Exploratory Data Analysis - Homework 5

Ganesh Nagarajan, gnagaraj@indiana.edu

October 30, 2015

1. Median Polish Column first

```
#Input the columns
c1<-c(25.3,32.1,38.8,25.4)
c2<-c(25.3,29.1,31.0,21.1)
c3<-c(18.2,18.8,19.3,20.3)
c4<-c(18.3,24.3,15.7,24.0)
c5<-c(16.3,19.0,16.8,17.5)
mat<-cbind(c1,c2,c3,c4,c5)

twoway.median2 <- function(mat){ # first column then row
  meff.MP <- median(mat)
  beff.MP <- apply(mat,2,median,na.rm=T)  # column medians
  mat.res <- mat - matrix(rep(beff.MP,each=nrow(mat)),byrow=F,nrow=nrow(mat));
  aeff.MP <- apply(mat.res,1,median,na.rm=T) # row effect
  beff.MP <- beff.MP - median(beff.MP)  # column effect
  res.MP <- mat.res - matrix(rep(aeff.MP,each=ncol(mat)),byrow=T,ncol=ncol(mat))
  list(overall=meff.MP, row=aeff.MP, col=beff.MP, res=res.MP)
}

lv1<-twoway.median2(mat)
lv1
```

```
## $overall
## [1] 20.7
##
## $row
## [1] -1.90  1.90  0.25  0.35
##
## $col
##    c1    c2    c3    c4    c5
##  7.60  6.05 -2.10  0.00 -4.00
##
## $res
##         c1    c2    c3    c4    c5
## [1,] -1.55  0.00  1.05 -0.95  1.05
## [2,]  1.45  0.00 -2.15  1.25 -0.05
## [3,]  9.80  3.55  0.00 -5.70 -0.60
## [4,] -3.70 -6.45  0.90  2.50  0.00
```

```
lv2<-twoway.median2(lv1$res)
lv2
```

```
## $overall
## [1] 0
##
```

```
## $row
## [1]  0.000  0.000 -0.450  0.025
##
## $col
##     c1     c2     c3     c4     c5
## -0.050  0.000  0.450  0.150 -0.025
##
## $res
##          c1     c2     c3     c4     c5
## [1,] -1.500  0.000  0.600 -1.100  1.075
## [2,]  1.500  0.000 -2.600  1.100 -0.025
## [3,] 10.300  4.000  0.000 -5.400 -0.125
## [4,] -3.675 -6.475  0.425  2.325  0.000
```

```
overallRow<-lv1$row+lv2$row
print(overallRow)
```

```
## [1] -1.900  1.900 -0.200  0.375
```

```
overallCol<-lv1$col+lv2$col
print(overallCol)
```

```
##     c1     c2     c3     c4     c5
##  7.550  6.050 -1.650  0.150 -4.025
```

```
overall<-lv1$overall+lv2$overall
print(overall)
```

```
## [1] 20.7
```

```
stem(c(lv2$res),2)
```

```
##
##   The decimal point is at the |
##
##   -6 | 5
##   -4 | 4
##   -2 | 76
##   -0 | 5110
##    0 | 000046115
##    2 | 3
##    4 | 0
##    6 |
##    8 |
##   10 | 3
```

It can be seen that after two iterations, both the row first and column first median polish results in not extremely different, but similar overall and row/column effects.

2.Personal Consumption Expenditures Dataset

```
diag.MP <- function(fit){
  fit.comp <- matrix(fit$row,ncol=1) %*% matrix(fit$col,nrow=1)/fit$overall
  plot(fit.comp, fit$res,xlab="Comparison value",ylab="Residual",cex=0.5)
  abline(v=0,h=0,lty=2)
  ls <- lm(c(fit$res)~c(fit.comp))
  abline(ls,col="red",lty=3)
  rr <- run.rrline(fit.comp,fit$res,iter=10)
  abline(rr$a, rr$b, col="red")
  pwr1 <- 1 - rr$b
  pwr2 <- 1 - ls$coef[2]
  title("",paste("Approximate power =",format(round(pwr1,2))," or ", format(round(pwr2,2))))
}

symbolPlot<-function(mat){
  result<-medpolish(mat)
  res<-c(result$residuals)
  genNos<-expand.grid(1:5,1:5)
  plotvar<-cbind(genNos$Var2,genNos$Var1,res)
  pos<-plotvar[plotvar[,3]>=0,]
  max<-sum(abs(pos[,3]))
  symbols(pos[,1],pos[,2],squares = 0.2*(abs(pos[,3]/(max))),inches = FALSE,xlab="Columns",ylab="Rows",
  pos<-plotvar[plotvar[,3]<0,]
  symbols(pos[,1],pos[,2],circles = 0.2*(abs(pos[,3]/(max))),inches = FALSE,add = TRUE)
}

rrline1 <- function(x,y) {
  n3 <- floor((length(x)+1.99)/3)
  x.order <- order(x)
  medxL <- median(x[x.order][1:n3])
  medxR <- median(rev(x[x.order])[1:n3])
  medyL <- median(y[x.order][1:n3])
  medyR <- median(rev(y[x.order])[1:n3])
  slope1 <- (medyR - medyL)/(medxR - medxL)
  int1 <- median(y - slope1 * x)
  # print(c(paste("Intercept = ", format(round(int1,5))),
  #   paste("Slope = ",format(round(slope1,5)))))
  newy <- y - slope1*x - int1
  sumres <- sum(abs(newy))
  list(a=int1, b=slope1, sumres = sumres, res=newy)
}

run.rrline <- function(x,y,iter=5) {
  out.coef <- matrix(0,iter,3)
  newy <- y
  for (i in 1:iter) {
    rr <- rrline1(x,newy)
    out.coef[i,] <- c(rr$a,rr$b,rr$sumres)
    newy <- rr$res
  }
  dimnames(out.coef) <- list(format(1:iter),c("a","b","|res|"))
  aa <- sum(out.coef[,1])
  bb <- sum(out.coef[,2])
  cc <- sum(abs(y - aa - bb*x))
```

```r
  res <- y - aa - bb*x
  out.coef <- rbind(out.coef,c(aa,bb,cc))
  print(round(out.coef,5))
  list(a = aa, b = bb, res = res, coef=out.coef)
}

rrline2 <- function(x,y) {
  n <- length(x)
  n3 <- floor((length(x)+1.99)/3)
  x.order <- order(x)
  medxL <- median(x[x.order][1:n3])
  medxR <- median(rev(x[x.order])[1:n3])
  medyL <- median(y[x.order][1:n3])
  medyR <- median(rev(y[x.order])[1:n3])
  medxM <- median(x[x.order][(n3+1):(n-n3)])
  medyM <- median(y[x.order][(n3+1):(n-n3)])
  slope1 <- (medyR - medyL)/(medxR - medxL)
  int1 <- median(y - slope1 * x)
  int2 <- mean(c(medyL,medyM,medyR) - slope1*c(medxL,medxM,medxR))
  newy <- y - slope1*x - int1
  sumres <- sum(abs(newy))
  newy2 <- y - slope1*x - int2
  sumres2 <- sum(abs(newy2))
  list(a=int1, a2=int2, b=slope1, sumres = sumres, sumres2=sumres2, res=newy2)
}

run.rrline2 <- function(x,y,iter=5) {
  out.coef <- matrix(0,iter,3)
  newy <- y
  for (i in 1:iter) {
    rr <- rrline2(x,newy)
    out.coef[i,] <- c(rr$a2,rr$b,rr$sumres2)
    newy <- rr$res
  }
  dimnames(out.coef) <- list(format(1:iter),c("a","b","|res|"))
  aa <- sum(out.coef[,1])
  bb <- sum(out.coef[,2])
  cc <- sum(abs(y - aa - bb*x))
  res <- y - aa - bb*x
  out.coef <- rbind(out.coef,c(aa,bb,cc))
  print(round(out.coef,5))
  list(a = aa, b = bb, res = res, coef=out.coef)
}
# Input Vector
C1<-c(22.2,44.5,59.6,73.2,86.8)
C2<-c(10.5,15.5,29.0,36.5,46.2)
C3<-c(3.53,5.76,9.71,14.0,21.1)
C4<-c(1.04,1.98,2.45,3.40,5.40)
C5<-c(.641,.974,1.80,2.60,3.64)
mat<-cbind(C1,C2,C3,C4,C4)
result<-medpolish(mat)
```

```
## 1: 138.44
```

```
## Final: 138.44
```

```r
result
```

```
##
## Median Polish Results (Dataset: "mat")
##
## Overall: 9.71
##
## Row Effects:
## [1] -6.18 -3.95  0.00  4.29 11.39
##
## Column Effects:
##    C1    C2    C3    C4    C4
## 49.89 19.29  0.00 -7.26 -7.26
##
## Residuals:
##           C1     C2 C3    C4    C4
## [1,] -31.22 -12.32  0  4.77  4.77
## [2,] -11.15  -9.55  0  3.48  3.48
## [3,]   0.00   0.00  0  0.00  0.00
## [4,]   9.31   3.21  0 -3.34 -3.34
## [5,]  15.81   5.81  0 -8.44 -8.44
```

Analog R square calcuation

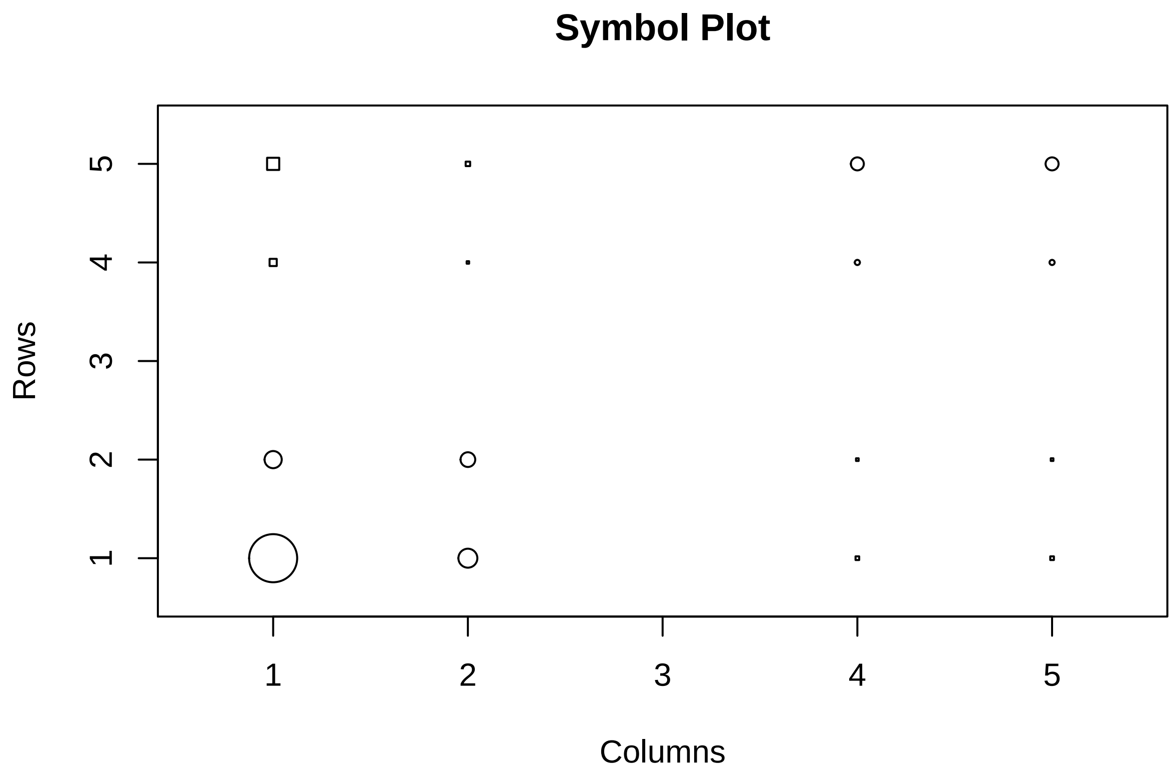
```r
Analog_R_Square<- 1-((sum(abs(result$residuals))) /(sum(abs(mat-result$overall))))
Analog_R_Square
```

```
## [1] 0.6713747
```

b) Symbol Plot. The positive are given square, negatives are given circle. Yes there are patterns. Main diagonal elements have positive residuals, other side has negative elements.

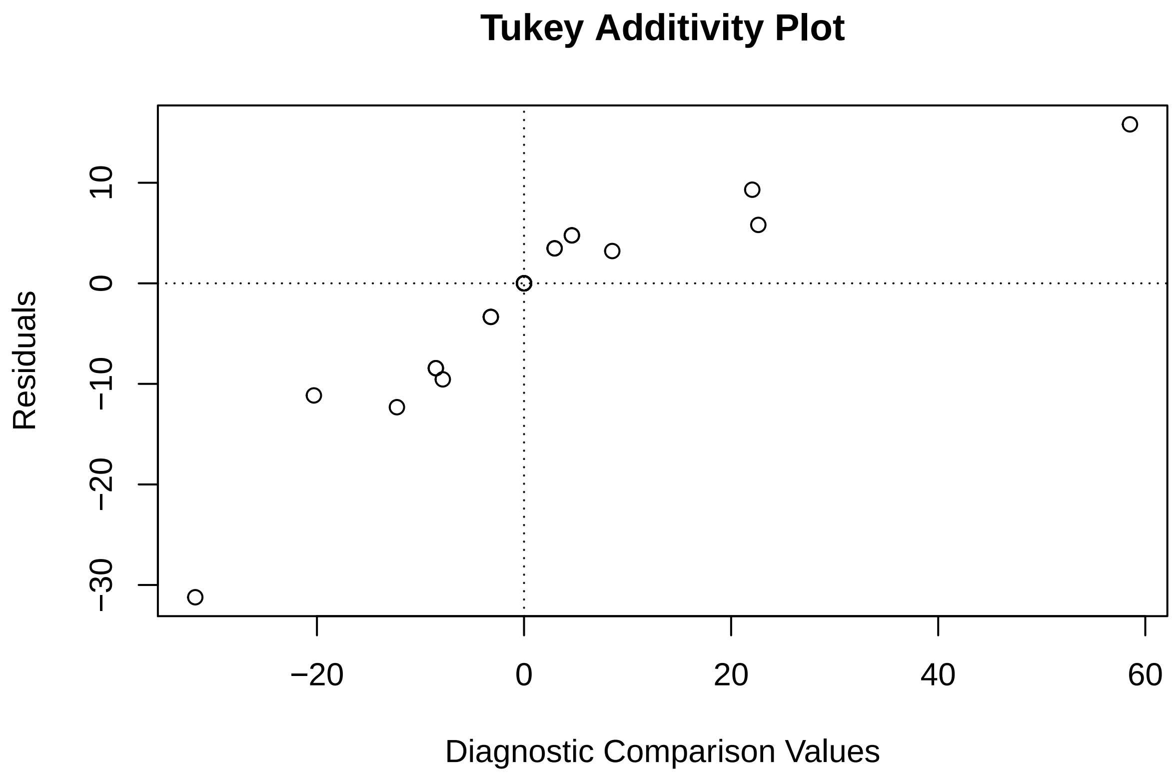
```r
symbolPlot(mat)
```

```
## 1: 138.44
## Final: 138.44
```

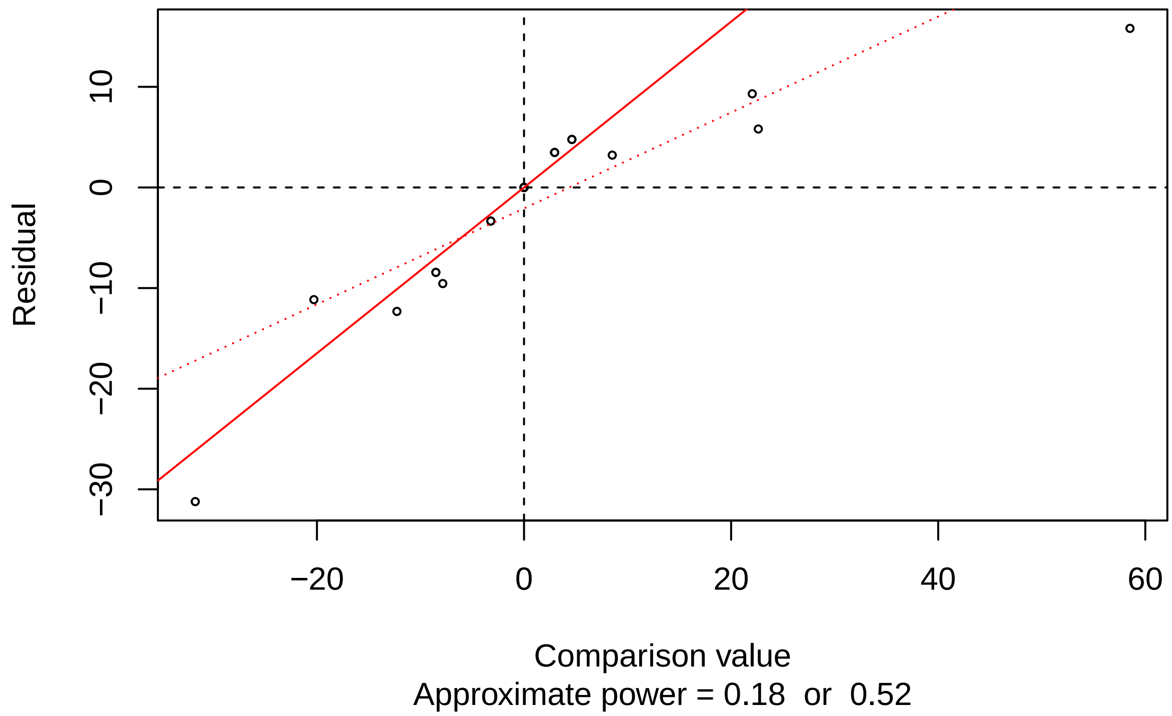


c) Diagnostic plot It can be noted the residuals are not along the zero axis. Hence must be transformed.

```
plot(result)
```




```
diag.MP(result)
```




```
##     a         b    |res|
##   1 0   0.91233 85.75497
##   2 0  -0.08848 82.07972
##   3 0   0.00000 82.07972
##   4 0   0.00000 82.07972
##   5 0   0.00000 82.07972
##   6 0   0.00000 82.07972
##   7 0   0.00000 82.07972
##   8 0   0.00000 82.07972
##   9 0   0.00000 82.07972
##  10 0   0.00000 82.07972
##      0   0.82385 82.07972
```

d) According to the Analog R2, log transform is suggested. Following is the log transform

```
mat<-log(mat)
result<-medpolish(mat)
```

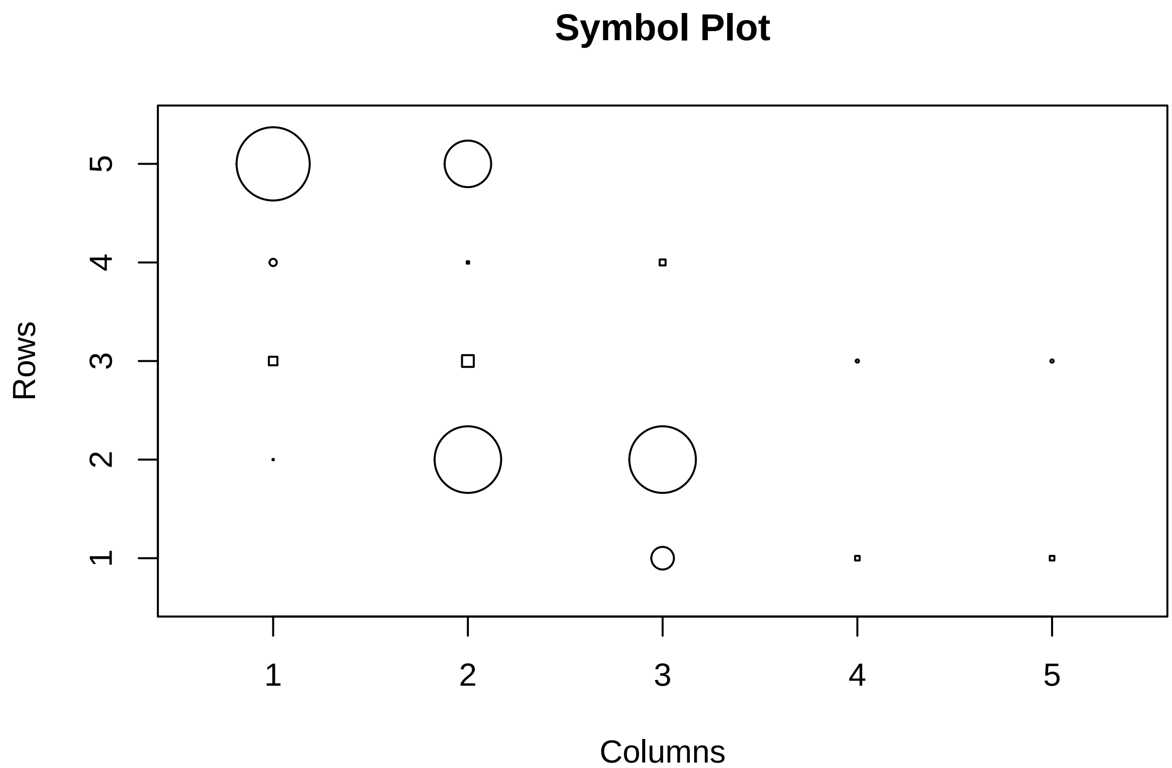
```
## 1: 2.261994
## 2: 1.691594
## 3: 1.626613
## Final: 1.626613
```

```r
result
```

```
##
## Median Polish Results (Dataset: "mat")
##
## Overall: 2.273156
##
## Row Effects:
## [1] -0.9116836 -0.2271854  0.0000000  0.3134932  0.7761168
##
## Column Effects:
##         C1         C2         C3         C4         C4
##  1.7386196  0.9899025  0.0000000 -1.3628741 -1.3628741
##
## Residuals:
##             C1       C2        C3        C4        C4
## [1,]  0.000000  0.00000 -0.100175  0.040622  0.040622
## [2,]  0.010899 -0.29503 -0.295033  0.000000  0.000000
## [3,]  0.075880  0.10424  0.000000 -0.014194 -0.014194
## [4,] -0.032074  0.02076  0.052408  0.000000  0.000000
## [5,] -0.324286 -0.20620  0.000000  0.000000  0.000000
```

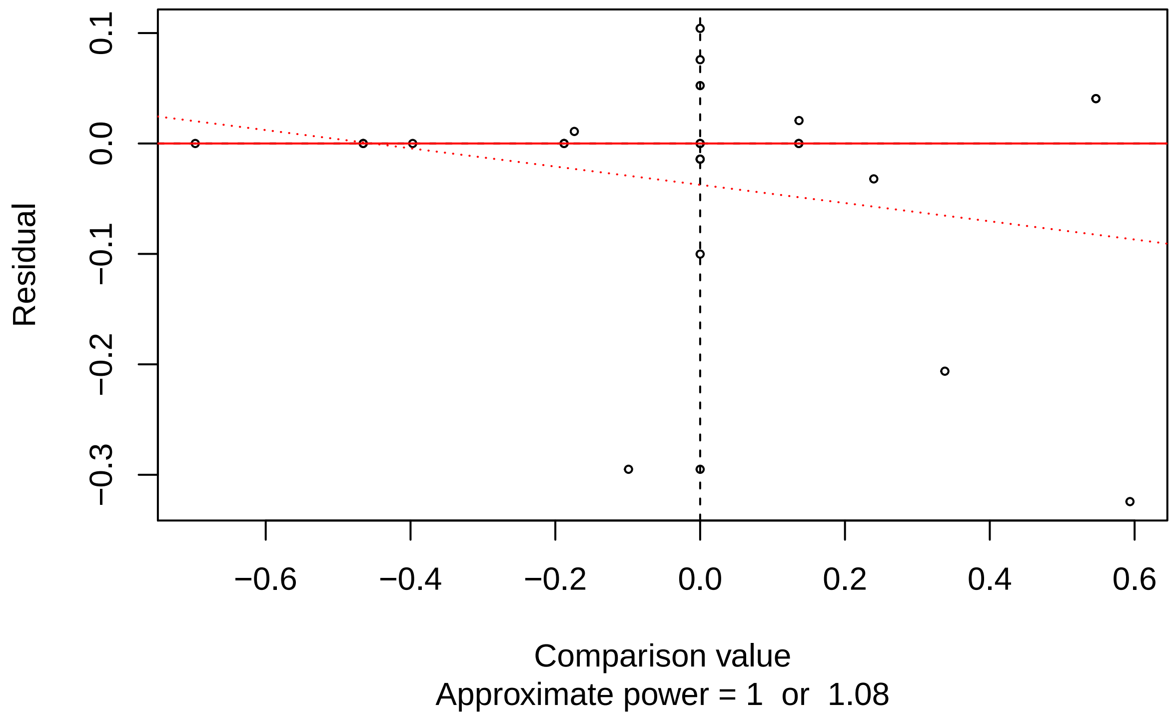
```r
symbolPlot(mat)
```

```
## 1: 2.261994
## 2: 1.691594
## 3: 1.626613
## Final: 1.626613
```



It can be clearly seen that after transformation, most of the residuals are along zeroes, hence is a good fit.

```
diag.MP(result)
```




```
##     a b    |res|
##   1 0 0 1.62661
##   2 0 0 1.62661
##   3 0 0 1.62661
##   4 0 0 1.62661
##   5 0 0 1.62661
##   6 0 0 1.62661
##   7 0 0 1.62661
##   8 0 0 1.62661
##   9 0 0 1.62661
##  10 0 0 1.62661
##     0 0 1.62661
```

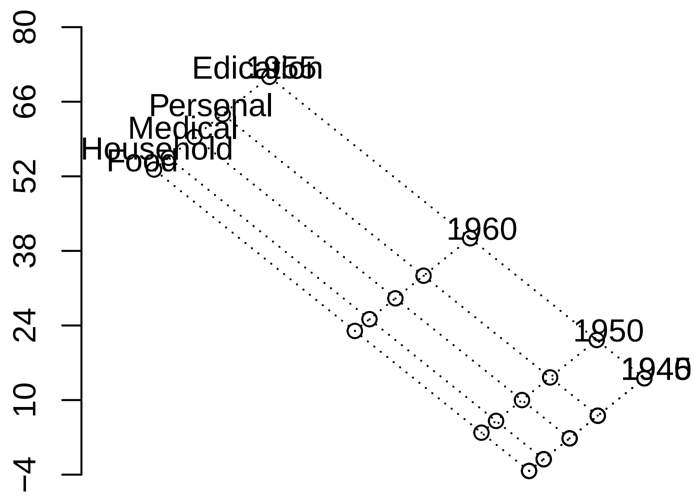
e) Forget it plot - Before and After transformation:

```
forgetitplot <- function(outmpol,outlim=0,...) {
  # outmpol is output of medpolish in library(eda) or library(stats)
  # be sure to assign dimnames to matrix being polished
  oldpar <- par()
  par(fig=c(0,.7,0,1))
  nc <- length(outmpol$col)
  nr <- length(outmpol$row)
  a <- rep(outmpol$row,nc)
 b <- rep(outmpol$col,rep(nr,nc))
  sqrt2 <- sqrt(2)
```

```
  ab <- cbind((a-b)/sqrt2,(a+b)/sqrt2)
  xrange <- range(ab[,1]) + c(-.1,.1)*(max(ab[,1])-min(ab[,1]))
  yrange <- range(ab[,2]) + c(-.1,.1)*(max(ab[,2])-min(ab[,2]))
  dx <- (xrange[2]-xrange[1])/50
  dy <- (yrange[2]-yrange[1])/50
  plot(ab[,1],ab[,2],axes=F,xlim=xrange,ylim=yrange,xlab="",ylab="",...)
  segments((min(a)-outmpol$col)/sqrt2, (min(a)+outmpol$col)/sqrt2,
           (max(a)-outmpol$col)/sqrt2, (max(a)+outmpol$col)/sqrt2,lty=3)
  segments((outmpol$row-min(b))/sqrt2, (outmpol$row+min(b))/sqrt2,
           (outmpol$row-max(b))/sqrt2, (outmpol$row+max(b))/sqrt2,lty=3)
  # segments((outmpol$row)/sqrt2-min(b), (outmpol$row)/sqrt2+min(b),
  #          (outmpol$row)/sqrt2-max(b), (outmpol$row)/sqrt2+max(b),lty=3)
  yrowloc <-  rep(max(b),nr)
  xrowloc <-  outmpol$row
  # text((xrowloc-yrowloc)/sqrt2-dx,dy+(xrowloc+yrowloc)/sqrt2,format(1:nr))
  text((xrowloc-yrowloc)/sqrt2-dx,dy+(xrowloc+yrowloc)/sqrt2,
       names(sort(outmpol$row)))
  xcolloc <- rep(max(a),nc)
  ycolloc <- outmpol$col
  # text(dx+(xcolloc-ycolloc)/sqrt2,dy+(xcolloc+ycolloc)/sqrt2,format(1:nc))
  text(dx+(xcolloc-ycolloc)/sqrt2,dy+(xcolloc+ycolloc)/sqrt2,
       names(sort(outmpol$col)))
  ynames <- format(round(outmpol$overall + sqrt2*pretty(ab[,2])))
  axis(2,at=pretty(ab[,2]),labels=ynames)
  # add vertical lines when there is an outlier
  if(abs(outlim) > 1e-4) {
    out.index <- which(abs(outmpol$res) > outlim, arr.ind=T)
    # find (r,c) for outlier indices
    zz.x <- outmpol$row[out.index[,1]]
    zz.y <- outmpol$col[out.index[,2]]
    # outlier points at (zz.x-zz.y)/sqrt2, (zz.x+zz.y)/sqrt2
    # draw segment from here to end of residual
    segments((zz.x-zz.y)/sqrt2, (zz.x+zz.y)/sqrt2,
             (zz.x-zz.y)/sqrt2, (zz.x+zz.y)/sqrt2 + outmpol$res[out.index])
  }
  par <- oldpar
  invisible()
}
C1<-c(22.2,44.5,59.6,73.2,86.8)
C2<-c(10.5,15.5,29.0,36.5,46.2)
C3<-c(3.53,5.76,9.71,14.0,21.1)
C4<-c(1.04,1.98,2.45,3.40,5.40)
C5<-c(.641,.974,1.80,2.60,3.64)
mat<-cbind(C1,C2,C3,C4,C4)
rownames(mat)<-c("Food","Household","Medical","Personal","Edication")
colnames(mat)<-c(1940,1945,1950,1955,1960)
result<-medpolish(mat)
```

```
## 1: 138.44
## Final: 138.44
```

```r
forgetitplot(result)
```



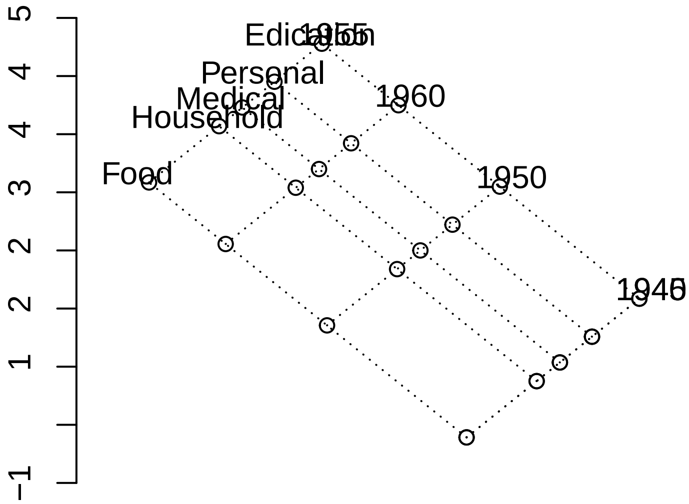
It can be seen that since year has lot of elongations and more distributed than type. Hence year/time has larger effect on the data.

After Transformation

```r
mat<-log(mat)
result<-medpolish(mat)
```

```
## 1: 2.261994
## 2: 1.691594
## 3: 1.626613
## Final: 1.626613
```

```r
forgetitplot(result)
```



It can be clear seen that after the transformation, the row and column effects are now spreaded than evenly than before the transformation. Hence gives a uniform effect rather than particular points influencing the median polish.

3.Creating the simulated dataset

```
set.seed(1234)
#ti generator
ti<-function(t){((2*t)-1)/100}
t<-sapply(1:50,ti)

#mu-i generator
mui<-function(t){t+0.5*exp(-50*(t-0.5)^2)}
mu<-sapply(t,mui)

#Random Noise
e<-rnorm(50,0,0.5)

#Total function y
y<-(mu*t)+e

#bind them
testDS<-as.data.frame(cbind(t,y))

#Meeting with Dr.king,
u = expression(t + 0.5*exp(-50*t^2 - 12.5 + 50*t))
```

```r
uder2 = D(D(u, 't'), 't')
ff=function(t){eval({t=t;uder2})}
gg=function(t){ff(t)^2}
j = unlist(integrate(gg,0,1))
j = j[[1]]
r = 1/(2*sqrt(pi))
l = 50^(-1/5)*((0.25*r)/(j*1))^(1/5)

#Print the computed l
print(l)
```

```
## [1] 0.08424765
```

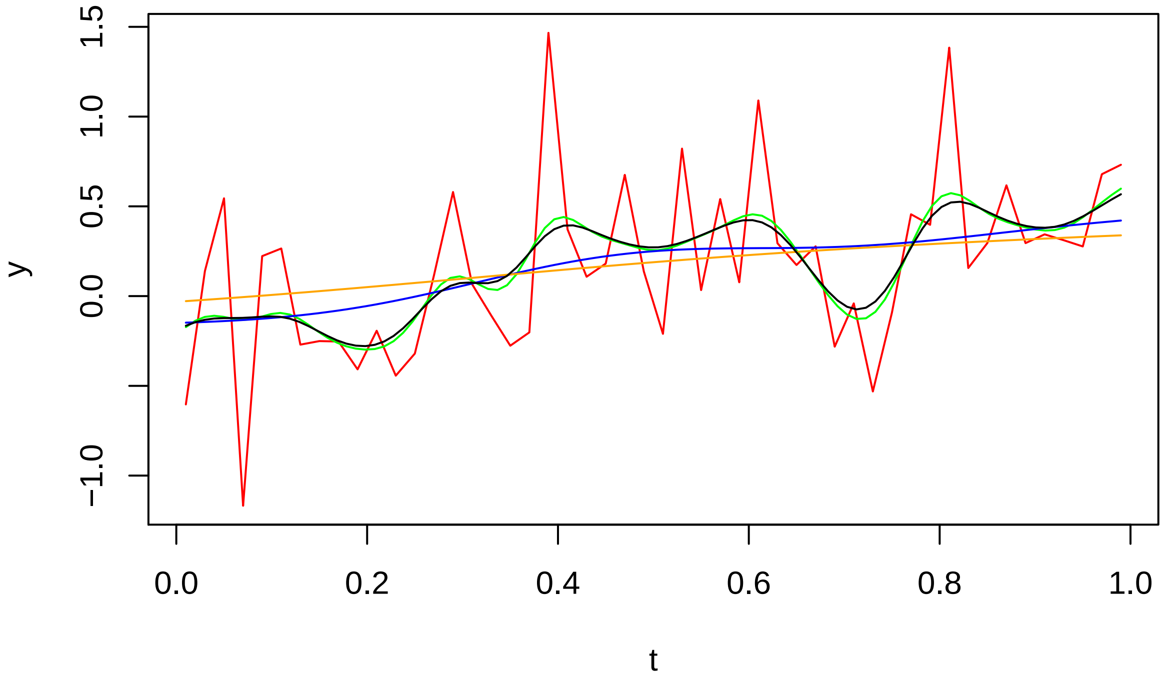
The above is the estimated lambda optimal.

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\left[t + 0.5\mathrm{e}^{-50(t-0.5)^2}\right] = \left(5000t^2 - 5000t + 1200\right)\mathrm{e}^{-\frac{100t^2-100t+25}{2}}$$

$$\int_0^1 \left(5000t^2 - 5000t + 1200\right)\mathrm{e}^{-\frac{100t^2-100t+25}{2}}\left(5000t^2 - 5000t + 1200\right)\mathrm{e}^{-\frac{100t^2-100t+25}{2}}\,\mathrm{d}t =$$

$$\frac{\sqrt{\pi}\,(288000\,\mathrm{erf}\,(5) - 287625) - 500\,\Gamma\left(\frac{5}{2}, 25\right) + 1000\,\Gamma\left(\frac{3}{2}, 25\right) + 287500\,\Gamma\left(\frac{1}{2}, 25\right)}{2} = 332.33$$

```r
a = ksmooth(t, y, kernel = "normal", l)
b = ksmooth(t, y, kernel = "normal", 0.4)
c = ksmooth(t, y, kernel = "normal", 0.8)
d = ksmooth(t, y, kernel = "normal", 0.1)
plot(t, y, type = "l", col = "red")
lines(a$x, a$y, col = "green", type = "l")
lines(b$x, b$y, col = "blue", type = "l")
lines(c$x, c$y, col = "orange", type = "l")
lines(d$x, d$y, col = "black", type = "l")
```



The green line is the actual value that was lambda optimal. The it is observed that when the lambda is less, the line seems to be curvy, however a higher lambda straightens the line.

Acknowledgements 1. Online integration and differential calculator http://www.derivative-calculator.net/ and http://www.integral-calculator.com/