Biostat/Stat 571 Exercise #3

Answer Key

Problem 1:

One important characteristic of "working activity" for bees is the number of bees that leave the hive per hour. The file `bees.data` contain bee count (`Number`) over 11 hours in the day (`Time`), and over several successive non-rainy days.

To characterise the number of bees leaving as a function of time, one might be tempted to assume that the observed counts come from some underlying Poisson process, where the rate is a function of the time of day. Fitting a Poisson regression would the be appropriate. It is important to consider that the data may exhibit extra variation, relative to a Poisson distribution (i.e. overdispersion). Here we are going to investigate the extent of overdispersion in the `Number` counts.

(a) Before we investigate the extent of overdispersion in the data it is important to ensure that we have adequately modelled the mean as a function of time. To do this we can entertain a series of mean models, which increase in their flexibility but at the same time increase in the number of parameters. We consider 7 possible mean models:

model 1	linear in `Time`
model 2	quadratic in `Time`
model 3	cubic in `Time`
model 4	linear splines in `Time` with knots at 9, 11, 13, 15
model 4	cubic splines in `Time` with knots at 9, 11, 13, 15
model 4	natural splines in `Time` with knots at 9, 11, 13, 15
model 7	saturated model using `factor(Time)`

Figure 1 shows fitted means for mean models 1-6. In addition, for each plot the fitted means for the saturated model is included. In terms of fitting the data the saturated model provides the best that we can hope for (given the data). It also includes the largest number of parameters. We can therefore compare the fitted means from models 1-6 in the hope of finding a relatively parsimonious model that adequately describes the trends. We can see that the closest correspondence comes with models 5 and 6, with models 2-4 not being too bad. Figure 2 provides the Pearson residuals from models 1-6. Using a lowess smoother we can assess any trends in the residuals. Here we see similar trends to Figure 1, where models 2-6 seems to be doing relatively well and model 1 does very poorly.

Taking what we can see from Figures 1 and 2 it seems that the "best" choice for a mean model might be either model 5 or model 6.

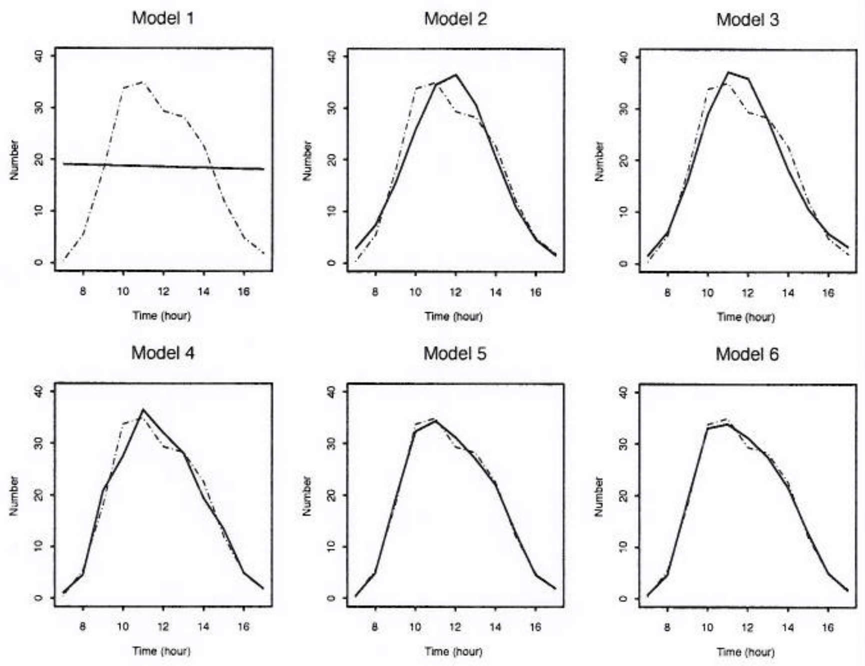


Figure 1: Solid lines: of the fitted means versus `Time` for each of models 1-6. Dashed lines: fitted means for the saturated mean model (model 7).

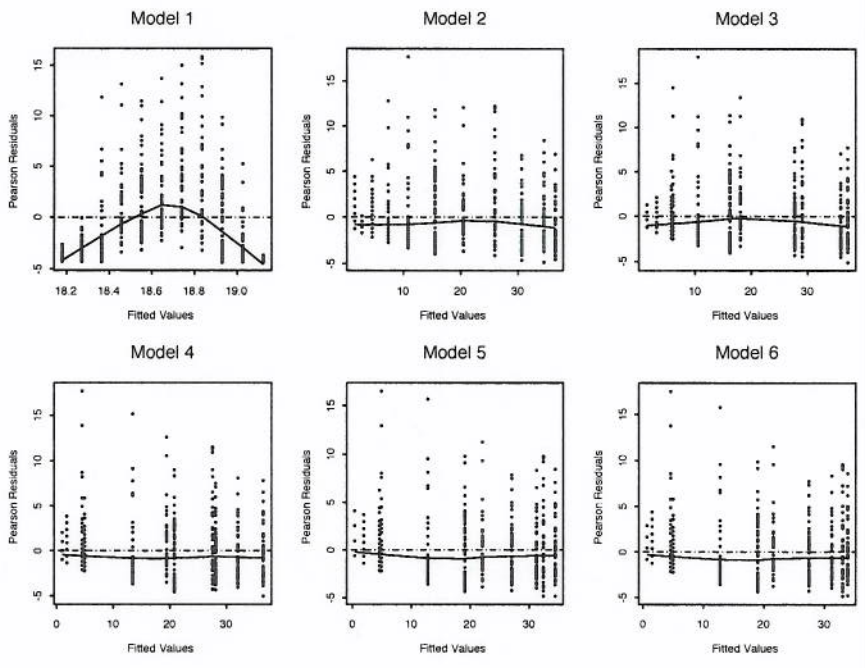


Figure 2: Pearson residuals versus fitted values. A lowess smoother has been applied to each plot to investigate mean model misspecification.

(b) Figure 3 provides a plot of the squared Pearson residuals versus the fitted means. Under the Poisson assumption, we would expect these quantities to have expectation equal to 1. A lowess smoother applied to the data provides evidence that this is not the case, indicating that the data are indeed overdispersed. It isn't clear whether a scaled variance (which would result in a flat line greater than 1) or a negative binomial variance (which would result in an increasing straight line) are appropriate here. Given the sparse data for low fitted values, I would be less inclined to trust the dips on the left-hand sides of the plots. Overall I am inclined to beleive that a scale form for the variance is more likely.

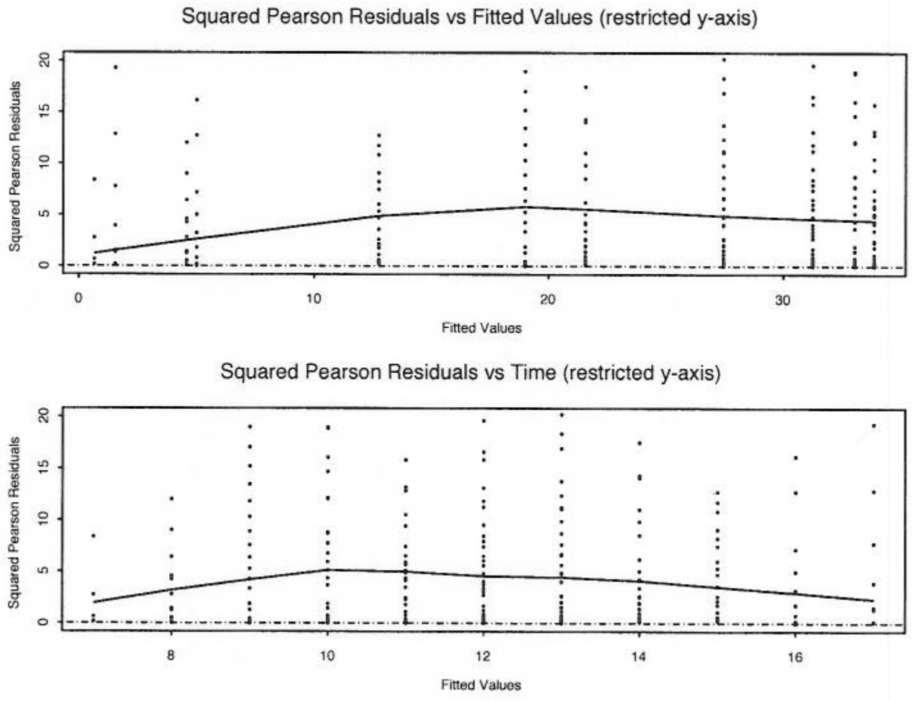


Figure 3: Residual plots to examine the extend of overdispersion (relative to a Poisson distribution) in the bee count data.

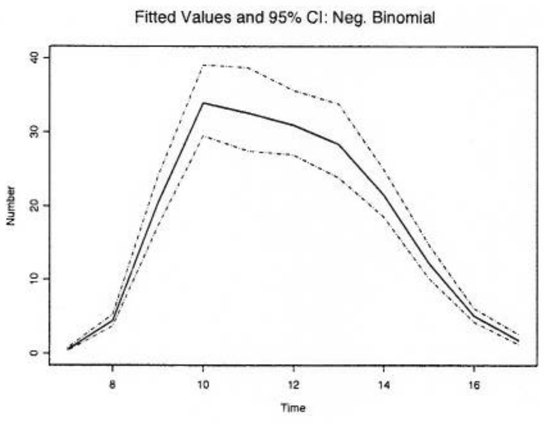
(c) Initially, we can consider fitting a negative binomial model to the overdispersed bee data, using each of the 7 mean models indicated above. These were performed in S-Plus using the `glm.nb` function, with the loglikelihood being reported as part of the output. Note that these results differ from the AIC values produced by the `AIC` function in S-plus.

Given this specification of the AIC, larger values indicate "better" models. We can see from Table 1 that the two models with the highest AIC values are models 5 and 6. Given that model 6 has fewer parameters (7 versus 9), for only a slight decrease (relatively) in the AIC value, I am going to use model 6 as a plausible parsimonious model. Figure 4(a) provides the fitted means, as well as pointwise 95% confidence intervals, as functions of time.

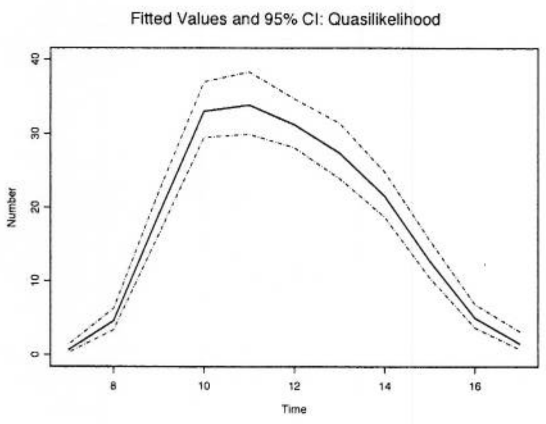
Mean Model	2 × LogLike	# of parameters	AIC
model 1	43578.8	3	43572.8
model 2	43922.2	4	43914.2
model 3	43953.4	5	43943.4
model 4	43964.1	7	43950.1
model 5	43986.2	9	43968.2
model 6	43979.9	7	43965.9
model 7	43988.0	12	43964.0

Table 1: Computed AIC values : 2 × LogLike - 2 × # of parameters

(d) Figure 4(b) examines the results from the fit of a quasilikelihood model using the same mean model (model 6) as in 1(c). In particular, the variance is chosen to be the scale variance form. That is, it assumes that the mean-variance relationship is of the following form : $V(\mu) = \phi\mu$. From the S-Plus fit of the model the scale parameter ϕ is estimated to be 10.11354.



(a) Negative Binomial



(b) Quasilikelihood

Figure 4: Results from two fits, each using mean model 6 (natural splines).

S-Plus Code

```
#####
##### Read the data
#####
#
bees <- read.table( "C:\\TA\\571_W03\\data\\bee_data.txt", header=T )
#
#####
##### Part (a) : fit a number of models to evaluate the mean form
#####
#
model1 <- glm( Number ~ Time, family = poisson, data = bees )
model2 <- glm( Number ~ Time + Time^2, family = poisson, data = bees )
model3 <- glm( Number ~ Time + Time^2 + Time^3, family = poisson, data = bees )
model4 <- glm( Number ~ bs( Time, knots = c(9,11,13,15), degree=1 ), family = poisson, data = bees )
model5 <- glm( Number ~ bs( Time, knots = c(9,11,13,15) ), family = poisson, data = bees )
model6 <- glm( Number ~ ns( Time, knots = c(9,11,13,15) ), family = poisson, data = bees )
model7 <- glm( Number ~ factor(Time), family = poisson, data = bees )
#
uT <- unique(bees$Time)
uT <- uT[order(uT)]
uf1 <- predict( model1, newdata = list(Time = uT), type = "response")
uf2 <- predict( model2, newdata = list(Time = uT), type = "response")
uf3 <- predict( model3, newdata = list(Time = uT), type = "response")
uf4 <- predict( model4, newdata = list(Time = uT), type = "response")
uf5 <- predict( model5, newdata = list(Time = uT), type = "response")
uf6 <- predict( model6, newdata = list(Time = uT), type = "response")
uf7 <- predict( model7, newdata = list(Time = uT), type = "response")
#
par( mfrow = c(2,3) )
plot( uT, uf1, xlab = "Time (hour)", ylab = "Number", main = "Model 1", ylim = c(0,40), type = "l" ); lines( uT, uf7, lty = 3 )
plot( uT, uf2, xlab = "Time (hour)", ylab = "Number", main = "Model 2", ylim = c(0,40), type = "l" ); lines( uT, uf7, lty = 3 )
plot( uT, uf3, xlab = "Time (hour)", ylab = "Number", main = "Model 3", ylim = c(0,40), type = "l" ); lines( uT, uf7, lty = 3 )
plot( uT, uf4, xlab = "Time (hour)", ylab = "Number", main = "Model 4", ylim = c(0,40), type = "l" ); lines( uT, uf7, lty = 3 )
plot( uT, uf5, xlab = "Time (hour)", ylab = "Number", main = "Model 5", ylim = c(0,40), type = "l" ); lines( uT, uf7, lty = 3 )
plot( uT, uf6, xlab = "Time (hour)", ylab = "Number", main = "Model 6", ylim = c(0,40), type = "l" ); lines( uT, uf7, lty = 3 )
#
par( mfrow = c(2,3) )
plot( model1$fitted, residuals(model1, type = "pearson"), xlab = "Fitted Values", ylab = "Pearson Residuals", main = "Model 1", pch = "." )
lines(lowess( model1$fitted, residuals(model1, type = "pearson") ))
abline( h = 0, lty = 3 )
plot( model2$fitted, residuals(model2, type = "pearson"), xlab = "Fitted Values", ylab = "Pearson Residuals", main = "Model 2", pch = "." )
lines(lowess( model2$fitted, residuals(model2, type = "pearson") ))
abline( h = 0, lty = 3 )
plot( model3$fitted, residuals(model3, type = "pearson"), xlab = "Fitted Values", ylab = "Pearson Residuals", main = "Model 3", pch = "." )
lines(lowess( model3$fitted, residuals(model3, type = "pearson") ))
abline( h = 0, lty = 3 )
plot( model4$fitted, residuals(model4, type = "pearson"), xlab = "Fitted Values", ylab = "Pearson Residuals", main = "Model 4", pch = "." )
lines(lowess( model4$fitted, residuals(model4, type = "pearson") ))
abline( h = 0, lty = 3 )
plot( model5$fitted, residuals(model5, type = "pearson"), xlab = "Fitted Values", ylab = "Pearson Residuals", main = "Model 5", pch = "." )
lines(lowess( model5$fitted, residuals(model5, type = "pearson") ))
abline( h = 0, lty = 3 )
plot( model6$fitted, residuals(model6, type = "pearson"), xlab = "Fitted Values", ylab = "Pearson Residuals", main = "Model 6", pch = "." )
lines(lowess( model6$fitted, residuals(model6, type = "pearson") ))
abline( h = 0, lty = 3 )
#
#####
##### Part (b) : fit flexible model for time and evaluate overdispersion
#####
#
model0 <- glm( Number ~ ns( Time, knots = c(9,11,13,15) ), family = poisson, data = bees )
#
summary( model0, cor = F )
#
par( mfrow = c(2,1) )
plot( model0$fitted, residuals(model0, type = "pearson")^2, xlab = "Fitted Values", ylab = "Squared Pearson Residuals",
 main = "Squared Pearson Residuals vs Fitted Values (restricted y-axis)", pch = ".", ylim = c(0,20) )
lines(lowess( model0$fitted, residuals(model0, type = "pearson")^2 ))
abline( h = 0, lty = 3 )
plot( bees$Time, residuals(model0, type = "pearson")^2, xlab = "Fitted Values", ylab = "Squared Pearson Residuals",
 main = "Squared Pearson Residuals vs Time (restricted y-axis)", pch = ".", ylim = c(0,20) )
lines(lowess( bees$Time, residuals(model0, type = "pearson")^2 ))
abline( h = 0, lty = 3 )
#
#####
##### Part (c) : use negative binomial model and AIC to identify adequate mean functions
#####
#
library( mass )
model1.nb <- glm.nb( Number ~ Time, data = bees )
```

```
model2.nb <- glm.nb( Number ~ Time + Time^2, data = bees )
model3.nb <- glm.nb( Number ~ Time + Time^2 + Time^3, data = bees )
model4.nb <- glm.nb( Number ~ bs( Time, knots = c(9,11,13,15), degree=1 ), data = bees )
model5.nb <- glm.nb( Number ~ bs( Time, knots = c(9,11,13,15) ), data = bees )
model6.nb <- glm.nb( Number ~ ns( Time, knots = c(9,11,13,15) ), data = bees )
model7.nb <- glm.nb( Number ~ factor(Time), data = bees )
#
## Compute the AIC : bigger is better
#
AIC1.nb <- c( model1.nb$twologlik, length(model1.nb$coef) + 1, model1.nb$twologlik - 2*(length(model1.nb$coef) + 1) )
AIC2.nb <- c( model2.nb$twologlik, length(model2.nb$coef) + 1, model2.nb$twologlik - 2*(length(model2.nb$coef) + 1) )
AIC3.nb <- c( model3.nb$twologlik, length(model3.nb$coef) + 1, model3.nb$twologlik - 2*(length(model3.nb$coef) + 1) )
AIC4.nb <- c( model4.nb$twologlik, length(model4.nb$coef) + 1, model4.nb$twologlik - 2*(length(model4.nb$coef) + 1) )
AIC5.nb <- c( model5.nb$twologlik, length(model5.nb$coef) + 1, model5.nb$twologlik - 2*(length(model5.nb$coef) + 1) )
AIC6.nb <- c( model6.nb$twologlik, length(model6.nb$coef) + 1, model6.nb$twologlik - 2*(length(model6.nb$coef) + 1) )
AIC7.nb <- c( model7.nb$twologlik, length(model7.nb$coef) + 1, model7.nb$twologlik - 2*(length(model7.nb$coef) + 1) )
#
round(rbind( AIC1.nb, AIC2.nb, AIC3.nb, AIC4.nb, AIC5.nb, AIC6.nb, AIC7.nb ), 1)
#
## Choose the natural splines model
#
results <- predict( model6.nb, newdata = list(Time = uT), type = "response", ci.fit = T )
plot( uT, results$fit, type = "l", xlab = "Time", ylab = "Number",
main = "Fitted Values and 95% CI: Neg. Binomial", ylim = c(0,40) )
lines( uT, results$ci.fit[,1], lty = 3 )
lines( uT, results$ci.fit[,2], lty = 3 )
#
#####
##### Part (d) : comparison with quasilikelihood
#####
#
model6.q <- glm( Number ~ ns( Time, knots = c(9,11,13,15) ), quasi(link = log, variance = mu), data = bees )
results <- predict( model6.q, newdata = list(Time = uT), type = "response", ci.fit = T )
plot( uT, results$fit, type = "l", xlab = "Time", ylab = "Number",
main = "Fitted Values and 95% CI: Quasilikelihood", ylim = c(0,40) )
lines( uT, results$ci.fit[,1], lty = 3 )
lines( uT, results$ci.fit[,2], lty = 3 )
```

Problem 2:

(a)

$$\text{E }[Y_i] = \text{ E }_\nu[E_Y(Y_i)] = \text{ E }_\nu[\lambda_i \nu_i] = \lambda_i \text{ E }_\nu[\nu_i] = \lambda_i p_i \equiv \mu_i$$

$$\begin{aligned}
\text{Var }[Y_i] &= \text{ E }_{nu}[\text{ Var }_Y(Y_i)] + \text{ Var }_\nu[\text{ E }_Y(Y_i)] \\
&= \text{ E }_\nu[\lambda_i \nu_i] + \text{ Var }_\nu[\lambda_i \nu_i] \\
&= \lambda_i \text{ E }_\nu[\nu_i] + \lambda_i^2 \text{ Var }_\nu[\nu_i] \\
&= \lambda_i p_i + \lambda_i^2 p_i(1 - p_i) \\
&= \mu_i + \frac{\mu_i^2}{p_i} - \mu_i^2 \\
&= \mu_i + \left(\frac{1 - p_i}{p_i}\right) \mu_i^2 \\
&\equiv \mu_i + \alpha_i \mu_i^2
\end{aligned}$$

(b) According to White (1982), the asymptotic distribution of the regression estimator is $\sqrt{n}(\hat{\beta} - \beta_0) \rightarrow_d \mathcal{N}(0, A^{-1}BA^{-1})$. Consequently, the variance of the estimator $\hat{\beta}$ is of the form :

$$\begin{aligned}
\text{Var }[\hat{\beta}] &= \frac{1}{n} A^{-1} B A^{-1}, \qquad \text{where} \\
A &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{v_i} \frac{\partial \mu_i}{\partial \beta} \frac{\partial \mu_i}{\partial \beta^t} |_{\beta_0}, \qquad \text{and} \\
B &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \frac{w_i}{v_i^2} \frac{\partial \mu_i}{\partial \beta} \frac{\partial \mu_i}{\partial \beta^t} |_{\beta_0}.
\end{aligned}$$

Now, for the Poisson family $v_i = \mu_i$, and for the mixture model of interest $w_i = \text{ Var }[Y_i] = \mu_i + \alpha_i \mu_i^2$. Noting that

$$\frac{\partial \mu_i}{\partial \beta} = \left(\frac{\partial \mu_i}{\partial \eta_i}\right)\left(\frac{\partial \eta_i}{\partial \beta}\right) = \mu_i \mathbf{X}_i,$$

and substituting in we have that,

$$\begin{aligned}
A &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mu_i \mathbf{X}_i^t \mathbf{X}_i \\
B &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (\mu_i + \alpha_i \mu_i^2) \mathbf{X}_i^t \mathbf{X}_i.
\end{aligned}$$

(c) First note that the A matrix given above remains the same. Now, assuming the scale model, ie. Var $[Y_i] = \phi\mu_i$, we have that $w_i = \phi\mu_i$ and hence

$$B = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \phi\mu_i \mathbf{X}_i^t \mathbf{X}_i.$$

(d) Again, the A matrix given in (b) remains the same. Assuming the negative-binomial variance model, ie. Var $[Y_i] = \mu_i + \alpha\mu_i^2$, we have that $w_i = \mu_i + \alpha\mu_i^2$ and hence

$$B = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (\mu_i + \alpha\mu_i^2) \mathbf{X}_i^t \mathbf{X}_i.$$

(e) As seen in part (a),

$$\text{Var } [Y_i] = \mu_i + \left(\frac{1-p_i}{p_i}\right)\mu_i^2,$$

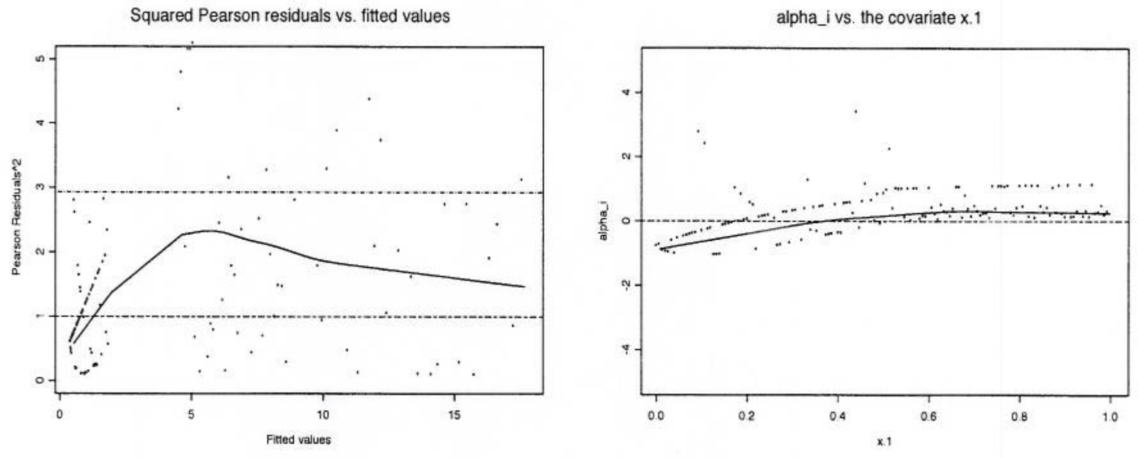
thus we will only obtain the scale variance in (c) if $\left(\frac{p_i}{1-p_i}\right) \propto \mu_i$, ie. if the odds of having potential for a positive count are proportional to the count rate given you have a the potential for a positive count. Also, the mixture model variance given in (d) is only obtained if $\left(\frac{1-p_i}{p_i}\right)$ is constant (independent) with respect to X_i.

(f) Code to simulate the data presented is given at the end of this problem. Figure 1(a) yields a plot of the squared Pearson residuals vs. the fitted values resulting from a Poisson regression fit. We can see that there is an irregular trend in the plot, suggesting that the mean-variance relationship is not as would be expected by the Poisson process. If the Poisson regression were correct, the expected squared Pearson residual value would be 1, but the smoothed line through the residuals does not look like the horizontal line $R^2 = 1$, and also doesn't resemble the horizontal line $R^2 = \hat{\phi} = 1.32$ also shown on the plot. Hence the scale model variance also does not seem very appropriate. If the smoother resembled a straight line with slope α and intercept 1, this would correspond to the variance function given in (d), since

$$\begin{aligned}\frac{(y-\mu)^2}{\mu} = 1 + \alpha\mu \to (y-\mu)^2 &= \mu + \alpha\mu^2 \\ \to \text{Var } [Y] &= \mu + \alpha\mu^2.\end{aligned}$$

In my plot, the smoothed line does not conform with what we would expect under either the scale or negative binomial variance forms.

To diagnose whether α_i was dependent on X_i, I plotted estimates of α_i, given by $\hat{\alpha}_i = \frac{R_i^2 - 1}{\mu_i}$, vs. X_i in Figure 1(b). The smoother is not a horizontal line suggesting that there is a relationship between α_i and X_i. It should be noted that there was great deal of variability in these plots and many of them reveal no relationship!



(a) (b)

Figure 5: Residual plots to examine departures from the Poisson variance assumption.

(g)-(h) For programming results, see the Splus code at the end of this problem. Parameter and standard error estimates are given below. We can see that for each parameter, the scale, mixture (beta-binomial) and empirical standard error estimates differ substantially from the naive Poisson-based estimates. It seems that the standard error estimates from both the scale and mixture models are relatively close to the empirical estimates for both the intercept and the coefficient associated with X_2. However, for X_1 the scale and mixture model standard error estimates are different from the empirical estimates. This is intuitive, since the underlying data generating mechanism depends on X_1 more substantially than we allow for.

```
               beta se.model se.scale se.mix se.empirical
(Intercept) -0.6870   0.1360   0.2326 0.2165       0.2040
        x.1  2.7169   0.2542   0.4349 0.5541       0.4835
        x.2  0.8429   0.1728   0.2957 0.3013       0.2948
```

(i) For programming results, see the Splus code at the end of this problem. The table below presents the 'true' standard error estimates of $\hat{\beta}$ computed across the 1000 simulations, as well as mean sandwich estimates. The simulation results agree with what was found in part (h). The model (Poisson) based standard errors are heavily biased (due to the overdispersion!), while for the intercept and X_2 the scale and mixture model estimates are approximately unbiased. However, these two variance forms perform poorly with respect to X_1. As we would expect the empirical estimates perform well, although there is an indication of slight negative bias.

```
          true.se se.model se.scale se.mix se.empirical
Intercept  0.2017   0.1334   0.2259 0.2155       0.2016
      x.1  0.5155   0.2529   0.4285 0.5625       0.4907
      x.2  0.2991   0.1715   0.2905 0.3079       0.2875
```

Simulated coverage probabilities for each of the considered standard error estimates are given below. We can see that for each of the sandwich estimators, the coverage probability remained very close to the nominal 95%, with the exception of the scale variance form coverage for the X_1 variable. The coverage probabilities for the naive model are very low in comparison to the nominal 95%.

```
          se.model se.scale se.mix se.empirical
Intercept    0.806    0.978  0.964        0.947
      x.1    0.682    0.895  0.964        0.935
      x.2    0.740    0.951  0.958        0.946
```

(j) For programming results, see the Splus code at the end of this problem. The table below provides simulation results for the negative binomial model. Firstly, we can see that the standard errors that are reported by the `glm.nb` function (i.e. assuming that the true variance form is that of a negative binomial) are biased. For each of the coefficients, the standard error estimates are too large. We also find that the coverage probabilities are too large as well. That is our inference will be too conservative (we would reject the null hypothesis too often).

```
          true.nb.se se.negbin  coverage
Intercept     0.2041    0.2428     0.984
      x.1     0.4832    0.6624     0.990
      x.2     0.2824    0.3759     0.987
```

Both the negative binomial and poisson regression estimators are biased, but in different directions. However, if you compare the "`true`" standard error estimates (i.e. the simulation-based standard error estimates), then the negative binomial distributional assumption results in estimators of β that are slightly more efficient than the estimators under the Poisson assumption.

Splus Code for Problem 2:

```
##
#####  Part f  #####
##
#####  Function to simulate overdispered data  #####
##
getODdata <- function( nobs, gamma, delta ){
    x.1 <- c(1:nobs) / nobs
    x.2 <- c( rep(0, (nobs/2)), rep(1, (nobs/2)) )
    X <- cbind( 1, x.1, x.2 )
    lambda <- exp( X %*% gamma )
    p <- exp( X[,1:2] %*% delta )
    nu <- rbinom( nobs, 1, p )
    nu[ nu == 0 ] <- 10e-10
    y <- rpois( nobs, lambda * nu )
    return( as.data.frame( cbind( y, x.1, x.2 ) ) )
}
simData <- getODdata( nobs=150, gamma=c( 0, 2, 1 ), delta=c( log(.6), log(.8/.6) ) )
##
####  GLM fit
##
fit <- glm( y ~ x.1 + x.2 , data = simData, family = poisson )
summary( fit )
##
resids <- residuals( fit, type = "pearson" )
fits <- fitted( fit )
plot(    fits, resids^2, pch="*", xlab="Fitted values", ylab="Pearson Residuals^2", main="Squared Pearson residuals vs. fitted values", ylim=c
lines(lowess(fits, resids^2))
abline( sum(resids^2) / (150 - 2 ), 0, lty=3)
abline(1, 0, lty = 6)
##
alpha.hat <- ( resids^2 - 1 ) / fits
summary( lm( alpha.hat ~ x.1, data=simData ) )
plot(    simData$x.1, alpha.hat, pch="*", xlab="x.1", ylab="alpha_i", main="alpha_i vs. the covariate x.1", ylim=c(-5,5) )
lines( lowess( simData$x.1, alpha.hat) )
abline( 0, 0, lty = 6 )
##
#####  Parts g and h  #####
##
#####  Function to calculate standard errors  #####
```

```
###
get.se <- function( data ){
    fit <- glm( y ~ x.1 + x.2, data = data, family = poisson )
    resids <- residuals( fit, type="pearson" )
    fits <- fitted( fit )
    nobs <- length( resids )
    y <- data$y
    ##
    ####  (1) estimate of scale parameter
    ##
    phi <- sum( resids^2 )/( nobs - 2 )
    ##
    ####  (2) estimate of alpha
    ##
    alpha <- mean( ( ( resids )^2 - 1 ) / fits )
    ##
    ####  Calculate standard errors...
    ##
    X <- cbind( 1, data$x.1, data$x.2 )
    ##
    ####  Fisher Information
    ##
    XtmuX <- t( X ) %*% ( fits * X )
    I.inv <- solve( XtmuX )
    se.model <- sqrt( diag( I.inv ) )
    ##
    ####  with linear combination of alpha
    ##
    cheese <- t( X ) %*% ( ( fits + alpha * fits^2 ) * X )
    se.mix <- sqrt( diag( I.inv %*% (cheese) %*% I.inv ) )
    ##
    #### with scale parameter
    #
    se.scale <- sqrt( phi ) * se.model
    ##
    #### with empirical variance of estimating function
    ##
    est.fnx <- X * ( y - fits )
    UUt <- t( est.fnx ) %*% est.fnx
    se.empirical <- sqrt( diag( I.inv %*% UUt %*% I.inv ) )
    ##
    beta <- fit$coef
    out <- cbind( beta, se.model, se.scale, se.mix, se.empirical )
    return( round( out, 4 ) )
}
get.se( simData )
##
#####  Parts i and j  #####
##
##
#####  Function to calculate negative-binomial estimates and standard errors  #####
###
library( mass )
get.nb.se <- function( data ){
    fit <- glm.nb( y ~ x.1 + x.2, data = data, maxit = 20 )
    value <- summary(fit)$coef[,1:2]
    return( round( value, 4 ) )
}
beta <- matrix(NA, 1000, 3)
modelSE <- matrix(NA, 1000, 3)
scaleSE <- matrix(NA, 1000, 3)
mixSE <- matrix(NA, 1000, 3)
empiricalSE <- matrix(NA, 1000, 3)
beta.nb <- matrix( NA, 1000, 3)
negbinSE <- matrix( NA, 1000, 3)
for ( i in 1:1000 ){
    simData <- getODdata( nobs=150, gamma=c( 0, 2, 1 ), delta=c( log(.6), log(.8/.6) ) )
    RSLT <- get.se( simData )
    beta[ i, ] <- t( RSLT[ , 1 ] )
    modelSE[ i, ] <- t( RSLT[ , 2 ] )
    scaleSE[ i, ] <- t( RSLT[ , 3 ] )
    mixSE[ i, ] <- t( RSLT[ , 4 ] )
    empiricalSE[ i, ] <- t( RSLT[ , 5 ] )
    RSLT.nb <- get.nb.se( simData )
    beta.nb[ i, ] <- t( RSLT.nb[ ,1 ] )
    negbinSE[ i, ] <- t( RSLT.nb[ ,2 ] )
    if( i %% 50 == 0 ) print( i )
}
##
#####  Compute summary statistics #####
##
```

```
se.betas <- sqrt( apply( beta, 2, var ) )
mean.se.model <- apply( modelSE, 2, mean )
mean.se.mix <- apply( mixSE, 2, mean )
mean.se.scale <- apply( scaleSE, 2, mean )
mean.se.empirical <- apply( empiricalSE, 2, mean )

mean.estimates <- cbind( se.betas, mean.se.model, mean.se.scale, mean.se.mix, mean.se.empirical )
dimnames(mean.estimates) <- list( c( "Intercept", "x.1", "x.2" ), c( "true.se", "se.model", "se.scale", "se.mix", "se.empirical" ) )
round(mean.estimates, 4)

se.nb.betas <- sqrt( apply( beta.nb, 2, var ) )
mean.se.negbin <- apply( negbinSE, 2, mean )

mean.nb.estimates <- cbind( se.nb.betas, mean.se.negbin )
dimnames(mean.nb.estimates) <- list( c( "Intercept", "x.1", "x.2" ), c( "true.nb.se", "se.negbin" ) )
round(mean.nb.estimates, 4)


##
#####  Compute coverage probabilities
##
beta0 <- 0 + log(.6)
beta1 <- 2 + log(.8/.6)
beta2 <- 1
#
cov.prob <- matrix(NA, 3, 5)
dimnames(cov.prob) <- list( c( "Intercept", "x.1", "x.2" ), c( "se.model", "se.scale", "se.mix", "se.empirical", "se.negbin" ) )
#
cov.prob[1,1] <- sum(     beta0 >= beta[,1] - qnorm(.975) * modelSE[,1] &
                            beta0 <= beta[,1] + qnorm(.975) * modelSE[,1] ) / 1000
cov.prob[2,1] <- sum(    beta1 >= beta[,2] - qnorm(.975) * modelSE[,2] &
                            beta1 <= beta[,2] + qnorm(.975) * modelSE[,2] ) / 1000
cov.prob[3,1] <- sum(    beta2 >= beta[,3] - qnorm(.975) * modelSE[,3] &
                            beta2 <= beta[,3] + qnorm(.975) * modelSE[,3] ) / 1000

cov.prob[1,2] <- sum(     beta0 >= beta[,1] - qnorm(.975) * scaleSE[,1] &
                            beta0 <= beta[,1] + qnorm(.975) * scaleSE[,1] ) / 1000
cov.prob[2,2] <- sum(    beta1 >= beta[,2] - qnorm(.975) * scaleSE[,2] &
                            beta1 <= beta[,2] + qnorm(.975) * scaleSE[,2] ) / 1000
cov.prob[3,2] <- sum(    beta2 >= beta[,3] - qnorm(.975) * scaleSE[,3] &
                            beta2 <= beta[,3] + qnorm(.975) * scaleSE[,3] ) / 1000

cov.prob[1,3] <- sum(     beta0 >= beta[,1] - qnorm(.975) * mixSE[,1] &
                            beta0 <= beta[,1] + qnorm(.975) * mixSE[,1] ) / 1000
cov.prob[2,3] <- sum(    beta1 >= beta[,2] - qnorm(.975) * mixSE[,2] &
                            beta1 <= beta[,2] + qnorm(.975) * mixSE[,2] ) / 1000
cov.prob[3,3] <- sum(    beta2 >= beta[,3] - qnorm(.975) * mixSE[,3] &
                            beta2 <= beta[,3] + qnorm(.975) * mixSE[,3] ) / 1000

cov.prob[1,4] <- sum(     beta0 >= beta[,1] - qnorm(.975) * empiricalSE[,1] &
                            beta0 <= beta[,1] + qnorm(.975) * empiricalSE[,1] ) / 1000
cov.prob[2,4] <- sum(    beta1 >= beta[,2] - qnorm(.975) * empiricalSE[,2] &
                            beta1 <= beta[,2] + qnorm(.975) * empiricalSE[,2] ) / 1000
cov.prob[3,4] <- sum(    beta2 >= beta[,3] - qnorm(.975) * empiricalSE[,3] &
                            beta2 <= beta[,3] + qnorm(.975) * empiricalSE[,3] ) / 1000

cov.prob[1,5] <- sum(     beta0 >= beta[,1] - qnorm(.975) * negbinSE[,1] &
                            beta0 <= beta[,1] + qnorm(.975) * negbinSE[,1] ) / 1000
cov.prob[2,5] <- sum(    beta1 >= beta[,2] - qnorm(.975) * negbinSE[,2] &
                            beta1 <= beta[,2] + qnorm(.975) * negbinSE[,2] ) / 1000
cov.prob[3,5] <- sum(    beta2 >= beta[,3] - qnorm(.975) * negbinSE[,3] &
                            beta2 <= beta[,3] + qnorm(.975) * negbinSE[,3] ) / 1000
cov.prob
```