Wave Damage Model: Poisson Regression
by Michael Escobar
February 12, 2018

1 Introduction

This document looks an example of Bayesian Poisson regresssion. Note that the R code that was used for this model is in the file: `wavedamROpenBugsPart1.txt` which has been sent to you with this file.

Here is a description of the data:

```
Data taken from page 204 of McCullagh Nelder (2nd ed)

The data was provided by J. Drilley and L.N. Hemingway of Lloyd's
Register of Shipping, concern a type of damage caused by waves to the forward
section of certain cargo-carrying vessels.  For he purpose of setting standards for hull
construction we need to know the risk of damage associated with the three
classifying factors show below:

column 1:  Ship type, 1-5
Column 2: Year of construction: 1=1960-64, 2=1965-69, 3=1970-74, 4=1975-79
Column 3: Period of operation: 1=1960-74, 2=1975-1979
Column 4: Aggregate months of service
Column 5: Number of reported damage incidents
```

Here is the data:

ship	yrcons	yrop	month	daminc
1	1	1	127	0
1	1	2	63	0
1	2	1	1095	3
1	2	2	1095	4
1	3	1	1512	6
1	3	2	3353	18
1	4	2	2244	11
2	1	1	44882	39
2	1	2	17176	29
2	2	1	28609	58
2	2	2	20370	53
2	3	1	7064	12
2	3	2	13099	44
2	4	2	7117	18
3	1	1	1179	1
3	1	2	552	1
3	2	1	781	0
3	2	2	676	1
3	3	1	783	6
3	3	2	1948	2
3	4	2	274	1
4	1	1	251	0

```
4   1  2    105    0
4   2  1    288    0
4   2  2    192    0
4   3  1    349    2
4   3  2   1208   11
4   4  2   2051    4
5   1  1     45    0
5   2  1    789    7
5   2  2    437    7
5   3  1   1157    5
5   3  2   2161   12
5   4  2    542    1
```

2 Modelling the Data

When modelling this data, the outcome variable is the number of damage incidents reported. Since the observed value is a "count", then one might consider modelling this observed value as a Poisson random variable. As discussed previously, a GLM type of model can then be used for this data. Therefore, consider Y_i to be the number of damage incidents under conditions i.

The i-th observation specifies a set of conditions corresponding to certain ship type, year of construction, and period of operation. So, for $i = 1$, this coresonds to ship type 1, year on constrution 1 which corresponds to construction between 1960 to 1964, and period of operation 1 which corresponds to operations between 1960 and 1974. For observation $i = 1$ there were 0 damage incidents reported in 127 total ship-months. The notion of "ship-months" means that the combined number of months that ships were exposed to being damaged.

For this model, one might expect a higher number of events if there is more exposure to being damaged. For example when considering the ships constructed between 1960-1964 and for the first period of operation, for ship type 1, there are 127 ship-months of exposure and for ship type 2 there are 44882 ship-months. Therefore, there where 350 times more months of ship time logged for ship type 2 versus ship type 1. Therefore, might not be surprised that since there were only 39 damage events for ship type 2 that seeing none for ship type 1 might just be due the less exposure time.

For the Poisson distribution, the common link function, $g(\cdot)$ is usually the $\log(\cdot)$ function althought one sometimes uses the indenty function. For this model, the log link function is used here.

When thinking about the covariates, the X variables, one might thing that the X variables have different rates of exposures. This leads to the following possible model:

$$\begin{aligned} E(\text{number of events}) &= (\text{rate of events—X}) \times (\text{months exposed}) \\ \log(E(\text{number of events})) &= \log(\text{rate of events—X}) + \log(\text{months exposed}) \\ &= \left(\sum_{j=0}^{p} \beta_j X_{ij} \right) + \log(\text{months exposed}) \end{aligned}$$

Note that in the above, this looks somewhat like the usual GLM model, but there is an additional term which is the log of the months exposed. This is called the offset term.

Also, note that the β parameters represent the log of the increase risk. So, usually, when one wants to discuss the the increase risk, then one back transforms the β terms and the increase risk (or the relative risk) is the $\exp \beta$ type of term.

Therefore, for this model, the following WinBug modelling file is used in this example:

```
model{
for(i in 1:34)
{
#ship   yrcons   yrop  month daminc

daminc[i]~dpois(lam[i])
log(lam[i]) <- log(month[i]) + beta0 + beta.o*yrop[i] + beta.s[ship[i]]+ beta.c[yrcons[i]]  + b[i]
b[i] ~dnorm(0,tau)
b.adj[i] <- b[i] - mean(b[])
}
for(is in 1:5){
beta.s[is]~dnorm(0,tau.s)
beta.s.adj[is] <- beta.s[is] -mean(beta.s[])
}
for(ic in 1:4){
beta.c[ic]~dnorm(0,tau.c)
beta.c.adj[ic] <- beta.c[ic] - mean(beta.c[])
}

# Using tigher priors
#  Note: total ship-years per categories is less than 50,000
#  ln(50000) ~ 2.3*4 + 1.6 = 10.8....
#  so rate has to be bigger than 1/50,000 and log(rate) > -10.8...
#  so log( base rate) should be between about -11 and 11.
# 1/11/11 is about .0082
beta0 ~ dnorm(0, .0082)
beta0.adj <- beta0 + mean(b[]) + mean(beta.s[])+ mean(beta.c[])


# for the <extra poisson variation> ...
# assume bounded by very big number... say 1000 times...
# so log(1000) is about 2.3*4 which is about 9.2
std ~ dunif(0, 9)
tau <- 1/std/std

# for the relative risk between groups... a very large number would be 100 times,
#  so, log(100) is about 2.3*2 or about 4.6
#  also, note that 1/5/5 is 0.04
#
std.s ~dunif(0, 5)
tau.s <- 1/std.s/std.s
std.c ~ dunif(0,5)
tau.c <- 1/std.c/std.c

beta.o~dnorm(0, .04)
```

Some notes about the above WinBug model:

- The offset is included in the model as: `log(month[i])`
- For this model, "redundant" parametrization is used and there is an overdispersion parameter.

- For the precision parameters, there is a uniform distribution put on the standard deviation parameters.
- For the range of the parameters, note the following assumptions:
  - Since for each of the categories, there are less than 50,000 "person-years" in each of categories with an event in each category, then the rate per person year is bigger than 1/50,000. Therefore, the base rate is more than 1 per 50,000 person years. This implies that the log(base rate) is bigger than -log(50000). Note, log(50000) is equal to about 4*2.3+1.6 which is about 10.8. So, the prior for `beta0` is a normal with mean 0 and standard deviation of 11. (Note: for the log function, I am assuming the natural log.)
  - For the overdispersion parameter, we still let the multiplier be on the order of 1000 times. This means that the the overdispersion parameter, `b`, fluctuates at an order of log(100) which is about 3*2.3 which means that the standard deviation of `b` is less than 9. So, the standard deviation is set to a prior which is uniform between 0 and 9.
  - For the variation of the categorical parameters, I am assuming that the relative risk is smaller than 100. (Note that a relative risk of 100 would be very noticeable.) So, this means that letting the standard deviation of the log relative risk to be less than about 5. (That is, log(100) is about 2.3*2 which is about 4.6, so I simplified that to be 5.)
  - For more information about how to approxiamate the logs, please see the separate handout.

This model was run throught R using R2WinBugs with the following commands:

```
params=c("beta.s.adj",  "std.s", "beta.c.adj", "std.c", "beta.o", "beta0.adj", "std")

bug.dat=list("ship","yrcons","yrop", "month","daminc")
init.fun=function(){list(
  beta.s=rnorm(5), std.s=runif(1,1,2),
  beta.c=rnorm(4), std.c=runif(1,1,2),
  beta.o=rnorm(1), std=runif(1,1,2), beta0=rnorm(1),
  b=rnorm(34,0,.1))}

WaveBug0=bugs(bug.dat, init.fun, params, model.file="waveModCentered.txt",
    n.chains=5, n.iter=30000, n.burnin=10000, n.thin=5  #for production
# n.chains=5, n.iter=2000, n.burnin=100, n.thin=1, debug=TRUE   #for testing
    )

print(WaveBug0, digits.summary = 3)
```

Note, the following about the above command:

- There are 5 chains run in the above.
- For the line above with the `debug=TRUE`, this line is ran first to check to see if the chain was ran enough. In the above, that line is "commented" out.
- The "burnin" was 10,000 iterations and a total of 30,000 iterations are run. I decided to run this 30,000 times after looking at the trace plots.
- In the production run, I actually ran the algorithm with `n.thin=5`. These means that only every 5th sampled value is kept. Also, note that when using `R2OpenBugs`, the it still keeps 20,000 iterations per chain. (That is, 30,000 - 10,000). Therefore, it actually runs (`n.thin` time `n.iter`) for each

chain. That is, in the above, it sampled 750,000 total samples. By thinning (and with the burnin), the computer only saved 100,000 samples. If one kept all the samples, they computer start running sluggishly.

3 Some results

Here are some of the results from the R2WinBug package:

```
>
> #######   look at some results    ######
>
> print(WaveBug0, digits.summary = 3)
Inference for Bugs model at "waveModCentered.txt",
Current: 5 chains, each with 30000 iterations (first 10000 discarded), n.thin = 5
Cumulative: n.sims = 1e+05 iterations saved
                 mean    sd    2.5%     25%     50%     75%   97.5%  Rhat n.eff
beta.s.adj[1]   0.137 0.186  -0.218   0.011   0.130   0.257   0.518 1.001 52000
beta.s.adj[2]  -0.274 0.178  -0.604  -0.395  -0.284  -0.159   0.072 1.001  5800
beta.s.adj[3]  -0.306 0.256  -0.851  -0.471  -0.289  -0.115   0.120 1.001 16000
beta.s.adj[4]   0.025 0.221  -0.437  -0.105   0.024   0.165   0.455 1.001  8100
beta.s.adj[5]   0.418 0.243  -0.022   0.251   0.422   0.581   0.897 1.001  7600
std.s           0.568 0.448   0.049   0.302   0.460   0.700   1.746 1.008  1900
beta.c.adj[1]  -0.437 0.234  -0.928  -0.578  -0.435  -0.288   0.004 1.002  5600
beta.c.adj[2]   0.161 0.163  -0.162   0.056   0.162   0.264   0.490 1.001  7100
beta.c.adj[3]   0.320 0.173  -0.004   0.209   0.316   0.424   0.686 1.001  8300
beta.c.adj[4]  -0.045 0.196  -0.441  -0.164  -0.040   0.073   0.346 1.001 27000
std.c           0.696 0.614   0.087   0.330   0.515   0.836   2.485 1.006  2800
beta.o          0.360 0.224  -0.104   0.225   0.364   0.500   0.796 1.001 11000
beta0.adj      -6.491 0.384  -7.268  -6.729  -6.488  -6.252  -5.733 1.001 20000
std             0.334 0.185   0.039   0.194   0.318   0.452   0.735 1.009   510
deviance      135.740 8.106 120.500 129.900 135.800 141.500 151.400 1.003  1600

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = Dbar-Dhat)
pD = 16.3 and DIC = 152.1
DIC is an estimate of expected predictive error (lower deviance is better).
```

Also, there are important plots which demonstrate the results of this analysis. The below code produces the boxplots comparing the β's for the ship type category. Remember the β's represent the log relative risk. The increase risk wehn going from ship type 2 to ship type 1 would beexp $\beta_{s1} - \beta_{s2}$.

```
SArray= WaveBugR$sims.array
pdf("BoxPlotBetaS.pdf",width=5, height=4, onefile=F)
boxplot(data.frame( (WaveBug0$sims.list)["beta.s.adj"]),
  xlab="beta.s.adj")
dev.off()
```

The following code generates the density plot estimates by the different sampled chains.

```
pdf("DensityPlotBetaS.pdf",width=5, height=4, onefile=F)
plot(c(-2,1),c(0,3), type="n",main="beta.s.adj[2]")
apply(SArray[,,"beta.s.adj[2]"], 2, function(x)lines(density(x)))
dev.off()
```

The following will produce a plot of the autocorrelation of a parameter. Remember, for this model, the chain is thinned by 10, so the reported lag of 1 in this plot is actually a lag of 10.

```
pdf("ACFPlotBetaS.pdf",width=5, height=4, onefile=F)
acf( SArray[,1,"beta.s.adj[2]"], main="beta.s.adj[2]")
dev.off()
```

The following will produce a trace of a parameter. Remember, for this model, the chain is thinned by 5, so the reported lag of 1 in this plot is actually a lag of 5.

```
pdf("TracePlotBetaS.pdf",width=5, height=4, onefile=F)
matplot(1:chainL,SArray[,,"beta.s.adj[2]"], main="beta.s.adj[2]",xlab="index",type="l")
dev.off()
```

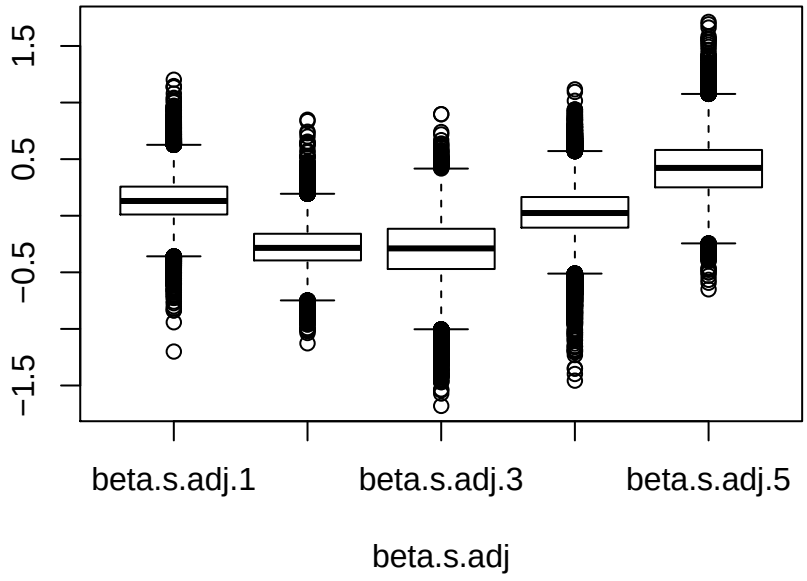

Figure 1: The box plot demonstrates the difference between the ship types.

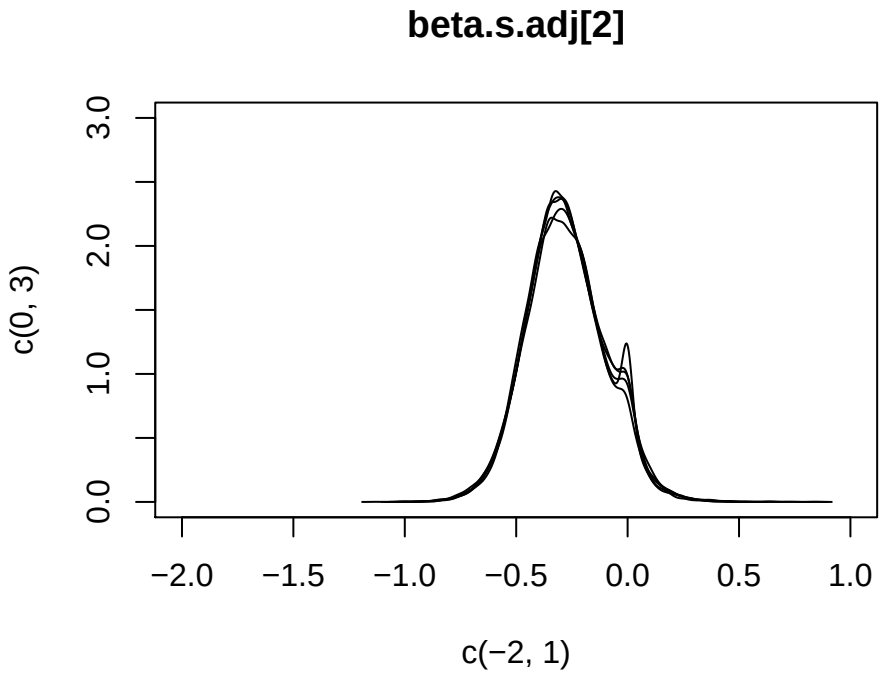

Figure 2: The cross correlation plot. This plot shows which parameters have a high cross correlation. This might indicate a possible problem with slow convergence.

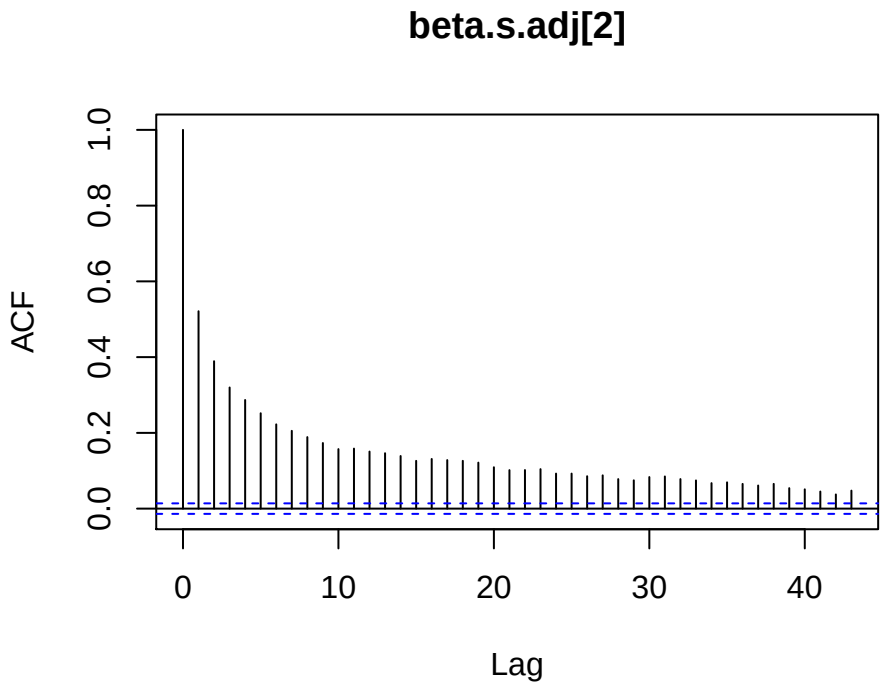

Figure 3: The autocorrelation plot. This plot shows which parameter has a high correlation with itself.

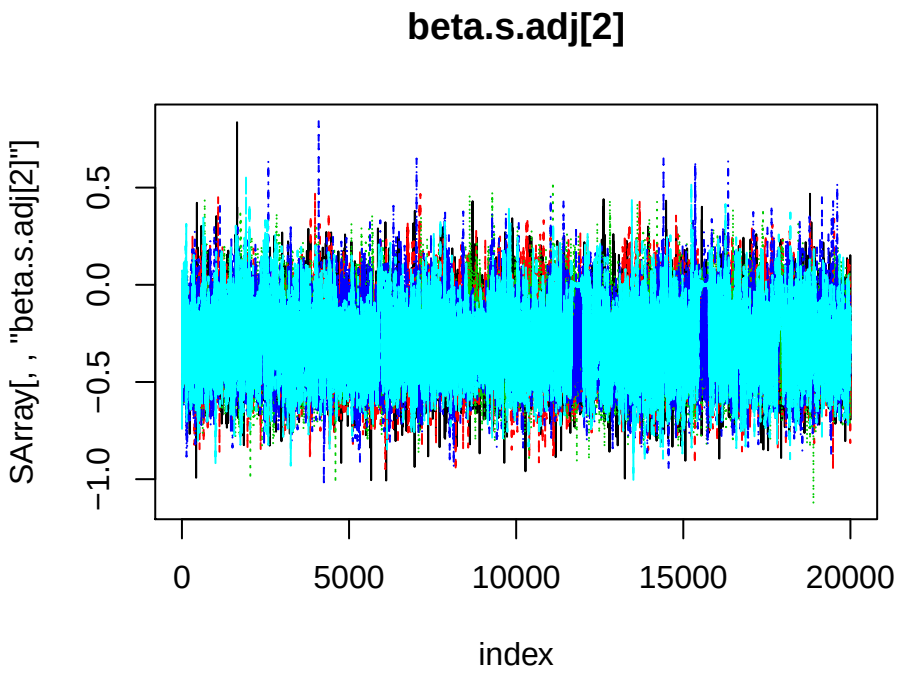

Figure 4: The trace plot.