* The simulation results from frequentist methods shows the fixed cubic spline with random intercept has very similar performance as the GAM. The, the Bayesian stan model, I used fixed spline with random intercept first since site-specific spline can be very challenging due to large number of parameters to be estimated.

**Simulation results (1) 200 iteration:**

**Frequentist GAM**

**fitgam <- gam(y ~ A + s(k, site, bs = "fs", k = 5), data = dd, method="REML")**

bias rmse true\_value.se est.se coverage

0.018 0.575 0.576 0.234 59.500

**Bayesian GAM using posterior median**

**brm(y ~ A + s(k, site, bs = "fs", k = 5), data = dd, family = gaussian(), #cores = 4,**

**iter = 2500, warmup = 500, refresh = 0,**

**control = list(adapt\_delta = 0.9))**

bias rmse coverage

0.018 0.574 60.000

**Bayesian GAM using posterior mean**

bias rmse coverage

0.018 0.574 60.000

09/11

Since spline can generate ns and bs, can I just update this step to be the cluster-specific spline and then fit the model, instead of define cluster in the stan code? But before that make sure the stan code with cluster specific spline is working or not.

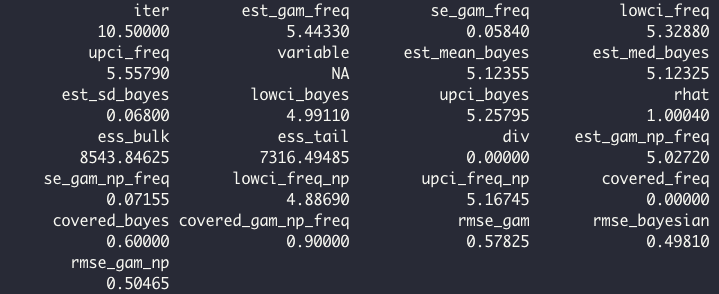
**Start with simple**

1. One\_x\_nonlinear.R: y~nonlinear x + linear A

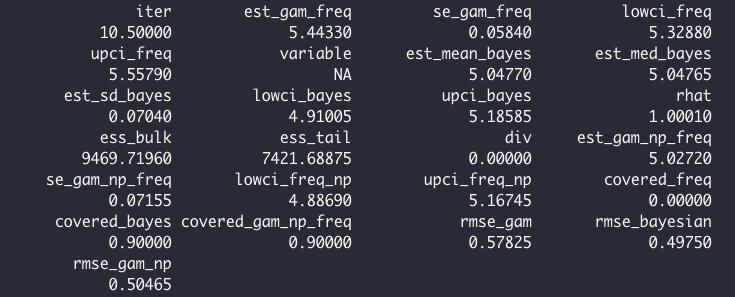
Frequentist gam cannot give a good coverage of 95%CI, repeat 20 times, almost always not covered

1. If fit linear GAM with same spline basis as the Bayesian non penalized version: the estimation is the same; so the problem comes from the penalization of GAM
2. Now we have a penalized Bayesian model and compare with the freqentist GAM, the coverage is better in Bayesian method. See 3. In gitkran

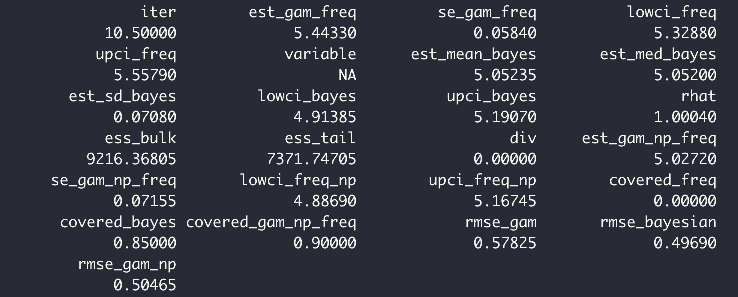
Problem: Bayesian model has similar results as the frequentist version without penalization

1. **Hyperprior Sensitivity**: Check the sensitivity of your results to the hyperpriors you've set (e.g., for **tau**, **sigma**, and **lambda**).
2. Lambda: set a fixed=10, coverage decreased, the other things are similar:
3. 

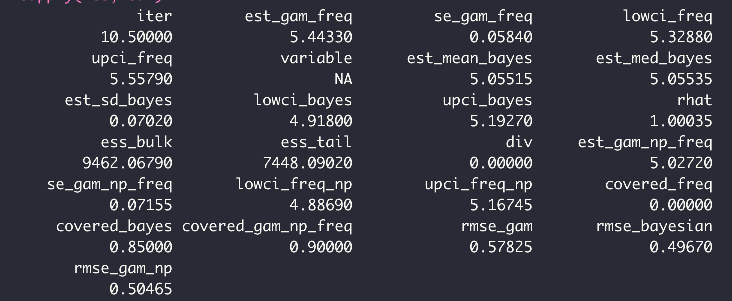
Set as 0:

****

1. **Sigma~cauchy (0,1)**

****

1. **Sigma~cauchy(0,0.25)**



* 9/24/2023
* When fit model: exp(-(x1 - 0.5 + A)^2) + 5\*A
* A screenshot of a computer program

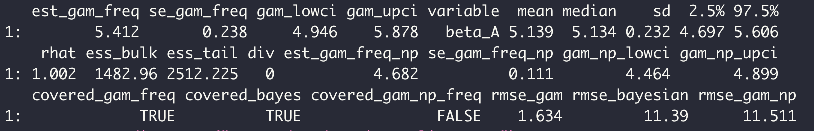
  Description automatically generated
* Coverage: similar, rmse bayesian seems lower
* When fit model: exp(-(x1 - 0.5 + A + x2)^2) + 5\*A
* A screenshot of a computer

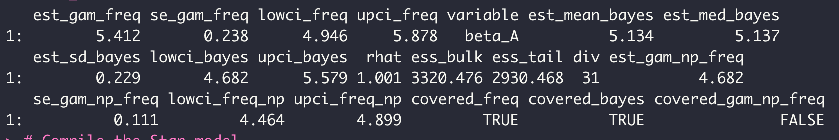
  Description automatically generated
* Performance similar, the div is too high. Consider remove: a[i] = a[i-1] + a\_raw[i]\*tau[i-1];then divergence =0
* A screen shot of a computer code

  Description automatically generated
* The Bayesian performance seems not outperform right, consider basis based on x1\*x2\*A instead of X1+X2+A
* A screenshot of a computer

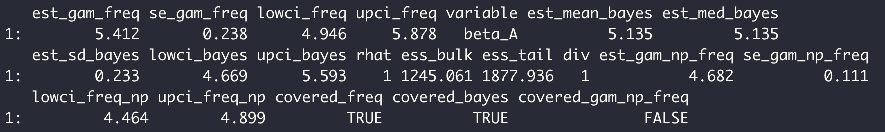
  Description automatically generated
* Back to step wedge:
* No a[i]~N(a[i-1],tau)
* A screen shot of a computer

  Description automatically generated
* Rhat~1,div=0
* With a[i]~N(a[i-1],tau)
* A screenshot of a computer

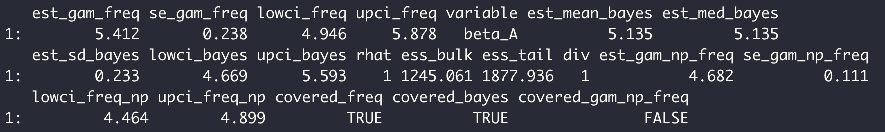
  Description automatically generated
* Rhat~1, div ~20
* 0925
* Step-wedge design
* Stepped\_wedge.r
  + Using normk as basis, trt effect~16
  + Using k as basis, trt effect ~5
  + In stan: if y\_pred\_test[i] = beta\_0[site\_test[i]] + A\_test[i] \* beta\_A + dot\_product(a\_site[site\_test[i]], B\_test[:,i]);
  + y\_pred\_test[i] = beta\_0[site\_test[i]] + A\_test[i] \* beta\_A + dot\_product(a\_site[site\_test[i]], B\_test[:,i]);
  + 
  + Comparing penalized and unpenalized version:
  + Penalized:
  + mu\_a[1] = mu\_a\_raw[1];
  + for (i in 2:num\_basis)
  + mu\_a[i] = mu\_a[i-1] + mu\_a\_raw[i]\*tau;



Nonpenalized:



The penalized one



Leads to very similar results