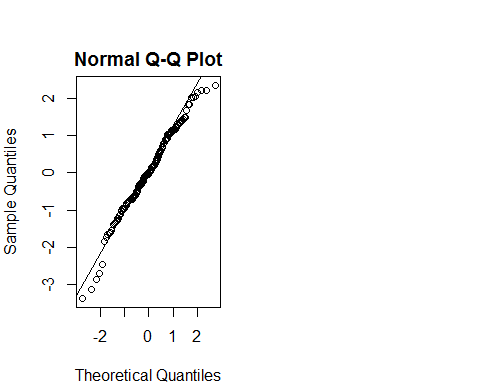
##   
## Anderson-Darling test of goodness-of-fit  
## Null hypothesis: Normal distribution  
##   
## data: z  
## An = 0.75306, p-value = 0.5159

cvm.test(z,"pnorm")#Cramer-von-Mises Test

##   
## Cramer-von Mises test of goodness-of-fit  
## Null hypothesis: Normal distribution  
##   
## data: z  
## omega2 = 0.088797, p-value = 0.6432

#Correlation of the QQ Data test  
qqnorm(z)  
qqline(z)  
  
#Shapiro Wilk's Test  
shapiro.test(z)

##   
## Shapiro-Wilk normality test  
##   
## data: z  
## W = 0.98436, p-value = 0.0667



The test above shows that distribution is normally distributed.

# Pearson's goodness of fit Test

Pearson's chi-squared test uses a measure of goodness of fit which is the sum of differences between observed and expected outcome frequencies (that is, counts of observations), each squared and divided by the expectation.The resulting value can be compared to the chi-squared distribution to determine the goodness of fit. In order to determine the degrees of freedom of the chi-squared distribution, one takes the total number of observed frequencies and subtracts the number of estimated parameters. The test statistic follows, approximately, a chi-square distribution with (k ??? c) degrees of freedom where k is the number of non-empty cells and c is the number of estimated parameters (including location and scale parameters and shape parameters) for the distribution. Sample with a large size is assumed. Observations are supposed to be independent.

# Shapiro Wilk's test:

The null-hypothesis of this test is that the population is normally distributed. Thus if the p-value is less than the chosen alpha level, then the null hypothesis is rejected and there is evidence that the data tested are not from a normally distributed population. In other words, the data are not normal. On the contrary, if the p-value is greater than the chosen alpha level, then the null hypothesis that the data came from a normally distributed population cannot be rejected.

# QQ plot:-

* is used to visualize the normality of the data.
* It is easy to compute.

# ECDF based statistics:

# Kolmogorov Test

* A feature of this test is that distribution of the K-S test statistics itself does not depend on underlying cumulative distribution function being tested.
* Another advantage is that it is an exact test.

# Limitations:

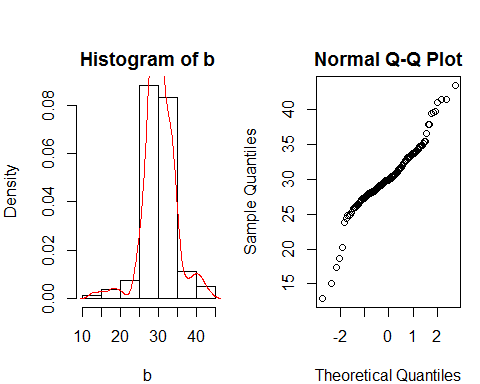
* It only applies to continuous distributions.
* It tends to be more sensitive near the center of distribution than at tails.
* Most serious limitation is that distribution must be fully specified.

# Anderson-Darling Test

Many statistical tests and procedures are based on specific distributional assumptions. The assumption of normality is particularly common in classical statistical tests. Much reliability modeling is based on the assumption that the data follow a Weibull distribution. There are many non-parametric and robust techniques that do not make strong distributional assumptions. However, techniques based on specific distributional assumptions are in general more powerful than non-parametric and robust techniques. Therefore, if the distributional assumptions can be validated, they are generally preferred.

# Solution 7

b =c(12.87,15.09,17.39,18.62,20.24,23.76,24.35, 24.74,24.81,24.96,25.19,25.75,25.89,25.97, 26.07,26.19,26.35,26.36,26.67,26.76,27.07, 27.12,27.26,27.28,27.30,27.31,27.46,27.49, 27.54,27.72,27.81,27.82,27.88,27.90,27.93, 28.03,28.05,28.06,28.07,28.07,28.17,28.19, 28.20,28.22,28.25,28.34,28.35,28.46,28.53,28.58,28.64,28.65,28.70,28.92,28.99,29.00, 29.07,29.16,29.16,29.17,29.18,29.22,29.23, 29.28,29.37,29.40,29.45,29.59,29.62,29.63, 29.71,29.74,29.81,29.82,29.85,29.86,29.86, 29.86,29.87,29.88,29.92,30.04,30.05,30.09, 30.09,30.10,30.19,30.34,30.37,30.38,30.39, 30.43,30.43,30.53,30.55,30.55,30.57,30.64, 30.68,30.77,30.86,30.93,30.98,31.08,31.22, 31.32,31.35,31.41,31.52,31.60,31.65,31.76, 31.76,31.77,31.96,31.98,32.28,32.33,32.39, 32.42,32.61,32.68,32.71,32.73,32.79,33.15, 33.18,33.19,33.20,33.24,33.33,33.35,33.43, 33.60,33.65,33.66,33.70,33.77,33.80,34.03, 34.03,34.26,34.33,34.44,34.68,34.71,34.91, 34.93,35.09,35.40,35.44,36.63,37.81,37.84, 39.47,39.58,39.72,41.00,41.49,41.52,43.50)  
par(mfrow=c(1,2),mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0))  
hist(b,prob=TRUE)  
lines(density(b),col="red")  
qqnorm(b)



lvals <- lval(b);   
#lvals  
len<-length(b)  
gh2.data <- b  
lvals.gh2 <- lval(gh2.data)  
yy.gh2 <- log(lvals.gh2[-1,6])  
xx.gh2 <- (qnorm((lvals.gh2[-1,1] - 1/3)/(161 + 1/3)))^2/2   
plot(xx.gh2,yy.gh2,main="Estimate h and B",   
 ylab="log(pseudo-sigma)", xlab=expression(z[p]^2/2),  
 sub="rrline: 2.71 + 0.24x => B = 2.71, h = 0.24")  
rr <- rrline1(xx.gh2,yy.gh2);  
  
abline(rr$a,rr$b)  
  
exp(rr$a) # estimate B

## [1] 2.948061

rr$b # estimate h

## [1] 0.2080906

median(b) #estimate A

## [1] 29.92

#Estimate of B  
data<-b  
fbootb <- function(d, i){  
 d=d[i]  
 fit=rrline1(xx.gh2,quantile(d,p))  
 a=fit$a  
 return(a)   
}  
  
boot\_corr\_1 <- boot(data,fbootb, R=1000)  
boot.ci(boot\_corr\_1, type = "all",conf=0.95) #Confidence Interval for B

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_corr\_1, conf = 0.95, type = "all")  
##   
## Intervals :   
## Level Normal Basic   
## 95% ( 6.84, 18.77 ) ( 5.50, 17.35 )   
##   
## Level Percentile BCa   
## 95% (10.98, 22.83 ) ( 8.93, 19.31 )   
## Calculations and Intervals on Original Scale  
## Some BCa intervals may be unstable

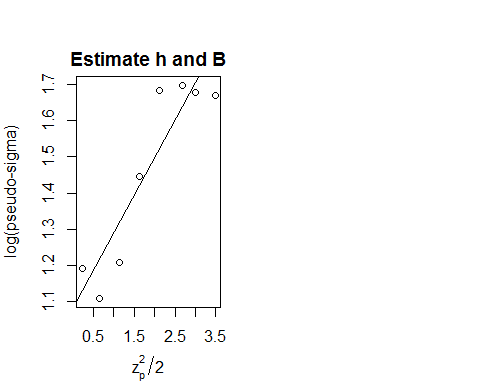
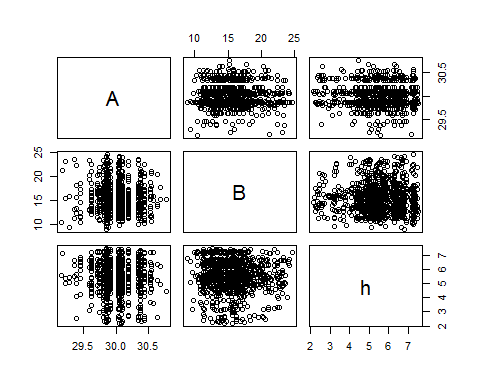
#Estimate of H  
fbootg <- function(d, i){  
 d=d[i]  
 fit=rrline1(xx.gh2,quantile(d,p))  
 g=fit$b  
 return(g)   
}  
  
boot\_corr\_2 <- boot(data,fbootg, R=1000)  
boot.ci(boot\_corr\_2, type = "all",conf=0.95)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_corr\_2, conf = 0.95, type = "all")  
##   
## Intervals :   
## Level Normal Basic   
## 95% ( 3.682, 8.151 ) ( 4.144, 8.867 )   
##   
## Level Percentile BCa   
## 95% ( 2.600, 7.324 ) ( 3.785, 7.486 )   
## Calculations and Intervals on Original Scale  
## Some BCa intervals may be unstable

#Estimate of A  
fboota <- function(d, i){  
 d=d[i]  
 return (median(d))}   
  
boot\_corr\_3 <- boot(data,fboota, R=1000)   
boot.ci(boot\_corr\_3, type = "all",conf=0.95)

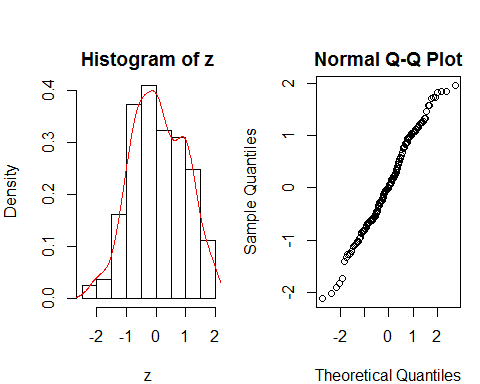
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_corr\_3, conf = 0.95, type = "all")  
##   
## Intervals :   
## Level Normal Basic   
## 95% (29.39, 30.30 ) (29.31, 30.25 )   
##   
## Level Percentile BCa   
## 95% (29.59, 30.53 ) (29.45, 30.43 )   
## Calculations and Intervals on Original Scale

# Pairs Plot  
D=data.frame(as.vector(boot\_corr\_1$t), as.vector(boot\_corr\_2$t),as.vector(boot\_corr\_3$t))  
  
A=D[,3]  
  
B=D[,1]  
  
h=D[,2]  
  
pairs(~A+B+h,data=D)

# Solution 8

HDistBackXform=function(h,A,B,data){  
 n=length(data)  
 #using Veleman's rule  
 output=numeric(n)  
 g=function(z){z\*exp(h\*z^2)-((x-A)/B)}  
 for(i in 1:n){  
 x=data[i]  
 obj=uniroot(g,interval=c(-6,6))  
 output[i]=obj$root  
 }  
 return(output)  
}  
  
h<-0.24  
A<-29.92  
B<-2.71  
  
z<-HDistBackXform(h,A,B,b)  
par(mfrow=c(1,2),mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0))  
hist(z,prob=TRUE)  
lines(density(z),col="red")  
qqnorm(z)



noofbins=2\*sqrt(length(z))  
out<-gof.pearson(z,noofbins)  
out

## $df  
## bin bincount prob expectedcount  
## 1 (-Inf,-1.41] 6 0.04767087 7.675010  
## 2 (-1.41,-1.13] 6 0.03994244 6.430732  
## 3 (-1.13,-1.02] 6 0.02116316 3.407268  
## 4 (-1.02,-0.827] 6 0.04651381 7.488724  
## 5 (-0.827,-0.717] 6 0.03103625 4.996837  
## 6 (-0.717,-0.633] 6 0.02630822 4.235623  
## 7 (-0.633,-0.588] 6 0.01519798 2.446875  
## 8 (-0.588,-0.485] 7 0.03624750 5.835847  
## 9 (-0.485,-0.331] 7 0.05935784 9.556613  
## 10 (-0.331,-0.254] 6 0.03131020 5.040942  
## 11 (-0.254,-0.11] 7 0.06161783 9.920471  
## 12 (-0.11,-0.0258] 6 0.03726980 6.000438  
## 13 (-0.0258,0.0443] 7 0.03122751 5.027629  
## 14 (0.0443,0.165] 7 0.05390391 8.678529  
## 15 (0.165,0.23] 6 0.02860502 4.605408  
## 16 (0.23,0.338] 6 0.04697354 7.562740  
## 17 (0.338,0.497] 6 0.06635819 10.683669  
## 18 (0.497,0.619] 6 0.04716925 7.594250  
## 19 (0.619,0.786] 6 0.05826451 9.380585  
## 20 (0.786,0.88] 6 0.02929340 4.716237  
## 21 (0.88,0.993] 6 0.03195898 5.145395  
## 22 (0.993,1.06] 6 0.01781587 2.868355  
## 23 (1.06,1.19] 7 0.02807358 4.519847  
## 24 (1.19,1.33] 7 0.02658769 4.280618  
## 25 (1.33, Inf] 7 0.08013265 12.901357  
##   
## $chisq  
## [1] 25.86585  
##   
## $pval  
## [1] 0.3804874

# Solution 9

#Question 9:  
data = rnorm(100, 3, 2)  
  
getGausEstimate = function(data){  
 d = density(data, kernel="gaussian")  
 index = which(d$y == max(d$y), arr.ind =TRUE)  
 ans = d$x[index]  
 return(ans)  
}  
  
calculatePseudoValues = function(data) {  
 n = length(data)  
 yall = getGausEstimate(data)  
 PV = numeric(n)  
 for( i in 1:n) {  
 yminusi = getGausEstimate(data[-i])  
 PV[i] = n\*yall - (n-1)\*yminusi  
 }  
 return(PV)  
}  
  
# We first use jackknife   
PVAll = calculatePseudoValues(data)  
n = length(PVAll)  
print('Jackknife estimate is')

## [1] "Jackknife estimate is"

mean(PVAll)

## [1] 3.611022

jackKnifeEstimate = mean(PVAll)  
varJK = sum((PVAll - jackKnifeEstimate)^2)/(n\*(n-1))  
seJK = sqrt(varJK)  
seJK

## [1] 0.3262691

getbootstrapestimate = function(data, nsim) {  
 theta = numeric(nsim)  
 varTheta = numeric(nsim)  
   
 n = length(data)  
 index = 1:n  
 for (i in 1:nsim){  
 sampleindex= sample(index,n,replace=TRUE)  
 theta[i] = mean(getGausEstimate(data[sampleindex]))  
 }  
   
 output = list(thetaBS = mean(theta), varBS = var(theta),  
 seBS = sqrt(var(theta)))  
 output  
}  
# Now we calculate the Bootstrap estimate of the statistic  
seBS = getbootstrapestimate(data, 100)$seBS   
seBS

## [1] 0.7324645

# Solution 10

# To fit a robust resistant line we do the following:

* Sort the values and divide the observation say n into 3 equal sized groups.
* Find the summary(median x and median y) of the extreme groups.
* Calculate the slope using two point formula.
* Calculate the intercept.

# Advantages:

* Easy to calculate.
* The RRline is more robust and resistant towards outliers.
* Slope and intercept can easily be calculated.

# Disadvantages:

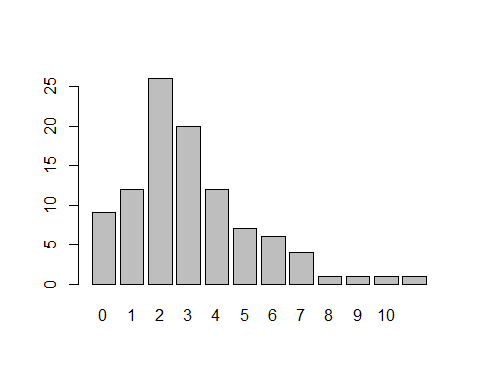
* It may take a lot time, depending on degree of resistance required.
* Unique solution is not guaranteed, depends on iterations.
* Bigger the dataset, more time the calculations take.

# Solution 11

* In bootstrapping we create the sample ,with replacements, of data from given data while in jackniffing we keep one data point at the back to test at the end when we do calculations.
* Bootstrapping is more relevant today than jacknifing.

# Solution 12

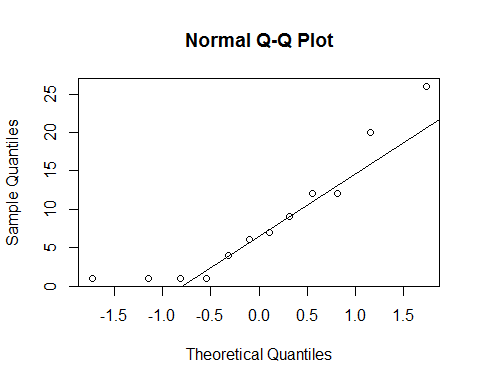
obs1 = c(5, 3, 0, 2, 0, 3, 2, 3, 6, 1, 2, 1, 2, 1, 3, 3, 3, 5, 2, 4)  
obs2 = c(4, 0, 2, 3, 7, 12, 3, 10, 9, 2, 3, 7, 7, 2, 3, 3, 6, 2, 4, 3)  
obs3 = c(5, 2, 2, 4, 0, 4, 2, 5, 2, 3, 3, 6, 5, 8, 3, 6, 6, 0, 5, 2)  
obs4 = c(2, 2, 6, 3, 4, 4, 2, 2, 4, 7, 5, 3, 3, 0, 2, 2, 2, 1, 3, 4)  
obs5 = c(2, 2, 1, 1, 1, 2, 1, 4, 4, 3, 2, 1, 4, 1, 1, 1, 0, 0, 2, 0)  
  
obs\_total <- c(obs1,obs2,obs3,obs4,obs5)  
  
years = c(1860:1959)  
  
observation\_year = cbind(obs\_total,years)  
  
combinedObservation = observation\_year  
Observations\_dataSet = data.frame(combinedObservation)  
colnames(Observations\_dataSet) <- c("Inventions","Year")  
uniqueOrderedDataFrame<-unique(Observations\_dataSet)  
count\_table\_data <- xtabs(~Inventions, data=uniqueOrderedDataFrame)  
  
#Looking at the table it is a discrete frequency distributions. It may belong to poisson family or binomial family.For this we will try plotting poissonness plot, if we can fit a straight line then we can say that a poisson distribution is a good fit. If not we have to resort to plotting binomial distribution plot.  
barplot(count\_table\_data)



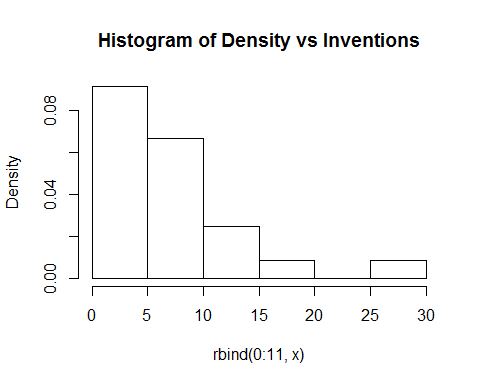
count\_table\_data

## Inventions  
## 0 1 2 3 4 5 6 7 8 9 10 12   
## 9 12 26 20 12 7 6 4 1 1 1 1

x<-as.vector(count\_table\_data)  
qqnorm(x)  
qqline(x)



#Looking at qqnorm plot the distribution appears to be right skewed distribution.  
hist(rbind(0:11,x),probability = TRUE, main = "Histogram of Density vs Inventions")



# skewness and kurtosis, they should be around (0,3)  
skewness(count\_table\_data)

## [1] 0.8758157

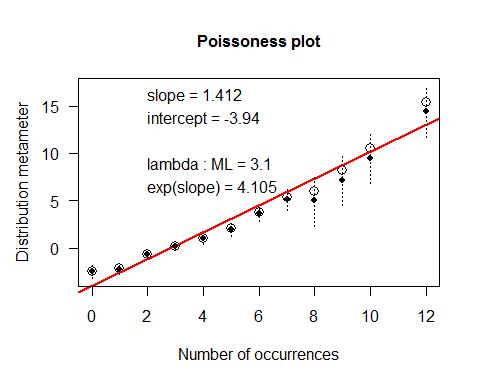
kurtosis(x)

## [1] -0.4820701

#b)  
  
#install.packages("vcd")  
library(vcd)

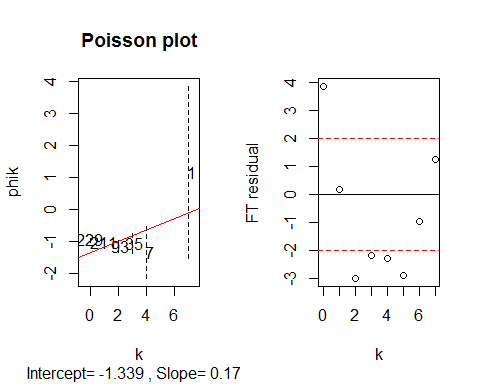
## Loading required package: grid

distplot(count\_table\_data)

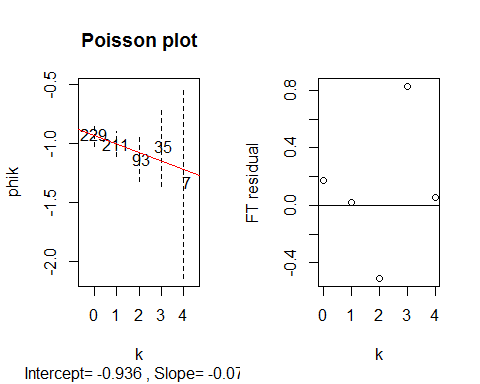


#Slope is 1.412, intercept is -3.94, lambda = 3.1, exp(slope) = 4.105  
#We then call y-axis a count/disribution metameter (by analogy with the use, in bioassay, of "response metameter" and "dose metameter"). The slope of such a theoretical line  
#identifies the main parameter of the theoretical distribution.  
  
source("C:/EDA/Final Exam/poisplot.R")

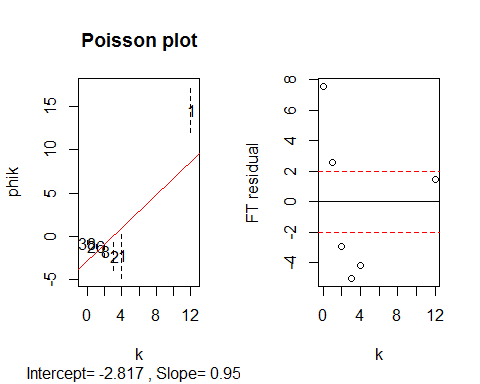
## a b |res|  
## 1 -1.33945 0.17703 2.91358  
## 2 0.00000 0.00000 2.91358  
## 3 0.00000 0.00000 2.91358  
## 4 0.00000 0.00000 2.91358  
## 5 0.00000 0.00000 2.91358  
## -1.33945 0.17703 2.91358



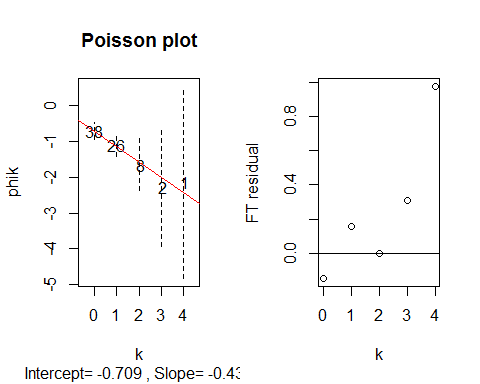
## a b |res|  
## 1 -0.93567 -0.07142 0.30258  
## 2 0.00000 0.00000 0.30258  
## 3 0.00000 0.00000 0.30258  
## 4 0.00000 0.00000 0.30258  
## 5 0.00000 0.00000 0.30258  
## -0.93567 -0.07142 0.30258



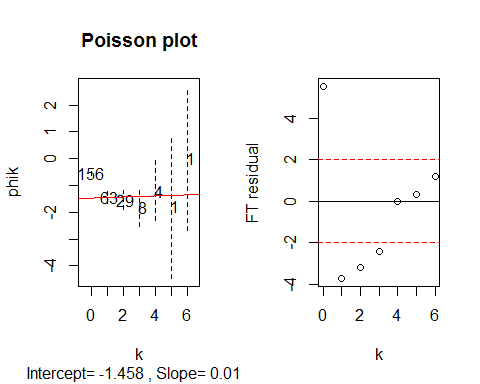
## a b |res|  
## 1 -2.8169 0.95567 15.08206  
## 2 0.0000 0.00000 15.08206  
## 3 0.0000 0.00000 15.08206  
## 4 0.0000 0.00000 15.08206  
## 5 0.0000 0.00000 15.08206  
## -2.8169 0.95567 15.08206



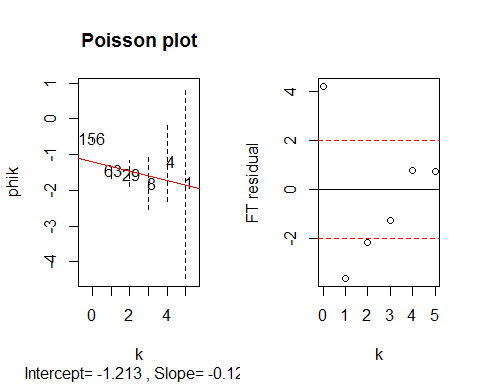
## a b |res|  
## 1 -0.70861 -0.43184 0.66485  
## 2 0.00000 0.00000 0.66485  
## 3 0.00000 0.00000 0.66485  
## 4 0.00000 0.00000 0.66485  
## 5 0.00000 0.00000 0.66485  
## -0.70861 -0.43184 0.66485



## a b |res|  
## 1 -1.45848 0.01949 3.39726  
## 2 0.00000 0.00000 3.39726  
## 3 0.00000 0.00000 3.39726  
## 4 0.00000 0.00000 3.39726  
## 5 0.00000 0.00000 3.39726  
## -1.45848 0.01949 3.39726

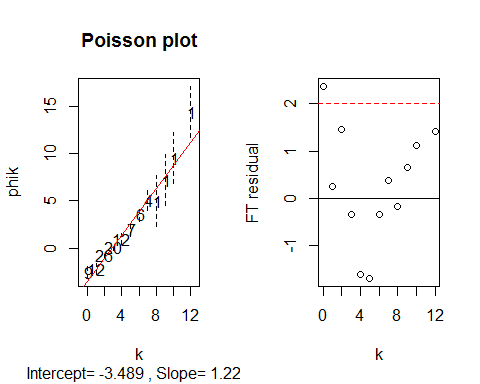


## a b |res|  
## 1 -1.21326 -0.12587 1.64127  
## 2 0.00000 0.00000 1.64127  
## 3 0.00000 0.00000 1.64127  
## 4 0.00000 0.00000 1.64127  
## 5 0.00000 0.00000 1.64127  
## -1.21326 -0.12587 1.64127



poisplot(as.integer(names(count\_table\_data)),x)

## a b |res|  
## 1 -3.48913 1.22315 8.31058  
## 2 0.00000 0.00000 8.31058  
## 3 0.00000 0.00000 8.31058  
## 4 0.00000 0.00000 8.31058  
## 5 0.00000 0.00000 8.31058  
## -3.48913 1.22315 8.31058



If the frequencies are Poisson distributed, then the Freeman-Tukey residuals are approximately normal distributed. When an isolated point strays from an apparently linear pattern of a Poissonness plot, we may want to judge more formally whether it is unlikely to have done so by chance. Potting the residuals against k shows that the fit is quite good, except for the isolated count at k = 0 which does not follow the poisson distribution.

# c)

For an observed frequency ni and the estimated frequency mi, the Freeman-Tukey residual FTi is defined as FTi = sqrt(ni) + sqrt(ni + 1) - sqrt(4mi + 1). Or Freeman-Tukey residuals = sqrt(4\* observed value of k + 2) - sqrt(4\* expected value of k +1) Freeman and Tukey suggest this for a variance stabalizing transformation for Poisson data that leads to using the quantities defined above as residules. For a Poisson random variable X with mean m, Freeman and Tukey (1949) point out that the expected value of sqrt(X) + sqrt(X+1) is well approximated by sqrt(4(n) + 1), and its variance is close to 1. Substituting n for fitted value leads to the residual. sqrt(x) + sqrt(x+1) - sqrt(4 \* fitted value + 1) whose behavior is approximately that of an observation from the standard Gaussian distribution.

All values except for k=0 are reasonable values, they follow poisson distribution because they are within the red-line in FT residual plot.

# Solution 13

r1<-c(16.0, 13.6, 16.2, 14.2, 9.3, 15.1, 10.6, 12.0, 11.3, 10.5, 7.7, 10.6)  
r2<-c(30.4, 27.3, 32.4, 24.1, 27.3, 21.0, 19.2, 22.0, 19.4, 14.9, 11.4, 18.0)  
r3<-c(34.8, 37.1, 40.3, 30.3, 35.0, 38.1, 26.2, 30.6, 25.8, 18.1, 12.3, 17.9)  
r4<-c(37.2, 41.8, 42.1, 34.6, 38.8, 34.0, 30.0, 31.8, 27.9, 18.9, 13.0, 17.9)  
r5<-c(35.3, 40.6, 42.9, 32.5, 38.6, 38.9, 30.9, 32.4, 28.5, 19.5, 12.5, 17.9)  
r6<-c(39.2, 41.4, 43.9, 35.4, 37.5, 39.6, 32.4, 31.1, 28.1, 22.2, 13.7, 18.9)  
r7<-c(39.7, 44.3, 45.5, 38.7, 42.4, 41.4, 35.5, 31.5, 27.8, 21.9, 14.4, 19.9)  
rowNames<-c(95,175,250,350,500,675,1000)  
colNames<-c(0111, 0211, 0311, 0412, 0512, 0612, 0721, 0821, 0921, 1022, 1122, 1222)  
CO2PlantTable <- rbind(r1,r2,r3,r4,r5,r6,r7)  
rownames(CO2PlantTable)<-rowNames  
colnames(CO2PlantTable)<-colNames  
  
  
#a)  
  
medPolished<-medpolish(CO2PlantTable)

## 1: 174.4  
## 2: 162.35  
## Final: 161.5375

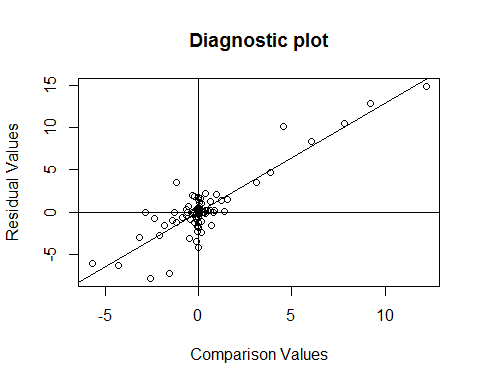
medPolished

##   
## Median Polish Results (Dataset: "CO2PlantTable")  
##   
## Overall: 33.0125  
##   
## Row Effects:  
## 95 175 250 350 500 675 1000   
## -20.1375 -9.9625 -2.0500 0.0000 0.2125 1.3750 3.0000   
##   
## Column Effects:  
## 111 211 311 412 512 612 721 821   
## 3.8375 7.0125 9.3500 1.0500 4.2500 5.2125 -2.3250 -1.0500   
## 921 1022 1122 1222   
## -5.1125 -12.8625 -20.0125 -15.1125   
##   
## Residuals:  
## 111 211 311 412 512 612 721 821  
## 95 -0.7125 -6.2875 -6.0250 0.2750 -7.8250 -2.9875 0.0500 0.1750  
## 175 3.5125 -2.7625 0.0000 0.0000 0.0000 -7.2625 -1.5250 0.0000  
## 250 0.0000 -0.8750 -0.0125 -1.7125 -0.2125 1.9250 -2.4375 0.6875  
## 350 0.3500 1.7750 -0.2625 0.5375 1.5375 -4.2250 -0.6875 -0.1625  
## 500 -1.7625 0.3625 0.3250 -1.7750 1.1250 0.4625 0.0000 0.2250  
## 675 0.9750 0.0000 0.1625 -0.0375 -1.1375 0.0000 0.3375 -2.2375  
## 1000 -0.1500 1.2750 0.1375 1.6375 2.1375 0.1750 1.8125 -3.4625  
## 921 1022 1122 1222  
## 95 3.5375 10.4875 14.8375 12.8375  
## 175 1.4625 4.7125 8.3625 10.0625  
## 250 -0.0500 0.0000 1.3500 2.0500  
## 350 0.0000 -1.2500 0.0000 0.0000  
## 500 0.3875 -0.8625 -0.7125 -0.2125  
## 675 -1.1750 0.6750 -0.6750 -0.3750  
## 1000 -3.1000 -1.2500 -1.6000 -1.0000

#b)  
  
AnalogRSqr<- 1-((sum(abs(medPolished$residuals))) /(sum(abs(CO2PlantTable-medPolished$overall))))  
AnalogRSqr

## [1] 0.8080648

#c)  
  
#The diagnostic plot is a transformation plot for the two way table. Let y\_ij be the response for row i and column j of a   
#a two way table. Decompose the data according to y\_ij = m + a\_i + b\_j +r\_ij where m, a\_i, and b\_j are   
#resistantly determined estimates for the common value, row effects, and column effects, respectively. The diagnostic plot  
#has the comparision values, (a\_i)(b\_j)/m on its horizontal axis and the residuals from the additive fit,  
# r\_ij = y\_ij -(m + a\_i + b\_j) on its vertical axis. When the pattern is roughly linear, 1-slope is the power transformation  
#for the y\_ij to promote additive structure.  
  
x<- vector()  
y<- vector()  
for(i in 1:length(medPolished$row)){  
 for(j in 1:length(medPolished$col)){  
 x<- c(x,(medPolished$row[i] \* medPolished$col[j])/medPolished$overall)  
 }  
   
}  
  
  
residuals<-vector()  
for (i in 1:7){  
 residuals<-c(residuals,medPolished$residuals[i,])  
}  
plot(x,residuals,xlab="Comparison Values",ylab="Residual Values",main="Diagnostic plot")  
abline(h=0,v=0)  
fit<-lm(residuals~x)  
abline(fit)



slope = fit$coefficients[[2]]  
p = 1- slope  
p

## [1] -0.2863723

#d) Yes,We need to do transformation.  
  
#After transformations  
  
CO2PlantTable.transform<-(CO2PlantTable)^(p)  
CO2PlantTable.transform<-matrix(CO2PlantTable.transform,c(7,12))  
dimnames(CO2PlantTable.transform)=list(rowNames,colNames)   
CO2PlantTable.transform.MP <- medpolish(CO2PlantTable.transform)

## 1: 0.6880347  
## 2: 0.5358098  
## Final: 0.5332187

CO2PlantTable.transform.MP

##   
## Median Polish Results (Dataset: "CO2PlantTable.transform")  
##   
## Overall: 0.3707469  
##   
## Row Effects:  
## 95 175 250 350 500   
## 0.107102606 0.035811166 0.006953811 0.000000000 -0.001440291   
## 675 1000   
## -0.004581040 -0.010479745   
##   
## Column Effects:  
## 111 211 311 412 512   
## -0.015845879 -0.022417444 -0.028097032 -0.006078114 -0.018033169   
## 612 721 821 921 1022   
## -0.017455923 0.006819518 0.006078114 0.018541603 0.054793425   
## 1122 1222   
## 0.106413441 0.064395854   
##   
## Residuals:  
## 111 211 311 412 512  
## 95 -0.00996727 0.01813974 0.00067864 -0.0040185 0.06820721  
## 175 -0.01457519 0.00376205 -0.00912497 0.0015224 -0.00062222  
## 250 0.00000000 0.00000000 -0.00263916 0.0048695 0.00159393  
## 350 0.00010848 -0.00497713 0.00000000 -0.0022162 -0.00195978  
## 500 0.00691884 -0.00066078 -0.00040186 0.0057818 0.00000000  
## 675 -0.00059474 0.00055066 0.00049739 0.0000000 0.00606116  
## 1000 0.00403690 -0.00016177 0.00294300 -0.0031758 -0.00028016  
## 612 721 821 921 1022 1122  
## 95 -0.00080034 0.0239362 0.0069264 0.0029847 -0.0226553 -0.0269058  
## 175 0.02906797 0.0156627 0.0000000 0.0026693 0.0000000 -0.0148541  
## 250 -0.00765734 0.0079781 -0.0083475 -0.0020109 0.0038566 0.0032812  
## 350 0.01098190 0.0000000 -0.0055066 -0.0037933 0.0054392 0.0025703  
## 500 -0.00135519 -0.0017423 -0.0060487 -0.0046948 0.0030395 0.0094292  
## 675 0.00000000 -0.0036493 0.0014488 0.0000000 -0.0093911 0.0000000  
## 1000 0.00148785 -0.0072897 0.0059824 0.0070830 -0.0018857 -0.0007974  
## 1222  
## 95 -0.03364018  
## 175 -0.03391038  
## 250 -0.00435522  
## 350 0.00259860  
## 500 0.00403889  
## 675 0.00041785  
## 1000 0.00000000

MedianPolishdata<-rbind(CO2PlantTable.transform,CO2PlantTable.transform.MP$col)  
MedianPolishdata<-cbind(MedianPolishdata,CO2PlantTable.transform.MP$row)  
colnames(MedianPolishdata)[13]<-"Row Effect"  
rownames(MedianPolishdata)[8]<-"Column Effect"  
MedianPolishdata[8,13]<-medPolished$overall  
  
#After transformation  
MedianPolishdata

## 111 211 311 412 512  
## 95 0.45203632 0.47357177 0.45043108 0.467752894 0.52802352  
## 175 0.37613696 0.38790264 0.36933603 0.402002299 0.38790264  
## 250 0.36185480 0.35528323 0.34696449 0.376492040 0.36126144  
## 350 0.35500947 0.34335229 0.34264983 0.362452556 0.35075392  
## 500 0.36037954 0.34622835 0.34080768 0.369010237 0.35127341  
## 675 0.34972520 0.34429905 0.33856619 0.360087712 0.35419381  
## 1000 0.34845814 0.33768790 0.33511309 0.351013230 0.34195379  
## Column Effect -0.01584588 -0.02241744 -0.02809703 -0.006078114 -0.01803317  
## 612 721 821 921 1022  
## 95 0.45959321 0.508605150 0.490854000 0.4993758 0.50998761  
## 175 0.41817008 0.429040229 0.412636145 0.4277689 0.46135146  
## 250 0.35258741 0.392498256 0.375431293 0.3942313 0.43635066  
## 350 0.36427284 0.377566384 0.371318357 0.3854952 0.43097952  
## 500 0.35049546 0.374383840 0.369336034 0.3831534 0.42713953  
## 675 0.34870990 0.369336034 0.373692778 0.3847074 0.41156813  
## 1000 0.34429905 0.359796943 0.372327652 0.3858918 0.41317485  
## Column Effect -0.01745592 0.006819518 0.006078114 0.0185416 0.05479343  
## 1122 1222 Row Effect  
## 95 0.5573571 0.50860515 0.107102606  
## 175 0.4981174 0.43704350 0.035811166  
## 250 0.4873953 0.43774131 0.006953811  
## 350 0.4797306 0.43774131 0.000000000  
## 500 0.4851492 0.43774131 -0.001440291  
## 675 0.4725793 0.43097952 -0.004581040  
## 1000 0.4658832 0.42466297 -0.010479745  
## Column Effect 0.1064134 0.06439585 33.012500000

sum\_res<-sum(abs(CO2PlantTable.transform.MP$residual))  
sum\_data<-sum(abs(CO2PlantTable.transform-CO2PlantTable.transform.MP$overall))  
  
Analogrsquare<-1-(sum\_res/sum\_data)  
#Analog R square after Transformation  
Analogrsquare

## [1] 0.8618704

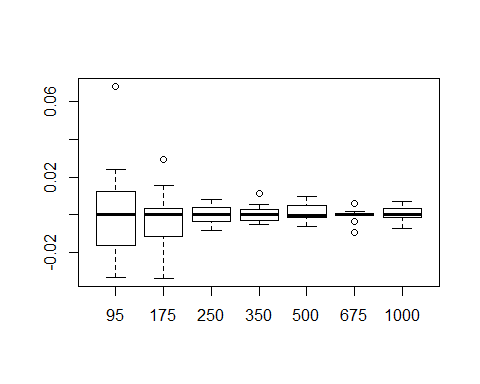
#e)  
library(aplpack)

## Loading required package: tcltk

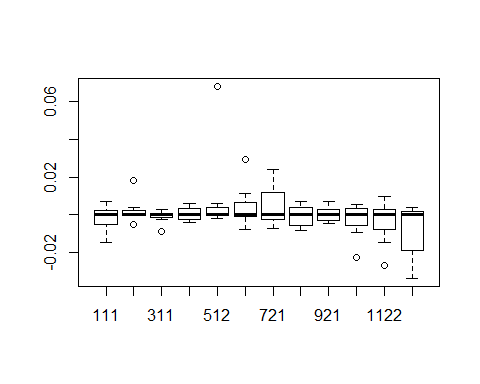
stem.leaf(CO2PlantTable.transform.MP$residuals, m=2)

## 1 | 2: represents 0.012  
## leaf unit: 0.001  
## n: 84  
## LO: -0.0339103826926934 -0.0336401751631705 -0.0269057703406803 -0.0226552878284819  
## 6 -1\* | 44  
## 14 -0. | 99987765  
## 36 -0\* | 4444333222111100000000  
## (33) 0\* | 000000000000000001111222223333444  
## 15 0. | 555666779  
## 6 1\* | 0  
## 5 1. | 58  
## 3 2\* | 3  
## HI: 0.0290679739452966 0.0682072145749605

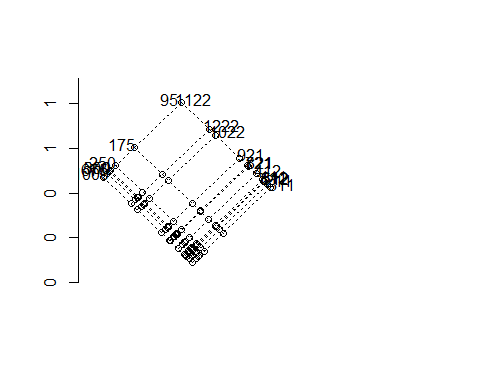
#Yes, there are few outliers.  
  
#f)  
#Boxplot along rows  
boxplot(t(CO2PlantTable.transform.MP$residuals))



#Boxplot along columns   
boxplot(CO2PlantTable.transform.MP$residuals)



#g)  
  
source("myplotfit.r")  
myplotfit(CO2PlantTable.transform.MP)



#1.Plant Combination has the largest effect than CO2 level.  
  
#2.The highest combination of influence is of (95,1122) with a value of 1.  
  
#h)  
vect<-function(res\_mp){  
 res\_ls <- c(res\_mp)  
 res\_ret <- c()  
 for (i in 1:nrow(res\_mp)){  
 res\_ret <- rbind(res\_ret,c(sample(res\_ls,ncol(res\_mp),replace = TRUE)))  
 }  
 return (res\_ret)  
}  
  
boot<-function(matrix,n){  
 nrows <- nrow(matrix)  
 ncols <- ncol (matrix)  
 row.est <- matrix(0,nrow=n,ncol=nrows)  
 col.est <- matrix(0,nrow=n,ncol=ncols)  
 overall.est <- c()  
 result<-medpolish(matrix)  
 sample.matrix <- result$residuals  
 temp.result <- result  
 for (j in 1:n){  
 new\_res <- vect(sample.matrix)  
 combinedResults<-rbind(temp.result$row,temp.result$row,temp.result$row,temp.result$row,temp.result$row,  
 temp.result$row,temp.result$row,temp.result$row,  
 temp.result$row,temp.result$row,temp.result$row,temp.result$row)  
 bs.matrix <- new\_res+t(combinedResults)+  
 sapply(c(temp.result$col),function(x) rep(x,nrow(temp.result$residuals)))+matrix(temp.result$overall,  
 nrow=length(temp.result$row),ncol=length(temp.result$col))  
 temp.result <- medpolish(bs.matrix,maxiter = 1000)  
 sample.matrix<-temp.result$residuals  
 row.est[j,]<-c(temp.result$row)  
 col.est[j,]<-c(temp.result$col)  
 overall.est[j]<-temp.result$overall  
 }  
 return(list(row.est=row.est,col.est=col.est,overall.est=overall.est))  
}  
  
b<-boot(CO2PlantTable,50)

## 1: 174.4  
## 2: 162.35  
## Final: 161.5375  
## 1: 118.0187  
## 2: 113.375  
## Final: 113.2609  
## 1: 119.3125  
## 2: 99.36523  
## Final: 99.21387  
## 1: 116.7289  
## 2: 106.919  
## Final: 106.6554  
## 1: 97.22419  
## 2: 68.43981  
## 3: 67.57158  
## Final: 67.42971  
## 1: 79.143  
## 2: 68.77692  
## 3: 67.52093  
## Final: 67.43951  
## 1: 79.84152  
## 2: 60.22981  
## 3: 59.44585  
## Final: 59.10548  
## 1: 71.99147  
## 2: 71.19046  
## Final: 71.16129  
## 1: 76.80047  
## 2: 66.47459  
## Final: 66.14265  
## 1: 57.91469  
## 2: 51.89977  
## Final: 51.82797  
## 1: 93.93413  
## 2: 61.4345  
## 3: 60.02752  
## Final: 59.71607  
## 1: 45.50909  
## 2: 41.93218  
## Final: 41.89986  
## 1: 50.9619  
## 2: 42.78889  
## Final: 42.60972  
## 1: 47.62396  
## 2: 33.5362  
## 3: 32.97154  
## Final: 32.88575  
## 1: 65.49937  
## 2: 50.17633  
## 3: 49.62051  
## Final: 49.43938  
## 1: 62.94318  
## 2: 50.01298  
## Final: 49.93553  
## 1: 107.4571  
## 2: 62.29278  
## 3: 59.74787  
## Final: 59.72408  
## 1: 103.9295  
## 2: 64.58268  
## Final: 64.2728  
## 1: 70.50497  
## 2: 45.36378  
## Final: 45.0568  
## 1: 72.92226  
## 2: 46.62698  
## Final: 46.20824  
## 1: 58.65827  
## 2: 44.619  
## Final: 44.509  
## 1: 54.49549  
## 2: 40.77627  
## 3: 39.99131  
## Final: 39.89506  
## 1: 43.10281  
## 2: 36.89106  
## Final: 36.72268  
## 1: 30.78649  
## 2: 26.78011  
## Final: 26.65243  
## 1: 18.53562  
## 2: 17.17211  
## Final: 17.10594  
## 1: 16.83933  
## 2: 15.95056  
## Final: 15.88654  
## 1: 15.21844  
## 2: 14.7597  
## Final: 14.74291  
## 1: 22.59122  
## 2: 14.07392  
## Final: 14.00145  
## 1: 10.13641  
## 2: 9.574255  
## 3: 9.473013  
## Final: 9.468716  
## 1: 12.05512  
## 2: 8.783373  
## Final: 8.735793  
## 1: 10.98701  
## 2: 8.369837  
## 3: 8.272271  
## Final: 8.264271  
## 1: 14.00538  
## 2: 7.396654  
## 3: 7.246312  
## Final: 7.216335  
## 1: 7.35692  
## 2: 6.487726  
## Final: 6.471049  
## 1: 7.238984  
## 2: 6.183737  
## Final: 6.157166  
## 1: 6.586515  
## 2: 4.625552  
## Final: 4.585475  
## 1: 6.114613  
## 2: 3.998895  
## Final: 3.970021  
## 1: 3.893418  
## 2: 3.715802  
## Final: 3.697746  
## 1: 3.823885  
## 2: 3.508286  
## 3: 3.465569  
## Final: 3.455238  
## 1: 4.087223  
## 2: 3.23861  
## Final: 3.2117  
## 1: 3.049722  
## 2: 2.657624  
## Final: 2.633944  
## 1: 3.388721  
## 2: 2.32255  
## Final: 2.315405  
## 1: 2.581473  
## 2: 1.936364  
## 3: 1.894659  
## Final: 1.889738  
## 1: 1.943159  
## 2: 1.687051  
## Final: 1.673187  
## 1: 1.93884  
## 2: 1.487529  
## Final: 1.47538  
## 1: 1.886293  
## 2: 1.225741  
## 3: 1.208136  
## Final: 1.202979  
## 1: 1.040581  
## 2: 1.022608  
## Final: 1.01816  
## 1: 0.9596957  
## 2: 0.8909307  
## Final: 0.8900149  
## 1: 1.760856  
## 2: 1.06277  
## Final: 1.062274  
## 1: 1.8667  
## 2: 1.05027  
## Final: 1.045945  
## 1: 1.882564  
## 2: 1.030434  
## Final: 1.024797  
## 1: 1.661901  
## 2: 1.249711  
## Final: 1.248223

overallSDerr<-sd(b$overall.est)/sqrt(length(CO2PlantTable))  
print(paste("overall : ",mean(b$overall.est)))

## [1] "overall : 30.8294399839823"

print(paste("Standard Error:",overallSDerr))

## [1] "Standard Error: 0.0562643384937636"