Stat 5421 (Geyer) Spring 2016
Homework Assignment 3
Due Wednesday, October 28, 2020, 11:59 PM

3.1. This problem uses the data in Table 4.3 in Agresti, which can be read into R as follows

```
> library(CatDataAnalysis)
> data(table_4.3)
> names(table_4.3)

[1] "color"  "spine"  "width"  "satell" "weight" "y"

> sapply(table_4.3, class)

    color     spine     width    satell    weight         y
"integer" "integer" "numeric" "integer" "integer" "integer"
```

This assumes you have R package ‘CatDataAnalysis‘ installed on your computer. If not, instructions for installing the package are found at

`https://github.com/cjgeyer/CatDataAnalysis`

Note that the table caption for Table 4.3 in Agresti makes it clear that the variables `color` and `spine` are categorical, not (as R has them at this point) `"integer"`. So we have to deal with that.

The response is `satell`, a count variable. Do a Poisson regression of `satell` on the other variables not including `y`, which apparently isn't part of the original data. Unlike the analysis recommended by Agresti in Section 4.3.2, use the default log link for the Poisson model (thus making the model being used an exponential family).

(a) Test whether any of the terms in the model can be dropped. use any one of Wald, Wilks, Rao for all the tests, your choice. (Do not do more than one of these. Pick one and use it.) Report the P-value for each test. What model do these tests suggest we use?

(b) Do a test of the model these tests suggest (little model) versus the original model with all predictors. What does this test say?

3.2. Do a Bayesian analysis of the model for these data specified by the formula

```
satell ~ 0 + color + weight
```

(when color has been converted to a factor). This gets everybody on the same page to start.

Since the MLE regression coefficients (canonical linear submodel canonical parameters) are of very different sizes, it may help to use a vector (or even a matrix) scale parameter. When doing MCMC with the R function `metrop` in the CRAN package `mcmc`, which is what was used for the Bayes examples, the following are things to do (TTD). Scale proportional to standard deviation componentwise, that is,

```
foo <- sqrt(diag(var(mout$batch)))
```

where `mout` is the result of a call to `metrop` that had `blen = 1` (no batching), and use `scale` proportional to `foo` for future runs. Scale proportional to matrix square root of (approximate) posterior variance matrix, that is,

```
foo <- var(mout$batch)
bar <- eigen(foo, symmetric = TRUE)
baz <- bar$vectors %*% diag(sqrt(bar$values)) %*% t(bar$vectors)
# check that this is indeed matrix square root
all.equal(foo, baz %*% baz)
```

where `mout` is the result of a call to `metrop` that had `blen = 1` (no batching), and use `scale` proportional to `baz` for future runs.

Like the Bayes examples, use the conjugate prior distribution that adds 1/2 of a count to each component of the response vector.

(a) Produce estimates of the posterior means of each of the parameters with Monte Carlo standard errors. Use long enough batches so there is no statistically significant autocorrelation of batch means. Base your estimate on a run that takes at least one minute of computing time. If `mout` is the result of a call to `metrop`, then `mout$time` is the amount of time it took.

(b) Figure out how much longer the run would have to be for the Monte Carlo standard errors to be less than 0.01 of the quantities being estimated. And the same except for 0.001.

(c) Make a plot of the posterior PDF for the parameter named `weight`.