Supplementary Materials for "Randomized Predictive P-values: A Versatile Model Diagnostic Tool with Unified Reference Distribution" (arXiv:1708.08527, v3)

Cindy Feng, Alireza Sadeghpour and Longhai Li

June 18, 2019

1 An Example of Checking Logistic Regression

In this Section, we present the results for the performance of NRPPs and NMPPs for checking GOF of the logistic regression model.

We repeatedly simulate 500 datasets from the true model: a logistic regression model with $\text{logit}(p_i) = \beta_1 \sin(5x_i)$, with the covariate $x_i \sim \text{Uniform}(0, \pi)$ and $\beta_1 = 2$. We consider a wrong logistic regression model with linear effect $\text{logit}(p_i) = \beta_0 + \beta_1 x_i$.

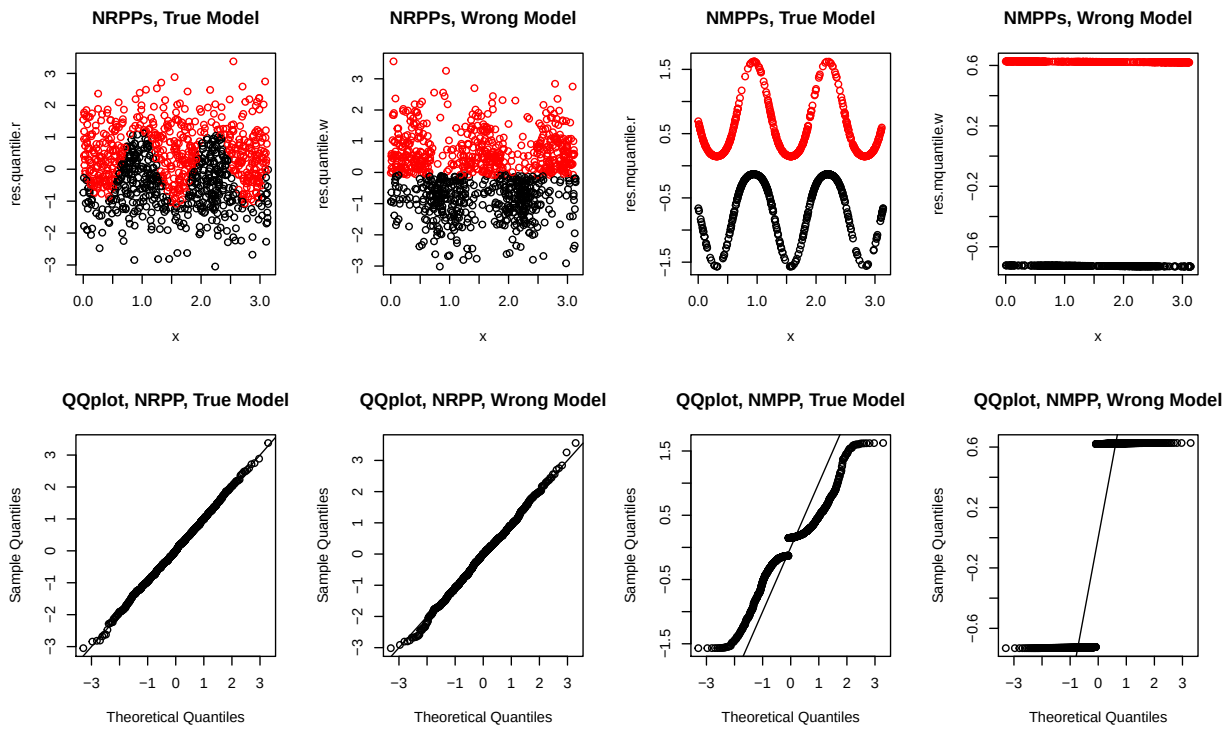

Figure S1: Residual plots of NRPPs and NMPPs versus x_i under the true vs. wrong models (top row), and the corresponding QQ plots (bottom row). The colours in the residual plot represent two values of y_i.

The residual plot of NRPPs for the wrong model clearly reveals the non-linearity. However, we see that the SW normality test applied to NRPPs has very low power in detecting the non-linearity. Another overall statistical test to measure the non-linearity pattern of NRPPs against the covariate is therefore desirable. We experiment a GOF test by testing the equal variances of NRPPs after sorted by x_i. We sort NRPPs by x_i and then divide them into 20 groups evenly from the smallest to the largest. Bartlett equal-variance test is applied to test whether the variances of the 20 groups of NRPPs are the same or not. Figure S2 shows the results of SW tests and Bartlett tests. It is clearly shown that the power of Bartlett test is very high, whereas the power of SW test is not good. Although the power of the Bartlett test is high, the results may depend on the choice of the number of groups. Hence, it remains an interesting topic to devise better GOF tests applied to NRPPs for checking model fit of logistic regression models.

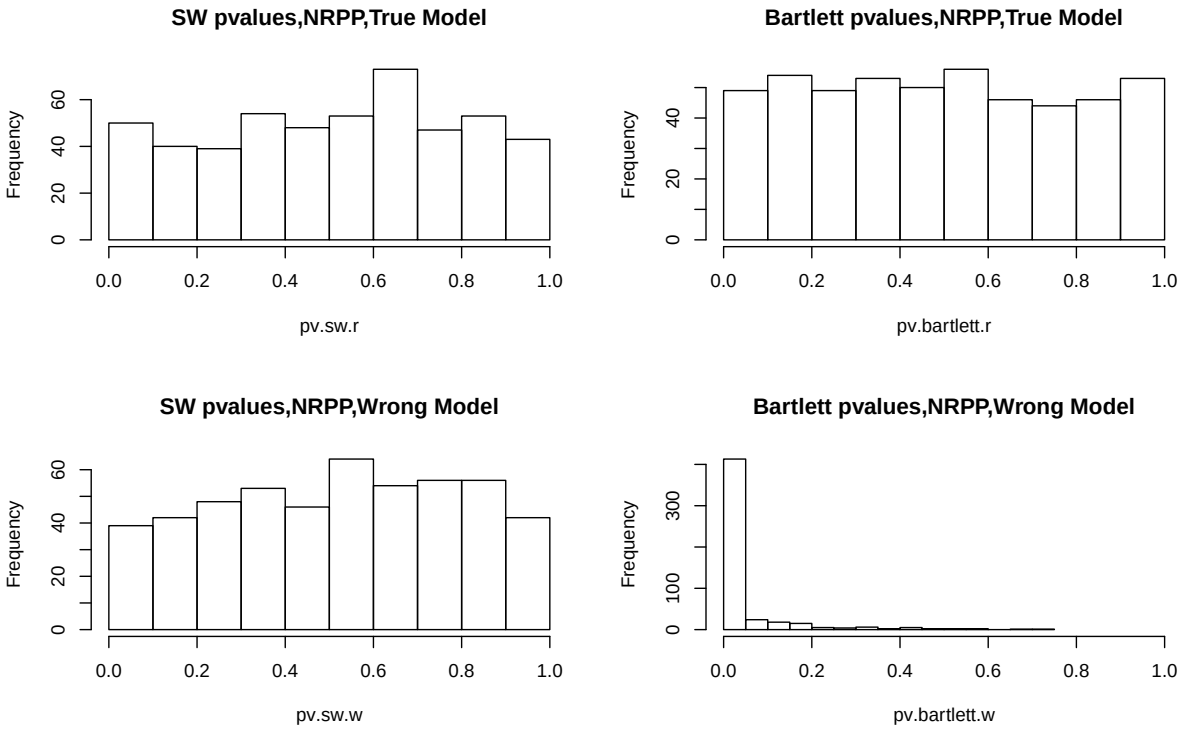

Figure S2: 500 replicated GOF test p-values for NRPPs with SW normality tests and Bartlett's equal-variance tests.

2 R Code

All the R code for the simulation studies and the real data analysis, as well as the dataset used in Section 5 of the main article can be also downloaded through the corresponding author's personal website, see here [the url will be inserted upon publication].

2.1 R Codes for Illustrative Examples

2.1.1 An Example with No Covariate

```
library(ggplot2)
library(ggExtra)
library(gridExtra)
#######################################################
# generate data
#######################################################
#sample size
n <- 2000
smp <- sample(1:n, 100)## randomly selected sample for drawing scatterplot
#Generating data
y <- rbinom(n,2,.5)
#######################################################
# RPP for the true model
#######################################################
pv.r <- pbinom(y-1,2,.5)+runif(n)*dbinom(y,2,.5)
# select a small subset of pv.r for drawing scatterplot
pv.r2 <- rep (NA, n)
pv.r2[smp] <- pv.r[smp]
#######################################################
# RPP for the wrong model
#######################################################
# functions for calculating the pmf and the cdf of the wrong distribution(.1,.8,.1)
pmf <- function(x)
{
  return((x==0)*(.1)+(x==1)*(.8)+(x==2)*(.1))
}
cdf <- function(x)
{
  return((x==0)*(.1)+(x==1)*(.9)+(x==2)*(1))
}

#Residuals for wrong model
pv.w <- cdf(y-1)+runif(n)*pmf(y)
# select a small subset of pv.w for drawing scatterplot
```

```
pv.w2 <- rep (NA, n)
pv.w2[smp] <- pv.w[smp]

#######################################################
# dataset for ggplot
#######################################################
#Saving all the results in a data frame to be used in ggplot
df <- data.frame(y=y,pv.r=pv.r,pv.r=pv.r2,pv.w=pv.w,pv.w2=pv.w2)
#######################################################
## plot settings
pch = 1
#a function for different colors of the CDF
clr <- function(x){return((x==0)*(1)+(x==1)*(2)+(x==2)*(4))}
#######################################################

#######################################################
# Drawing RPPs  for the true model
#######################################################
p.r <- ggplot(df, aes(as.factor(y), pv.r2)) +
  geom_point(pch=pch,colour=clr(y),size=3) +
  theme_classic() +
  ylab("RPP") +
  xlab("y") +
  theme(axis.text=element_text(size=rel(1.5))) +
  theme(axis.title = element_text(size = rel(1.5)))

#histogram for the true model
hist_right.r <- ggplot(data=df, aes(pv.r)) +
  geom_histogram(breaks=seq(0, 1, by=.1),
                 fill=I("blue"),col=I("red"),alpha=I(.2)) +
  coord_flip() +
  scale_y_reverse() +
  xlab("RPP") +
  theme(axis.text=element_text(size=rel(1.5))) +
  theme(axis.title = element_text(size = rel(1.5)))

#######################################################
# Drawing RPPs  for the wrong model
#######################################################
p.w <- ggplot(df, aes(as.factor(y), pv.w2)) +
    geom_point(pch=pch,colour=clr(y), size=3) +
    theme_classic() +
    ylab("RPP") +
    xlab("y") +
```

```
    theme(axis.text=element_text(size=rel(1.5))) +
    theme(axis.title = element_text(size = rel(1.5)))
#histogram for the wrong model
hist_right.w <- ggplot(data=df, aes(pv.w)) +
    geom_histogram(breaks=seq(0, 1, by=.1),
                   fill=I("blue"),col=I("red"),alpha=I(.2)) +
    coord_flip() +
    scale_y_reverse() +
    xlab("RPP") +
    theme(axis.text=element_text(size=rel(1.5))) +
    theme(axis.title = element_text(size = rel(1.5)))
grid.arrange(hist_right.r,p.r, ncol=2, nrow=1)
grid.arrange(hist_right.w,p.w, ncol=2, nrow=1)
```

2.1.2 An Example with Covariate

```
# generate a dataset
n <- 1000
x <- seq (0, 2*pi, length = n)
mu <- exp (-1 + 2*sin(2*x))
y <- rpois(length (x), mu)
uy <- sort(unique(y)) ## unique values in y

# define col representation for different values of y
col <-  rainbow(length (uy), s=1, start=0, end = 0.7)
# fit the true model
fit.r <- glm(y~sin(2*x), family = "poisson")
mu.r <- fit.r$fitted.values
# fit a wrong model
fit.w <- glm(y~x, family = "poisson")
mu.w <- fit.w$fitted.values
#draw plots
par(mfrow = c(4,2),mar=c(4.5, 4.5, 2, 1))
#=============================================
# RPP
#=============================================
## RPP for the true model:
RPP <- ppois(y-1, lambda = mu.r)+runif(n)*dpois(y, lambda = mu.r)
plot (x,rep (0.5, length (x)), ylim = c(0, 1), type ="n", ylab="RPP", xlab="x")
title("RPP, True Model")
k <- length (uy)
upper <- matrix (0,k, n)
for (i in 1:length (uy))
{
```

```
  upper[i,] <- ppois(uy[i], lambda = mu.r)
  lines(x, upper[i,], lty = 1, lwd=1,col = "black" )
}
points(x, RPP, col = col[match (y, uy)], pch = 20, cex=0.6)
###RPP for the wrong model
RPP <- ppois(y-1, lambda = mu.w)+runif(n)*dpois(y, lambda = mu.w)
plot (x, rep (0.5, length (x)), ylim = c(0, 1), type="n",  ylab="RPP", xlab="x")
title("RPP, Wrong Model")
k <- length (uy)
upper <- matrix (0,k, n)
for (i in 1:length (uy))
{
  upper[i,] <-  ppois(uy[i], lambda = mu.w)
  lines(x, upper[i,], lty = 1, col = "black" )
}
points (x, RPP, col = col[match (y, uy)], pch = 20, cex=0.6)
#============================================
# MQR
#============================================
### MQR for the True model:
MPP <- ppois(y-1, lambda = mu.r)+0.5*dpois(y, lambda = mu.r)
plot (x,rep (0.5, length (x)), ylim = c(0, 1), type = "n", ylab="MPP", xlab="x" )
title("MPP, True Model")
k <- length (uy)
upper <- matrix (0,k, n)
for (i in 1:length (uy))
{
  upper[i,] <- ppois(uy[i], lambda = mu.r)
  lines(x, upper[i,], lty = 1, col = "black" )

}
points(x, MPP, col = col[match (y, uy)], pch = 20, cex=0.6)

### MQR for the Wrong model:
MPP<- ppois(y-1, lambda = mu.w)+0.5*dpois(y, lambda = mu.w)
plot (x, rep (0.5, length (x)), ylim = c(0, 1),  type="n",  ylab="MPP", xlab="x")
title("MPP, Wrong Model")
k <- length (uy)
upper <- matrix (0,k, n)
for (i in 1:length (uy))
{
  upper[i,] <-  ppois(uy[i], lambda = mu.w)
  lines(x, upper[i,], lty = 1, col = "black" )
}
```

```
points (x, MPP, col = col[match (y, uy)], pch = 20, cex=0.6)

#============================================
# Deviance
#============================================
res.deviance.r <- resid(fit.r,"deviance")
res.deviance.w <- resid(fit.w,"deviance")
plot (x, res.deviance.r, ylim = range (res.deviance.r),
      col = col[match (y, uy)], pch = 20, ylab="Deviance",
      main="Deviance, True Model", cex=0.6)
plot (x, res.deviance.w, ylim = range (res.deviance.w),
      col = col[match (y, uy)], pch = 20, ylab="Deviance",
      main="Deviance, Wrong Model", cex=0.6)
#============================================
# Pearson
#============================================
res.pearson.r <- resid(fit.r,"pearson")
res.pearson.w <- resid(fit.w,"pearson")
plot (x, res.pearson.r, ylim = range (res.pearson.r),
      col = col[match (y, uy)], pch = 20, ylab="Pearson",
      main="Pearson, True Model", cex=0.6)
plot (x, res.pearson.w, ylim = range (res.pearson.w),
      col = col[match (y, uy)], pch = 20, ylab="Pearson",
      main="Pearson, Wrong Model", cex=0.6)
```

2.2 R codes for Simulation Study

2.2.1 Simulation Setting #1: Detection of Non-Linear Covariate Effect

```
library("MASS")
n<-500 #sample size
x <- runif(n,-1.5,1.5)
lambda <- x^2
y <- rnbinom(n,size = 2, mu = exp(lambda))
############################################################
#Model fitting
############################################################
fit.r <- glm.nb(y~lambda)  # true model (nb: x^2)
fit.w <- glm.nb(y~x)       # wrong model(nb: x)
mu.hat.r <- fitted.values(fit.r)
mu.hat.w <- fitted.values(fit.w)
############################################################
#Residuals
############################################################
```

```
### deviance residual
res.deviance.r <- resid(fit.r,"deviance")
res.deviance.w <- resid(fit.w,"deviance")
### Pearson residual
res.pearson.r <- resid(fit.r,"pearson")
res.pearson.w <- resid(fit.w,"pearson")
### RQR (NRPP)
size.hat.r <- fit.r$theta
size.hat.w <- fit.w$theta
res.quantile.r <- qnorm(pnbinom (y-1,size = size.hat.r, mu = mu.hat.r) +
                          dnbinom (y,size = size.hat.r, mu = mu.hat.r) * runif(n))
res.quantile.w <- qnorm(pnbinom (y-1,size = size.hat.w, mu = mu.hat.w) +
                          dnbinom (y,size = size.hat.w, mu = mu.hat.w) * runif(n))
#################################################################
### Residual plot and QQplot
#################################################################
par (mfcol=c(2,2))
plot(x,res.quantile.r)
plot(x,res.quantile.w)
qqnorm(res.quantile.r);abline(a=0,b=1)
qqnorm(res.quantile.w);abline(a=0,b=1)
### Shapiro Wilk test
shapiro.test(res.quantile.r)
shapiro.test(res.quantile.w)
```

2.3 Simulation Setting #2: Detection of Overdispersion

```
library("MASS")
n<-100 #sample size
beta0 <- 1
beta1 <- 2
x <- runif(n,-1,2)
y <- rnbinom(n,size = 2, mu = exp(beta0+beta1*x))
#################################################################
#Model fitting
#################################################################
fit.r <- glm.nb(y~x)
fit.w <- glm(y~x, family = "poisson")
mu.hat.r <- fitted.values(fit.r)
mu.hat.w <- fitted.values(fit.w)
size.hat.r <- fit.r$theta
#################################################################
#Residuals
#################################################################
```

```
### deviance residual
res.deviance.r <- resid(fit.r,"deviance")
res.deviance.w <- resid(fit.w,"deviance")
### Pearson residual
res.pearson.r <- resid(fit.r,"pearson")
res.pearson.w <- resid(fit.w,"pearson")
### RQR (NRPP)
res.quantile.r <- qnorm(pnbinom (y-1,size = size.hat.r, mu = mu.hat.r)+
dnbinom (y,size = size.hat.r, mu = mu.hat.r) * runif(n))
pvalue.w <- ppois (y-1,mu.hat.w) + dpois (y, mu.hat.w) * runif(n)
pvalue.w[pvalue.w==1] <- .999999999
res.quantile.w <- qnorm(pvalue.w)
############################################################
### Residual plot and QQplot
############################################################
par (mfcol=c(2,2))
plot(x,res.quantile.r)
plot(x,res.quantile.w)
qqnorm(res.quantile.r);abline(a=0,b=1)
qqnorm(res.quantile.w);abline(a=0,b=1)
### Shapiro Wilk test
shapiro.test(res.quantile.r)
shapiro.test(res.quantile.w)
```

2.4 Simulation Setting #3: Detecting Zero-Inflation

```
library("pscl")
dzpois <- function(x,lambda,p){return((1-p)*dpois(x,lambda)+p*(x==0))}
pzpois <- function(x,lambda,p){return((1-p)*ppois(x,lambda)+p*(x>=0))}
n<-100 #sample size
beta0 <- 1
beta1 <- 2
p <- .3
x <- runif(n,-1,2)
mu <- exp(beta0+beta1*x)
y <- rpois(n,mu)*rbinom(n,1,1-p)
############################################################
#Model fitting
############################################################
fit.r <- zeroinfl(y~x|1,dist="poisson")
fit.w1 <- glm(y~x,family="poisson")
mu.hat.r <- exp(coef(fit.r)[1]+coef(fit.r)[2]*x)
mu.hat.w1 <- fitted.values(fit.w1)
p.hat.r <- exp(coef(fit.r)[3])/(1+exp(coef(fit.r)[3]))
```

```
#############################################################
#Residuals
#############################################################
#deviance Residuals
res.deviance.r <- sign(y-mu.hat.r*(1-p.hat.r))*sqrt(2)*sqrt(log(dpois(y,y))
-log(dzpois(y,mu.hat.r,p.hat.r)))
res.deviance.w1 <- resid(fit.w1,"deviance")
#Pearson residuals
res.pearson.r <- resid(fit.r,"pearson")
res.pearson.w1 <- resid(fit.w1,"pearson")
### RQR (NRPP)
pvalue.r <- pzpois (y-1,mu.hat.r,p.hat.r) + dzpois (y, mu.hat.r,p.hat.r) * runif(n)
pvalue.w1 <- ppois (y-1,mu.hat.w1) + dpois (y, mu.hat.w1) * runif(n)
pvalue.r[pvalue.r==1] <- .999999999
pvalue.w1[pvalue.w1==1] <- .999999999
res.quantile.r <- qnorm(pvalue.r)
res.quantile.w1 <- qnorm(pvalue.w1)
#############################################################
### Residual plot and QQplot
#############################################################
par (mfcol=c(2,2))
plot(x,res.quantile.r)
plot(x,res.quantile.w1)
qqnorm(res.quantile.r);abline(a=0,b=1)
qqnorm(res.quantile.w1);abline(a=0,b=1)
### Shapiro Wilk test
shapiro.test(res.quantile.r)
shapiro.test(res.quantile.w1)
```

2.5 Simulation Setting #4: Detecting of Non-Linear Covariate Effect in Logistic Regression

```
library("MASS")
#-----------------------------------------------------------------------------
# Test for equal variances of RQR against m groups of x
#  n: sample size
#  y: residuals
#  x: covariate
#  m: groups (split sample into m groups)
#-----------------------------------------------------------------------------
equavarance.test<-function(n, y, x, m){
  y.ordered<-y[order(x)] #order RQR by the covaraite x
  test.eqvan.p<-bartlett.test(y.ordered~as.factor(rep(1:(n/m), each=m)))$p.value
  return(test.eqvan.p)
```

```
}
n<-1000 #sample size
x <- runif(n,0,pi)
xsq<- sin(5*x)
beta0 <- 0
beta1 <- 2
xbeta <- beta0+beta1*xsq
prob <- exp(xbeta)/(1+exp(xbeta))
y <- rbinom(n, 1,prob=prob)
############################################################
#Model fitting
############################################################
fit.r <- glm(y~xsq, family = "binomial")
fit.w <- glm(y~x, family ="binomial")
mu.hat.r <- fitted.values(fit.r)
mu.hat.w <- fitted.values(fit.w)
############################################################
#Residuals
############################################################
### deviance residual
res.deviance.r <- resid(fit.r,"deviance")
res.deviance.w <- resid(fit.w,"deviance")
### Pearson residual
res.pearson.r <- resid(fit.r,"pearson")
res.pearson.w <- resid(fit.w,"pearson")
### NMPP
mpvalue.r <- pbinom (y-1,1,prob=mu.hat.r)+dbinom (y,1,prob =mu.hat.r)*0.5
mpvalue.w <- pbinom (y-1,1,prob=mu.hat.w)+dbinom (y,1,prob=mu.hat.w)*0.5
res.mquantile.r<- qnorm (mpvalue.r)
res.mquantile.w<- qnorm (mpvalue.w)
### NRPP
pvalue.r <- pbinom (y-1,1,prob = mu.hat.r)+dbinom (y,1,prob=mu.hat.r)*runif(n)
pvalue.w <- pbinom (y-1,1,prob = mu.hat.w)+dbinom (y,1,prob=mu.hat.w)*runif(n)
res.quantile.r<- qnorm (pvalue.r)
res.quantile.w<- qnorm (pvalue.w)
############################################################
### residual analysis with RQR (NRPP)
############################################################
par (mfrow=c(2,4))
plot(x,res.quantile.r, main="NRPPs, True Model", col = y+1)
plot(x,res.quantile.w, main="NRPPs, Wrong Model", col = y+1)
plot(x,res.mquantile.r, main="NMPPs, True Model", col = y+1)
plot(x,res.mquantile.w, main="NMPPs, Wrong Model", col = y+1)
qqnorm(res.quantile.r, main="QQplot, NRPP, True Model");abline(a=0,b=1)
```

```
qqnorm(res.quantile.w, main="QQplot, NRPP, Wrong Model");abline(a=0,b=1)
qqnorm(res.mquantile.r, main="QQplot, NMPP, True Model");abline(a=0,b=1)
qqnorm(res.mquantile.w, main="QQplot, NMPP, Wrong Model");abline(a=0,b=1)
shapiro.test(res.quantile.r)
shapiro.test(res.quantile.w)
shapiro.test(res.mquantile.r)
shapiro.test(res.mquantile.w)
equavarance.test(n, res.quantile.r, x, m=20)
equavarance.test(n, res.quantile.w, x, m=20)
################################################
## Replicate RQR for R times (power analysis)
################################################
R<-500
pv.sw.r<-rep(NA, R)
pv.sw.w<-rep(NA, R)
pv.bartlett.r<-rep(NA, R)
pv.bartlett.w<-rep(NA, R)
for(r in 1:R){
  x <- runif(n,0,pi)
  xsq<- sin(5*x)
  beta0 <- 0
  beta1 <- 2
  xbeta <- beta0+beta1*xsq
  prob <- exp(xbeta)/(1+exp(xbeta))
  y <- rbinom(n, 1,prob=prob)
  ##############################################################
  #Model fitting
  ##############################################################
  fit.r <- glm(y~xsq, family = "binomial")
  fit.w <- glm(y~x, family ="binomial")
  mu.hat.r <- fitted.values(fit.r)
  mu.hat.w <- fitted.values(fit.w)
 pvalue.r <- pbinom (y-1,1,prob = mu.hat.r)+dbinom (y,1,prob=mu.hat.r)*runif(n)
pvalue.w <- pbinom (y-1,1,prob = mu.hat.w)+dbinom (y,1,prob=mu.hat.w)*runif(n)
  res.quantile.r<- qnorm (pvalue.r)
  res.quantile.w<- qnorm (pvalue.w)
  pv.sw.r[r]<-shapiro.test(res.quantile.r)$p.value
  pv.sw.w[r]<-shapiro.test(res.quantile.w)$p.value
  pv.bartlett.r[r]<-equavarance.test(n, res.quantile.r, x, m=20)
  pv.bartlett.w[r]<-equavarance.test(n, res.quantile.w, x, m=20)
}
par(mfcol = c(2,2))
hist(pv.sw.r, main = "SW pvalues,NRPP,True Model")
hist(pv.sw.w, main = "SW pvalues,NRPP,Wrong Model")
```

```
hist(pv.bartlett.r,main = "Bartlett pvalues,NRPP,True Model")
hist(pv.bartlett.w,main = "Bartlett pvalues,NRPP,Wrong Model", nclass=20,xlim=c(0,1))
mean(pv.sw.w<=0.05)
mean(pv.sw.r<=0.05)
mean(pv.bartlett.w<=0.05)
mean(pv.bartlett.r<=0.05)
```

2.6 R code for real data analysis

```
#############################################################################
#    Case study: number of hospital visits
#
#Demand for medical care by the elderly Deb and Trivedi (1997)
#analyze data on 4406 individuals, aged 66 and over, who are
#covered by Medicare, a public insurance program. Originally
#obtained from the US National Medical Expenditure Survey (NMES)
#for 1987/88, the data are available from the data archive of
#the Journal of Applied Econometrics at
#http://www.econ.queensu.ca/jae/1997-v12.3/deb-trivedi/.
#
#The objective is to model the demand for medical care-as captured
#by the number hospital stays-by the covariates available for the patients.
#
#Outcome: hosp (number of hospital stays)
#Covariates:
#age:      age in years (divided by 10)
#gender:   =1 if the person is male
#numchron: number of chronic conditions (cancer, heart attack,
#          gall bladder problems, emphysema, arthritis, diabetes,
#          other health disease)
#health:   self-perceived health status (excellent, average and poor),
#adldiff:  =1 if the person has a condition that limits activity of daily living
#############################################################################
library("pscl")
library("MASS")
#############################################################################
# CDF and PMF used in RPP
#############################################################################
#PDF of ZIP distribemervisitson;
dzpois <- function(x,xbeta,p)
{
return((1-p)*dpois(x,xbeta)+p*(x==0))
}
#CDF of ZIP distribemervisitson;
```

```
pzpois <- function(x,xbeta,p)
{
  return((1-p)*ppois(x,xbeta)+p*(x>=0))
}
#PDF of ZINB distribemervisitson;
dznbinom <- function(x,size,mu,p) #size: dispersion parameter
{
  return((1-p)*dnbinom(x,size = size, mu = mu)+p*(x==0))
}
#CDF of ZINB distribemervisitson;
pznbinom <- function(x,size,mu,p)
{
  return(
    (1-p)*pnbinom(x,size = size, mu = mu)+p*(x>=0)
  )
}
##############################################################################
# plotting function
##############################################################################
plotf <- function(a,b,c,y,z)
{
  plot(a, b, ylab = y, xlab = "x", main = z)
  lines(lowess(a, b),col="blue",lwd=3)
  plot(c,b, ylab = y, xlab = "Fitted Values", main = z)
  lines(lowess(c,b),col="blue",lwd=3)
}
qqplotf <- function(res,x)
{
  qqnorm (res,main = paste(x),cex.main=2, cex.lab=1.5)
  abline(a=0,b=1, lty=2)
}
##############################################################################
## data analysis
##############################################################################
### load and process data
load("DebTrivedi.rda")
dt <- DebTrivedi
attach(dt)
health_poor<-as.numeric(health=="poor")
health_average<-as.numeric(health=="average")
health_excellent<-as.numeric(health=="excellent")
gendermale<-as.numeric(gender=="male")
adldiffyes<-as.numeric(adldiff=="yes")
y<-emer
```

```
par (mfrow = c(1,1))
hist(y, xlab="Number of emergency room visits", main="", col="blue", breaks=30)

### fit Poission, NB, ZIP, ZINB models
fit.pois <-glm(y~ black+numchron+ health_excellent+health_average+
adldiffyes+school, data = dt, family = poisson)
summary(fit.pois)
fit.nb <- glm.nb(y~ numchron+health_excellent+health_average+
adldiffyes+school, data = dt)
summary(fit.nb)
fit.zip <- zeroinfl(y~ black+numchron+ health_excellent+health_average+
adldiffyes+school|1, data = dt, dist="poisson")
summary(fit.zip)
fit.zinb <- zeroinfl(y~ numchron+ health_excellent+health_average+
adldiffyes+school|1, data = dt, dist="negbin")
summary(fit.zinb)
### compute AICs
AIC(fit.pois)
AIC(fit.nb)
AIC(fit.zip)
AIC(fit.zinb)
### Parameter estimates and fitted values
mu.hat.pois<-fitted.values(fit.pois)
mu.hat.nb <- fitted.values(fit.nb)
mu.hat.zip<-fitted.values(fit.zip)
mu.hat.zinb<-fitted.values(fit.zinb)
p.zip <- coef(fit.zip)[8]
p.zinb <- coef(fit.zinb)[7]
p.hat.zip<-exp(p.zip)/(1+exp(p.zip))
p.hat.zinb<-exp(p.zinb)/(1+exp(p.zinb))
size.hat.nb<- fit.nb$theta
size.hat.zinb <- fit.zinb$theta
## define colorized representation of values in y
uy <- sort (unique(y))
col <-  rainbow(length (uy), s=1, start=0, end = 0.7)#[rm.col]
col.y <- col[match(y, uy)]
#######################################################
#   Pearson residuals
#######################################################
res.pearson.pois <- resid(fit.pois,"pearson")
res.pearson.nb <- resid(fit.nb,"pearson")
res.pearson.zip <- resid(fit.zip,"pearson")
res.pearson.zinb <- resid(fit.zinb,"pearson")
#Residuals vs. fitted values
```

```
par (mfrow = c(2,2))
plot(mu.hat.pois,res.pearson.pois,xlab="Fitted values",
ylab="Pearson Residuals",main="Poisson", cex.main=2, cex.lab=1.5, log="x", col=col.y)
plot(mu.hat.nb,res.pearson.nb,xlab="Fitted values",
ylab="Pearson Residuals",main="NB", cex.main=2, cex.lab=1.5, log="x", col=col.y)
plot(mu.hat.zip,res.pearson.zip,xlab="Fitted values",
ylab="Pearson Residuals",main="ZIP", cex.main=2, cex.lab=1.5, log="x", col=col.y)
plot(mu.hat.zinb,res.pearson.zinb,xlab="Fitted values",
ylab="Pearson Residuals",main="ZINB", cex.main=2, cex.lab=1.5, log="x", col=col.y)
#QQ plot
par (mfrow = c(2,2))
qqplotf(res.pearson.pois,"Poisson")
qqplotf(res.pearson.nb,"NB")
qqplotf(res.pearson.zip,"ZIP")
qqplotf(res.pearson.zinb,"ZINB")
#Shaprio-wilk normality test p.value
pv_pearson.pois<-shapiro.test(res.pearson.pois)$p.value;pv_pearson.pois
pv_pearson.nb<-shapiro.test(res.pearson.nb)$p.value; pv_pearson.nb
pv_pearson.zip<-shapiro.test(res.pearson.zip)$p.value; pv_pearson.zip
pv_pearson.zinb<-shapiro.test(res.pearson.zinb)$p.value; pv_pearson.zinb

#####################################################
#   Deviance residuals
#####################################################
res.deviance.pois <- resid(fit.pois,"deviance")
res.deviance.nb <- resid(fit.nb,"deviance")
res.deviance.zip <- sign(y-mu.hat.zip*(1-p.hat.zip))*sqrt(2)*
sqrt(log(dpois(y,y))-log(dzpois(y,mu.hat.zip,p.hat.zip)))
res.deviance.zinb <- sign(y-mu.hat.zinb*(1-p.hat.zinb))*sqrt(2)*
sqrt(log(dnbinom (y,size = size.hat.zinb, mu = y))-
log(dznbinom(y,size.hat.zinb, mu.hat.zinb,p.hat.zinb)))
#Residuals vs. fitted values
par (mfrow = c(2,2))
plot(mu.hat.pois,res.deviance.pois,xlab="Fitted values",
ylab="Deviance Residuals",main="Poisson", cex.main=2, cex.lab=1.5, log="x", col=col.y)
plot(mu.hat.nb,res.deviance.nb,xlab="Fitted values",
ylab="Deviance Residuals",main="NB", cex.main=2, cex.lab=1.5, log="x", col=col.y)
plot(mu.hat.zip,res.deviance.zip,xlab="Fitted values",
ylab="Deviance Residuals",main="ZIP", cex.main=2, cex.lab=1.5, log="x", col=col.y)
plot(mu.hat.zinb,res.deviance.zinb,xlab="Fitted values",
ylab="Deviance Residuals",main="ZINB", cex.main=2, cex.lab=1.5, log="x", col=col.y)
#QQ plot
par (mfrow = c(2,2))
qqplotf(res.deviance.pois,"Poisson")
```

```
qqplotf(res.deviance.nb,"NB")
qqplotf(res.deviance.zip,"ZIP")
qqplotf(res.deviance.zinb,"ZINB")
#Shaprio-wilk normality test p.value
pv_deviance.pois<-shapiro.test(res.deviance.pois)$p.value; pv_deviance.pois
pv_deviance.nb<-shapiro.test(res.deviance.nb)$p.value; pv_deviance.nb
pv_deviance.zip<-shapiro.test(res.deviance.zip)$p.value; pv_deviance.zip
pv_deviance.zinb<-shapiro.test(res.deviance.zinb)$p.value; pv_deviance.zinb

#######################################################
#   Randomized Quantile Residuals (NRPP)
#######################################################
set.seed(3)
#CDF
n<-length(y)
pvalue.pois <- ppois (y-1,mu.hat.pois) + dpois (y, mu.hat.pois)*runif(n)
pvalue.nb <- pnbinom (y-1,size = size.hat.nb, mu = mu.hat.nb) +
             dnbinom (y,size = size.hat.nb, mu = mu.hat.nb) * runif(n)
pvalue.zip <- pzpois (y-1,mu.hat.zip,p.hat.zip) +
              dzpois (y, mu.hat.zip, p.hat.zip) * runif(n)
pvalue.zinb <- pznbinom (y-1,size.hat.zinb, mu.hat.zinb, p.hat.zinb) +
                dznbinom (y,size.hat.zinb,mu.hat.zinb,p.hat.zinb) * runif(n)
pvalue.nb_linear <- pnbinom (y-1,size = size.hat.nb, mu = mu.hat.nb) +
dnbinom (y,size = size.hat.nb, mu = mu.hat.nb) * runif(n)
pvalue.pois[pvalue.pois==1]<-0.999999999
pvalue.nb[pvalue.nb==1]<-0.999999999
pvalue.zip[pvalue.zip==1]<-0.999999999
pvalue.zinb[pvalue.zinb==1]<-0.999999999
#randomized quantile residuals (inverse CDF)
res.new.pois <- qnorm (pvalue.pois)
res.new.nb <- qnorm (pvalue.nb)
res.new.zip <- qnorm (pvalue.zip)
res.new.zinb <- qnorm (pvalue.zinb)
res.new.nb_linear <- qnorm (pvalue.nb_linear)
#Residuals vs. fitted values
par (mfrow = c(2,2))
plot(mu.hat.pois,res.new.pois,xlab="Fitted values", ylab="NRPP",
main="Poisson", cex.main=2, cex.lab=1.5,log="x",  ylim=c(-6, 6))
abline(h=c(-3, 3), lty=2, col="blue")
plot(mu.hat.nb,res.new.nb,xlab="Fitted values",ylab="NRPP",
main="NB", cex.main=2, cex.lab=1.5,log="x",  ylim=c(-6, 6))
abline(h=c(-3, 3), lty=2, col="blue")
plot(mu.hat.zip,res.new.zip,xlab="Fitted values",ylab="NRPP",
main="ZIP", cex.main=2, cex.lab=1.5,log="x", ylim=c(-6, 6))
```

```
abline(h=c(-3, 3), lty=2, col="blue")
plot(mu.hat.zinb,res.new.zinb,xlab="Fitted values",ylab="NRPP",
main="ZINB", cex.main=2, cex.lab=1.5,log="x", ylim=c(-6, 6))
abline(h=c(-3, 3), lty=2, col="blue")

#QQ plot
par (mfrow = c(2,2))
qqplotf(res.new.pois,"Poisson")
qqplotf(res.new.nb,"NB")
qqplotf(res.new.zip,"ZIP")
qqplotf(res.new.zinb,"ZINB")
#Shaprio-wilk normality test p.value
shapiro.test(res.new.pois)
shapiro.test(res.new.nb)
shapiro.test(res.new.zip)
shapiro.test(res.new.zinb)
##############################################################################
### replicate RQR (NRPP) for R times to examine the impact of randomization on RQR
##############################################################################
set.seed(1)
R <- 1000
pv.pois<-rep(NA, R)
pv.nb<-rep(NA, R)
pv.zip<-rep(NA, R)
pv.zinb<-rep(NA, R)
for (i in 1:R){
  #RPP
  pvalue.pois <- ppois (y-1,mu.hat.pois) + dpois (y, mu.hat.pois)*runif(n)
  pvalue.nb <- pnbinom (y-1,size = size.hat.nb, mu = mu.hat.nb) +
  dnbinom (y,size = size.hat.nb, mu = mu.hat.nb) * runif(n)
  pvalue.zip<-pzpois(y-1,mu.hat.zip,p.hat.zip)+dzpois (y,mu.hat.zip,p.hat.zip)*runif(n)
  pvalue.zinb<-pznbinom(y-1,size.hat.zinb,mu.hat.zinb,p.hat.zinb) +
   dznbinom (y,size.hat.zinb,mu.hat.zinb,p.hat.zinb)*runif(n)

  pvalue.pois[pvalue.pois==1]<-0.999999999
  pvalue.nb[pvalue.nb==1]<- 0.999999999
  pvalue.zip [pvalue.zip==1]<-0.999999999
  pvalue.zinb [pvalue.zinb==1]<-0.999999999

  #randomized quantile residuals (NRPP)
  res.new.pois <- qnorm (pvalue.pois)
  res.new.nb <- qnorm (pvalue.nb)
  res.new.zip <- qnorm (pvalue.zip)
  res.new.zinb <- qnorm (pvalue.zinb)
```

```
  pv.pois[i]<-shapiro.test(res.new.pois)$p.value
  pv.nb[i]<-shapiro.test(res.new.nb)$p.value
  pv.zip[i]<-shapiro.test(res.new.zip)$p.value
  pv.zinb[i]<-shapiro.test(res.new.zinb)$p.value
}
par (mfrow = c(2,2))
hist(pv.pois, main="Poisson", xlab="SW p-value", xlim=c(0, 1))
hist(pv.nb, main="NB", xlab="SW p-value",  xlim=c(0, 1))
hist(pv.zip, main="ZIP", xlab="SW p-value",  xlim=c(0, 1))
hist(pv.zinb, main="ZINB", xlab="SW p-value", xlim=c(0, 1))
```