* 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.

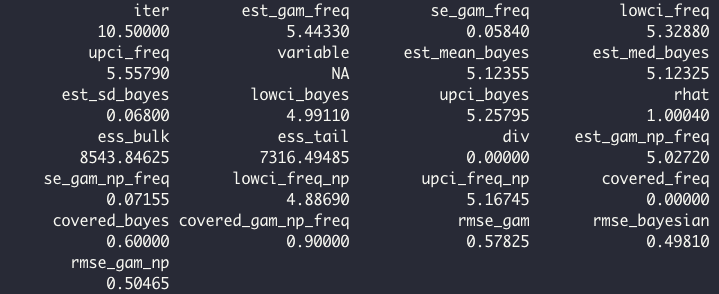
**Start with simple**

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

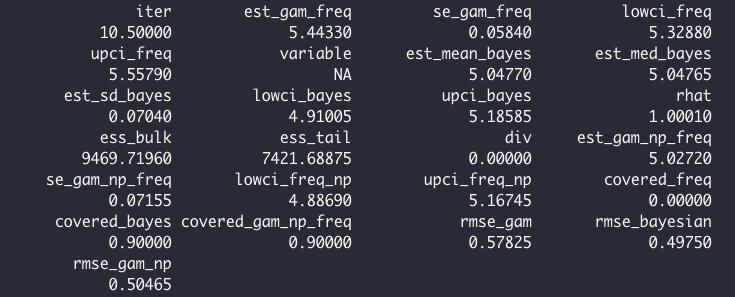
Frequentist gam cannot give a good coverage of 95%CI, repeat 20 times, almost always not covered

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

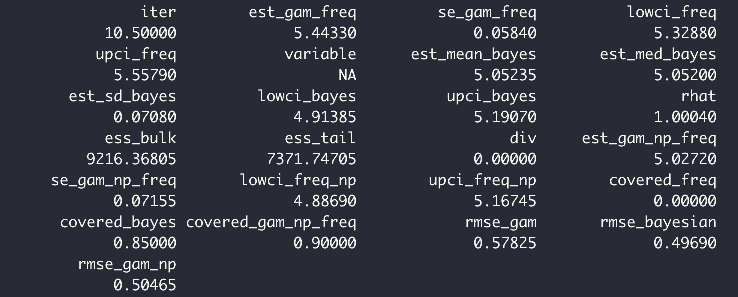
Problem: Bayesian model has similar results as the frequentist version without penalization

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. 

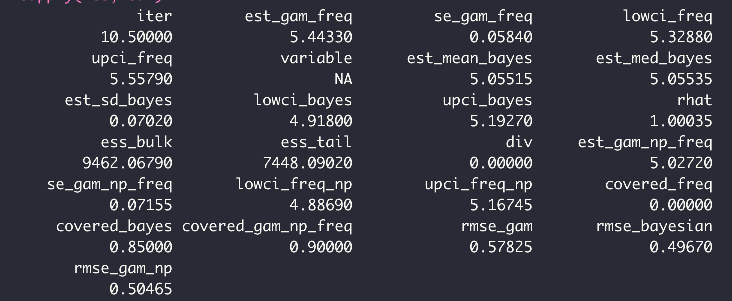
Set as 0:

****

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

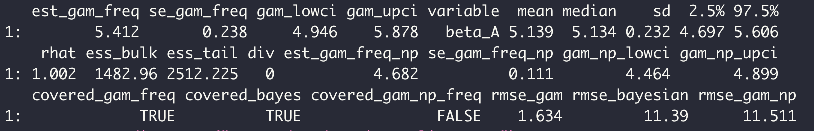
  Description automatically generated
* Coverage: similar, rmse bayesian seems lower
* When fit model: exp(-(x1 - 0.5 + A + x2)^2) + 5\*A
* A screenshot of a computer

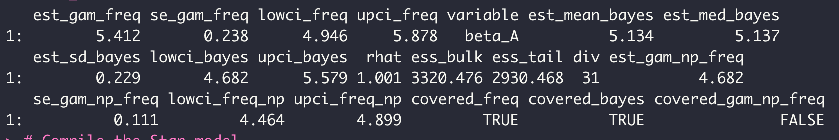
  Description automatically generated
* Performance similar, the div is too high. Consider remove: a[i] = a[i-1] + a\_raw[i]\*tau[i-1];then divergence =0
* A screen shot of a computer code

  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

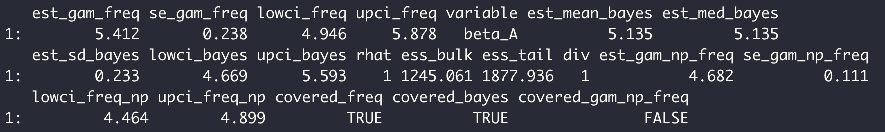
  Description automatically generated
* Back to step wedge:
* No a[i]~N(a[i-1],tau)
* A screen shot of a computer

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

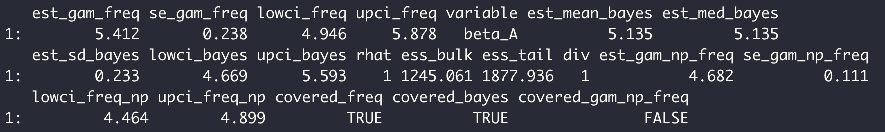
  Description automatically generated
* Rhat~1, div ~20
* 0925
* Step-wedge design
* Stepped\_wedge.r
  + Using normk as basis, trt effect~16
  + Using k as basis, trt effect ~5
  + In stan: if y\_pred\_test[i] = beta\_0[site\_test[i]] + A\_test[i] \* beta\_A + dot\_product(a\_site[site\_test[i]], B\_test[:,i]);
  + 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

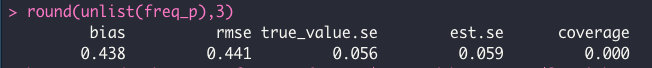


Leads to very similar results

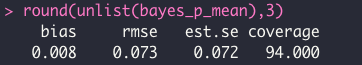
**Nonlinear simple model:** y~N(**exp(-(x1 - 0.5)^2) + 5 \* A,sigma^2=0.25); 200 patients RCT**

**Frequentist gam**

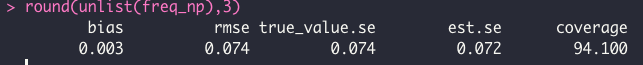
**Gam tends to over-estimate treatment effect**

****

**Bayesian**

****

**Frequentist non penalized**

****

**09/26 stepped wedged simulation across 1000 iterations; 10 clusters, 1 cluster 1 time period switching treatment; cluster sample size=25**

**#cluster-specific intercept**

**def <- defData(varname = "a", formula = 0, variance = 1)**

**#cluster specific time trend**

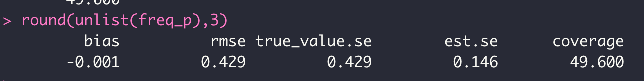
**def2 <- defDataAdd(varname = "b", formula = "(k - 0.5)^2", variance =0.4)**

**#A: trt for each cluster and time period**

**defOut <- defDataAdd(varname = "y", formula = "a + b + 5 \* A", variance = 1)**

GAM:

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



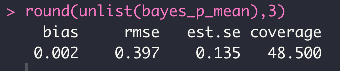
Bayesian model: Y\_hat[i] = beta\_0[site[i]] + beta\_A \* A[i] + dot\_product(a\_site[site[i]], B[:,i])

for (j in 1:num\_basis) {

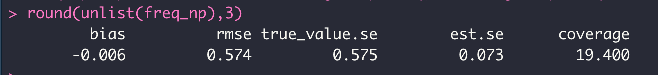
a\_site[:,j] ~ normal(mu\_a[j], sigma\_a);

}

mu\_a ~ normal(0, 2.5);



GAM none penalized



Consider cluster specific treatment effect + cluster specific time spline in the data generation,

def <- defData(varname = "a", formula = 0, variance = 1)

def <- defData(def, varname = "a\_trt", formula = 0, variance = 1)

def2 <- defDataAdd(varname = "b", formula = "(k - 0.5)^2", variance =0.4)

defOut <- defDataAdd(varname = "y", formula = "a + b + (5 + a\_trt) \* A", variance = 1)

**Gam:**

fitgam <- mgcv::gam(y ~ A + s(site, bs = "re") + s(site, A, bs = "re") + s(k, site, bs = "fs", k = 5), data = train\_data, method="REML")

**Bayesian**: hierarchical model with hierarchical structure for treatment effect and spline coefficient

for (i in 1:num\_data) {

Y\_hat[i] = beta\_0[site[i]] + beta\_A\_site[site[i]] \* A[i] + dot\_product(a\_site[site[i]], B[:,i]);//

beta\_A\_site ~ normal(beta\_A, 3);

beta\_A ~ normal(0, 20);

A screenshot of a computer program

Description automatically generated

Increase the number of clusters to 24, 410 iterations

def <- defData(varname = "a", formula = 0, variance = 1)

def <- defData(def, varname = "a\_trt", formula = 0, variance = 4)

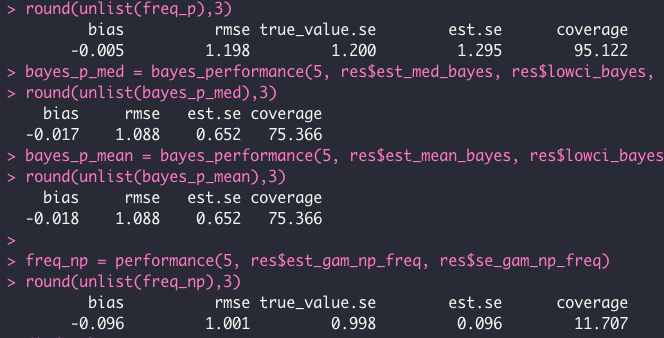
def2 <- defDataAdd(varname = "b", formula = "(k - 0.5)^2", variance =9)

#A: trt for each cluster and time period

defOut <- defDataAdd(varname = "y", formula = "a + b + (5 + a\_trt) \* A", variance = 1)

Fit gam: mgcv::gam(y ~ A + s(site, A, bs = "re") + s(k, site, bs = "fs", k = 5), data = train\_data, method="REML")

Bayesian: Y\_hat[i] = beta\_0 + beta\_A\_site[site[i]] \* A[i] + dot\_product(a\_site[site[i]], B[:,i]);



In frequentist, why do not consider site specific intercept? Reason: reduce the number of parameter to be estimated? In Keith post

What if the shrinkage of spline increased, and no hierarchical treatment effect.

Go back to previous setting, what if add random intercept to the model

To do: try this model in previous hierarchical settings, beta\_A\_site~normal(beta\_A, tau); tau has a prior, beta\_A~normal(0,10)

Next consider beta\_0 site specific or an overall is enough.

09/27: logistic model

Data generation: def <- defData(varname = "a", formula = 0, variance = 1)

def2 <- defDataAdd(varname = "b", formula = "(k - 0.5)^2", variance =0.4)

#A: trt for each cluster and time period

defOut <- defDataAdd(varname = "y", formula = "a + b + 5 \* A", dist = "binary", link="logit")

fit the Bayesian model: for (i in 1:num\_data) {

Y\_hat[i] = beta\_0[site[i]] + beta\_A \* A[i] + dot\_product(a\_site[site[i]], B[:,i]);//B[:,i] will give you a vector that consists of all elements from the ith column of that matrix.

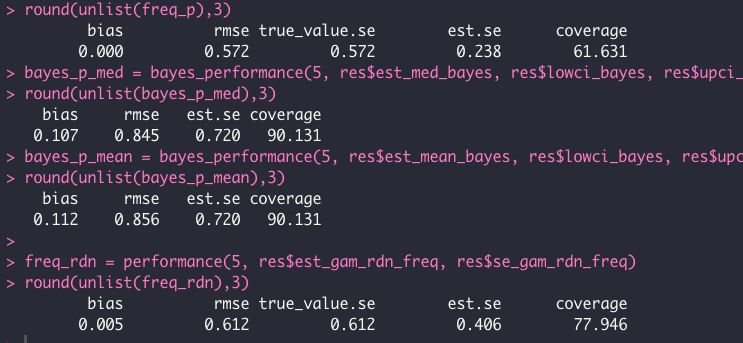
}

Divergence too high

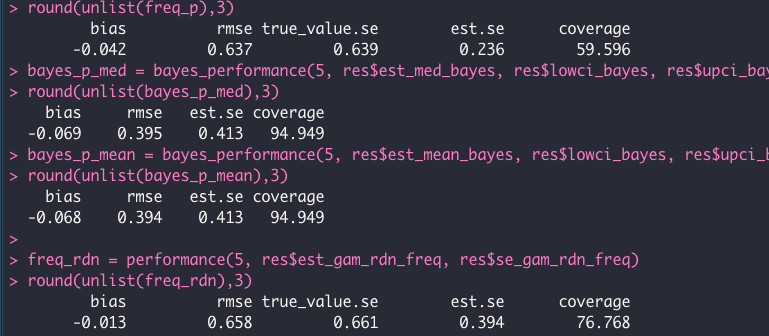
1. See if use norm k would help: y outcome distribution becomes reasonable, but the performance is poor for Bayesian model fitting

09/28 Bayesian hierarchical model vs. frequentist: here is just 10 sites

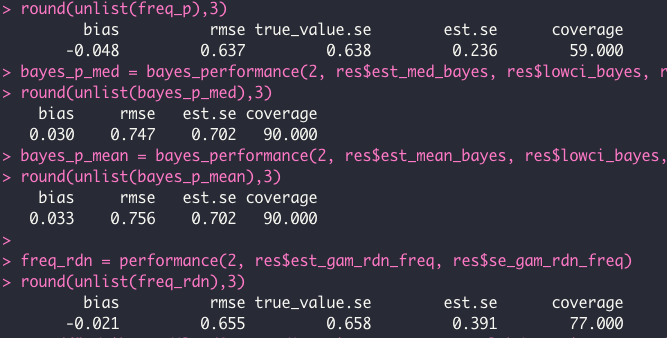
1000 iterations



100 iterations



Now, I change the prior a little bit and set true treatment effect as 2:



Let’s think of the possibilities of reduce bias:

1. Increasing the chain length: default for chain length for stan is 2000(includes 1000 warm-up), 4 chains. Now I use 2500 per chain, 500 warm up, not much difference

30 sites, now we do, 200 iterations