New Statistical Methods That Improve on MLE and GLM – Including for Reserve Modeling

GARY G VENTER

MLE Going the Way of the Buggy Whip

- **Used to be gold standard of statistical estimation**
  - Minimum variance unbiased estimate – estimation and predictive variance
- **But even by 1956, Stein's Paradox was that if you are estimating 3 or more means, shrinking them all towards overall mean reduces variance**
  - Paradox part was the means don't have to be related conceptually
  - This shrinking ends up with lower variances, but some biased up, some down
  - Having lower errors generally better than having bigger unbiased errors
  - Method is same as credibility – shrink using within and between variances
  - Famous example is estimating year-end batting averages by player from early season averages
  - In regression or GLM, every fitted value is an estimated mean, so lots of them

Now Regularization

- Not very informative name for another way of shrinking
- Minimizes negative log likelihood (NLL) plus parameter penalty
  - Examples of penalties, with $\lambda>0$, parameters b_i
  - Ridge regression: $\Sigma\lambda b_i^2$: in 1970 this proved to have lower error than MLE for some λ, but method of determining λ not clear
  - Lasso: $\Sigma\lambda|b_i|$: some parameters go to zero, so variable selection as well
  - Cauchy mode: $\Sigma\log(\lambda + b_i^2)$. Cauchy is t-distribution with one dof.
- Shrinking parameters tends to shrink estimates towards overall mean
  - Constant term is not included in parameters that are shrunk
  - Typically variables are scaled to have mean zero, variance one by a linear transformation, so really shrinking towards mean.
    - Coefficients and constant term adjust for the linear scaling
    - Takes away size of the variable from influencing the shrinkage

How Much to Shrink: Choosing λ

- Instead of within and between variances, keep a holdout sample
- Measure NLL of the holdout sample for various λ's
- Some shrinkage always better than MLE
- Typically divide the data into several subsets and leave each out in turn, rotating through all of them
  - Choose λ that best predicts holdout samples
  - In case of a tie, pick one with the most shrinkage
- Best case considered to be leave one out – loo – where every point is used as a holdout sample of 1.
- NLL sum of those holdouts a good estimate of NLL of a new sample
  - Fitting the population vs. fitting the sample

Bayesian Version

- Give each parameter a prior distribution symmetric around zero.
- Posteriors are shrunk towards zero. E.g., normal and double exponential priors give ridge regression and lasso as posterior modes.
- MCMC estimation simulates a sample of the posteriors – doesn't need to have the form of the posterior – just needs prior and density
- Advantages:
  - Loo likelihood for a point well estimated as Pareto-smoothed harmonic mean of the point's likelihood across the sample parameters – more weight for worse fits. Fast.
  - Parameter uncertainty already there from posterior sample
  - Can put a prior on λ too – usually fairly small uniform prior works, and gives a good posterior sample for λ. So don't need a lot of runs.
  - Good software packages available
  - Not restricted to distribution choices from GLM – better for runoff ranges
  - Makes posterior mean available – frequentist versions like lasso only have mode

Posterior Mean vs. Posterior Mode

- Mean uses all parameter sets that could have generated the model, weighted by probability of being the right set
- Mode looks at one sample only – the one with the highest probability
  - That probability is still quite low
- If the mean is very different from the mode, there is a risk that the mode is over-responsive to the particular sample.
- Issue of trying to fit to the population instead of fitting to the sample
- Mode can be computed as a maximization of prior * likelihood, so can be done as a frequentist calculation, but where prior is reinterpreted as the distribution of the effect being measured, not of the parameter.
- Frequentist regularization like lasso and ridge regression compute mode

Easy Application Example – Regression

- Just shrinks regression coefficients
- Straightforward for estimating pricing factors when there are many variables
- Reserving can be set up as a regression too
  - Put triangle into a column vector
  - Use dummy variables for row and column factors
  - Usually make an additive row – column model for log of cell means
  - Dummy variables are 1 for cells in that row or column, 0 elsewhere

But Shrinkage Complicates This

- Shrinking row and column factors isn't the same thing as shrinking a typical regression coefficient
- Something more like smoothing would be better
- One way to do this is to put the factors on piecewise linear curves
  - Then shrink the slope changes between segments – gives a kind of smoothing
  - Also can be done with cubic splines across the parameters
- Formally, make parameters for these slope changes, which are 2nd differences of the row and column factors or log factors
- Factors are cumulative sums of 2nd differences, so still can use dummies
- The dummy for the 2nd difference parameter for row j is:
- $d_{j,k} = \max(0, 1 + k - j)$ for a cell from row k. Same for columns. First row is row 1.

Loss Reserve Modeling – General

- **Over-parameterization reduces predictive accuracy**
  - **Look up "overfitting" in Wikipedia. One quote from a widely-cited source:**
  - **The essence of overfitting is to have unknowingly extracted some of the residual variation (i.e. the noise) as if that variation represented underlying model structure.**
- **Cumulative triangles violate the assumption of independent observations.**
- **Incremental triangles do not – they are not negatively correlated (empirically)**
- **If you are modeling cumulative triangles, a factor is significant if it is two or more standard deviations away from 1.0, not from 0.**

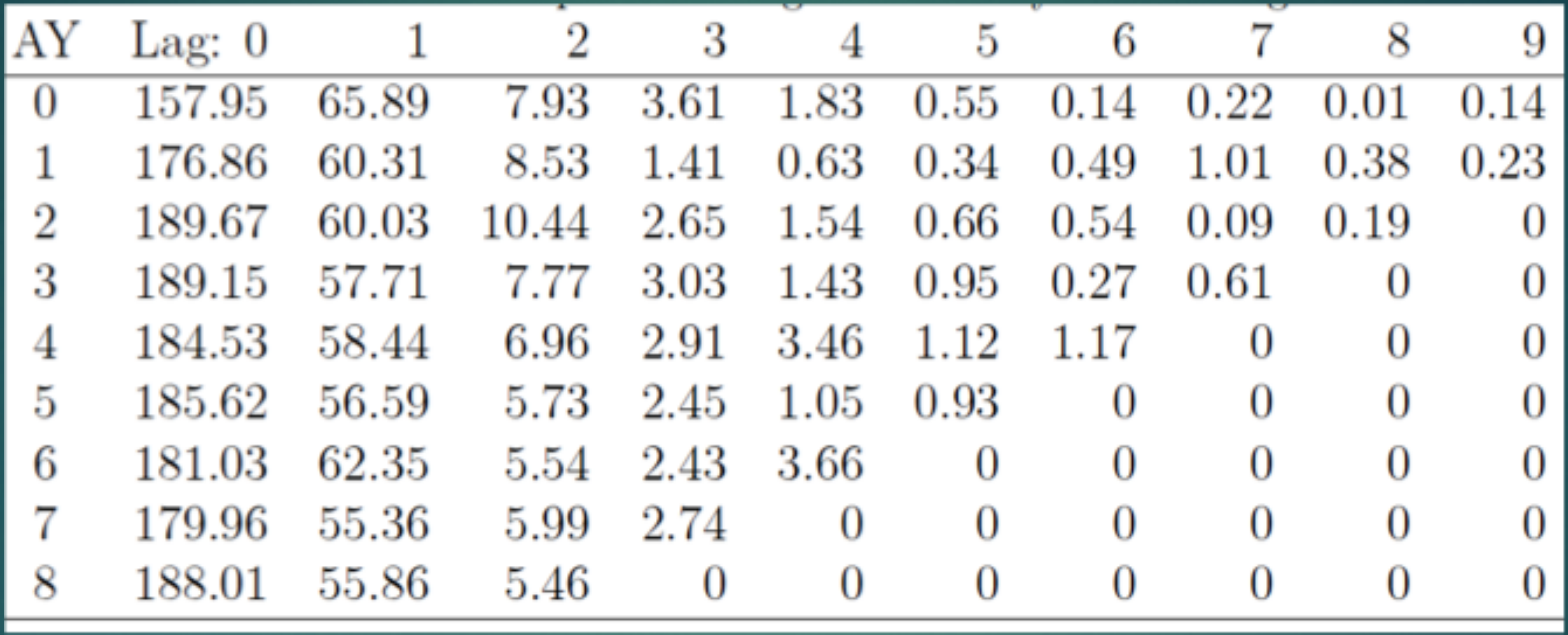
Row – Column Factor Model

- **For a cell in row u, column w, the mean is a constant times row and column factors**
  - $\mu_{w,u} = A_w B_u C$
    - Row factors A, column factors B, constant C
    - Factors for first row and column are both 1.0
- **Parameters for us are 2nd differences in logs**
  - $A_w = \exp(p_w)$, $B_u = \exp(q_u)$, $\mu_{w,u} = exp(p_w + q_u + c)$
  - $p_1 = 0$, $p_2 = a_2$, $p_3 = 2a_2 + a_3$, $p_4 = 3a_2 + 2a_3 + a_4$, …
  - D_R = design matrix row section consists of all these coefficients on the a's, so
  - $(p_1, p_2, \ldots)^T = D_R{*}(a_1, a_2, \ldots)^T$, similar for q and b.
  - That is linear model part

Distributions

- Losses in cell j are assumed gamma distribution with mean $\mu_j = \alpha_j\beta_j$, variance = $\alpha_j\beta_j^2$.
- GLM assumes α is fixed across the cells, but here assume β is. Then $\mu_j = \alpha_j\beta$, variance = $\alpha_j\beta^2 = \beta\mu_j$. This is like ODP assumption – variance proportional to mean
- Assume that the a and b 2nd difference parameters are double exponential distributed in s. This implies that:
- Prior mean of a_w or b_u is zero, with variance a function of s
- Instead of trying a lot of values of s, assumed that log of s is uniform on [-5,-0.2]. That lets s go up to 0.8, which was ok.
- Too high an s can give convergence problems

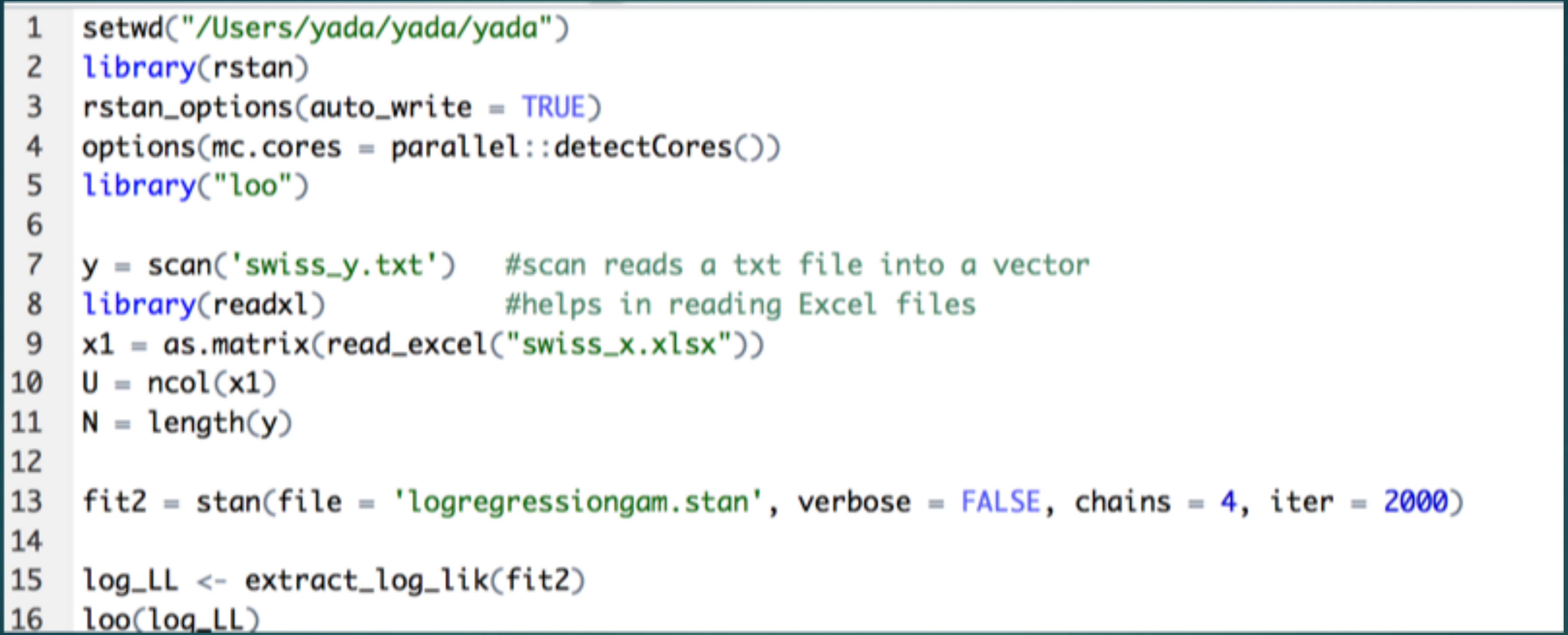
Example, from Wüthrich, Mario V. 2003. Astin Bulletin 33:2: 331–46.



AY	Lag: 0	1	2	3	4	5	6	7	8	9
0	157.95	65.89	7.93	3.61	1.83	0.55	0.14	0.22	0.01	0.14
1	176.86	60.31	8.53	1.41	0.63	0.34	0.49	1.01	0.38	0.23
2	189.67	60.03	10.44	2.65	1.54	0.66	0.54	0.09	0.19	0
3	189.15	57.71	7.77	3.03	1.43	0.95	0.27	0.61	0	0
4	184.53	58.44	6.96	2.91	3.46	1.12	1.17	0	0	0
5	185.62	56.59	5.73	2.45	1.05	0.93	0	0	0	0
6	181.03	62.35	5.54	2.43	3.66	0	0	0	0	0
7	179.96	55.36	5.99	2.74	0	0	0	0	0	0
8	188.01	55.86	5.46	0	0	0	0	0	0	0

- **Incremental paid losses, 62 data points**
- **Pretty fast paying – losses get very small in later columns**
- **Will start with row-column model**

R code to set up Stan run



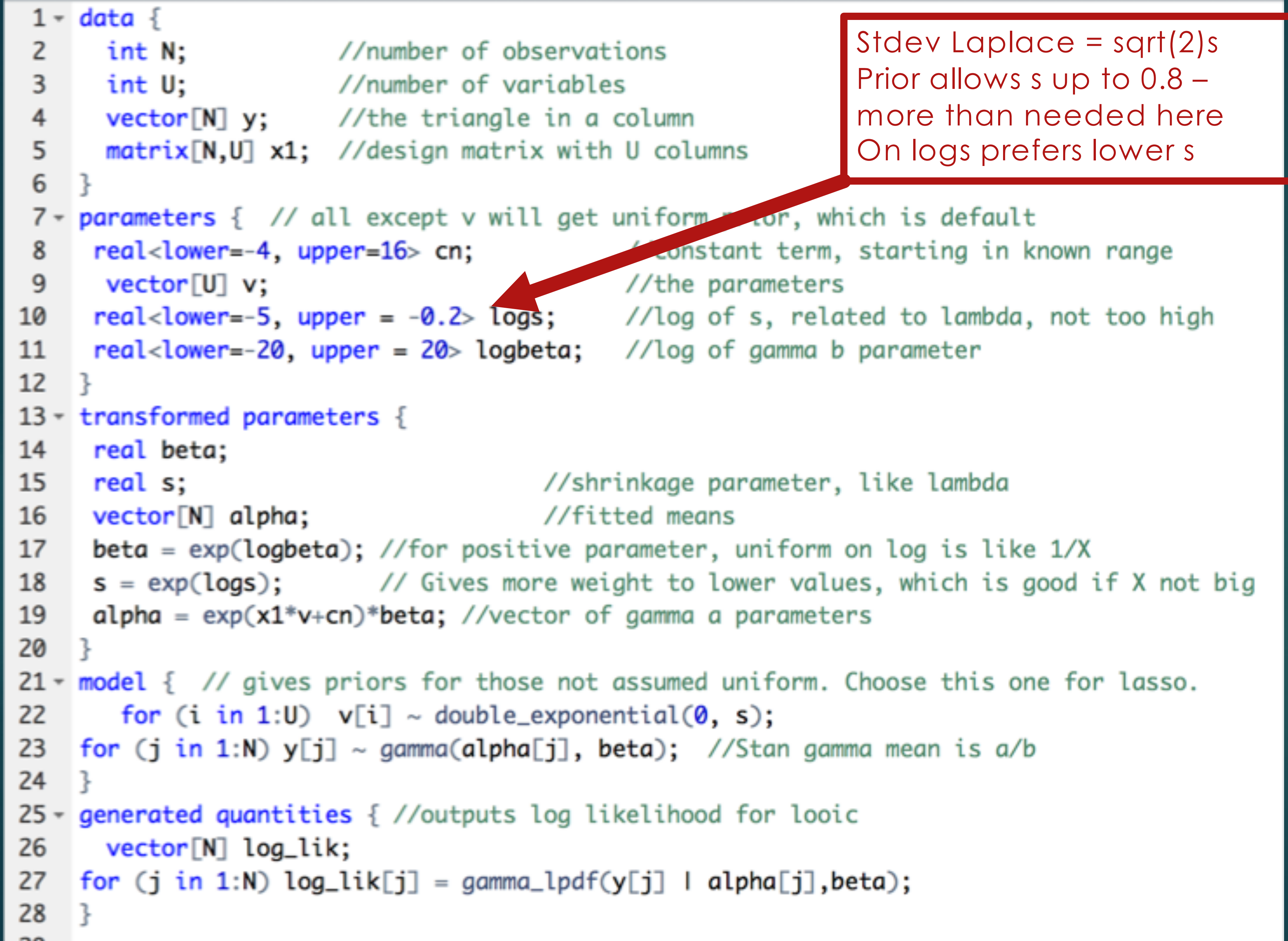
```r
setwd("/Users/yada/yada/yada")
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library("loo")

y = scan('swiss_y.txt')    #scan reads a txt file into a vector
library(readxl)            #helps in reading Excel files
x1 = as.matrix(read_excel("swiss_x.xlsx"))
U = ncol(x1)
N = length(y)

fit2 = stan(file = 'logregressiongam.stan', verbose = FALSE, chains = 4, iter = 2000)

log_LL <- extract_log_lik(fit2)
loo(log_LL)
```

Assumes triangle is in a column in a file swiss_y.txt and the dummy variables are in swiss_x.xlsx

Sets up and runs Stan model in logregrssiongam.stan

Then computes loo

Example Stan gamma code



```
data {
  int N;          //number of observations
  int U;          //number of variables
  vector[N] y;    //the triangle in a column
  matrix[N,U] x1; //design matrix with U columns
}
parameters {  // all except v will get uniform prior, which is default
 real<lower=-4, upper=16> cn;                //constant term, starting in known range
  vector[U] v;                              //the parameters
 real<lower=-5, upper = -0.2> logs;        //log of s, related to lambda, not too high
 real<lower=-20, upper = 20> logbeta;      //log of gamma b parameter
}
transformed parameters {
 real beta;
 real s;                            //shrinkage parameter, like lambda
 vector[N] alpha;                   //fitted means
 beta = exp(logbeta); //for positive parameter, uniform on log is like 1/X
 s = exp(logs);        // Gives more weight to lower values, which is good if X not big
 alpha = exp(x1*v+cn)*beta; //vector of gamma a parameters
}
model {  // gives priors for those not assumed uniform. Choose this one for lasso.
   for (i in 1:U)  v[i] ~ double_exponential(0, s);
for (j in 1:N) y[j] ~ gamma(alpha[j], beta);  //Stan gamma mean is a/b
}
generated quantities { //outputs log likelihood for looic
  vector[N] log_lik;
for (j in 1:N) log_lik[j] = gamma_lpdf(y[j] | alpha[j],beta);
}
```

Stdev Laplace = sqrt(2)s
Prior allows s up to 0.8 –
more than needed here
On logs prefers lower s

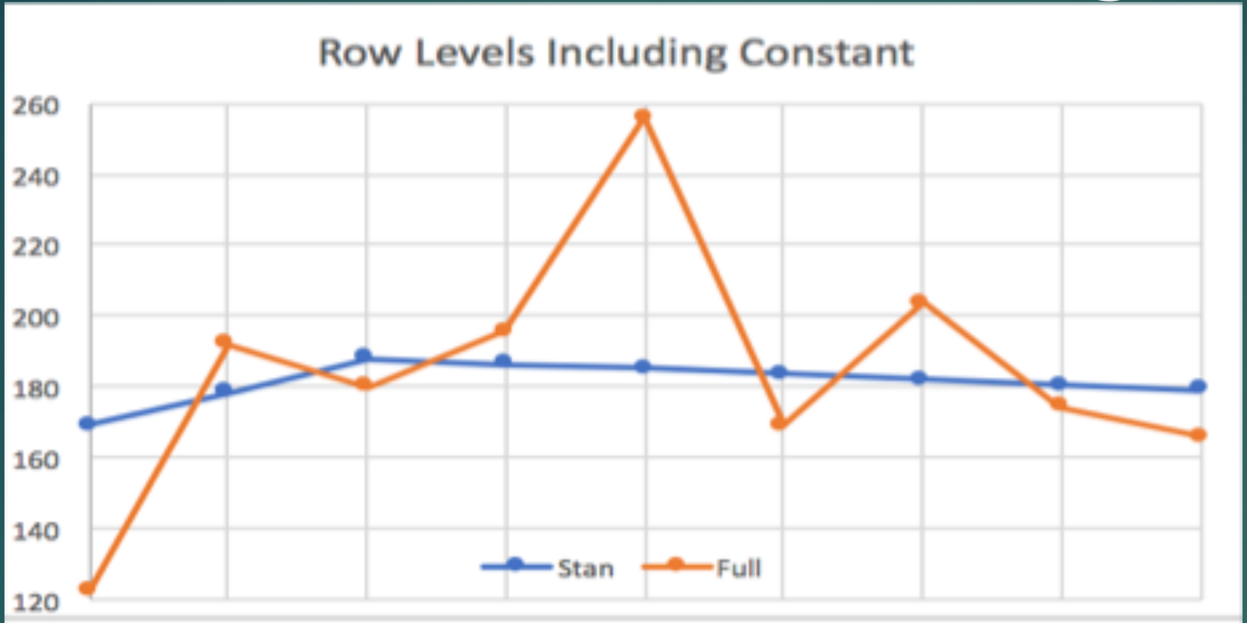
Start with defining data, parameters

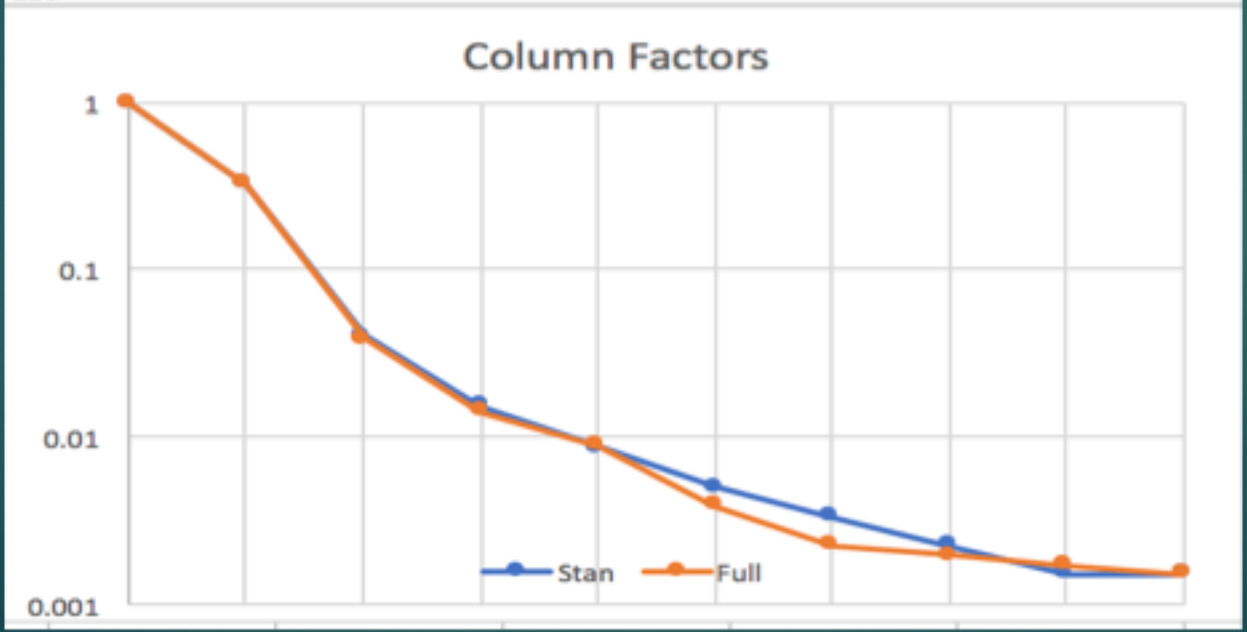
Model has priors to use then has distribution for data points y_j

Eliminating Some Parameters

- Shrinking parameters towards zero makes some of them very close to zero
- Eliminating those simplifies the model and may improve loo measure
- Print and plot functions in rstan run on Stan output gives mean and any desired percentiles of the variables – here slope changes – and plots posterior distribution of each as a bar
- Look to eliminate parameters near zero with wide ranges
- Try and see if loo improves – even if stays the same, leaving them out simplifies the model
- Eliminating a slope changes continues the previous slope so results in longer linear segments

Factors from Stan and Regression





Heteroscedasticity in Reserves

- Variance varies across cells – maybe CV does too
- If large losses pay later, later cells have lower count, higher severity
- Variance decreases slower than mean does: severity variance ~ μ^2
- A way to address this is to make variance proportional to a power of the mean that is estimated – takes two variance parameters instead of one across the triangle
  - $\text{Variance}_j = s(\text{mean}_j)^k$ where s and k are estimated
  - For any assumed distribution, solve for 2 parameters for cell by matching moments. Called k version of that distribution.
- For gamma distribution with mean $\mu_j = \alpha_j\beta_j$, variance = $\alpha_j\beta_j^2$, fixing α across cells makes k = 2, fixing β makes k = 1, but with any estimated k can solve for α and β separately for every cell
- Can do that for any distribution – select which one by skewness, other shape characteristics – using goodness of fit measures

Distribution Fits Compared by Loo

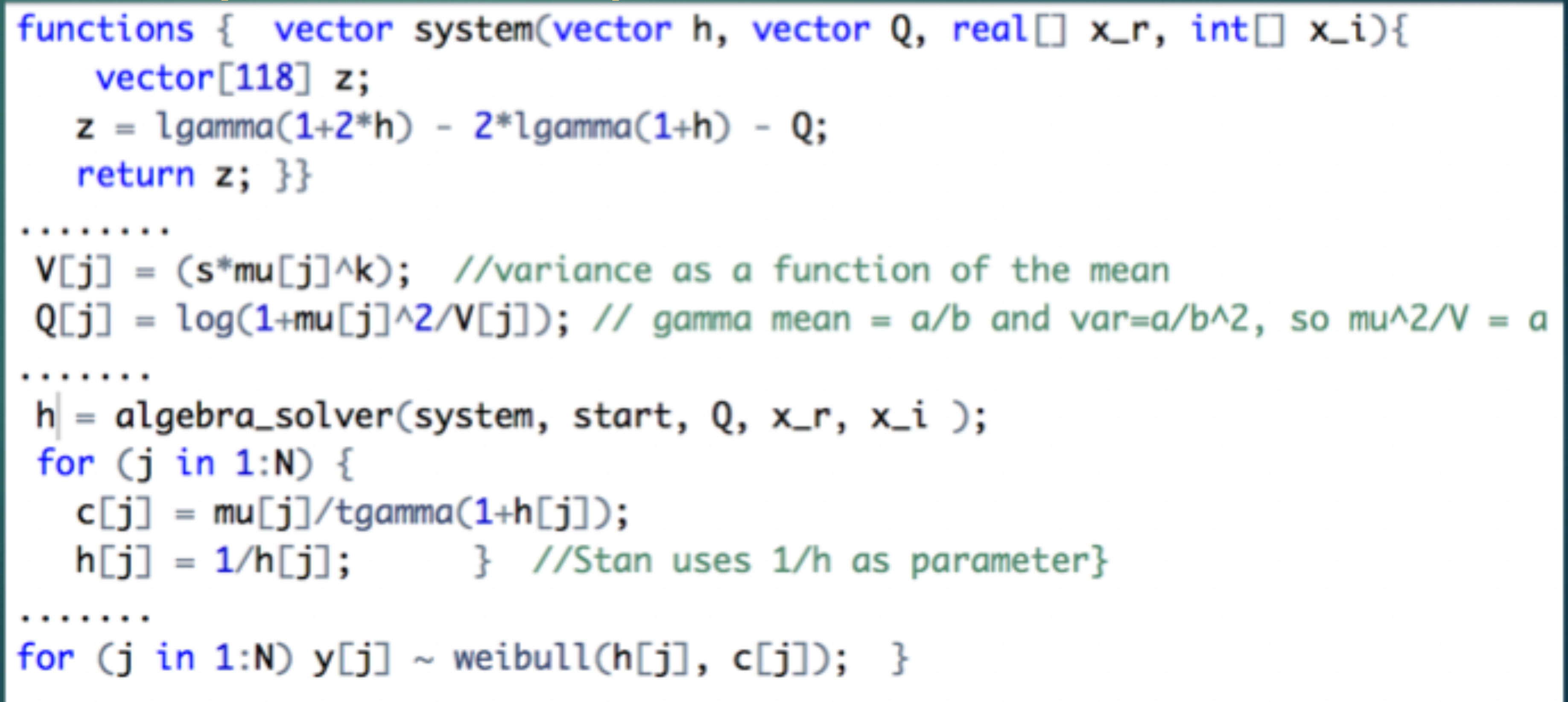
- **Triangle Model Fits**

Distribution	looic	NLL	Penalty
Normal-k	111.2	98.9	12.3
GiG	106.2	94.7	11.5
Gamma	103.6	93.8	9.8
Weibull-k	101.8	92.3	9.5

  - Looic is NLL + parameter penalty
  - Distribution with s, k fit by cell called the k form
  - GIG is weighted average of Gaussian and Inverse Gaussian, weight a parameter
  - Gamma k parameter near 1.0, so just made β constant, saving a parameter
  - Weibull best – skewness varies across cells more than gamma, but still increases with CV. Sometimes better, sometimes not

Issues with Weibull – Method of Moments Not Closed Form, Also Slow Fitting

- Using notation $n! = \Gamma(1+n)$, Weibull with $F(x) = 1 - \exp[(x/c)^{1/h}]$ has mean = $ch!$ Var = $c^2[(2h)! - (h!)^2]$. Then:
- $1+CV^2 = (2h)! / (h!)^2 = 1 + s{*}mean^{k-2}$. Solve for h as function of s, k.
- Solve in logs, using Stan's finicky solver vector system
- 118 data points here
- x_r, x_i: 0-dimensional
- Empty but required



```
functions {  vector system(vector h, vector Q, real[] x_r, int[] x_i){
    vector[118] z;
   z = lgamma(1+2*h) - 2*lgamma(1+h) - Q;
   return z; }}
........
 V[j] = (s*mu[j]^k);  //variance as a function of the mean
 Q[j] = log(1+mu[j]^2/V[j]); // gamma mean = a/b and var=a/b^2, so mu^2/V = a
.......
 h = algebra_solver(system, start, Q, x_r, x_i );
 for (j in 1:N) {
   c[j] = mu[j]/tgamma(1+h[j]);
   h[j] = 1/h[j];      }  //Stan uses 1/h as parameter}
.......
for (j in 1:N) y[j] ~ weibull(h[j], c[j]);  }
```

Going Beyond Row-Column Model

- Müller in 2016 Variance suggests adding an exposure adjustment
- Each row has the annual exposure, each column has a factor for how much of the exposure to use for that column
- Idea is that some emerging losses are a % of exposure, not of losses emerged so far
- For a cell in row w=1,2,.., column u =1,2,..,, fitted parameter:
- $\mu_{w,u} = A_w B_u C + D_u E_w$, with E_w the known exposure and D_u the column % of exposure factor to be estimated
  - µ could be cell mean, or a parameter proportional to the mean, …

Fitting $\mu_{w,u} = A_w B_u C + D_u E_w$

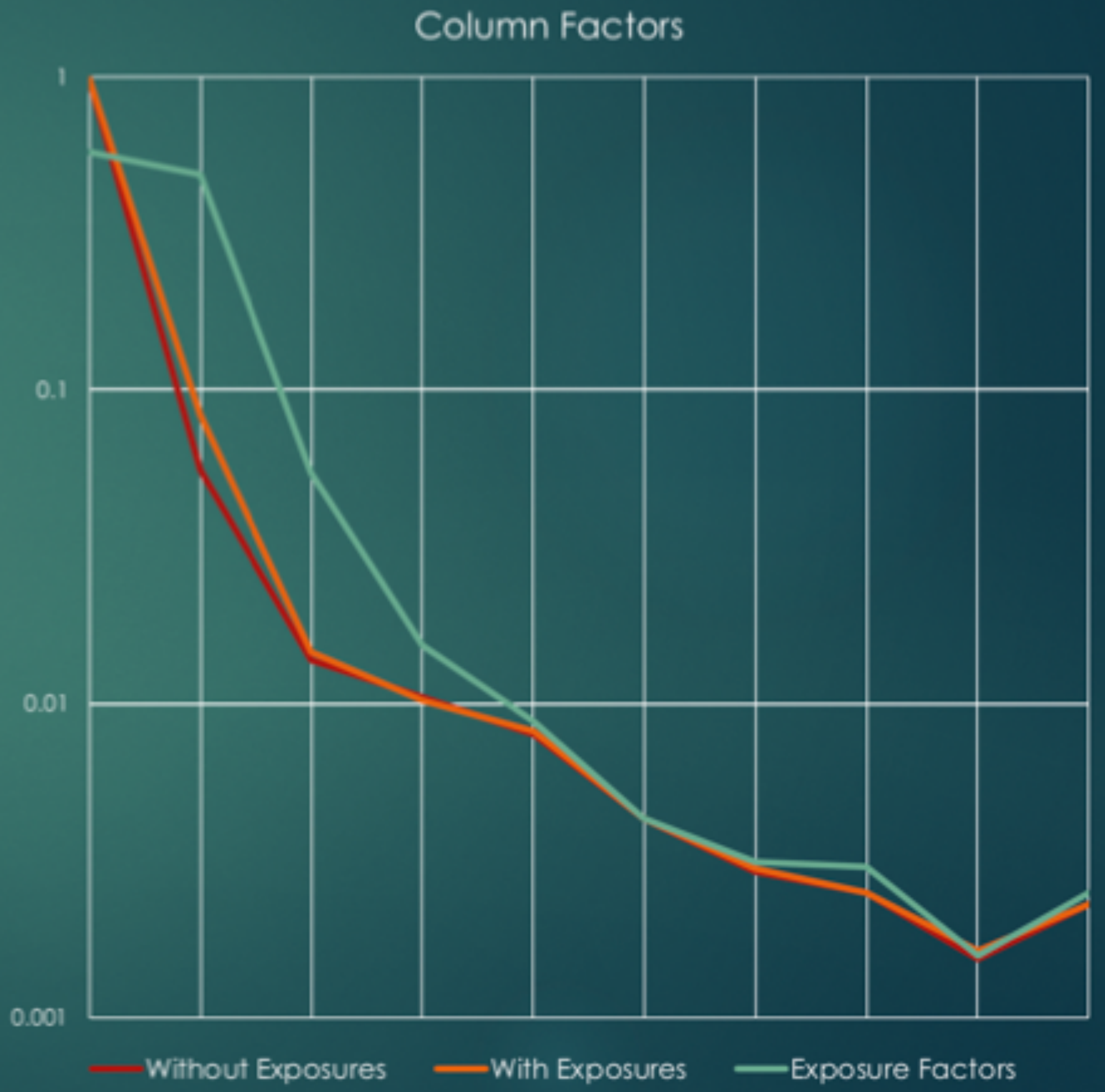
- Still use piecewise linear curve across columns for D_u
- Make a separate design matrix for the slope change parameters
- Design matrix times vector of fitted parameters is D_u for each element of the triangle when it is strung out in a single vector
- All cells from the same column will get the same D
- Need E_w also in a vector for the strung out data – will be constant for all elements from the same row
- Then dot product of those two vectors gives $D_u E_w$ as a vector
- Add that to vector of row*column*C means to get new mean for each cell – can be done in same line of Stan code
- Multiply that by beta to get the gamma alpha parameter by cell

Gamma Model Both Ways

- Swiss data had exposures by row

Model	looic	NLL	Penalty
Row-Column	103.6	93.8	9.8
With Exposure	99.9	90.1	9.8

- Extra parameters did not increase penalty as they helped with prediction
- Usually including exposure term improves model fit and predictions
- Exposure factors don't have to start at 1
- Assuming exposure = 1 often enough, especially in loss ratio triangles



Summary

- Parameter shrinkage reduces estimation and prediction variances
- Similar to credibility in shrinking fitted values towards overall mean
- Bayesian version more flexible, easier to determine how much to shrink, provides parameter distributions
- Not hard to implement in Stan package
- Can be very fast, depending on distributions, size of triangle
- One way to implement for reserving is to make the parameters to shrink the slope changes of piecewise linear fits to the row and column factors
- Mean – variance relationship across cells of triangle can be more complicated than GLM allows
- Can make variance proportional to any power of the mean with just one more parameter – then choice of distribution gives shape features, like skewness
- Simple but useful distribution is gamma with beta parameter fixed across the cells, which makes the variance proportional to the mean, as in ODP
- Including an additive term by column can and usually does improve fit