Bayesian Approach to Spline Smoothing

Gary Venter *

* Columbia University; gv2112@columbia.edu; Tel.: +1-646-266-2495

Abstract:

Regressions using variables categorized or listed numerically, like 1st one, 2nd one, etc. – such as age, weight group, year measured, etc., are often modeled with a dummy variable for each age, etc. Cubic splines are used to smooth the fitted values along age curves, year curves, etc. This can give nearly as good a fit as straight regression but with fewer variables. Spline smoothing adds a smoothing constant times a smoothness measure, often the integral of the curve's squared-second derivative, to the negative loglikelihood, which is then minimized. The smoothing constant is estimated by cross validation. Picking the knots (curve-segment connection points) is a separate estimation. Here we look at using simpler measures of curve smoothness like sum of squares of the needed parameters. This gives very similar curves across the fitted values, both in goodness of fit and visual smoothness. It also allows the fitting to be done with more standard fitting methods, like Lasso or ridge regression, with the knot selection optimized in the process. This helps modelers incorporate spline smoothing into their own more complex models. It also makes it possible to smooth using Bayesian methods. That is slower but it gives distributions for each fitted parameter and a direct estimate of the probability distribution of the smoothing constant. Cross-validation is a good method to compare models but has problems if used for estimation, discussed. Also linear splines can be modeled this way as well, and after smoothing look similar to cubic splines and are often easier and faster to fit. Modern Bayesian methods do not rely on Bayesian interpretations of probability and can be done within frequentist random effects, liberally interpreted.

Keywords: smoothing splines; shrinkage priors; MCMC; Bayesian methods

1. Introduction

Complex trends in data can be modeled with spline curves pieced together across the data range. Linear splines are line segments that join up at the points, while cubic splines use adjoining curve segments that match their first two derivatives at the joints (knots). This tends to give more realistic interpolations near the knots. Design matrices have been worked out to create these splines by regression.

Splines can also be fit across sets of variables in more general models. For instance, APC (age-period-cohort) models could have splines across each of the three sets of variables. The three spline design matrices combine into one for the whole model. Variables are need for each A, P, and C, but if adjacent fitted parameters are similar, or lie on the existing curve, those spline parameters can be set to zero, as there is no change in the curve shape at those points. This also eliminates those knots.

Smoothing the splines further increases model parsimony. Then the curves, and so the level parameters on the curves, do not swing as widely. This is a form of parameter shrinkage. Measures of curve smoothness are added to the negative loglikelihood (NLL) and are shrunk along with it. Before smoothing, the splines produce the same fitted parameters as the usual dummy matrices. Smoothing can improve predictive accuracy, even though not fitting every value as closely, because of using fewer parameters. Here parameter-shrinkage methods like ridge regression, which are known to improve predictive accuracy, are investigated as smoothing criteria. Bayesian shrinkage using shrinkage priors is tried as well. This gives a wider range of smoothing choices.

Ridge regression, Lasso, and traditional spline smoothing cannot use the usual test of NLL penalized by parameter counts to compare models. This is because shrunk parameters use up fewer degrees of freedom. Instead, cross validation (CV) is used for model comparison for model comparison.

For CV, the dataset is divided into several subsets, the NLL on each is calculated with the parameters estimated from only the remaining data, and the sum of the NLLs over the data sets is the CV NLL. CV is thus an alternative method of penalizing the likelihood, and the degree of smoothing that optimizes the CV NLL is chosen. Estimating parameters by penalized likelihood has issues, however. The penalty is effectively an estimate of sample bias – the overstatement of the likelihood due to measuring it on the sample used for fitting. But the best penalized likelihood could well be the one with the greatest under-estimation of sample bias. From a Bayesian perspective, CV estimates are often close to the posterior mode. The Bayesian posterior mean of the shrinkage parameter comes from Bayesian shrinkage, and usually the mean is a better estimator than the mode, at least as measured by the estimation variance.

In a frequentist framework, parameters are constants that do not have distributions. But frequentists can use random effects, which are a lot like parameters with distributions. Typically, the random-effect (parameter-like) distributions are postulated, not subjective as in traditional Bayesian estimation. But the same is true for priors in contemporary Bayesian analysis. They

can be tested by model results just like other postulated distributions, such as that of the data given the parameters.

The main Bayesian engine is MCMC (Markov chain Monte Carlo) estimation, which samples from the joint likelihood of the parameters and the data. By the definition of conditional distribution this is the conditional distribution of the parameters given the data times the probability of the data. The latter is an unknown constant, and MCMC can sample without knowing what it is. In frequentist terms, this gives a sample of the conditional distribution of the random effects given the data. Whether or not this is consistent with frequentist methods is largely a philosophical question for frequentists to decide. Some specifics are in the examples.

[1] give regression basis functions for cubic splines, with a derivation posted at [2]. Simplifying it a little, assume the spline has knots at the points *1:K*. They assume the spline is linear in [1,2] and [*K–1, K*]. Dummy variables a_j are defined for *j = 1:K*. For an observation from any real $z \in [1, K]$ they set $a_1(z) = 1$ and $a_2(z) = z$. Then for *j* > 2 it works out that:

$$z > K - 1\text{: } a_j(z) = -(z + 1 - K)^3 + {(z + 2 - j)^3}\Big/{K + 2 - j}$$

$$z \le K - 1\text{: } a_j(z) = {(z + 2 - j)_+^3}\Big/{K + 2 - j}$$

[3] introduced a smoothing criterion for splines in general a century ago. The curve is differenced three times at each knot, and the sum of the third difference constrained. The integral of the squared second derivative of the curve is a popular criterion lately. With parameters β_j the value of the cubic spline at any point is $\sum_j \beta_j a_j(z)$. The integral of the second derivative squared comes out to be $\left(\sum_j c_j \beta_j\right)^2$ for some values c_j. This is a weighted sum of all the cross products of the parameters. It generally increases simultaneously with $\sum \beta_j^2$ and $\sum |\beta_j|$, so constraining either of those, as in ridge regression or Lasso (least absolute shrinkage and selection operator), also constrains the squared second derivative, giving similar smoothing.

[4] provide a regression basis for linear splines. Again, $a_1(z) = 1$. For *j*, *z* > *1*, $a_j(z) = (1 + z - j)_+$. This gives a design matrix for the second differences of the level parameters. Then constraining $\sum \beta_j^2$ constrains the sum of the squared second differences along the curve, analogously to the integral of the squared second derivatives of the cubic spline.

[5] introduced ridge regression, minimizing NLL + $\lambda \sum \beta_j^2$. They showed that there is always some $\lambda > 0$ that gives a lower error variance than $\lambda = 0$, which is the MLE (maximum likelihood estimation) case. MLE gives the minimum variance unbiased estimate. Ridge regression biases the estimates, generally by shrinking them towards the overall mean. Thus it ties in to Stein's Theorem ([6]), which says that with 3 or more means being estimated, the error variance is reduced by some degree of shrinkage towards the overall mean. With spline design matrices smoothing the splines generally improves the fit. There remains the issue of how much to shrink.

Lasso, from [7], popularized by [8] instead minimizes NLL $+\lambda\sum|\beta_j|$. This is used more because it sets some parameters to exactly zero, thus taking them out of the model. That is the selection part. It still reduces the error variance by shrinkage towards the overall mean, verified for each model by cross validation for λ.

2. Materials and Methods

This section describes the methodology applied, with specific details in the next section. The first example looks at a sample of USA workers compensation benefit payments arranged by year of accident (rows = cohorts) by time to payment (columns = ages), as in Table 1. The years of payment (periods) are the upward sloping diagonals. The logs of the payments are fit by APC regression models.

Table 1. Workers compensation benefits paid by accident year and lag

129124	217397	118421	63984	39196	31450	19809	14556	8420	8507	7319	8937	10274	5383	5594
216427	325705	171013	92336	65834	41287	28358	22541	15742	16934	12498	17058	10786	7834	0
190698	207945	132229	74989	41989	32347	26640	21860	26870	17868	15149	15073	10549	0	0
117415	166555	95082	55577	40785	28737	16396	15102	12993	16123	10323	8979	0	0	0
133903	196230	110738	68289	52485	31817	23922	25356	17653	13431	14458	0	0	0	0
201997	289239	165234	103281	70280	57042	39964	34252	25683	21589	0	0	0	0	0
293284	435794	242665	159174	108122	75959	53191	42359	35453	0	0	0	0	0	0
409026	605225	381820	235599	162399	113710	81298	70265	0	0	0	0	0	0	0
514682	799168	437650	281391	203616	143531	109599	0	0	0	0	0	0	0	0
674265	886002	512631	364672	263383	215784	0	0	0	0	0	0	0	0	0
692986	905049	546113	396918	266963	0	0	0	0	0	0	0	0	0	0
700094	929014	590184	366655	0	0	0	0	0	0	0	0	0	0	0
627820	834302	517097	0	0	0	0	0	0	0	0	0	0	0	0
449942	552715	0	0	0	0	0	0	0	0	0	0	0	0	0
326810	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The second example is log mortality rates for Danish women found in the Human Mortality Database for ages 50 – 89 for years of death 1967 – 2016. The mortality rates are computed as the number of deaths divided by the applicable population. The data used is age by year, with cohort computed as year of death – age at death. These year:age cohorts approximate the year-of-birth cohorts. With 50 ages and 40 periods there are 40 + 50 -1 = 89 cohorts, 1878 – 1966. The variables for each direction are numbered consecutively 1, 2, 3…

For the linear and cubic spline design matrices, the logs of payments and of the mortality rates are the dependent variables, put into column vectors, with parallel vectors set up to track the age, period, and cohort of each observation to help create the design matrices. The first age, period, and cohort are left out for the linear splines, with a constant term represented by a

column of all 1's. The three individual-dimension spline design matrices are combined into larger ones for the datasets, with the age, period, and cohort variables grouped in columns.

These combined design matrices are singular because of the overlap among the three directions, which is a well-known problem in APC modeling. There is no apparent way to eliminate the overlap and get true effects for each direction. Even leaving out a direction does not help. Fitting an age-period model with no cohorts implicitly assumes that there are no cohort effects, and the parameters use this assumption. It is testable by reviewing residuals by cohort, and usually fails. For spline models, some variables will quickly be eliminated in the fitting – those for ages, periods, and cohorts where the curve shape changes minimally. Leaving out such variables makes the design matrix invertible. In the first example, the age-cohort model was fit first, and the least-significant curve-shape variables were taken out, and then the period variables were put in the model. For the second example, the cohorts were left out initially.

The variables easy to eliminate were identified by a Lasso fit on the initial spline design matrices, using the R package glmnet. The function cv.glmnet does a cross-validation exercise for λ, using customized algorithms to find a range of values for λ. One output is $lambda.min, the resulting smallest reasonable degree of shrinkage. The variables whose parameters it sets to zero were the ones eliminated. This was done again once the third direction variables were put back in, leaving a starting set of variables for the analysis. Some of those latter also went to zero. For the cubic splines, the first variable in each direction is the constant term. Only one of those is kept. The second variable gives the age, period, or cohort number for the direction. Because cohort = period – age, these variables are linearly dependent, and one must be left out. In these examples, that already happened at the first Lasso stage. Also, glmnet adds a constant that is not shrunk, so the constant term is left out of the design matrix for this step. Usually you do not want to shrink it.

Ridge regression parameter estimates for a given λ have a closed-form expression that simplifies cross validation and Bayesian analysis. With design matrix x and data y, standard regression parameters are given by:

$$\beta = (x'x)^{-1}x'y$$

With U parameters, let J be the UxU identity matrix with $J[1,1]$ replaced by zero. Then the ridge-regression estimate is:

$$\beta = (x'x + \lambda J)^{-1}x'y$$

The sum of squared residuals is $\sum(y - x\beta)^2$. This computes quickly enough to easily do leave-one-out (loo) cross validation for λ, using only a few lines of R code. Just loop through the observations, computing for each its squared residual, and so its loglikelihood, with the parameters fit without that observation. Standard R non-linear optimization software can then quickly find the cross-validation λ with the highest loo loglikelihood. This is done for each example using cubic and linear splines.

The implementation of MCMC is done here in the Stan software system, using the Rstan and CmdStanR packages. The Stan code is run from R then gets converted to C++ and compiled. The Stan code for a model mainly takes in the data and records the prior distributions. In these examples, the priors for the parameters β, but not for the constant, were taken as mean-zero normals with standard deviation s, which is estimated. These priors produce Bayesian ridge regression. The NLL of the parameters is a constant $+\beta^2/2s^2$, so their NLL sum is the ridge regression penalty $\lambda\sum\beta_j^2$, $\lambda = 1/2s^2$. The residuals are postulated to be normal with standard deviation *sig*. Both log(s) and log(*sig*) are assumed to be uniform distributed in [–6, 6], and the constant is uniform on $\pm 1.8\times 10^{308}$, which is the limit of double-precision numbers on most computers. The range *{-6,6]* is fairly wide for these parameters, and this was checked by finding that the posterior ranges were well within this interval. For a positive parameter, a wide range on the parameter can bias it upward but putting the prior on the log helps.

Bayesian Lasso puts double-exponential priors on the parameters. The NLL is then a constant$+|\beta|/2s^2$, so gives Lasso. It does not, however, give any parameters identically zero over the entire sample. It does for some parameters in most samples, but this can vary across the samples, so the means are never exactly zero.

The double exponential distribution is similar to a Students t-distribution with six degrees of freedom. With matching scale parameters, these distributions agree in variance and kurtosis, and so in all existing positive moments of the t-distribution, as the odd moments are zero for both. The observed different shapes close to zero are due to differing moments of 1/X for the positive half of the distributions. The normal is the limit of the t as the degrees of freedom increases, and for degrees of freedom less than 6, the t gets more heavy-tailed. The t with 2 degrees of freedom is sometimes a good shrinkage prior. It has:

$$F(x) = 0.5 + 0.5x[2s^2 + x^2]^{-0.5}$$

$$f(x) = (2 + [x/s]^2)^{-1.5}/s$$

Then for each β, -log$\big(f(\beta)\big) = 1.5\log\big(2 + [\beta/s]^2\big) + \log(s)$, so for a fixed s, the constrained estimation would minimize NLL$+\sum_j \log(2 + [\beta_j/s]^2)$. Just finding parameters to minimize that by direct search is similar to Lasso and ridge regression, but it would constrain larger parameters less than the sum of squares or absolute values does, so would allow greater relative deviation from zero for larger parameters than in either Lasso or ridge regression. It also would shrink the smaller parameters more.

Intermediate between cross validation and fully Bayesian estimation of λ is a hybrid constrained Bayesian estimation. This is Bayesian estimation of the loo cross-validation λ. It estimates λ using the full spline design matrix. The fitted value for each point for a given λ is the ridge-regression fitted value excluding that point, calculated by the formula above applied to each point individually. The residuals from those fitted values are taken as normally distributed with standard deviation *sig*, which is the other parameter estimated in the model.

This can run much faster and generally gets mean β parameters close to those from the full Bayesian analysis, but with unrealistically low standard deviations. It generates a distribution of λ, and often its mode is close to the cross-validation λ.

The full Bayesian estimation builds a sample distribution of all the parameter sets that could have generated the data. The hybrid method includes just the parameter sets that are ridge-regression parameters from some λ. That would not be as useful for distributional projections from the model but would provide a look at the parameters and gives a distribution for λ.

In the full Bayesian estimation, Stan uses MCMC to create a sample of the conditional distribution of the parameters given the data. It also provides an estimate of the loo cross-validation loglikelihood for the whole sample. For that it uses Pareto-smoothed importance sampling [9]. The entire distribution of *s* would then be provided for risk analysis. A point estimate could be selected as the mean of *s*. This avoids using cross-validation for parameter estimation. Also the cross-validation estimate of *s*, under some common assumptions, is more like the mode than the mean of the conditional distribution.

The distributions are also postulated for the β parameters in the frequentist random effects estimation, but in that case the constant, *s*, and *sig* are parameters. There is no apparent reason that these three could not also be taken as random effects with the postulated distributions. With that step, MCMC could be used in a frequentist estimation.

3. Results

Table 2 shows the resulting λs and loo cross-validation sum of squared residuals (SSR) for linear and cubic splines for the two examples.

Table 2. Ridge-regression loo cross-validation results

	Example 1 Linear, Cubic		Example 2 Linear, Cubic	
λ	0.342	0.004	25.36	0.202
SSR	2.163	3.138	8.046	11.21

A higher λ for a given model means that more shrinkage was applied. The linear models smoothed more and had better fits by loo cross-validation SSR than did the cubic models but this might not be true in general.

The variables remaining for each model after the Lasso step with MCMC-fitted parameters are given in Table 3. The variables are identified as *a, y,* or *c* for age, year, or cohort. In the linear splines, the first age, year, and cohort are left out for identifiability, and this is also done for the cubic splines as those variables are all constant terms.

The hybrid MCMC – cross-validation estimated λ for Example 1, linear splines is 0.402, with a mode of 0.291. The cross-validation estimate of 0.342 is between these. The cubic-spline estimate is 0.0066, with mode of 0.0048,

which is still higher than 0.0040 from cross-validation. The means are higher than the modes as is usual for positively skewed parameter distributions, and they were higher than the CV estimates. Calculating modes for sample distributions is a bit involved. The R modeest package was used here with the half-sample mode estimator. That finds the smallest interval with half the sample in it, then repeats for that half, etc. until it narrows down on the estimated mode.

For Example 2, the MCMC linear spline λ is 22.0 with standard deviation 7.7 and mode 24.4, compared to λ of 25.4 from cross validation. The mode is close to the CV estimate, but now the distribution seems to be negatively skewed. The MCMC cubic-spline λ is 0.0033, which gives less shrinkage than the λ = 0.2 from cross validation. The MCMC and hybrid estimations here followed some of the cohort variations in more detail than did cross validation – which had a better SSR than the hybrid did.

In both examples for both splines, the hybrid β parameters are close to those from the full MCMC, and the resulting level parameters on the splines are difficult to distinguish in a graph. Naturally, though, the parameter standard deviations from the hybrid fit are too small, as the parameters are constrained to be ridge-regression results.

The loo penalized loglikelihood for the linear spline fit in Example 1 is 74.9, with a standard error of 11.9. For the cubic spline it is 52.0 with a standard error of 8.5. They are about 2 standard errors apart, so the linear spline fit is probably but not definitively better in this example. In Example 2, the linear spline loo is 2676 with a standard error of 39.3. The cubic loo is 2376.5 with standard error 42.2. The linear spline fit is somewhat better there as well.

Table 3. Variables left after Lasso with MCMC parameter estimates

Ex. 1 Linear		Ex. 1 Cubic		Ex.2 Cohort Linear		Ex.2 Cohort Cubic		Ex. 2 All Other	
cn	11.87	cn	11.47	c2	-0.0056	c3	-0.0033	Linear	
y2	0.289	y2	0.092	c3	-0.0060	c9	0.02559	cn	-5.1208
y4	-0.098	y6	0	c30	0.0028	c10	-0.0773	a2	0.0825
y6	0.074	y8	0	c31	0.0017	c11	-0.0199	a10	-0.0043
y8	0.036	y9	0	c32	0.0018	c12	0.14383	a11	0.0071
y9	0	y10	0.013	c33	-0.0028	c13	0.0283	a12	-0.0022
y10	0	y11	-0.025	c34	0.0004	c14	-0.0339	a19	0.0087
y11	0.061	y14	0.008	c35	0.0042	c15	-0.1202	a25	-0.0052
y14	-0.107			c36	0.0033	c16	-0.0177	a26	0.0181
y15	0.071			c37	0.0020	c17	0.08452	y30	-0.0063
c2	0.119	c2	0.415	c38	0.0034	c18	0.03676	y31	-0.0044
c3	-0.484	c3	-1.848	c50	-0.0015	c19	-0.0374	y34	-0.008
c4	-0.232	c4	3.163	c51	0.0026	c20	-0.0984		
c5	0.507	c6	-1.769	c52	-0.0003	c21	0.10715	Cubic	
c6	0.159	c7	0	c53	0.0029	c24	-0.0175	cn	-5.16
c9	-0.157	c8	0.425	c54	-0.0026	c40	0.08974	a3	0.1285
c10	-0.025	c11	0	c55	-0.0069	c41	-0.1814	a7	-1.4956
c11	-0.192	c12	0.012	c56	-0.0220	c42	-0.0154	a8	2.2623
c12	-0.032			c63	0.0004	c43	0.25192	a9	-0.5971
c13	-0.113			c64	0.0120	c44	-0.2621	a10	-0.7382
c14	-0.168			c65	0.0073	c45	0.09924	a11	0.5747
a3	-0.785	a3	-0.748	c66	0.0019	c46	0.24626	a12	-0.1344
a5	0.105	a5	2.904	c67	-0.0031	c47	-0.1843	y3	-0.0002
a6	0.023	a6	-3.018	c68	-0.0053	c48	-0.4701		
a8	0.13	a7	0.916	c69	0.0023	c49	0.65241		
a10	0.061	a10	-0.071			c50	-0.2266		
a12	0.143								
a13	-0.188								
a14	-0.029								

There are splines fit to each of age, year, and cohort components for each example. The sum of these components plus the constant term gives the fitted value for a data point. Figure 1 graphs these six splines for the Workers Comp payment example, Example 1, for the MCMC fits. The other fits give splines that are visually indistinguishable from these. Even though the cross-validation estimation shrinks less, the splines are robust to small changes in λ. This makes the weakness of cross validation less important but of course MCMC gives better interval estimates.

In this example there are 15 intervals in each direction, so they work on one graph. The cubic splines are smoother. The linear splines show corners

where the slope changes, but the parameter shrinkage keeps them reasonably smooth. The cubic and linear splines in each direction look a bit like rotations of each other. These offset to give similar fitted values.

The cubic splines are graphed with dots at intervals of 0.2 to show the smoothness of interpolation. They were fit at intervals of 1.0, or possibly even longer if some parameters went to zero. The jumps in the last interval are a bit misleading. The basis functions were set up to make the splines linear within the first and last intervals with the same slope, given by the coefficient of the second spline variable. That's what looks like a jump near the right side of the graphs. But only the ending point of the interval was used in the fitting. Visually those final points appear to continue the curves established in the prior intervals. It would probably not be difficult to devise a new spline basis that just continues the current spline without making it linear in the last interval.

For Example 2, Danish female mortality, there are different numbers of points in each direction so separate graphs are used. Also the MCMC and cross-validation curves are slightly different, especially for cohort variables for cubic splines. Figure 2 graphs the age and year splines and Figure 3 shows the cohort curves.

The cross-validation and MCMC linear splines again look the same, and the cubic splines are smoother, but with more twists and turns. The cubic spline fit by MCMC was shrunk less and shows more detailed movements than the one fit by cross validation, especially for the cohort splines. The cross-validation cohort spline, where those details were smoothed over, was better.

In the linear spline fit, the cohort trend overall was almost double the time trend. This included a sharp increase starting for cohorts born in 1912, peaking around 1932, and not getting back to the previous trend until those born in the early 1940s. This is not a typical pattern. [9] and [10] studied Danish mortality drivers and found that smoking behavior is a cohort effect that explains much of the historical mortality pattern. Smoking tends to be a lifelong habit, so would be a cohort effect. They found that there had been a strong increase in smoking among Danish women born between the two world wars, which is consistent with the model results here.

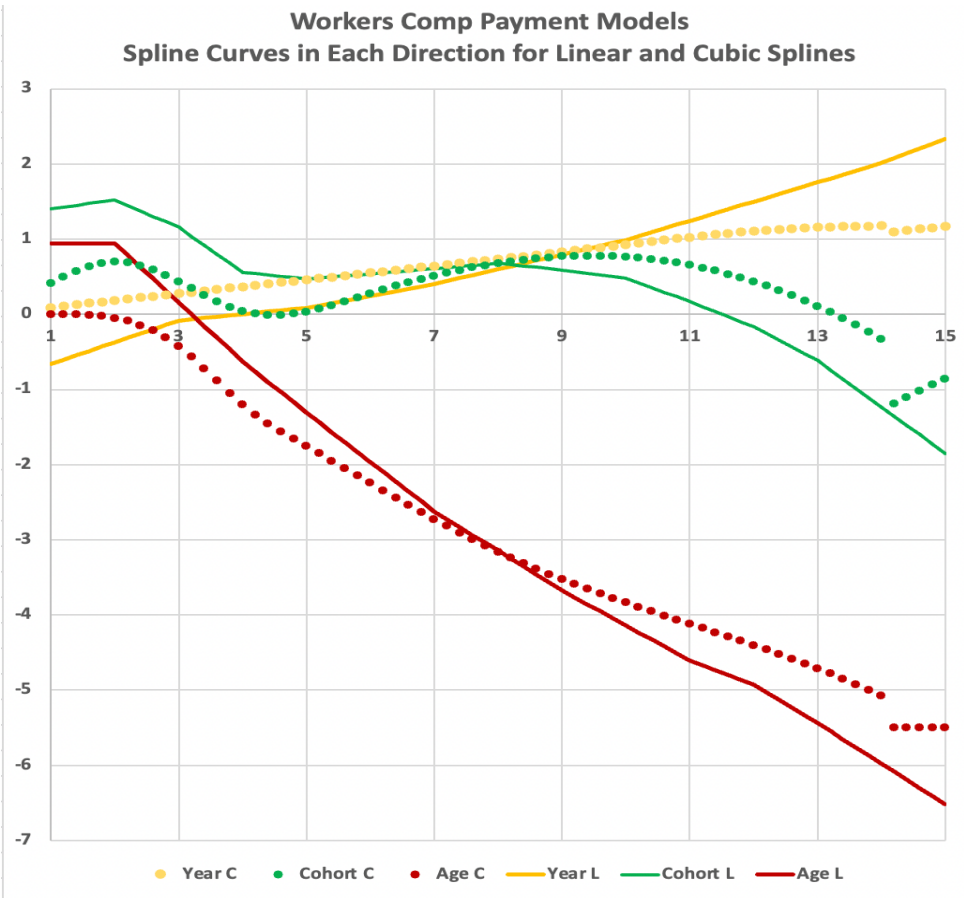

Figure 1. Spline curves Example 1, denoted as C or L for cubic or linear

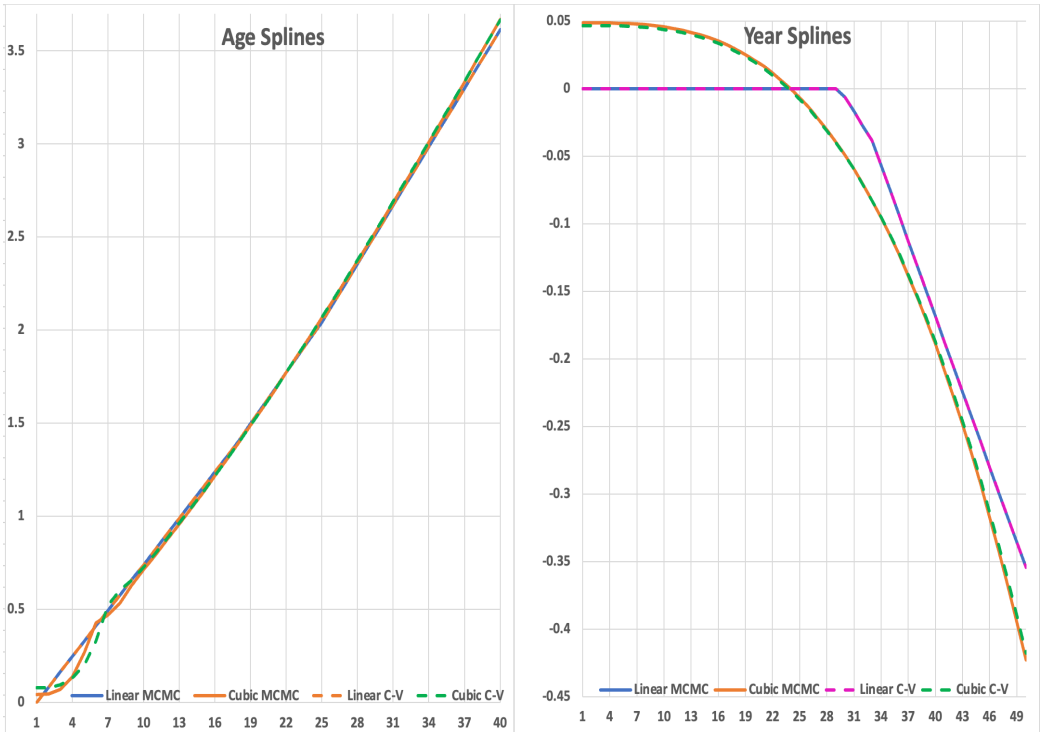

Figure 2. Age and year spline curves Example 2, denoted as MCMC or C-V.

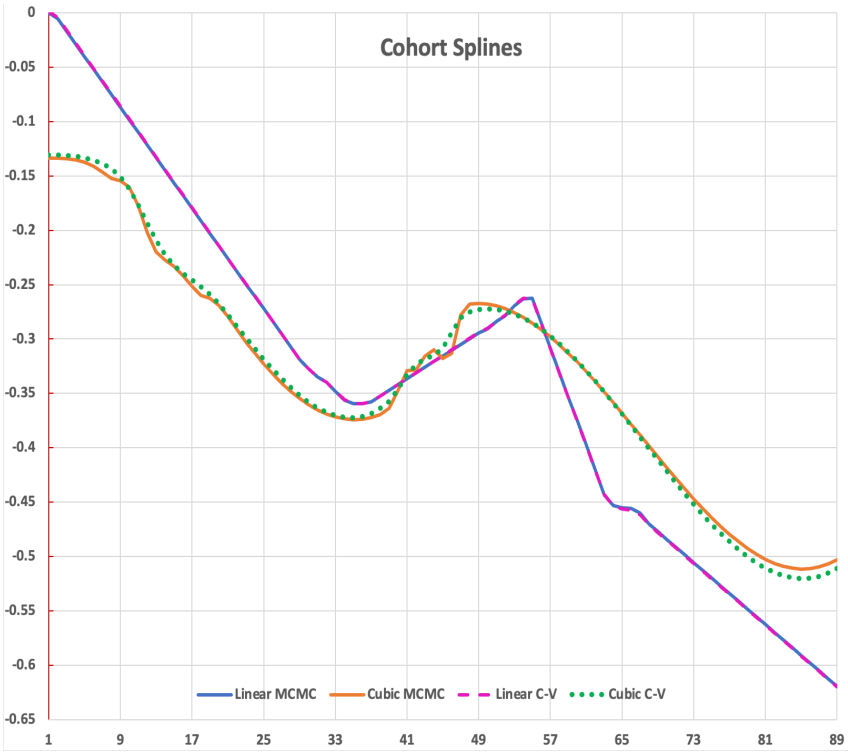

Figure 3. Cohort spline curves Example 2, MCMC or C-V for cross validation.

4. Discussion

Using parameter shrinkage for spline smoothing results in reasonably smooth splines, and shrinkage has known advantages, like reduction in error variance. Lasso software can be used to do the estimation, and the Lasso-selected minimal degree of shrinkage usually eliminates a few variables, and with them their related knots. These are at points where the curve shape does not change much, and Lasso makes that change zero. The remaining knots are where the shape changes the most.

Ridge-regression smoothing for a given λ can be fit quickly with a single numeric calculation. This makes it fast and easy to use simple cross-validation for estimation of the smoothing constant λ. An advantage of instead using Bayesian shrinkage priors in MCMC is that they give full distributions of the parameters and the resulting spline curves, and of λ as well. They also avoid problems with cross-validation parameter estimation understating sample bias and estimating λ by the mode. The overall fits to the data are not particularly sensitive to the degree of shrinkage, however, so the mode might not be too bad.

Linear splines had largely been abandoned for being too jagged, but spline smoothing helps with that. In these examples, the cubic spline curves are smoother, but the linear splines are not too jagged and give better fits. They may have more flexibility in matching the changes in curve shapes needed here but that is not known more generally. A hybrid using MCMC in

cross validation averts some problems with CV estimation, and is faster than straight MCMC, but does not produce the full distribution of possible parameter sets. For one of the cubic splines, CV seems to give a more useful model than do the hybrid or Bayesian approaches.

Having simple formulas for spline smoothing enables modelers to put smoothing spline fitting into their own code for model development. MCMC does not require linear models, and spline parameter shrinkage can be included as needed in a wide variety of models. Some nonlinear extensions of Example 1 using linear splines are briefly covered next as illustration.

It is common to start with a model for logs of the data, and then use a log link for fitting to the original unlogged values. Fitting a gamma distribution with a log link is widely used in General Linear Models (GLM). The j^{th} observation is postulated to be gamma distributed in a_j, b_j, which has mean and variance $a_j b_j, a_j b_j^2$. Usually some simplifying relationship among these parameters across the observations is assumed. In GLM the assumption is $a_j = a$, so this parameter is constant across the observations. Then the log link gives the b_j parameters. The gamma's mean squared divided by its variance is a, so is then constant for all observations. Thus the variance is proportional to the square of the mean. But if $b_j = b$ is taken to be constant across the observations, then b gives the variance divided by the mean, so that is made constant. In general, you could get the b_j parameters from the log link and then for a constant c, to be estimated, and any power p, set a_j to make the variance = $c\mu^p$ with:

$$a_j = \left(c b_j^{p-2}\right)^{1/(1-p)}$$

Then:

$$c\left(a_j b_j\right)^p = \left(c b_j^{-p}\right)^{1/(1-p)} = a_j b_j^2$$

Another popular generalization is to include interactions among the directions in the model. In mortality this started with the model of [11] that has the period trend modified by age. This is reasonable as medical improvements, etc., affect some ages more than others. Later [12] added cohorts to this model. The mean log mortality rate in year i for age j is:

$$m_{ij} = cn + a_j + y_i^{b_j} + c_{i-j}$$

[13] and [14] use this with linear splines in MCMC for a negative binomial fit with a log link. Like the gamma, there are two forms of the negative binomial that can be used, although GLM only uses one of them.

The model of [12] also works for Example 1. The period trends in workers compensation in the US are driven by medical cost inflation. The payments include wage replacement as well but that is usually not inflation adjusted. The wage portion also caps out in a few years for most workers. Medical becomes a growing part for later payment ages (which are the payment times since injury), and so the period trend affects later payments more.

The MCMC loo cross-validation loglikelihood is 74.9 for the APC log model above. This improves to 85.9 by including these interaction effects.

Assuming a lognormal distribution for the actual losses using the log link produces all the same parameters as the log fit but with a loo of –1266.6. Likelihoods are lower for the unlogged values as the numbers are more spread out and the density is lower. For the gamma distribution with variance proportional to mean-squared, as in GLM, the loo is slightly worse, at –1268.3. With variance proportional to mean, it is better, at –1257.7. Finally, using the t-distribution with two degrees of freedom as the shrinkage prior brings this up to –1244.1, with a standard deviation of 13.6. Each step gives a possibly better result, and they add up to a meaningful improvement. The interaction term, choice of gamma distribution, and t-2 shrinkage prior are examples of the nonlinear modeling methods that can be built with spline models.

One limitation of Stan is that there are some probability distributions it cannot handle, including the Tweedie and the Poisson-Inverse Gaussian, which is a heavier-tailed negative binomial. These have good R apps but those are specialized, and Stan hasn't included them yet. The Poisson-Inverse Gaussian uses modified Bessel functions which move very slowly. It can take 40-50 digits of accuracy to compute their changes, and double-precision computers need special software for that. Stan cannot just call the R apps because it converts to C++ and compiles. There are MCMC packages that can call anything from R, but right now they are significantly slower.

The discussion here has emphasized what Bayesian shrinkage can do for smoothing splines, but the flip side of that is what smoothing splines can do for Bayesian shrinkage. Nearby parameters that are similar, or change in a regular way, can be put on splines that use fewer parameters. That can increase the parsimony and predictive accuracy of shrinkage models. Also it can help address the overlap problem of APC modeling. In APC models it is not always possible to eliminate variables with low t-statistics, as variables are needed in every position. Splines can deal with that as the omitted variables are just points where the curve shape doesn't change. The two methodologies work well in combination.

5. Conclusion

Constraining the parameters themselves produces smoothed splines like those coming other smoothing criteria. Then closed-form or standard regularized regression apps can do spline smoothing, but this still uses cross-validation for estimating the smoothing constant. Also, knot-selection can be done along with the fitting in an integrated manner. Often the knots end up widely spaced in some regions and concentrated in others when that works better than the usual evenly-spaced knots.

Bayesian shrinkage uses shrinkage priors on the spline parameters. There is a wide choice of such priors, including analogues of Lasso and ridge-regression. Putting another prior on the smoothing constant provides an estimate of its conditional distribution given the data, which avoids problems from using cross validation for parameter estimation. The saving grace for cross validation is that fitted and projected data values are robust to small changes in the smoothing constant. Linear and cubic smoothing splines give comparable smoothness and fit quality. Bayesian estimation uses MCMC,

and most or all of that can be done with frequentist random effects to produce samples of the distribution of the random effects given the data.

Modelers can customize these methods for their own models. As an example of the kind of things that can be done using MCMC, a model was fit to one of the examples in several steps, adding an interaction of age and period effects, a log link for a distribution not in GLM, and a Students-t shrinkage prior. Each step improved the fit in this case.

Further research possibilities include:

- Adjusting the cubic-spline basis functions so the spline is not linear in the final segment.
- Exploring the effects of the choice of shrinkage prior on goodness of fit.
- Comparative analysis of linear vs. cubic splines on various types of data.

Data Availability Statement: The Danish mortality data is from the Human Mortality Database, University of California, Berkeley (USA), Max Planck Institute for Demographic Research (Germany) and United Nations, https://www.mortality.org/. For Danish data, go to the country section of https://www.mortality.org/cgi-bin/hmd/hmd_download.php. Download the zip file (free account needed) and then find Exposures_lexis.txt and Deaths_lexis.txt under STATS.

References

1. Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, Corrected 12th Printing; Springer, 2017. Available online: https://web.stanford.edu/~hastie/ElemStatLearn//printings/ESLII_print12.pdf. (accessed on 14 2 2022).
2. Why are the basis functions for natural cubic splines expressed as they are? Available online: https://stats.stackexchange.com/questions/172217/why-are-the-basis-functions-for-natural-cubic-splines-expressed-as-they-are-es . (accessed on 13 2 2022).
3. Whittaker, E. W. On a New Method of Graduation. *Proceedings of the Edinburgh Mathematical Society* 1922, Volume 41 February, 63 – 75.
4. Barnett, G. and Zehnwirth, B. Best estimates for reserves. *Proceedings of the Casualty Actuarial Society* 2000, LXXXVII(167), 245–321.
5. Hoerl, Arthur E. and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* 1970, 12 (1): 55–67. https://doi.org/10.1080/00401706.1970.10488634.
6. Stein, C. Inadmissibility of the Usual Estimator for the Mean of a Multivariate Normal Distribution. *Contributions to the Theory of Statistics* 1956, 197–206.
7. F. Santosa and W.W. Symes. Linear inversion of band-limited reflection seismograms. *SIAM J. Sci. Stat. Comput.* 1986, 7(4), 1307–1330.
8. Tibshirani, Robert. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 1996, 58 (1): 267–88.
9. Vehtari, Aki, Andrew Gelman, and Jonah Gabry. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and computing* 2017, 27.5, 1413-1432.

10. Christensen, Kaare, Michael Davidsen, Knud Juel, Laust Mortensen, Roland Rau, and James W Vaupel. The Divergent Life-Expectancy Trends in Denmark and Sweden—and Some Potential Explanations. In Crimmins, E. M., Preston, S. H., & Cohen, Panel on Understanding Divergent Trends in Longevity in High-Income Countries Committee on Population Division of Behavioral and Social Sciences and Education. 2010. https://www.ncbi.nlm.nih.gov/books/NBK62592/pdf/Bookshelf_NBK62592.pdf.

11. Juel, K., J. Sorensen and H. Bronnum-Hansen. Risk Factors and Public Health in Denmark. *Scandinavian Journal of Public Health (Supplement 1)* 2008, 36, 112–227. https://journals.sagepub.com/doi/pdf/10.1177/1403494800801101.

12. Lee, R. and L. Carter. Modeling and Forecasting U.S. Mortality. *Journal of the American Statistical Association* 1992, 87, 659–75.

13. Renshaw, A. E. and Haberman, S. (2006). A cohort-based extension to the Lee-Carter model for mortality reduction factors. *Insurance: Mathematics and Economics 2006* 38(3), 556–570.

14. Gary Venter and Şule Şahin (2021) Semiparametric Regression for Dual Population Mortality. *North American Actuarial Journal*, Published online: 30 Jun 2021.

15. Venter, Gary. A Mortality Model for Pandemics and Other Contagion Events. In *BT - Pandemics: Insurance and Social Protection*; M. del C. Boado-Penas, J. Eisenberg, and Ş. Şahin Eds.; Springer International Publishing, 2022; pp. 75–94.

Appendix A

This appendix has segments of selected R and Stan code used in the study. They can be incorporated into modelers own code but are not stand-alone programs. The first one is for Lasso.

```
library(glmnet)  # Lasso app
library(readxl) # Allows reading Excel filees
y=scan('logLoss.txt')  # reads in txt file as vector
xl = as.matrix(read_excel('lin_spline_dummy.xlsx')) # use either one of these
xc = as.matrix(read_excel('cub_spline_dummy.xlsx')) # For lasso leave constant out of design m
atrix

lin_fit = cv.glmnet(xl,y,standardize = FALSE)
lin_fit$lambda.min #lminimal lambda from cross validation
out = coef(lin_fit, s = "lambda.min")
out= as.matrix(out)
write.csv(out, file="out_lin.csv")
out
```

Now the R function for cross validation with ridge regression.

```
y1=scan('comp_log_y.txt') #then design matrices with constant term
xl = as.matrix(read_excel("comp_lin_lasso.xlsx"))
xc = as.matrix(read_excel("comp_cub_lasso.xlsx"))
x1 = xc # note x1 is with number 1, not l
n = ncol(x1)
m = nrow(x1)
Il = diag(n)
Il[1,1] = 0
# define function to calculate loo for given lambda
loo_sse <- function(lam){
  loo = 0
  J = lam*Il #Il is n x n identity matrix with Il[1,1]=0
  for (j in 1:m){
    x = x1[-j,] # x1 and y1 have all rows
    y = y1[-j] # Negative indexes in R leave out the row or column
    beta = solve(t(x)%*%x + J)%*%(t(x)%*%y)
```

```
    loo = loo + (y1[j] - x1[j,]%*%beta)^2
  }
  return(loo)
}
ridge = as.numeric(optimize(loo_sse, lower = 0, upper = 30))
lambda =  ridge[1]
ridge
lambda # minimum solved for
beta = solve(t(x1)%*%x1 + lambda*Il)%*%(t(x1)%*%y1)
beta #parameters at minimum lambda
```

This is the R code to set up the Stan run for the hybrid CV-MCMC fit. It uses design matrices that include the constant term.

```
library("loo")
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
# last two  lines are some recommended Stan options
dmat = as.matrix(read_excel('lin_spline_dummy_cn.xlsx'))
dmat = as.matrix(read_excel('cub_spline_dummy_cn.xlsx'))
y = scan('logLoss.txt')
N <- nrow(dmat)
U <- ncol(dmat)
c(N,U)
xpx = array(0,dim = c(N, U, U))  #in R make 3D array of x'x missing one data point
for (i in (1:N)) {xpx[i,,] = t(dmat[-i,]) %*% dmat[-i,]}
xpy = array(0,dim = c(N,U))  # 2D array of x'y
for (i in (1:N)) {xpy[i,] = t(dmat[-i,]) %*%y[-i]}
Iminus = diag(U)
Iminus[1,1] = 0
df = list(N=N,U=U,dmat=dmat,y=y,xpx=xpx,xpy=xpy, Iminus=Iminus)
#now run stan with the arrays fed to it
set.seed(4)
fit_2 <- stan(file = 'lam_ridge_mcmc.stan', data=df, verbose = FALSE, chains = 2, iter = 3000
0, warmup = 5000, control = list(adapt_delta = 0.9, max_treedepth = 18))
```

This is the Stan code it calls – `'lam_ridge_mcmc.stan'`.

```
data {
  int N;           //number of observations
  int U;           //number of variables
  matrix[N,U] dmat;  //design matrix with U columns
  vector[N] y;     //the triangle in a column
  matrix[U,U] xpx[N];  //acts like a vector[N] of UxU matrices xpx
  vector[U] xpy[N];  //acts like a vector[N] of U vectors xpy
  matrix[U,U] Iminus;  //Identity matrix with 1,1 element = 0
}
parameters {  // both get uniform prior on defined ranges, the default
 real<lower=0, upper = 1> sig;
 real<lower=0, upper = 35> lam;
}
transformed parameters {
 vector[N] yhat; //fitted values of each observation when it is left out
 vector[N] yyhat; //fitted values using all points
 vector[U] beta;  //parameters from lambda using all points
 beta  = ((dmat'*dmat)+lam*Iminus)\(dmat'*y); //In Stan, matrix multiplication is *
 yyhat = dmat*beta; //not used but these fitted values are in output files
 for (j in 1:N) yhat[j] = dmat[j,]*((xpx[j] + lam*Iminus)\xpy[j]);
} //here xpx[j] is xpx leaving out the j observation; xpy similar
```

```
model { //fitting y using the left-out regression estimates
for (j in 1:N) y[j] ~ normal(yhat[j], sig);
}
generated quantities { //outputs log likelihood for looic
  vector[N] log_lik;
for (j in 1:N) log_lik[j] = normal_lpdf(y[j] | yyhat[j],sig);
}
```

The straight MCMC run uses the design matrix without the constant term and calls the Stan code.

```
x = dmat
n = N
u = U
df = list(n=n,u=u,x=x,y=y)
set.seed(4)
fit_3 <- stan(file = 'beta_ridge.stan', data=df, verbose = FALSE, chains = 2, iter = 2000, war
mup = 1000, control = list(adapt_delta = 0.84, max_treedepth = 20))
```

This is `'beta_ridge.stan'` that it calls.

```
data {
  int n;    // number of obs
  int u;    // number of variables
  vector[n] y;
  matrix[n,u] x;
}
parameters {  // all except v will get uniform prior, which is default
 real cn;
 real<lower=-6, upper = 6> logs;   //log of s, related to lambda, not too high
 real<lower=-6, upper = 6> logsig;   //log of sig
 vector[u] v; //parameters to fit
}
transformed parameters {
 real s;                  // shrinkage parameter
 real sig;   //sigma
 vector[n] mu;  //means
 s = exp(logs);
 sig = exp(logsig);
 mu = cn+x*v;
}
model {  // gives priors for those not assumed uniform. Choose this one for Bayesian ridge reg
ression.
   for (i in 1:u)  v[i] ~ normal(0, s);  // more weight to close to 0
  y ~ normal(mu,sig);
}
generated quantities { //outputs log likelihood for computing loo
  vector[n] log_lik;
 for (j in 1:n) log_lik[j] = normal_lpdf(y[j] | mu[j], sig);
}
```

The code including an age-year interaction term uses two design matrices – one for cohorts and ages, in the rows and columns of the original dataset, and one for ages and year, the columns and diagonals. That one is later split into the two components, where the column factors are the trend weights applied to the year trends by age. The trend weights are standardized to be positive with a maximum at 1.0. The code here includes the log link and the gamma with variance proportional to mean. It is called lc for Lee-Carter.

```
dmat = as.matrix(read_excel('dsgn_tri_row_col.xlsx'))
coldiag = as.matrix(read_excel('dsgn_tri_col_diag.xlsx'))
y = scan('comp_log_y.txt')
dml = dmat[,-14]
cols = c(1:14)
diags = c(15:28)
diagm = coldiag[,diags]
```

```
wghtm = coldiag[,cols]
n <- ncol(diagm)
u <- ncol(wghtm)
c(n,u)
N <- nrow(dml)
U <- ncol(dml)
c(N,U)
yy = exp(y)
df = list(N=N,U=U,dmat=dml,n=n,u=u,diagm=diagm,wghtm=wghtm,y=yy) #now run stan
set.seed(4)
fit_5 <- stan(file = 'ln_tri_spline_lc.stan', data=df, verbose = FALSE, chains = 1, iter = 100
00, control = list(adapt_delta = 0.94, max_treedepth = 20))
```

This is the Stan code `'ln_tri_spline_lc.stan'`.

```
data {
  int N;           //number of observations
  int U;           //number of variables
  matrix[N,U] dmat;  //row col design matrix with U columns
  int n;           //number of diag params
  int u;           //number of wghts params
  matrix[N,n] diagm;  //diag design matrix
  matrix[N,u] wghtm;  //col wght dsgn mat
  vector[N] y;     //the triangle listed in a column
}
parameters {  // all except v will get uniform prior, which is default
 real<lower=-20, upper=20> lncn;//ln constant term, starting in known range
  vector[U] v;   //row-col parameters
  vector[u] w;  //trend-wghts parameters
  vector[n] d;  //diag parameters
 real<lower=-5, upper = 5> 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;                        //variance in Stan = mean/beta
 real s;                           //shrinkage parameter, like lambda
 vector[N] alpha;                  //fitted alpha parameter vector
 real cn;
 real m;  // max weight
 vector[N] colwghts;
 vector[N] trend;
 vector[N] yhat;
 beta = exp(logbeta);
 s = exp(logs);       // Gives more weight to lower values, which is good if X not big
 cn = exp(lncn);
 colwghts = wghtm*w;
 m = max(colwghts);
 colwghts = exp(colwghts-m); //makes max = 1, all >0
 trend = diagm * d;
 for (j in 1:N) trend[j] = trend[j]*colwghts[j];
 yhat = exp(trend+dmat*v+cn);
 alpha = yhat*beta; //In Stan, mean = a/b
}
model {  // gives priors for those not assumed uniform. Choose this one for lasso.
   for (i in 1:U)  v[i] ~ student_t(2, 0, s);
   for (i in 1:u)  w[i] ~ student_t(2, 0, s);
   for (i in 1:n)  d[i] ~ student_t(2, 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);
}
```