Wave Damage Model: Poisson Regression

by Michael Escobar

February 11, 2013

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: `wavedamRRedOnly.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)
{
daminc[i]~dpois(lam[i])
log(lam[i]) <- log(month[i]) +b0 + beta.s[ship[i]]  + beta.c[yrcons[i]] + beta.o*(yrop[i]*2-3) + b[i]
b[i] ~dnorm(0,tau)
b.adj[i]<-b[i]-mean(b[])
}

b0~dnorm(0,.001)
b0.adj<-b0+mean(b[])+mean(beta.s[])+mean(beta.c[])

for(is in 1:5){
beta.s[is]~dnorm(mu.s,tau.s)
beta.s.adj[is]<-beta.s[is]-mean(beta.s[])
}
mu.s~dnorm(0,.001)

for(ic in 1:4){
beta.c[ic]~dnorm(mu.c,tau.c)
beta.c.adj[ic]<-beta.c[ic]-mean(beta.c[])
}
mu.c~dnorm(0,.001)

beta.o~dnorm(0, .001)

tau<-1/std/std
tau.s<-1/std.s/std.s
tau.c<-1/std.c/std.c
std~dunif(0,20)
std.s~dunif(0,20)
std.c~dunif(0,20)
```

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.

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

```
bug.dat=list("ship","yrcons","yrop", "month","daminc")
init.fun=function(){list(
  beta.s=rnorm(5), mu.s=rnorm(1), std.s=runif(1,.5,2),
  beta.c=rnorm(4), std.c=runif(1,.5,2),mu.c=rnorm(1),
  beta.o=rnorm(1), std=runif(1,.5,2),
  b=rnorm(34,0,1), b0=rnorm(1) )}
```

```
params=c("beta.s.adj",  "std.s", "beta.c.adj", "std.c", "beta.o", "b0.adj")

WaveBugR=bugs(bug.dat, init.fun, params, model.file="waveModR.txt",
    n.chains=3, n.iter=200000, n.burnin=10000,
    n.thin=10, bugs.directory=WinDir #, debug=TRUE
        )
```

Note, the following about the above command:

- There are 3 chains run in the above.
- The "burnin" was 10,000 iterations and a total of 200,000 iterations are run. I decided to run this 200,000 times because of the convergence diagnostics that I ran. For a smaller number of runs, I noticed that for several of the parameters for different chains, the Geweke statistics were some times bigger (and sometimes much bigger) than 2. That happened when I only ran about 20,000 or 30,000 iterations. However, when I ran 200,000 iterations, as seen below, the convergence diagnostics looked fairly good.
- Since I had run such a large number of iterations, I decided to thin by 10 because my computer seemed a bit sluggish with so many samples.

3 Some results

Here are some of the results from the R2WinBug package:

```
>
> #######   look at some results and diagnostics  ######
>
> print(WaveBugR)
Inference for Bugs model at "waveModR.txt", fit using WinBUGS,
 3 chains, each with 2e+05 iterations (first 10000 discarded), n.thin = 10
 n.sims = 57000 iterations saved
               mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
beta.s.adj[1]   0.1 0.2  -0.2   0.0   0.1   0.3   0.5    1 48000
beta.s.adj[2]  -0.3 0.2  -0.6  -0.4  -0.3  -0.2   0.1    1 23000
beta.s.adj[3]  -0.3 0.3  -0.9  -0.5  -0.3  -0.1   0.1    1 57000
beta.s.adj[4]   0.0 0.2  -0.4  -0.1   0.0   0.2   0.5    1 37000
beta.s.adj[5]   0.4 0.2   0.0   0.3   0.4   0.6   0.9    1 53000
std.s           0.6 0.5   0.1   0.3   0.5   0.7   1.7    1 13000
beta.c.adj[1]  -0.4 0.2  -0.9  -0.6  -0.4  -0.3   0.0    1 45000
beta.c.adj[2]   0.2 0.2  -0.2   0.1   0.2   0.3   0.5    1 22000
beta.c.adj[3]   0.3 0.2   0.0   0.2   0.3   0.4   0.7    1 57000
beta.c.adj[4]   0.0 0.2  -0.4  -0.2   0.0   0.1   0.3    1 57000
std.c           0.8 1.0   0.1   0.3   0.5   0.9   3.0    1 33000
beta.o          0.2 0.1   0.0   0.1   0.2   0.3   0.4    1 57000
b0.adj         -6.0 0.1  -6.2  -6.0  -5.9  -5.9  -5.7    1 41000
deviance      135.9 8.2 120.5 129.9 136.0 141.8 151.6    1 57000

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 = 152.2
DIC is an estimate of expected predictive error (lower deviance is better).
>
>
```

Also, one can obtained some summaries of the MCMC chain via the coda package. Below are the commands and the summaries from the coda package.

```
> library(coda)
>
> make.mcmc.list=function(x){
+     aa=x$sims.array
+     zz=list(list())
+     for(i in 1:(dim(aa)[2]) ){
+         tmp=mcmc(aa[,i,])
+         zz=c(zz,list(tmp))   }
+         res=mcmc.list(zz[-1])
+         res}
>
>
> waveR=make.mcmc.list(WaveBugR)
> summary(waveR)

Iterations = 1:19000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 19000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                   Mean     SD  Naive SE Time-series SE
beta.s.adj[1]   0.13920 0.1853 0.0007763      0.0009664
beta.s.adj[2]  -0.27610 0.1760 0.0007372      0.0010934
beta.s.adj[3]  -0.30911 0.2537 0.0010627      0.0013274
beta.s.adj[4]   0.02463 0.2225 0.0009319      0.0009125
beta.s.adj[5]   0.42137 0.2405 0.0010073      0.0013228
std.s           0.57319 0.4873 0.0020409      0.0024872
beta.c.adj[1]  -0.44009 0.2343 0.0009813      0.0014733
beta.c.adj[2]   0.16292 0.1631 0.0006830      0.0009031
beta.c.adj[3]   0.32278 0.1720 0.0007205      0.0010165
beta.c.adj[4]  -0.04562 0.1955 0.0008189      0.0008174
std.c           0.77917 0.9759 0.0040878      0.0045154
beta.o          0.18329 0.1117 0.0004679      0.0005153
b0.adj         -5.95421 0.1215 0.0005089      0.0006109
deviance      135.92296 8.2042 0.0343636      0.0822894

2. Quantiles for each variable:

                  2.5%       25%       50%       75%      97.5%
```

```
beta.s.adj[1]  -0.214800   0.01440   0.13290   0.25800   0.518200
beta.s.adj[2]  -0.605102  -0.39450  -0.28550  -0.16370   0.072641
beta.s.adj[3]  -0.850603  -0.47190  -0.29225  -0.12340   0.123703
beta.s.adj[4]  -0.442405  -0.10753   0.02473   0.16680   0.455203
beta.s.adj[5]  -0.019932   0.25780   0.42455   0.58090   0.899500
std.s           0.067130   0.30638   0.46160   0.69633   1.727000
beta.c.adj[1]  -0.934802  -0.57970  -0.43855  -0.29250   0.003428
beta.c.adj[2]  -0.160600   0.05860   0.16430   0.26470   0.491700
beta.c.adj[3]  -0.001793   0.21310   0.31800   0.42520   0.689703
beta.c.adj[4]  -0.439400  -0.16370  -0.04230   0.07174   0.348400
std.c           0.093340   0.33640   0.53050   0.87010   3.015000
beta.o         -0.049101   0.11690   0.18590   0.25230   0.403700
b0.adj         -6.209000  -6.03100  -5.94900  -5.87100  -5.729000
deviance      120.500000 129.90000 136.00000 141.80000 151.600000
```

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$sims.array
pdf("BoxPlotBetaS.pdf",width=5, height=4, onefile=F)
boxplot(data.frame( (WaveBugR$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()
```

4 Some Convergence Diagnostics

Below are some examples of convergence diagnostics.

First, the following code produced a plot of the cross correlation plots.

```
pdf("CrossCorr.pdf", width=5, height=4, onefile=F)
crosscorr.plot(waveR)
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 next series of results use the different coda commands to produce convergence diagnostics for this model:

```
>
> gelman.diag(waveR)
Potential scale reduction factors:

              Point est. 97.5% quantile
beta.s.adj[1]       1.00           1.00
beta.s.adj[2]       1.00           1.00
beta.s.adj[3]       1.00           1.00
beta.s.adj[4]       1.00           1.00
beta.s.adj[5]       1.00           1.00
std.s               1.00           1.00
beta.c.adj[1]       1.00           1.00
beta.c.adj[2]       1.00           1.00
beta.c.adj[3]       1.00           1.00
beta.c.adj[4]       1.00           1.00
std.c               1.00           1.00
beta.o              1.00           1.00
b0.adj              1.00           1.00
deviance            1.00           1.00

Multivariate psrf

1
> geweke.diag(waveR)
[[1]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
      -0.5881       -0.1583       -0.9700        1.4189        0.4911
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
      -1.2808       -0.5013        0.4784        0.6762       -0.5096
        std.c        beta.o        b0.adj      deviance
       0.8384        0.6130       -1.0415        1.2969


[[2]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
     -0.59059       1.36541       0.04993       0.37943      -1.26467
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
      0.40546      -0.67220       0.66490       0.26014       0.09359
        std.c        beta.o        b0.adj      deviance
     -0.70459       0.58033      -1.51851      -1.10746


[[3]]
```

```
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
     -2.19698       1.09690       1.37109      -0.73072      -0.18359
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
     -0.68137      -0.06777      -0.55375       0.06629       0.55300
        std.c        beta.o        b0.adj      deviance
     -0.70988      -0.14515      -0.33199      -0.84625


> geweke.diag.mod(waveR)
[[1]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
      -0.6425       -0.1833       -1.0504        1.4075        0.5058
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
      -1.2839       -0.4887        0.5381        0.6313       -0.4525
        std.c        beta.o        b0.adj      deviance
       0.8800        0.5969       -1.1078        1.4143


[[2]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
     -0.63814       1.54318       0.04561       0.38474      -1.28994
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
      0.41220      -0.62796       0.72213       0.25752       0.09437
        std.c        beta.o        b0.adj      deviance
     -0.74140       0.61100      -1.70334      -1.16190


[[3]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
     -2.34220       1.16779       1.33756      -0.80090      -0.19758
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
     -0.67148      -0.07781      -0.58801       0.06841       0.55466
        std.c        beta.o        b0.adj      deviance
     -0.71638      -0.15027      -0.35022      -0.91925
```

```
>
> batchSE(waveR)
beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
 0.0009258370  0.0011476102  0.0013147284  0.0009931315  0.0013509009
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
 0.0024675929  0.0014552821  0.0008324100  0.0009629065  0.0008549122
        std.c        beta.o        b0.adj      deviance
 0.0046392282  0.0005797272  0.0006321972  0.0746093826
> effectiveSize(waveR)
beta.s.adj[1] beta.s.adj[2] beta.s.adj[3] beta.s.adj[4] beta.s.adj[5]
     44963.65      22429.04      39092.74      48569.58      30663.92
        std.s beta.c.adj[1] beta.c.adj[2] beta.c.adj[3] beta.c.adj[4]
     41624.02      26334.29      37137.66      30944.94      51595.03
        std.c        beta.o        b0.adj      deviance
     44105.47      40942.63      34910.15      12169.76
>
```

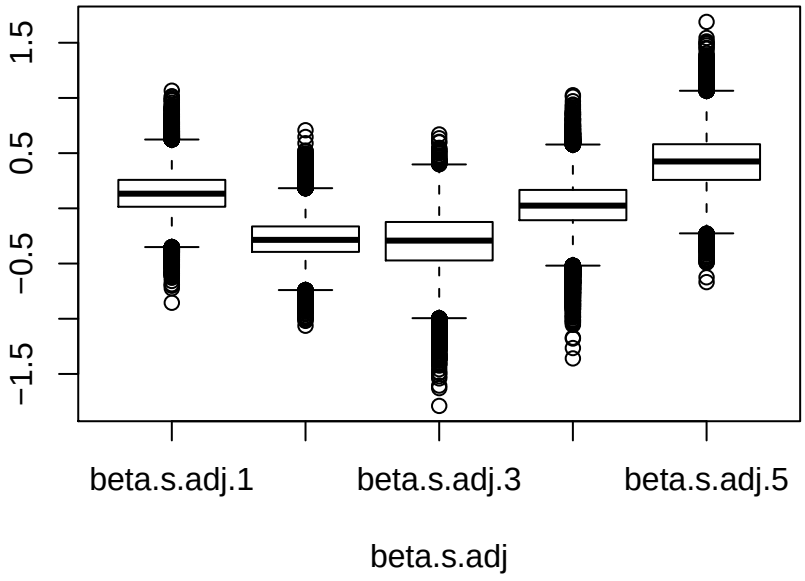

Figure 1: The box plot demonstrates the difference between the ship types.

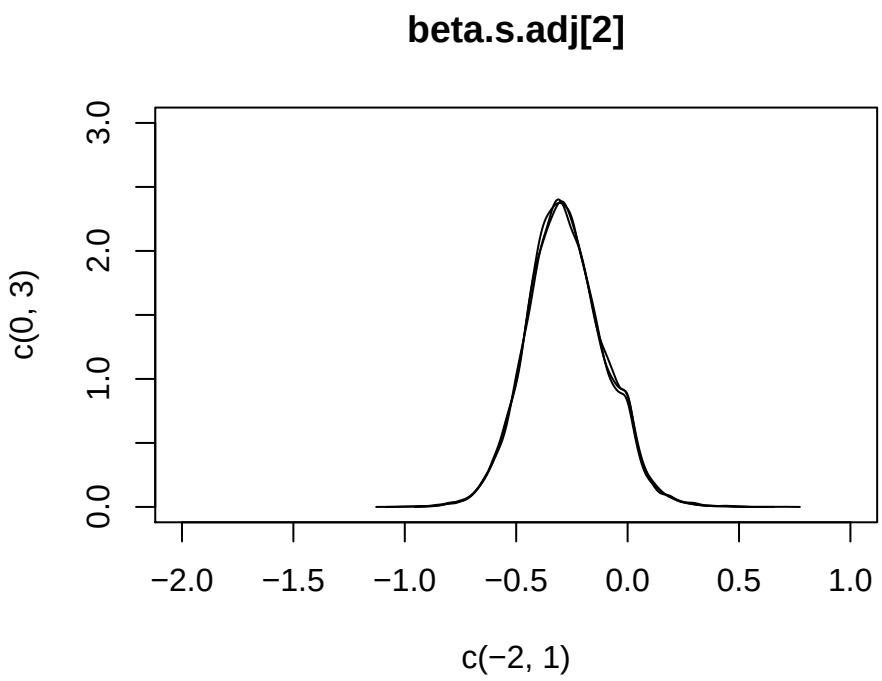

Figure 2: The cross correlation plot. This plot shows which parameters have a high cross correlation. This might indicate a possible problem with slow convergence.

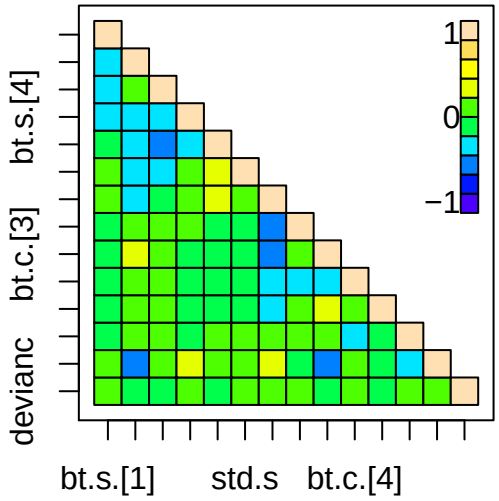

Figure 3: The cross correlation plot. This plot shows which parameters have a high cross correlation. This might indicate a possible problem with slow convergence.

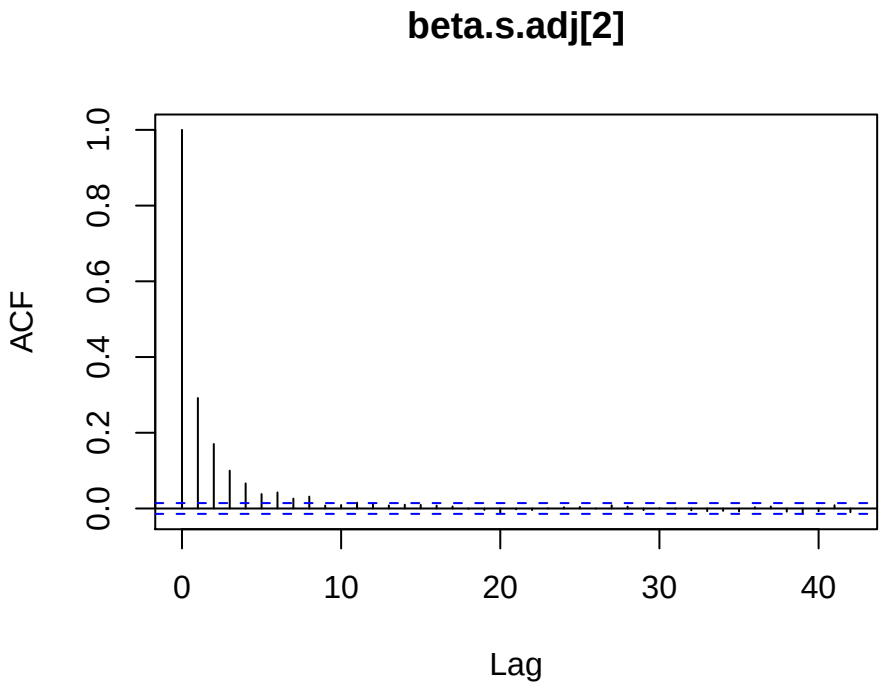

Figure 4: The autocorrelation plot. This plot shows which parameter has a high correlation with itself.