Poisson Regression Lab

Introduction

Poisson regression can be used when we want to model a response variable Y given a predictor or predictors X, where Y represents count data. The GLM model framework is applicable here, and the log-link function is the most appropriate since it maps positive data to the real line. Therefore the two components for GLM are

1. The response belongs to an exponential family (works for a poisson random variable), and
2. the link-function is $\ln(\lambda) = X^T\beta$.

Solving for λ in our link function we find that

$$\lambda = e^{x^T\beta},$$

and therefore our proposed poisson model is

$$P(y \mid x) = \frac{exp(-e^{x^T\beta})e^{x^T\beta y}}{y!}$$

Dataset set from Agresti, 2007.

- Data: Study of nesting horseshoe crabs (J. Brockman, Ethology, 1996). Each female crab in the study had a male crab attached to her in her nest. The study investigated factors that affect whether the female crab had any other males, called satellites, residing nearby her. Explanatory variables thought possibly to affect this included the female crab's color, spine condition, weight, and carapace width. The response outcome for each female crab is her number of satellites.
- Variable descriptions:
  - crab: crab id
  - color: 1 - light medium, 2 - medium, 3 - dark medium, 4 - dark
  - spine: 1 - both good, 2 - one worn or broken, 3 - both worn or broken
  - width: carapace width in cm
  - sat: number of satellites
  - weight: weight in kg

```r
crabs <- read.table("http://users.stat.ufl.edu/~aa/cat/data/Crabs.dat",
                    header = TRUE)
```

Problem 1

Fit a Poisson regression model using `weight` and `spine` as predictors for the number of satellites a female crab has. Comment on the results. How do the coefficients relate to the geometric mean of the data?

```r
crabs$spine <- as.factor(crabs$spine)
crabs$color <- as.factor(crabs$color)

fit1 <- glm(sat~weight+as.factor(spine),family=poisson,data=crabs)
summary(fit1)
```

```
##
## Call:
## glm(formula = sat ~ weight + as.factor(spine), family = poisson,
##     data = crabs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9823  -1.9861  -0.5900   0.9292   4.8768
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.32913    0.22949  -1.434    0.152
## weight              0.56918    0.06962   8.175 2.96e-16 ***
## as.factor(spine)2  -0.23894    0.20878  -1.144    0.252
## as.factor(spine)3  -0.04465    0.10777  -0.414    0.679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 632.79  on 172  degrees of freedom
## Residual deviance: 559.49  on 169  degrees of freedom
## AIC: 922.79
##
## Number of Fisher Scoring iterations: 5
```

```r
round(exp(coef(fit1)), 2)
```

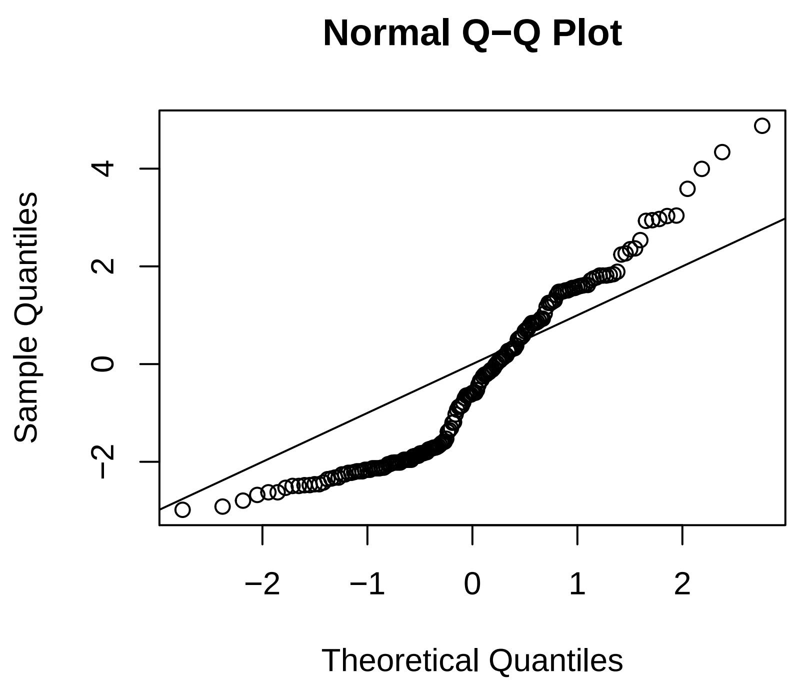
```
##       (Intercept)            weight as.factor(spine)2 as.factor(spine)3
##              0.72              1.77              0.79              0.96
```

Answer: The geometric mean of number of satellites is 0.77. Weight is highly significant in this model and increases the geometric mean by a factor of 1.73 for each one unit increase in weight. We don't find spine condition to be significant.

Problem 2

The deviance residuals can be obtained using the `resid` function on our model fit. Do these residuals follow a normal distribution? Perform a deviance test for goodness of fit. Comment on the results.

```r
qqnorm(resid(fit1))
abline(0,1)
```



```r
with(fit1, cbind(res.deviance = deviance, df = df.residual, p = 1-pchisq(deviance, df.residual)))
```

```
##      res.deviance  df p
## [1,]     559.4876 169 0
```

Answer: The residuals do not follow a normal distribution and the goodness of fit test produces a p-value less than 0.05 indicating that the current model is not a good fit.

Problem 3

It is very common in practice to find that the variance of the data is greater than the mean of the data, but the poisson model dictates that the mean and variance are both equal to λ. Therefore we can introduce a *dispersion parameter* ϕ, where now we model $V(Y) = \phi\lambda$. In R use `family = quasipoisson`. Fit a model with a dispersion parameter and comment on the results. Is there evidence of overdispersion?

```r
fit2 <- glm(sat~weight+as.factor(spine),family=quasipoisson,data=crabs)
summary(fit2)
```

```
##
## Call:
## glm(formula = sat ~ weight + as.factor(spine), family = quasipoisson,
##     data = crabs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9823  -1.9861  -0.5900   0.9292   4.8768
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.32913    0.40701  -0.809    0.420
## weight             0.56918    0.12349   4.609 7.93e-06 ***
## as.factor(spine)2 -0.23894    0.37030  -0.645    0.520
## as.factor(spine)3 -0.04465    0.19115  -0.234    0.816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 3.145652)
##
##     Null deviance: 632.79  on 172  degrees of freedom
## Residual deviance: 559.49  on 169  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

Answer: The dispersion parameter ϕ is estimated to be 3.15 (it's greater than one, indicating we may have some overdispersion in our data).

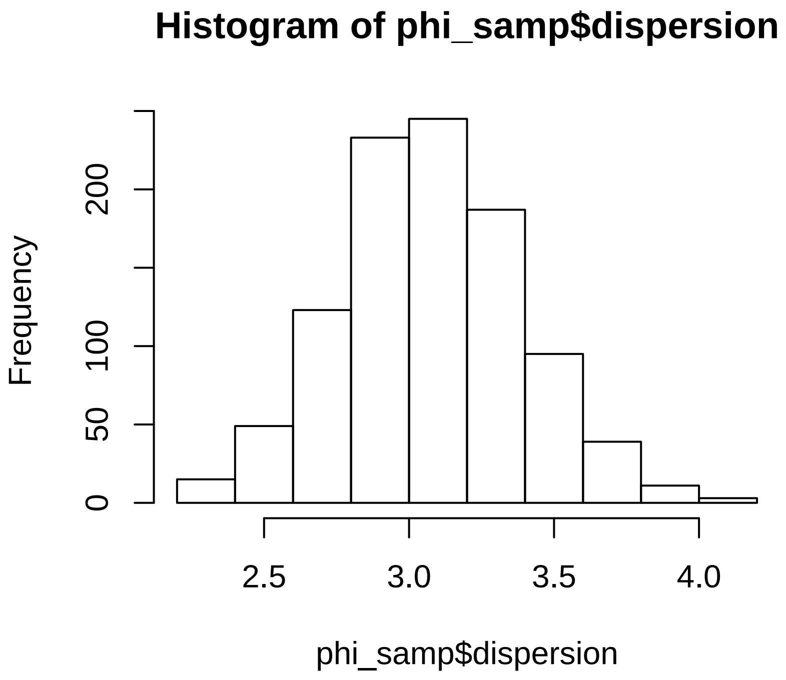
Problem 4

We might consider bootstrapping the distribution of the dispersion parameter to see if we should really include it. Calculate a bootstrap 95% confidence interval for the dispersions parameter using 1000 bootstrap replications. The following function will be helpful if you want to use `modelr` for your bootstrap.

```
getdispersion <- function(x){unlist(x)$dispersion}

phi_samp <- crabs %>%
  bootstrap(1000) %>%
  mutate(model = map(strap, ~glm(sat~weight+as.factor(spine),family=quasipoisson,data=.)),
         summary = map(model, summary),
         dispersion = map(summary, getdispersion)) %>%
  select(.id, dispersion) %>%
  unnest()

hist(phi_samp$dispersion)
```



```
quantile(phi_samp$dispersion, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 2.444939 3.716065
```

The bootstrap CI does seem to indicate some significant overdispersion ($\phi > 1$), so we believe we have overdispersed data, so we should use quasipoisson. However, we don't believe that spine condition is significant, so let's remove it and work with a model using only weight as a predictor.

Problem 5

Use the anova function in R with test = "Chisq" to compare the model without `spine` to the larger model with spine. As in logistic regression, the anova test statistic is the difference in deviance values, and the degrees of freedom for the χ^2 distribution is the difference in number of parameters. Comment on the results.

```
fit3 <- glm(sat~weight,family=quasipoisson,data=crabs)
summary(fit3)
```

```
##
## Call:
## glm(formula = sat ~ weight, family = quasipoisson, data = crabs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9307  -1.9981  -0.5627   0.9298   4.9992
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4284     0.3168  -1.352    0.178
## weight        0.5893     0.1151   5.120 8.17e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 3.13414)
##
##     Null deviance: 632.79  on 172  degrees of freedom
## Residual deviance: 560.87  on 171  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
anova(fit3, fit2, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: sat ~ weight
## Model 2: sat ~ weight + as.factor(spine)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       171     560.87
## 2       169     559.49  2   1.3788   0.8032
```

We fail to reject the more simple model and conclude that we do not need to include spine condition.

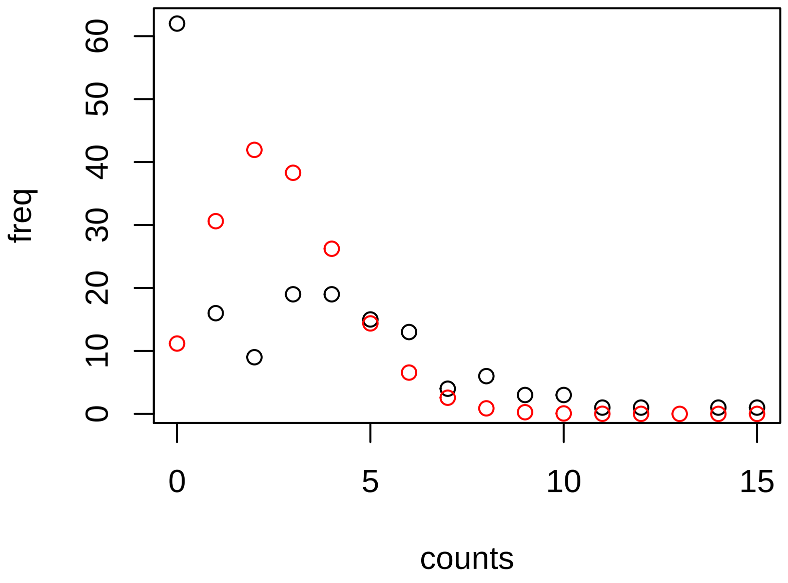
Problem 6

The following code fits shows the predicted counts (red) and observed counts (black) for the mean value of weight. Comment on the fit.

```r
# Calculate observed counts
ns.table <- table(crabs$sat)
counts <- as.data.frame(ns.table)[,1] %>%
  as.character(.) %>% as.numeric(.)
freq <- as.data.frame(ns.table)[,2]

# Calculate predicted counts for the mean value of weight
x.p <- mean(crabs$weight)
lambda.x.p <- exp(coef(fit3)[1]+coef(fit3)[2]*x.p)
vals <- 0:15
fits <- rep(0,length(vals))
for (i in 1:length(fits)){
  fits[i] <- dpois(i-1,lambda.x.p)*nrow(crabs)
}
```

```r
plot(freq~counts)
points(fits~vals,col="red")
```



The predictions are not very good, in particular our poisson model is struggling mightily with the observed number of zeros in the data. This is another common problem in modeling count data.

Problem 7

A 'zero-inflated' model splits the process into two pieces: we now assume that the response will be zero with probability π, and will follow a poisson distribution with probability $(1 - \pi)$. Notice that the zero could occur in two ways now, either with probability π or it could be generated by the poisson distribution. Furthermore, the π probability will be modeled using logistic regression, so our proposed distribution now is

$$p(y \mid x) = \pi I\{y = 0\} + (1 - \pi)\frac{e^{-\lambda}\lambda^y}{y!}I\{y > 0\},$$

where $\lambda = e^{x^T\beta}$ and

$$\pi = \frac{e^{x^T\alpha}}{1 + e^{x^T\alpha}}.$$

Here α represents the coefficients for the logistic regression piece, and β represents the coefficients for the poisson piece. In pracice the covariates used in each piece can be different, and I will point out how to do this in R. For example, the probability that $Y = 0$ is

$$P(y = 0) = \pi + (1 - \pi)e^{-\lambda}.$$

Zero-inflated models in R

These models can be fit in R using the *zeroinfl* function in the *pscl* package.

```
library(pscl)
```

```
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
fit_zero <- zeroinfl(sat~weight, data=crabs)
summary(fit_zero)
```

```
##
## Call:
## zeroinfl(formula = sat ~ weight, data = crabs)
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.7217 -0.8732 -0.3752  0.6097  4.3522
##
## Count model coefficients (poisson with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.00130    0.20793   4.816 1.47e-06 ***
## weight       0.19042    0.07572   2.515   0.0119 *
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.6778     0.9028   4.074 4.62e-05 ***
## weight       -1.8261     0.3879  -4.707 2.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 9
## Log-likelihood: -363.6 on 4 Df
```

Notice that the output now gives us output for the poisson piece and output for the logistic piece. By default R uses the same predictors for both pieces, but this can be easily modified. Notice that the direction of the weight coefficients agree for the two pieces. The positive coefficient for the poisson portion indicates that the expected number of satellites increases as weight increases; the negative coefficient for the logistic piece indicates that the probability of observing zero satellites decreases as weight increases. Plot the observed counts and predicted counts for this zero-inflated model using the mean weight value to fix λ. Comment on the results.

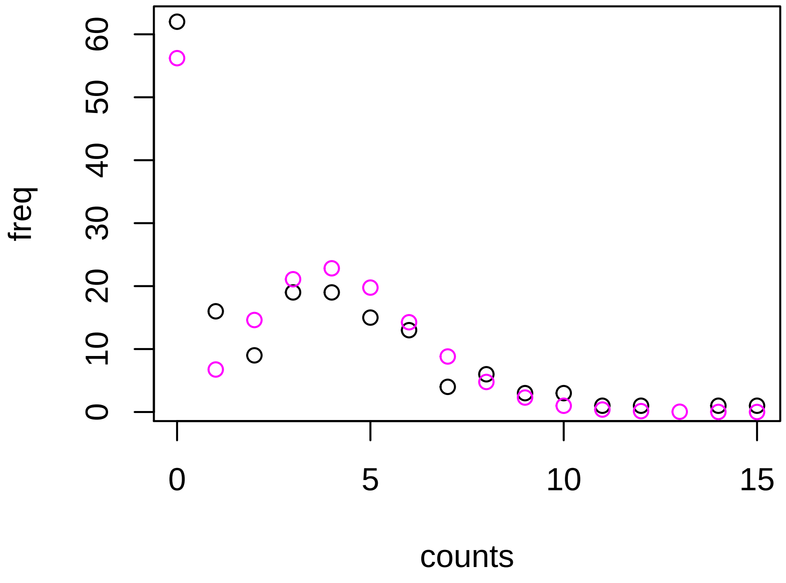
```r
x.p <- mean(crabs$weight) ##Change this max, min, or mean or whatever you want

odds <- exp(coef(fit_zero)[3] + coef(fit_zero)[4]*(x.p))
pi <- odds/(1+odds)

lambda.x.p.z <- exp(coef(fit_zero)[1]+coef(fit_zero)[2]*x.p)
vals <- 0:15
fits.z <- rep(0,length(vals))
fits.z[1] <- (pi + (1-pi)*dpois(0,lambda.x.p.z))*nrow(crabs)

for (i in 1:(length(fits.z)-1)){
  fits.z[i+1] <- (1-pi)*dpois(i,lambda.x.p.z)*nrow(crabs)
}

plot(freq~counts)
points(fits.z~vals,col=6)
```



Visually, this model seems to do a much better job of capturing the random process we are trying to model! In the R markdown file you can change the weight value we are using and see how the estimated poisson distribution changes with varying values of the predictor.