Runs Test Paper

Lindesay Scott-Hayward

2016-09-02

Contents

Simulated Data

Make some simulated toy data using the following equation:

$$y = \beta_0 \ + \ \beta_1 X_1$$

where $\beta_0 = 1$ and $\beta_1 = 0.3$

```
dat<-makeToyData(200, length.panels=5)
head(dat)
```

```
         x evph       mu panels
1 1.203441    0 3.900217      1
2 1.367273    0 4.096701      1
3 1.374475    0 4.105561      1
4 1.387077    0 4.121113      1
5 1.527506    0 4.298438      1
6 1.556311    0 4.335745      2
```

Uncorrelated data

1000 sets of noisy data are simulated from this truth using a Poisson distribution.

```
newdat<-generateNoise(nsim, dat$mu, family='poisson', d=1)
```

To test, fit a glm to one of the sets of data.

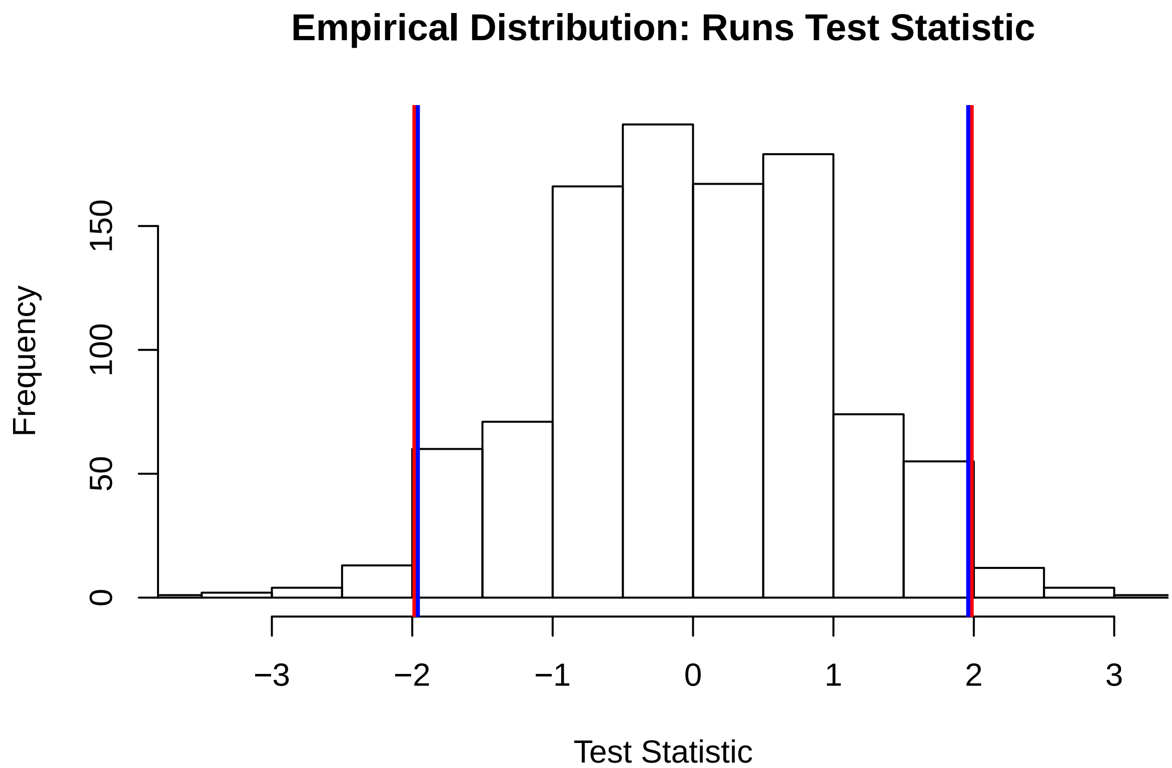
```
init_glm<-glm(newdat[,1] ~ x, data=dat, family='quasipoisson')
summary(init_glm)
```

```
Call:
glm(formula = newdat[, 1] ~ x, family = "quasipoisson", data = dat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1438  -0.7758  -0.1473   0.5711   3.2688

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.055506   0.059082   17.86   <2e-16 ***
x           0.292845   0.007655   38.26   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 1.011285)

    Null deviance: 1990.24  on 199  degrees of freedom
Residual deviance:  196.48  on 198  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4
```

Models were fitted to all datasets generated above and the runs test was evaluated for each one. The distribution of the test statistics is returned. A plot is also produced, which shows the lower 2.5% and upper 97.5% critical values of the empirical distribution (red) and from the Normal ($N(0, 1)$) distribution.

```
empdistribution<-getRunsCritVals2(n.sim = nsim, simData=newdat,
                                  model = init_glm, data = dat, plot=TRUE,
                                  returnDist = TRUE)
```

```
........................................................................................................
........................................................................................................
```



Use the data generated to assess the 5% error rate for this test when using the Normal distribition or the empirical distribution generated above.

```
newdat<-generateNoise(nsim, dat$mu, family='poisson', d=1)
ps<-matrix(NA, nrow=nsim, ncol=2)

for(i in 1:nsim){
  sim_glm<-glm(newdat[,i] ~ x , data=dat, family='quasipoisson')
   d<-summary(sim_glm)$dispersion
  resid<-residuals(sim_glm, type="response")/sqrt(fitted(sim_glm)*d)
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value
}
```

```
(length(which(ps[,1]<0.05))/nsim)*100
```

```
[1] 4.7
```

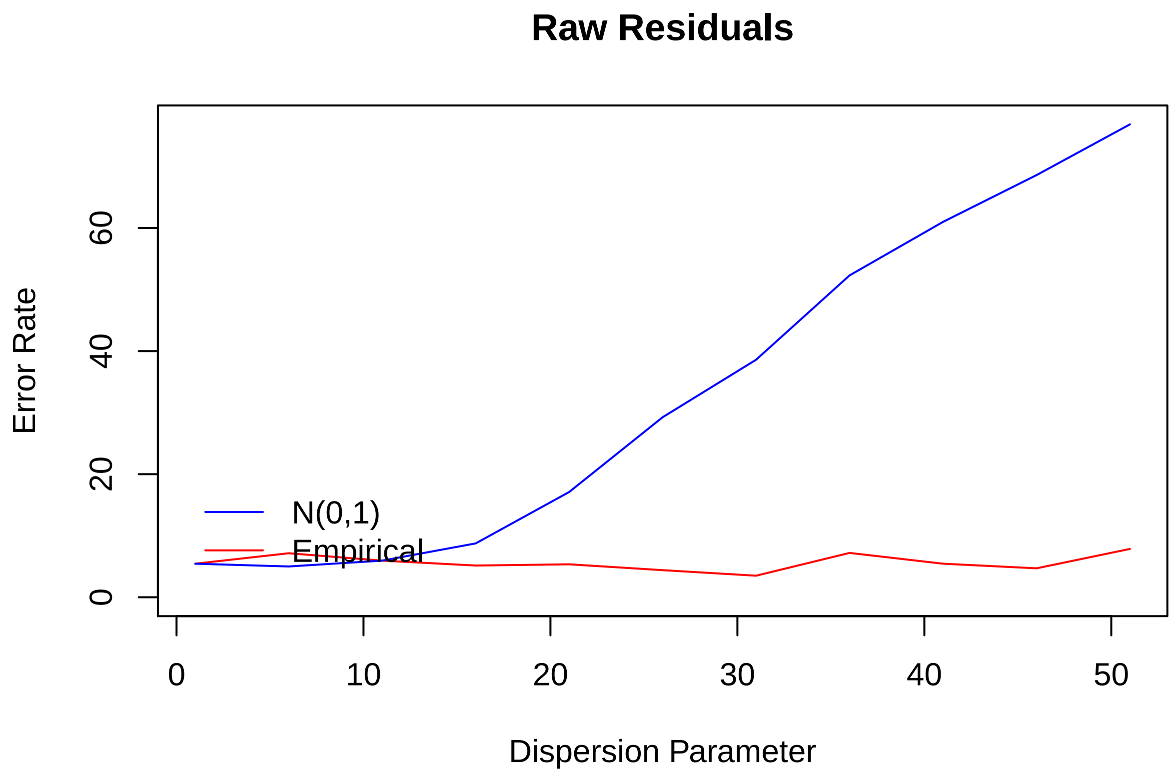
```
(length(which(ps[,2]<0.05))/nsim)*100
```

```
[1] 4.7
```

How does this change if we vary the dispersion parameter for the data?

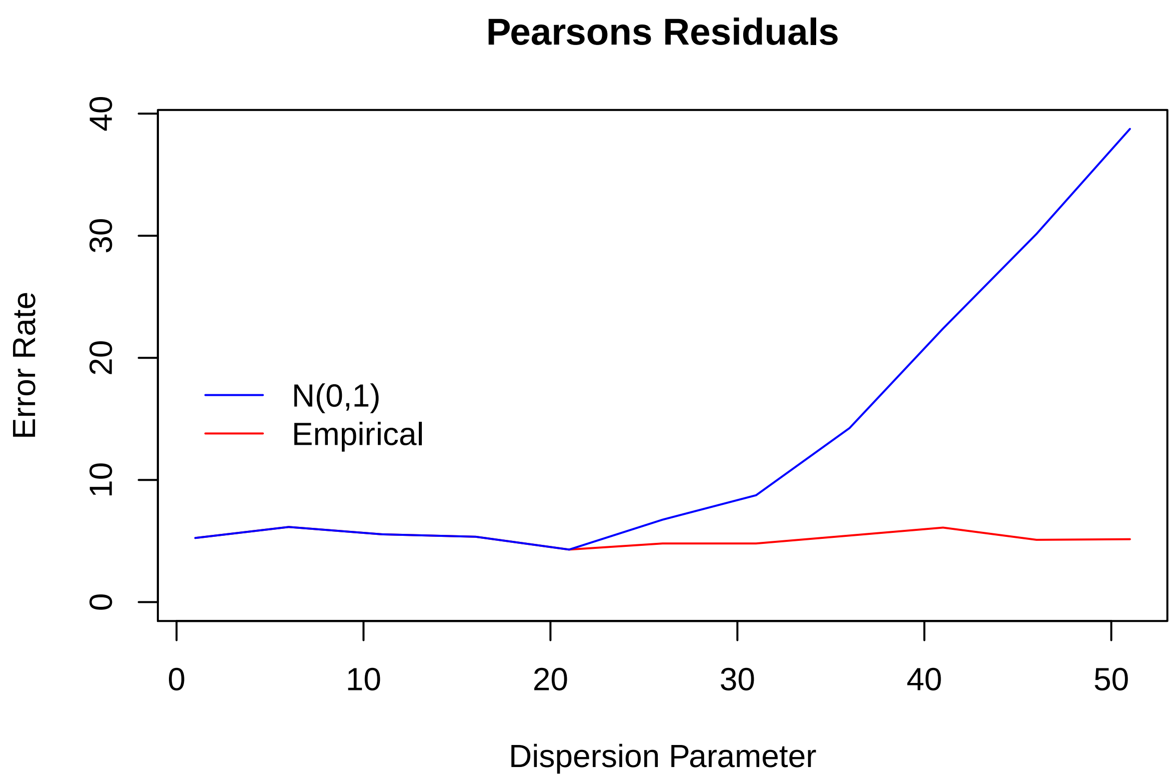
```r
nphi=seq(1,51, by=5)
errrate<-matrix(NA, nrow=length(nphi), ncol=2)
counter=1
nsim=2000
for(p in nphi){
  newdat<-generateNoise(nsim, dat$mu, family='poisson', d=p)
  # for each change in phi, update the empirical distribution
  empdistribution<-getRunsCritVals.raw(n.sim = nsim, simData=newdat,
                                  model = init_glm, data = dat, plot=FALSE,
                                  returnDist = TRUE, dots=FALSE)

  ps<-matrix(NA, nrow=nsim, ncol=2)
  for(i in 1:nsim){
    sim_glm<-glm(newdat[,i] ~ x + as.factor(evph), data=dat, family='quasipoisson')
    # find both the empirical and Normal p-values
    resid<-residuals(sim_glm, type='response')
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value
  }

  errrate[counter,1]<-(length(which(ps[,1]<0.05))/nsim)*100
  errrate[counter,2]<-(length(which(ps[,2]<0.05))/nsim)*100
  counter=counter+1
}
```

```r
plot(nphi, errrate[,1], type='l', col='red', ylim=c(0, range(errrate)[2]), ylab='Error Rate', xlab='Dis
lines(nphi, errrate[,2], col='blue')
legend(0, 20.1, legend = c('N(0,1)', 'Empirical'), col = c('blue', 'red'), lty = c(1,1), bty = 'n')
```




```
nphi=seq(1,51, by=5)
errrate<-matrix(NA, nrow=length(nphi), ncol=2)
counter=1
nsim=2000
for(p in nphi){
  newdat<-generateNoise(nsim, dat$mu, family='poisson', d=p)
  # for each change in phi, update the empirical distribution
  empdistribution<-getRunsCritVals(n.sim = nsim, simData=newdat,
                                   model = init_glm, data = dat, plot=FALSE,
                                   returnDist = TRUE, dots=FALSE)

  ps<-matrix(NA, nrow=nsim, ncol=2)
  for(i in 1:nsim){
    sim_glm<-glm(newdat[,i] ~ x + as.factor(evph), data=dat, family='quasipoisson')
    # find both the empirical and Normal p-values
    resid<-residuals(sim_glm, type='pearson')
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value
  }

  errrate[counter,1]<-(length(which(ps[,1]<0.05))/nsim)*100
  errrate[counter,2]<-(length(which(ps[,2]<0.05))/nsim)*100
  counter=counter+1
}
```

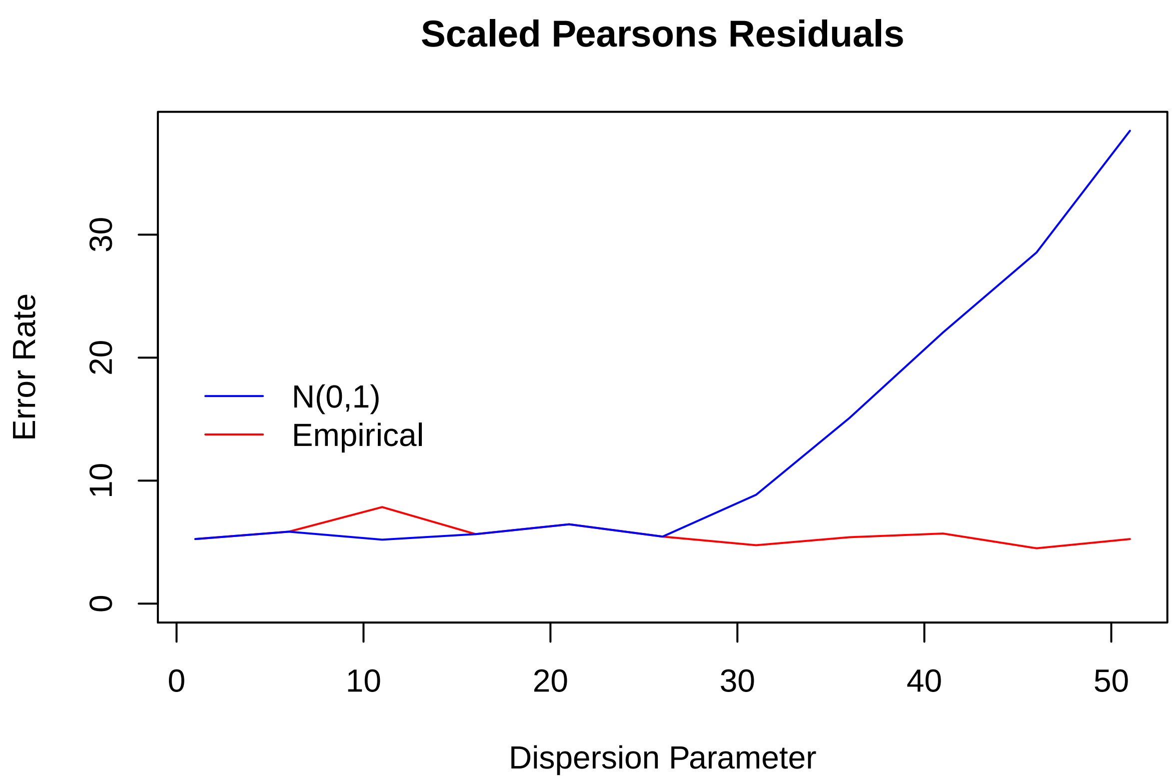
```
plot(nphi, errrate[,1], type='l', col='red', ylim=c(0, range(errrate)[2]), ylab='Error Rate', xlab='Disp
lines(nphi, errrate[,2], col='blue')
legend(0, 20.1, legend = c('N(0,1)', 'Empirical'), col = c('blue', 'red'), lty = c(1,1), bty = 'n')
```




```
nphi=seq(1,51, by=5)
errrate<-matrix(NA, nrow=length(nphi), ncol=2)
counter=1
nsim=2000
for(p in nphi){
  newdat<-generateNoise(nsim, dat$mu, family='poisson', d=p)
  # for each change in phi, update the empirical distribution
  empdistribution<-getRunsCritVals2(n.sim = nsim, simData=newdat,
                                   model = init_glm, data = dat, plot=FALSE,
                                   returnDist = TRUE, dots=FALSE)

  ps<-matrix(NA, nrow=nsim, ncol=2)
  for(i in 1:nsim){
    sim_glm<-glm(newdat[,i] ~ x + as.factor(evph), data=dat, family='quasipoisson')
    # find both the empirical and Normal p-values
    d<-summary(sim_glm)$dispersion
  resid<-residuals(sim_glm, type="response")/sqrt(fitted(sim_glm)*d)
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value
  }
```

```
  errrate[counter,1]<-(length(which(ps[,1]<0.05))/nsim)*100
  errrate[counter,2]<-(length(which(ps[,2]<0.05))/nsim)*100
  counter=counter+1
}
```

```
plot(nphi, errrate[,1], type='l', col='red', ylim=c(0, range(errrate)[2]), ylab='Error Rate', xlab='Dis
lines(nphi, errrate[,2], col='blue')
legend(0, 20, legend = c('N(0,1)', 'Empirical'), col = c('blue', 'red'), lty = c(1,1), bty = 'n')
```



Toy data with smaller mean

The Falls of Warness data has a very small mean (1.01) compared with the toy data used so far. Here we reduce the mean of the toy data inline with that seen in the Falls of Warness data to see if we can replicate results

```
datlow<-makeToyData(200, length.panels=5, b0=-2)
head(datlow)
```

```
         x evph        mu panels
1 1.131426    0 0.1900302      1
2 1.162032    0 0.1917831      1
3 1.176435    0 0.1926136      1
4 1.241248    0 0.1963954      1
```

```
5 1.534707    0 0.2144695      1
6 1.595919    0 0.2184443      2
```

```r
mean(datlow$mu)
```

```
[1] 0.8872261
```

```r
init_glm<-glm(newdat[,1] ~ x, data=datlow, family='quasipoisson')
summary(init_glm)
```

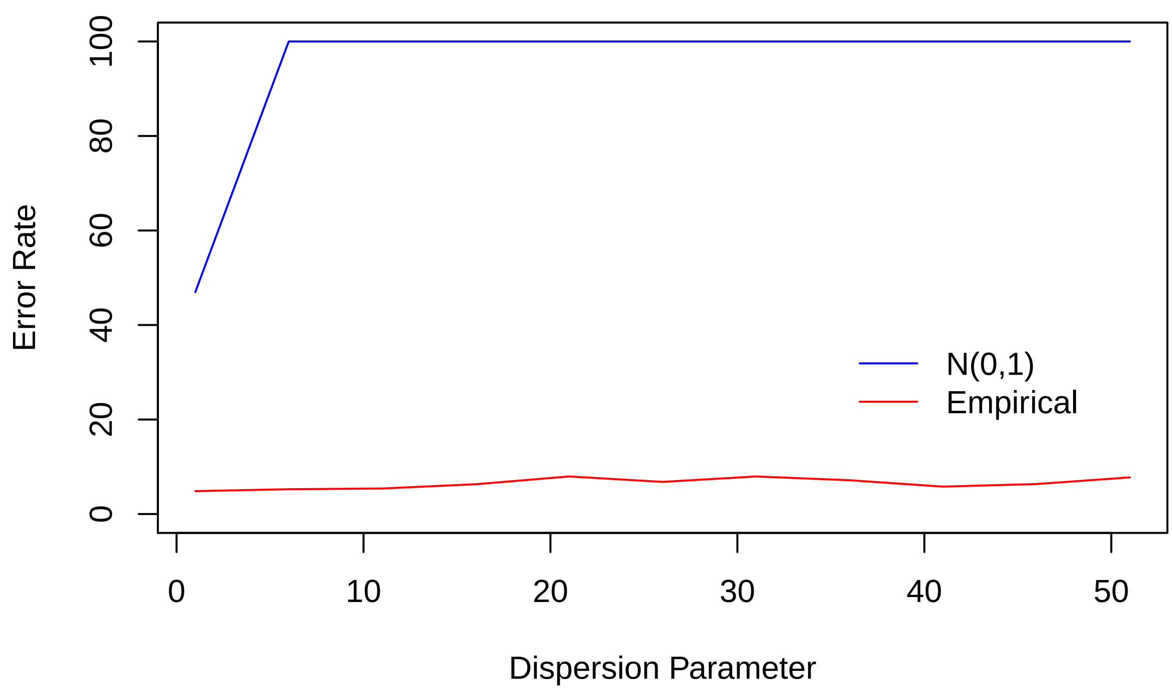
```
Call:
glm(formula = newdat[, 1] ~ x, family = "quasipoisson", data = datlow)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-9.196  -4.597  -2.896   0.767  34.930

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.32817    0.40778   3.257  0.00132 **
x            0.27923    0.05552   5.029  1.1e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 66.93312)

    Null deviance: 8189.0  on 199  degrees of freedom
Residual deviance: 6327.5  on 198  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 6
```

```r
newdat<-generateNoise(nsim, datlow$mu, family='poisson', d=20)
empdistribution<-getRunsCritVals2(n.sim = nsim, simData=newdat,
                                  model = init_glm, data = datlow, plot=TRUE,
                                  returnDist = TRUE)
```

```r
nphi=seq(1,51, by=5)
errrate<-matrix(NA, nrow=length(nphi), ncol=2)
counter=1
nsim=2000
for(p in nphi){
  #print(p)
  newdat<-generateNoise(nsim, datlow$mu, family='poisson', d=p)
  # for each change in phi, update the empirical distribution
  empdistribution<-getRunsCritVals2(n.sim = nsim, simData=newdat,
                                    model = init_glm, data = datlow, plot=FALSE,
                                    returnDist = TRUE, dots=FALSE)

  ps<-matrix(NA, nrow=nsim, ncol=2)
  for(i in 1:nsim){
```

```
    sim_glm<-glm(newdat[,i] ~ x + as.factor(evph), data=datlow,
                 family='quasipoisson')
    # find both the empirical and Normal p-values
    d<-summary(sim_glm)$dispersion
  resid<-residuals(sim_glm, type="response")/sqrt(fitted(sim_glm)*d)
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value

  }

  errrate[counter,1]<-(length(which(ps[,1]<0.05))/nsim)*100
  errrate[counter,2]<-(length(which(ps[,2]<0.05))/nsim)*100
  counter=counter+1
}
```

```
plot(nphi, errrate[,1], type='l', col='red', ylim=c(0, range(errrate)[2]), ylab='Error Rate', xlab='Dis
lines(nphi, errrate[,2], col='blue')
legend(35, 40, legend = c('N(0,1)', 'Empirical'), col = c('blue', 'red'), lty = c(1,1), bty = 'n')
```



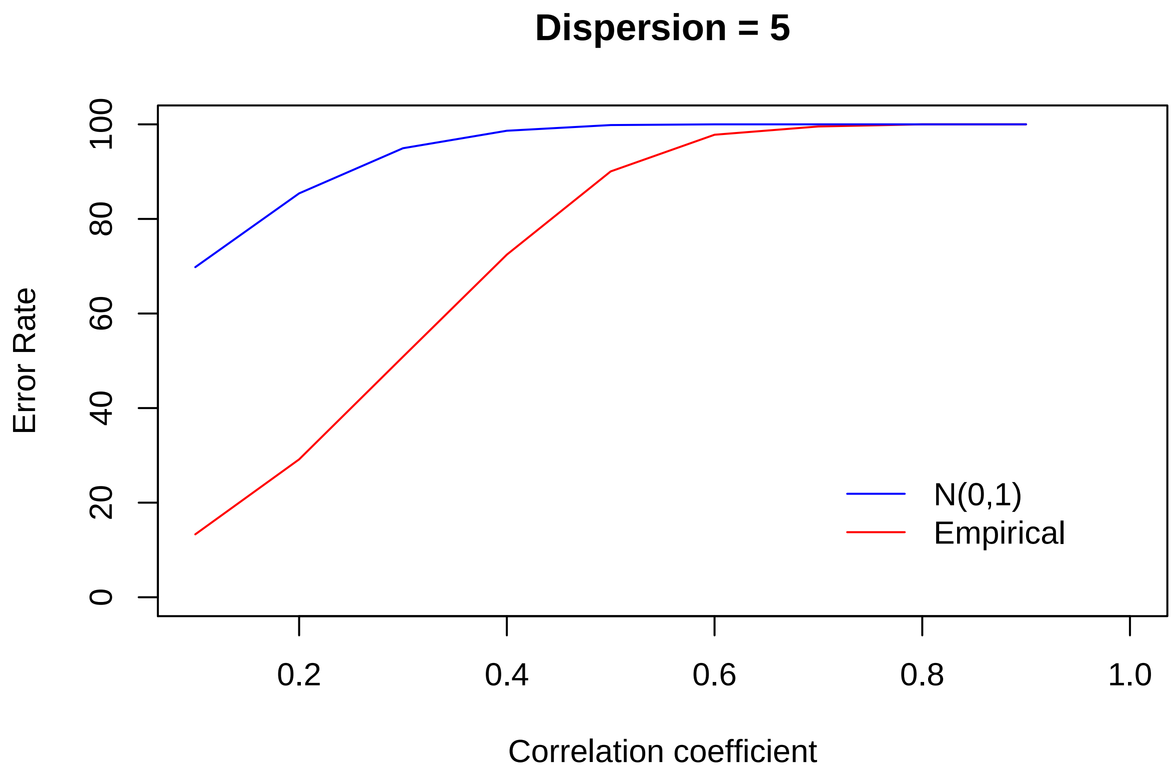
Correlated data low mean

```r
datlow<-makeToyData(200, length.panels=5, b0=-2)
head(dat)
```

```
         x evph       mu panels
1 1.203441    0 3.900217      1
2 1.367273    0 4.096701      1
3 1.374475    0 4.105561      1
4 1.387077    0 4.121113      1
5 1.527506    0 4.298438      1
6 1.556311    0 4.335745      2
```

```r
nsim=2000
newdat<-generateNoise(nsim, datlow$mu, family='poisson', d=1)
rho=seq(0.1, 0.9, by=0.1)
errrate<-matrix(NA, nrow=length(rho), ncol=2)
counter=1

for(r in rho){
  newdatcorr<-generateIC.toy(datlow, c(1, r, r^2, r^3, r^4), 'panels', newdat, ncol(newdat), dots=FALSE

  # for each change in phi, update the empirical distribution
  empdistribution<-getRunsCritVals2(n.sim = nsim, simData=newdat,
                                   model = init_glm, data = datlow, plot=FALSE,
                                   returnDist = TRUE, dots=FALSE)

  ps<-matrix(NA, nrow=nsim, ncol=2)
  for(i in 1:nsim){
    sim_glm<-glm(newdatcorr[,i] ~ x , data=datlow, family='quasipoisson')
    # find both the empirical and Normal p-values
    d<-summary(sim_glm)$dispersion
  resid<-residuals(sim_glm, type="response")/sqrt(fitted(sim_glm)*d)
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value
  }

  errrate[counter,1]<-(length(which(ps[,1]<0.05))/nsim)*100
  errrate[counter,2]<-(length(which(ps[,2]<0.05))/nsim)*100
  counter=counter+1
}
```

```r
plot(rho, errrate[,1], type='l', col='red', ylim=c(0, range(errrate)[2]), xlim=c(0.1,1), ylab='Error Ra
lines(rho, errrate[,2], col='blue')
legend(0.7, 30, legend = c('N(0,1)', 'Empirical'), col = c('blue', 'red'), lty = c(1,1), bty = 'n')
```



As expected, when the dispersion in the data is high, there is a greater discrepancy between the N(0,1) and empirical distributions.

Real Data: Falls of Warness

```
init_glm<-glm(response ~  TideState + WindStrength + SeaState + SimpPrecipitation + CloudCover, data=da
```

```
nsim=500
newdat<-generateNoise(nsim, fitted(init_glm), family='poisson', d=summary(init_glm)$dispersion)
```

500 sets of noisy data are simulated from this truth using a Poisson distribution.

To test, fit a glm to one of the sets of data.

```
fowsim_glm<-glm(newdat[,1] ~  TideState + WindStrength + SeaState + SimpPrecipitation + CloudCover, data
summary(fowsim_glm)
```

```
Call:
glm(formula = newdat[, 1] ~ TideState + WindStrength + SeaState +
    SimpPrecipitation + CloudCover, family = quasipoisson, data = dat)

Deviance Residuals:
```

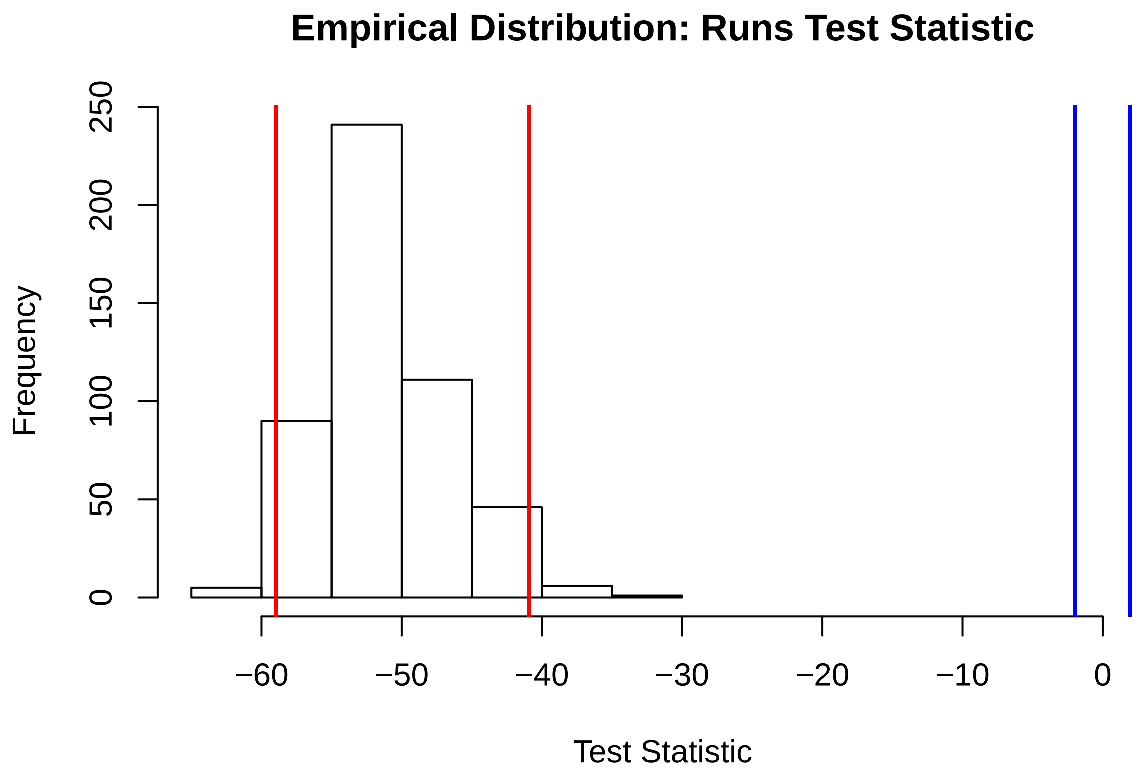
```
   Min      1Q  Median      3Q     Max
-2.654  -1.588  -1.404  -1.238  50.308

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              -0.56287    0.41798  -1.347 0.178106
TideState                -0.12765    0.06634  -1.924 0.054327 .
WindStrength             -0.29713    0.09524  -3.120 0.001811 **
SeaState                  0.36984    0.09629   3.841 0.000123 ***
SimpPrecipitationHEAVY   -1.07898    1.00552  -1.073 0.283253
SimpPrecipitationLIGHT    0.37331    0.41957   0.890 0.373612
SimpPrecipitationNONE     0.24571    0.36378   0.675 0.499408
SimpPrecipitationSHOWERS  0.77617    0.36782   2.110 0.034853 *
CloudCover                0.08231    0.02213   3.719 0.000200 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 60.76616)

    Null deviance: 206450  on 26334  degrees of freedom
Residual deviance: 201901  on 26326  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 8
```

Runs Test Check

Models were fitted to all datasets generated above and the runs test was evaluated for each one. The distribution of the test statistics is returned. A plot is also produced, which shows the lower 2.5% and upper 97.5% critical values of the empirical distribution (red) and from the Normal ($N(0,1)$) distribution.

```
empdistribution<-getRunsCritVals2(n.sim = nsim, simData=newdat,
                                  model = fowsim_glm, data = dat, plot=TRUE,
                                  returnDist = TRUE)
```

```
..........................................................................................................
```



Use the data generated to assess the 5% error rate for this test when using the Normal distribition or the empirical distribution generated above.

```
newdat<-generateNoise(nsim, fitted(init_glm), family='poisson', d=1)
ps<-matrix(NA, nrow=nsim, ncol=2)

for(i in 1:nsim){
  sim_glm<-update(fowsim_glm, newdat[,i]~.)
  # find both the empirical and Normal p-values
   d<-1
  resid<-residuals(sim_glm, type="response")/sqrt(fitted(sim_glm)*d)
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value
}
```

```
(length(which(ps[,1]<0.05))/nsim)*100
```

```
[1] 0
```

```
(length(which(ps[,2]<0.05))/nsim)*100
```

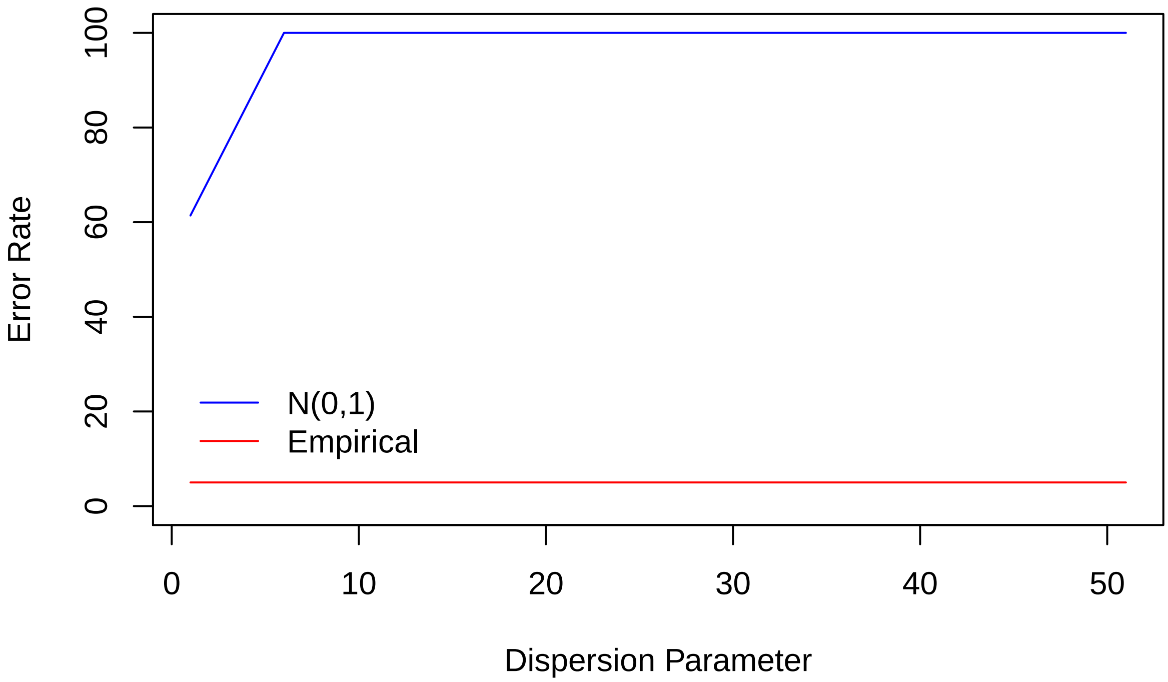
```
[1] 60.6
```

How does this change if we vary the dispersion parameter for the data?

```r
nphi=seq(1,51, by=5)
errrate<-matrix(NA, nrow=length(nphi), ncol=2)
counter=1
nsim=500
for(p in nphi){
  print(p)
  newdat<-generateNoise(nsim, fitted(init_glm), family='poisson', d=p)
  # for each change in phi, update the empirical distribution
  empdistribution<-getRunsCritVals2(n.sim = nsim, simData=newdat,
                                    model = init_glm, data = dat, plot=FALSE,
                                    returnDist = TRUE, dots=FALSE)

  ps<-matrix(NA, nrow=nsim, ncol=2)
  for(i in 1:nsim){
    sim_glm<-update(fowsim_glm, newdat[,i]~.)
    # find both the empirical and Normal p-values
     d<-summary(sim_glm)$dispersion
  resid<-residuals(sim_glm, type="response")/sqrt(fitted(sim_glm)*d)
  # find both the empirical and Normal p-values
  ps[i,1]<-runs.test(resid, critvals = empdistribution)$p.value
  ps[i,2]<-runs.test(resid)$p.value
  }

  errrate[counter,1]<-(length(which(ps[,1]<0.05))/nsim)*100
  errrate[counter,2]<-(length(which(ps[,2]<0.05))/nsim)*100
  counter=counter+1
}
```

```
[1] 1
[1] 6
[1] 11
[1] 16
[1] 21
[1] 26
[1] 31
[1] 36
[1] 41
[1] 46
[1] 51
```

```r
plot(nphi, errrate[,1], type='l', col='red', ylim=c(0, range(errrate)[2]), ylab='Error Rate', xlab='Disp
lines(nphi, errrate[,2], col='blue')
legend(0, 30, legend = c('N(0,1)', 'Empirical'), col = c('blue', 'red'), lty = c(1,1), bty = 'n')
```



Error Rate
0
20
40
60
80
100
N(0,1)
Empirical
0
10
20
30
40
50
Dispersion Parameter