

Biostat/Stat 571 Exercise #4

Answer Key

February, 2010

Question 1

Part a

Table 1 presents the asymptotic variance of $\hat{\beta}^{(0)}$ (the Poisson regression estimator) and $\hat{\beta}^{(2)}$ (the negative-binomial weighted estimator). We can see that for any value of α , the negative-binomial weighted estimator remained more efficient than the Poisson estimator for each coefficient although the ARE continued to decrease as the value of α increased. The variances from each of the estimators were closest with respect to the intercept parameter, while the largest discrepancies occurred with respect to β_1 .

Table 1: Asymptotic relative efficiencies comparing the effect of using a Poisson weighting scheme when the true variance function takes a negative-binomial form.

α	Coefficient	Poisson	N-B Weighted	ARE
0.25	β_0	0.1193	0.1171	0.9641
	β_1	0.3622	0.3340	0.8804
	β_2	0.2037	0.1966	0.9320
0.50	β_0	0.1464	0.1418	0.9369
	β_1	0.4737	0.4269	0.8122
	β_2	0.2595	0.2446	0.8885
1.00	β_0	0.1895	0.1806	0.9089
	β_1	0.6408	0.5551	0.7504
	β_2	0.3450	0.3176	0.8472

Part b

Table 2 yields asymptotic relative efficiencies comparing the effect of covariate magnitudes (with respect to β_1). We can see that the magnitude of β_1 is inversely related to the asymptotic relative efficiency of the two estimators. When $\beta_1 = 0$, the Poisson estimator is 87.9% as efficient as the negative-binomial weighted estimator, compared to 79.1% when $\beta_1 = 1$.

Table 2: Asymptotic relative efficiencies comparing the effect of covariate magnitudes.

β_1	Poisson	N-B Weighted	ARE
0.0	0.5820	0.5457	0.8791
0.5	0.5997	0.5480	0.8350
1.0	0.6196	0.5512	0.7914

Part c

Table 3 presents the asymptotic variance of the negative-binomial weighted estimator and the Poisson regression estimator (scaled) when the true variance form is assumed to be a scale model. Again, we can see that if we use the correct weighting function (ie. the Poisson estimator), this estimator yields a smaller asymptotic variance than that attained by using an inappropriate weighting scheme (ie. the negative-binomial estimator). Similar to part (a), the ARE comparing the two estimators continued to decrease as the value of the scale parameter increased.

Part d

Simulation study results to verify the calculations given in (a) and (c) are given in Table 4. Data was generated as follows:

- Part (a):

$$\begin{aligned}\nu_i &\sim \text{Gamma} \left(\frac{1}{\alpha}, 1 \right) \\ Y_i | \nu_i &\sim \text{Poisson}(\mu_i \nu_i).\end{aligned}$$

Table 3: Asymptotic relative efficiencies comparing the effect of using a negative-binomial weighting scheme when the true variance function is given by a scale adjustment.

ϕ	Coefficient	Poisson	N-B Weighted	ARE
2.0	β_0	0.1195	0.1183	0.9802
	β_1	0.2917	0.2759	0.8950
	β_2	0.1812	0.1768	0.9520
3.0	β_0	0.1480	0.1449	0.9586
	β_1	0.3786	0.3380	0.7967
	β_2	0.2280	0.2166	0.9022
4.0	β_0	0.1723	0.1673	0.9421
	β_1	0.4559	0.3902	0.7328
	β_2	0.2687	0.2501	0.8662

Thus,

$$\begin{aligned}
 E[Y_i] &= E_{\nu_i}[E_{Y_i|\nu_i}(Y_i)] \\
 &= E_{\nu_i}[\mu_i \nu_i] \\
 &= \mu_i E_{\nu_i}[\nu_i] \\
 &= \mu_i \\
 \text{Var}[Y_i] &= E_{\nu_i}[\text{Var}_{Y_i|\nu_i}(Y_i)] + \text{Var}_{\nu_i}[E_{Y_i|\nu_i}(Y_i)] \\
 &= \mu_i E_{\nu_i}[\nu_i] + \mu_i^2 \text{Var}_{\nu_i}[\nu_i] \\
 &= \mu_i + \alpha \mu_i^2.
 \end{aligned}$$

- Part (c):

$$\begin{aligned}
 \nu_i &\sim \text{Gamma}\left(\frac{\mu_i}{\phi - 1}, 1\right) \\
 Y_i|\nu_i &\sim \text{Poisson}(\mu_i \nu_i).
 \end{aligned}$$

Thus,

$$\begin{aligned}
E[Y_i] &= E_{\nu_i}[E_{Y_i|\nu_i}(Y_i)] \\
&= E_{\nu_i}[\mu_i \nu_i] \\
&= \mu_i E_{\nu_i}[\nu_i] \\
&= \mu_i \\
\text{Var}[Y_i] &= E_{\nu_i}[\text{Var}_{Y_i|\nu_i}(Y_i)] + \text{Var}_{\nu_i}[E_{Y_i|\nu_i}(Y_i)] \\
&= \mu_i E_{\nu_i}[\nu_i] + \mu_i^2 \text{Var}_{\nu_i}[\nu_i] \\
&= \mu_i + \mu_i^2 \left(\frac{\phi - 1}{\mu_i} \right) \\
&= \phi \mu_i
\end{aligned}$$

For both cases, we can see that the simulation results agreed quite nicely with the theoretical results presented earlier.

Table 4: Simulation study to verify the theoretical results presented in parts (a) and (c).

Variance Function	Coefficient	Poisson	Weighted	ARE
N-B: $\alpha = 1.0$	β_0	0.1927	0.1871	0.9424
	β_1	0.6423	0.5716	0.7921
	β_2	0.3485	0.3284	0.8882
Scale: $\phi = 4.0$	β_0	0.1693	0.1777	0.9079
	β_1	0.3923	0.4989	0.6184
	β_2	0.2513	0.2807	0.8014

Part e

Mathematically, we would not expect to lose too much efficiency by having to estimate α since the $\hat{\alpha}$ obtained using maximum likelihood estimation is orthogonal to the $\hat{\beta}$ obtained using maximum likelihood estimation. Hence, we have that $\text{Var}(\hat{\beta}(\hat{\alpha})) = \text{Var}(\hat{\beta}(\alpha))$. Note that we can also argue this fact using Slutsky's theorem.

Simulation results investigating the effect of estimating α in the negative-binomial case are given in Table 5. In comparison to the results obtained in (a), we can see that by estimating α , only a modest loss of efficiency (relative to the Poisson estimator) was witnessed.

Table 5: Simulation results to investigate the effect of estimating α .

Variance Function	Coefficient	Poisson	Weighted	ARE
	Var($\hat{\beta}(\alpha)$)			
N-B: $\alpha = 1.0$	β_0	0.1807	0.1732	0.9187
	β_1	0.6130	0.5487	0.8011
	β_2	0.3241	0.3044	0.8822
Simulation Variance = $\frac{1}{1000} \sum_{i=1}^{1000} (\boldsymbol{\beta}^{(i)}(\hat{\alpha}^{(i)}) - \bar{\boldsymbol{\beta}})^2$				
N-B: $\alpha = 1.0$	β_0	0.1806	0.1731	0.9187
	β_1	0.6127	0.5484	0.8011
	β_2	0.3239	0.3042	0.8822

Question 2

Let the number of observations per subject be $n_i = 10$, evaluated at times t_{ij} for $j = 1, 2, \dots, 10$. Consider the mean model:

$$E[Y_{ij}|\mathbf{X}_i] = \beta_0 + \beta_1 t_{ij}. \quad (1)$$

For each of the scenarios below we consider generating data for $m = 25$ subjects, and the parameter values $\boldsymbol{\beta} = (70, 10)$.

Part a

Assume each subject has their own intercept. The complete model is given by (1) and :

$$\begin{aligned} Y_{ij} &= \beta_0 + \beta_1 t_{ij} + b_{0,i} + \epsilon_{ij} \\ b_{0,i} &\sim N(0, \tau^2) \\ \epsilon_{ij} &\sim N(0, \sigma^2), \end{aligned}$$

where $b_{0,i}$ and ϵ_{ij} are mutually independent. We can give the general form of the covariance matrix $\Sigma = \text{cov}[\mathbf{Y}_i]$:

$$\begin{aligned} \text{cov}[Y_{ij}, Y_{ij}] &= \text{Var}[Y_{ij}] \\ &= \text{Var}[\beta_0 + \beta_1 t_{ij} + b_{0,i} + \epsilon_{ij}] \\ &= \tau^2 + \sigma^2 \\ \text{cov}[Y_{ij}, Y_{ik}] &= E[(Y_{ij} - E[Y_{ij}|\mathbf{X}_i])(Y_{ik} - E[Y_{ik}|\mathbf{X}_i])] \\ &= E[(b_{0,i} + \epsilon_{ij})(b_{0,k} + \epsilon_{ik})] \\ &= E[b_{0,i}^2] + E[b_{0,i}\epsilon_{ik}] + E[b_{0,i}\epsilon_{ij}] + E[\epsilon_{ij}\epsilon_{ik}] \\ &= \tau^2 \end{aligned}$$

So we finally find that

$$\Sigma = \begin{bmatrix} \tau^2 + \sigma^2 & \tau^2 & \dots & \tau^2 \\ \tau^2 & \tau^2 + \sigma^2 & \dots & \tau^2 \\ \vdots & \vdots & \ddots & \vdots \\ \tau^2 & \tau^2 & \dots & \tau^2 + \sigma^2 \end{bmatrix}.$$

Part b

Now assume that each subject has their own intercept and their own slope. The complete model is given by (1) and :

$$\begin{aligned} Y_{ij} &= \beta_0 + \beta_1 t_{ij} + b_{0,i} + b_{1,i} t_{ij} + \epsilon_{ij} \\ &= \beta_0 + \beta_1 t_{ij} + T_{ij}\mathbf{b} + \epsilon_{ij}, \quad T_{ij} = (1 \ t_{ij}) = (1 \ j), \\ \mathbf{b} &\sim N(0, \mathbf{D}) \\ \epsilon_{ij} &\sim N(0, \sigma^2) \end{aligned}$$

where $\mathbf{b} = (b_{0,i}, b_{1,i})$ and ϵ_{ij} are mutually independent. The general form of the covariance matrix is given by:

$$\begin{aligned} \text{cov}[Y_{ij}, Y_{ij}] &= \text{Var}[Y_{ij}] \\ &= \text{Var}[\beta_0 + \beta_1 t_{ij} + T_{ij}\mathbf{b} + \epsilon_{ij}] \\ &= \text{Var}[T_{ij}\mathbf{b} + \epsilon_{ij}] \\ &= T_{ij}\mathbf{D}T_{ij}^T + \sigma^2 \\ \text{cov}[Y_{ij}, Y_{ik}] &= E[(Y_{ij} - E[Y_{ij}|\mathbf{X}_i])(Y_{ik} - E[Y_{ik}|\mathbf{X}_i])] \\ &= E[(T_{ij}\mathbf{b} + \epsilon_{ij})(T_{ik}\mathbf{b} + \epsilon_{ik})] \\ &= E[(T_{ij}\mathbf{b})(T_{ik}\mathbf{b})] + E[(T_{ij}\mathbf{b})\epsilon_{ik}] + E[\epsilon_{ij}(T_{ik}\mathbf{b})] + E[\epsilon_{ij}\epsilon_{ik}] \\ &= E[(T_{ij}\mathbf{b})(T_{ik}\mathbf{b})^T] \quad \text{since } T_{ik}\mathbf{b} \text{ is scalar} \\ &= E[T_{ij}(\mathbf{b}\mathbf{b}^T)T_{ik}^T] \\ &= T_{ij}\mathbf{D}T_{ik}^T \end{aligned}$$

So we finally find that

$$\Sigma = \begin{bmatrix} T_{i1}\mathbf{D}T_{i1}^T + \sigma^2 & T_{i1}\mathbf{D}T_{i2}^T & \dots & T_{i1}\mathbf{D}T_{i10}^T \\ T_{i2}\mathbf{D}T_{i1}^T & T_{i2}\mathbf{D}T_{i2}^T + \sigma^2 & \dots & T_{i2}\mathbf{D}T_{i10}^T \\ \vdots & \vdots & \ddots & \vdots \\ T_{i10}\mathbf{D}T_{i1}^T & T_{i10}\mathbf{D}T_{i2}^T & \dots & T_{i10}\mathbf{D}T_{i10}^T + \sigma^2 \end{bmatrix}.$$

Part c

Now, we can assume that each subject has their own “process” that is serially correlated. The complete model is given by (1) and :

$$\begin{aligned} Y_{ij} &= \beta_0 + \beta_1 t_{ij} + \mathbf{W}_i(t_{ij}) + \epsilon_{ij} \\ \mathbf{W}_i &\sim N(0, \mathbf{G}) \\ \epsilon_{ij} &\sim N(0, \sigma^2), \end{aligned}$$

where \mathbf{W}_i and ϵ_{ij} are mutually independent. The diagonal and off-diagonal elements of \mathbf{G} are given by:

$$\begin{aligned} \text{Var}[\mathbf{W}_i(t_{ij})] &= \tau^2 \\ \text{cov}[\mathbf{W}_i(t_{ij}), \mathbf{W}_i(t_{ik})] &= \tau^2 \rho^{|t_{ij}-t_{ik}|} \end{aligned}$$

The general form of the covariance matrix is given by:

$$\begin{aligned} \text{cov}[Y_{ij}, Y_{ik}] &= \text{Var}[Y_{ij}] \\ &= \text{Var}[\beta_0 + \beta_1 t_{ij} + \mathbf{W}_i(t_{ij}) + \epsilon_{ij}] \\ &= \text{Var}[\mathbf{W}_i(t_{ij}) + \epsilon_{ij}] \\ &= \tau^2 + \sigma^2 \\ \text{cov}[Y_{ij}, Y_{ik}] &= E[(Y_{ij} - E[Y_{ij}|\mathbf{X}_i])(Y_{ik} - E[Y_{ik}|\mathbf{X}_i])] \\ &= E[(\mathbf{W}_i(t_{ij}) + \epsilon_{ij})(\mathbf{W}_i(t_{ik}) + \epsilon_{ik})] \\ &= E[\mathbf{W}_i(t_{ij})\mathbf{W}_i(t_{ik})] + E[\mathbf{W}_i(t_{ij})\epsilon_{ik}] + E[\epsilon_{ij}\mathbf{W}_i(t_{ik})] + E[\epsilon_{ij}\epsilon_{ik}] \\ &= \tau^2 \rho^{|t_{ij}-t_{ik}|} \\ &= \tau^2 \rho^{|j-k|}. \end{aligned}$$

So we finally find that

$$\Sigma = \begin{bmatrix} \tau^2 + \sigma^2 & \tau^2 \rho & \dots & \tau^2 \rho^9 \\ \tau^2 \rho & \tau^2 + \sigma^2 & \dots & \tau^2 \rho^8 \\ \vdots & \vdots & \ddots & \vdots \\ \tau^2 \rho^9 & \tau^2 \rho^8 & \dots & \tau^2 + \sigma^2 \end{bmatrix}.$$

Part d

Finally, we will assume that the random effects are non-normal. The complete model is given by (1) and :

$$\begin{aligned} Y_{ij} &= \beta_0 + \beta_1 t_{ij} + b_{0,i} + \epsilon_{ij} \\ b_{0,i} &\sim F_b \\ \epsilon_{ij} &\sim N(0, \sigma^2), \end{aligned}$$

where $b_{0,i}$ and ϵ_{ij} are mutually independent.

If the random effects distribution has mean 0 and variance τ^2 , the general form of the covariance matrix $\Sigma = \text{cov}[\mathbf{Y}_i]$ is given by:

$$\begin{aligned}\text{cov}[Y_{ij}, Y_{ij}] &= \text{Var}[Y_{ij}] \\ &= \text{Var}[\beta_0 + \beta_1 t_{ij} + b_{0,i} + \epsilon_{ij}] \\ &= \tau^2 + \sigma^2 \\ \text{cov}[Y_{ij}, Y_{ik}] &= E[(Y_{ij} - E[Y_{ij}|\mathbf{X}_i])(Y_{ik} - E[Y_{ik}|\mathbf{X}_i])] \\ &= E[(b_{0,i} + \epsilon_{ij})(b_{0,i} + \epsilon_{ik})] \\ &= E[b_{0,i}^2] + E[b_{0,i}\epsilon_{ik}] + E[b_{0,i}\epsilon_{ij}] + E[\epsilon_{ij}\epsilon_{ik}] \\ &= \tau^2\end{aligned}$$

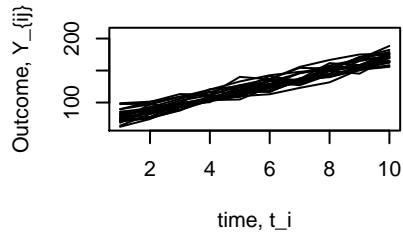
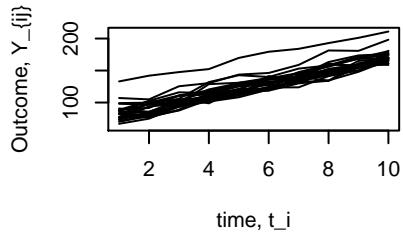
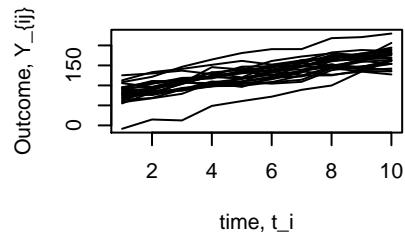
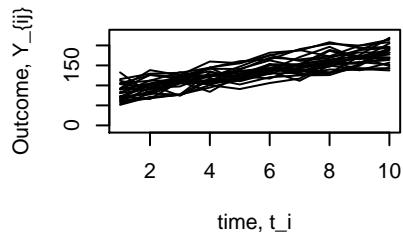
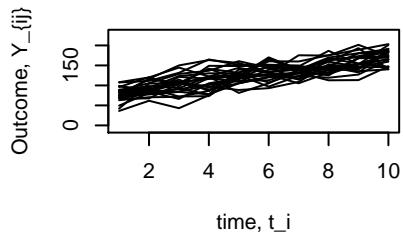
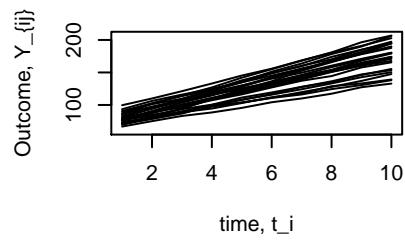
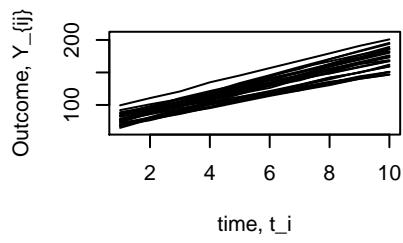
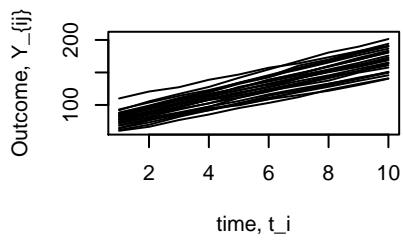
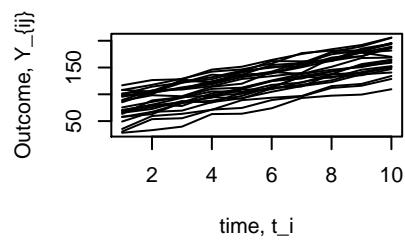
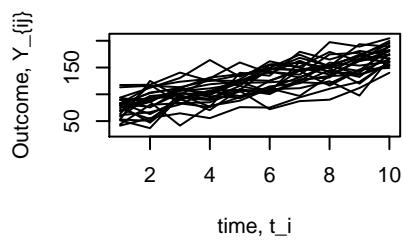
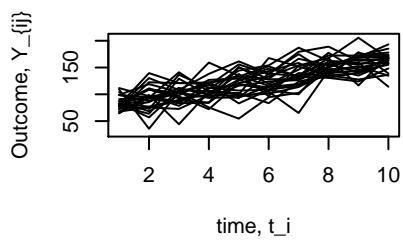
So we finally find that

$$\Sigma = \begin{bmatrix} \tau^2 + \sigma^2 & \tau^2 & \dots & \tau^2 \\ \tau^2 & \tau^2 + \sigma^2 & \dots & \tau^2 \\ \vdots & \vdots & \ddots & \vdots \\ \tau^2 & \tau^2 & \dots & \tau^2 + \sigma^2 \end{bmatrix}.$$

If we had 10 observations on each of 100 subjects, we can investigate the assumptions of random intercepts and slopes by performing linear regression on each subject and creating a histogram of the estimated intercepts and slopes. From the histogram, we would hopefully be able to identify the appropriate distribution. QQ-plots of the residuals could be helpful in determining whether the standard assumption of normality appears to be satisfied.

The table below serves as a legend to the 11 plots below:

Random Intercepts	$(\sigma, \tau) = (20, 5)$	$(\sigma, \tau) = (\sqrt{425/2}, \sqrt{425/2})$	$(\sigma, \tau) = (5, 20)$
Random Interc/Slopes	$(\sigma, \mathbf{D}) = (1, \begin{bmatrix} 100 & 0 \\ 0 & 2 \end{bmatrix})$	$(\sigma, \mathbf{D}) = (1, \begin{bmatrix} 100 & -2 \\ -2 & 2 \end{bmatrix})$	$(\sigma, \mathbf{D}) = (1, \begin{bmatrix} 50 & 0 \\ 0 & 4 \end{bmatrix})$
Serial Correlation	$(\sigma, \tau, \rho) = (10, 20, 0.7)$	$(\sigma, \tau, \rho) = (10, 20, 0.9)$	$(\sigma, \tau, \rho) = (5, 20, 0.9)$
Non-Normal Rand Eff	$(\sigma, \tau, F_b) = (5, 10, \text{Exp}(1))$	$(\sigma, \tau, F_b) = (5, 10, \chi^2(4))$	



R code

```
#####
##### Question 1 #####
#####
#
##### Function to sample data
#
getData <- function( beta ){
nobs <- 200
#
##### a linear predictor
#
x1 <- (c(1:nobs)-nobs/2)/nobs
#
##### a binary predictor
#
x2 <- rep( c(0,1), c(nobs,nobs)/2 )
#
X <- cbind( 1, x1, x2 )
mu <- exp( X %*% beta )
data <- list( X, mu )
names( data ) <- c( "X", "mu" )
data
}
#
#####
#####
##### Part (a): ARE when true model is N-B
#####
#
##### Function to calculate ARE when true model is N-B
#
areNB <- function( beta, alpha ){
data <- getData( beta )
X <- as.matrix( data$X )
mu <- as.vector( data$mu )
A.model <- t(X)%*%diag(mu)%*%X
B.true <- t(X)%*%diag( mu*(1+alpha*mu) )%*%X
A.optimal <- t(X)%*%diag( mu/(1+alpha*mu) )%*%X
#
se.poisson <- sqrt( diag( solve(A.model)%*%B.true%*%solve(A.model) ) )
se.optimal <- sqrt( diag( solve(A.optimal) ) )
#
cbind( se.poisson, se.optimal, (se.optimal)^2 / (se.poisson)^2 )
}
#
areNB( beta=c( 1, 1.5, 1 ), alpha=0.25 )
```

```

areNB( beta=c( 1, 1.5, 1 ), alpha=0.50 )
areNB( beta=c( 1, 1.5, 1 ), alpha=1.0 )
#
#####
##### Part (b): Effect of changing beta
#####
#
areNB( beta=c( 1, 0.0, 1 ), alpha=1.0 )
areNB( beta=c( 1, 0.5, 1 ), alpha=1.0 )
areNB( beta=c( 1, 1.0, 1 ), alpha=1.0 )
#
#####
##### Part (c): What if really scale and we fit negbin?
#####
#
##### Function to calculate ARE when true model is scale, but N-B is fit
#
areScale <- function( beta, scale ){
  data <- getData( beta )
  X <- as.matrix( data$X )
  mu <- as.vector( data$mu )
  A.model <- t(X) %*% diag(mu) %*% X
  A.optimal <- t(X) %*% diag( mu ) %*% X / scale
  #
  alpha <- mean( (scale-1)/mu )
  alpha <- (scale-1)/mean( mu )
  A.negbin <- t(X) %*% diag( mu/(1+alpha*mu) ) %*% X
  B.negbin <- t(X) %*% diag( scale*mu/(1+alpha*mu)^2 ) %*% X
  se.negbin <- sqrt( diag( solve(A.negbin) %*% B.negbin %*% solve(A.negbin) ) )
  se.optimal <- sqrt( diag( solve(A.optimal) ) )
  #
  cbind( se.negbin, se.optimal, (se.optimal)^2 / (se.negbin)^2 )
}
#
areScale( beta=c( 1, 1.5, 1 ), scale=2.0 )
areScale( beta=c( 1, 1.5, 1 ), scale=3.0 )
areScale( beta=c( 1, 1.5, 1 ), scale=4.0 )
#
#####
##### Part (d): Verification via simulation
#####
#
##### (1) alpha = 1.0
#
mu <- exp( 1.0 + 1.5*x1 + 1.0*x2 )
alpha <- 1.0
#
Nsim <- 1000

```

```

beta0 <- matrix( NA, Nsim, 3 )
beta1 <- matrix( NA, Nsim, 3 )
#
for( j in 1:Nsim ){
#
##### generate data
#
bbb <- rgamma( rep(1,nobs), shape=1/alpha ) * alpha
y <- rpois( rep(1,nobs), lambda=mu*bbb )
#
fit0 <- glm( y ~ x1 + x2, family=poisson )
beta0[j,] <- fit0$coef
#
maxits <- 10
converge <- F
fit <- fit0
count <- 0
while( !converge & count<maxits ){
old.beta <- fit$coef
mmm <- fitted(fit)
r2 <- (y-mmm)^2 / mmm
www <- 1/(1+alpha*mmm)
fit <- glm( y ~ x1 + x2, family=poisson, weight=www )
new.beta <- fit$coef
if( max( abs( new.beta-old.beta) ) < 1e-5 ) converge <- T
count <- count + 1
}
beta1[j,] <- new.beta
}
#
s0 <- sqrt( diag( var( beta0 ) ) )
s1 <- sqrt( diag( var( beta1 ) ) )
print( cbind( s0, s1, (s1/s0)^2 ) )
#
##### (2) scale = 4.0
#
scale <- 4.0
#
Nsim <- 1000
beta0 <- matrix( NA, Nsim, 3 )
beta1 <- matrix( NA, Nsim, 3 )
#
for( j in 1:Nsim ){
#
##### generate data
#
bbb <- rgamma( rep(1,nobs), shape=mu/(scale-1) ) * ( (scale-1)/mu )
y <- rpois( rep(1,nobs), lambda=mu*bbb )

```

```

#
fit0 <- glm( y ~ x1 + x2, family=poisson )
mu0 <- fitted(fit0)
beta0[j,] <- fit0$coef
#
maxits <- 10
converge <- F
fit <- fit0
count <- 0
while( !converge & count<maxits ){
old.beta <- fit$coef
mmm <- fitted(fit)
aaa <- mean( (scale - 1)/mu0 )
www <- 1/(1+aaa*mmm)
fit <- glm( y ~ x1 + x2, family=poisson, weight=www )
new.beta <- fit$coef
if( max( abs( new.beta-old.beta) ) < 1e-5 ) converge <- T
count <- count + 1
}
beta1[j,] <- new.beta
}
#
s0 <- sqrt( diag( var( beta0 ) ) )
s1 <- sqrt( diag( var( beta1 ) ) )
print( cbind( s0, s1, (s0/s1)^2 ) )
#
#####
##### Part (e): Effect of estimation of alpha
#####
#
alpha <- 1.0
#
Nsim <- 1000
beta0 <- matrix( NA, Nsim, 3 )
scale0 <- rep( NA, Nsim )
beta1 <- matrix( NA, Nsim, 3 )
alpha1 <- rep( NA, Nsim )
#
for( j in 1:Nsim ){
#
##### generate data
#
bbb <- rgamma( rep(1,nobs), shape=1/alpha ) * alpha
y <- rpois( rep(1,nobs), lambda=mu*bbb )
#
fit0 <- glm( y ~ x1 + x2, family=poisson )
mu0 <- fitted(fit0)
beta0[j,] <- fit0$coef
}

```

```

r2 <- (y-mu0)^2 / mu0
scale0[j] <- mean(r2)
#
maxits <- 10
converge <- F
aaa <- mean( (r2 - 1)/mu0 )
fit <- fit0
count <- 0
while( !converge & count<maxits ){
old.beta <- fit$coef
mmm <- fitted(fit)
r2 <- (y-mmm)^2 / mmm
aaa <- mean( (r2 - 1)/mu0 )
www <- 1/(1+aaa*mmm)
fit <- glm( y ~ x1 + x2, family=poisson, weight=www )
new.beta <- fit$coef
if( max( abs( new.beta-old.beta) ) < 1e-5 ) converge <- T
count <- count + 1
}
beta1[j,] <- new.beta
alpha1[j] <- aaa
}
#
s0 <- sqrt( diag( var( beta0 ) ) )
s1 <- sqrt( diag( var( beta1 ) ) )
print( cbind( s0, s1, (s1/s0)^2 ) )

sim.var1 <- sqrt(apply((apply(beta0, 1, function(x) { (x-apply(beta0, 2, mean))^2 } )), 1, sum)/1000)
sim.var2 <- sqrt(apply((apply(beta1, 1, function(x) { (x-apply(beta1, 2, mean))^2 } )), 1, sum)/1000)
print( cbind( sim.var1, sim.var2, (sim.var2/sim.var1)^2 ) )
#####
##### Question 2 #####
#####
library(mvtnorm)
#
##
##### Fixed scalars/vectors
##
#
n <- 10
m <- 25
beta <- c( 70, 10 )
t <- 1:10
MeanModel <- beta[1] + beta[2] * t
#
##
##### Random Intercepts
##

```

```

#
GenDataRI <- function( sigma, tau ) {
  b0i <- rnorm( m, mean = 0, sd = tau ) ## 25 random effects
  b0i <- rep( b0i, rep( n, m ) ) ## expand out the intercept random effects
  epsilon <- rnorm( (n*m), mean = 0, sd = sigma ) ## residual error
  Y <- rep( MeanModel, m ) + b0i + epsilon
  Y <- matrix( Y, nrow = m, byrow = T )
}

#
## 
##### Random Intercepts and Slopes
##
#
GenDataRIS <- function( sigma, D ) {
  bi <- rmvnorm( m, mean = c(0, 0), sigma = D ) ## 25 sets of random effects
  b0i <- rep( bi[,1], rep( n, m ) ) ## expand out the intercept random effects
  b1i <- rep( bi[,2], rep( n, m ) ) ## expand out slope random effects
  epsilon <- rnorm( (n*m), mean = 0, sd = sigma ) ## residual error
  Y <- rep( MeanModel, m ) + b0i + ( b1i * rep(t, m) ) + epsilon
  Y <- matrix( Y, nrow = m, byrow = T )
}

#
## 
##### Serial Correlation
##
#
GenDataSC <- function( sigma, tau, rho ) {
  Sigma <- matrix( tau^2, nrow = n, ncol = n ) + diag( rep(sigma^2, n) )
  for( i in 1:n ){
    for( j in 1:n ){
      Sigma[i,j] <- Sigma[i,j] * rho^(abs(i-j))
    }
  }
  Y <- rmvnorm( m, MeanModel, sigma = Sigma )
}

#
## 
##### Non-normal random effects
##
#
GenDataNN <- function( sigma, tau, type ) {
  if(type==1) { # exponential random effects
    zi <- rexp( m, 1 )
    b0i <- tau*(zi-1) ## 25 random effects
    b0i <- rep( b0i, rep( n, m ) ) ## expand out the intercept random effects
    epsilon <- rnorm( (n*m), mean = 0, sd = sigma ) ## residual error
    Y <- rep( MeanModel, m ) + b0i + epsilon
    Y <- matrix( Y, nrow = m, byrow = T )
  }
}

```

```

}

if(type==2) { #chi-squared random effects
zi <- rchisq( m, 4 )
b0i <- tau*(zi-4)/sqrt(8) ## 25 random effects
  b0i <- rep( b0i, rep( n, m ) ) ## expand out the intercept random effects
  epsilon <- rnorm( (n*m), mean = 0, sd = sigma ) ## residual error
  Y <- rep( MeanModel, m ) + b0i + epsilon
  Y <- matrix( Y, nrow = m, byrow = T )
}

return(Y)
}
#
##
##### Generate all the data
##
#
RI.a <- GenDataRI( 20, 5 )
RI.b <- GenDataRI( sqrt(425/2), sqrt(425/2) )
RI.c <- GenDataRI( 5, 20 )
#
RIS.a <- GenDataRIS( 1, matrix( c(100,0,0,2), nrow = 2 ) )
RIS.b <- GenDataRIS( 1, matrix( c(100,-2,-2,2), nrow = 2 ) )
RIS.c <- GenDataRIS( 1, matrix( c(50,0,0,4), nrow = 2 ) )
#
SC.a <- GenDataSC( 10, 20, 0.7 )
SC.b <- GenDataSC( 10, 20, 0.9 )
SC.c <- GenDataSC( 5, 20, 0.9 )
#
NN.a <- GenDataNN( 5, 10, 1 )
NN.b <- GenDataNN( 5, 10, 2 )
#
##
#####
# Produce Plots
##
#
pdf( file="exer4-Q2-plot.pdf" )
par( mfrow = c(4,3) )
#
PlotLimits.RI <- range( RI.a, RI.b, RI.c )
plot( t, RI.a[1], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.RI, type = "l" )
for( i in 2:m ) lines( t, RI.a[i] )
plot( t, RI.b[1], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.RI, type = "l" )
for( i in 2:m ) lines( t, RI.b[i] )
plot( t, RI.c[1], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.RI, type = "l" )
for( i in 2:m ) lines( t, RI.c[i] )
#
PlotLimits.RIS <- range( RIS.a, RIS.b, RIS.c )
plot( t, RIS.a[1], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.RIS, type = "l" )

```

```

for( i in 2:m ) lines( t, RIS.a[i,] )
plot( t, RIS.b[1,], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.RIS, type = "l" )
for( i in 2:m ) lines( t, RIS.b[i,] )
plot( t, RIS.c[1,], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.RIS, type = "l" )
for( i in 2:m ) lines( t, RIS.c[i,] )
#
PlotLimits.SC <- range( SC.a, SC.b, SC.c )
plot( t, SC.a[1,], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.SC, type = "l" )
for( i in 2:m ) lines( t, SC.a[i,] )
plot( t, SC.b[1,], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.SC, type = "l" )
for( i in 2:m ) lines( t, SC.b[i,] )
plot( t, SC.c[1,], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.SC, type = "l" )
for( i in 2:m ) lines( t, SC.c[i,] )
#
PlotLimits.NN <- range( NN.a, NN.b )
plot( t, NN.a[1,], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.NN, type = "l" )
for( i in 2:m ) lines( t, NN.a[i,] )
plot( t, NN.b[1,], xlab = "time, t_i", ylab = "Outcome, Y_{ij}", ylim = PlotLimits.NN, type = "l" )
for( i in 2:m ) lines( t, NN.b[i,] )
#
dev.off()
#

```