Bayesian Gaussian Process Regression

STAT8810, Fall 2017

M.T. Pratola

October 7, 2017

Today

Bayesian Gaussian Process Regression

Bayesian GP Regression

- Recall we had observations from our expensive simulator, $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n)$, where $\mathbf{x} \in \chi \subset \mathbb{R}^d$.
- We assumed the data is observed error-free, $y(\mathbf{x}) = z(\mathbf{x})$ and we also assumed that the simulator output is a realization of a GP, $z(\mathbf{x}) \sim GP(0, c(\cdot; \lambda^{-1}, \boldsymbol{\rho}))$.
- Our modeling assumption means that

$$\mathbf{y} \sim N(0, \lambda^{-1}\mathbf{R}(\boldsymbol{\rho})).$$

  For simplicity, we will assume our correlation matrix is defined using the separable Gaussian correlation function,

$$Cor(\mathbf{x}, \mathbf{x}'; \rho) = \prod_{i=1}^{d} \rho_i^{(x_i - x_i')^2}.$$

- So our likelihood function is

$$L(\mathbf{y}|\lambda^{-1}, \boldsymbol{\rho}) = \frac{\lambda^{n/2}}{\sqrt{2\pi}det(\mathbf{R})^{1/2}} exp\left(-\frac{\lambda}{2}\mathbf{y}^T\mathbf{R}^{-1}\mathbf{y}\right)$$

Bayesian GP Regression

- To complete our Bayesian model, we need to specify prior distributions on the parameters λ and $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d)$. We will assume $\pi(\lambda, \boldsymbol{\rho}) = \pi(\lambda)\pi(\boldsymbol{\rho})$.
- Taking advantage of conjugacy, we will use a Gamma prior for the precision,

$$\lambda \sim \text{Gamma}(a, b),$$

which means

$$\pi(\lambda) \propto \lambda^{a-1} exp\left(-b\lambda\right).$$

Bayesian GP Regression

- The correlation parameters have no conjugate form (which means we will have to resort to the Metropolis-Hastings sampler for these parameters).
- However we will make a further simplifying assumption of prior independence of the correlation parameters, $\pi(\boldsymbol{\rho}) = \prod_{i=1}^{d} \pi(\rho_i)$.
- Since each $\rho_i \in (0, 1)$, an appropriate prior to use is the Beta distribution,

$$\rho_i \sim \text{Beta}(\alpha, \beta),$$

which means

$$\pi(\rho_i) \propto \rho_i^{\alpha-1}(1-\rho_i)^{\beta-1}.$$

Bayesian GP Regression

- We are interested in the posterior distribution,

$$\pi(\lambda, \boldsymbol{\rho}|\mathbf{y}) \propto L(\lambda, \boldsymbol{\rho}|\mathbf{y})\pi(\lambda) \prod_{i=1}^{d} \pi(\rho_i),$$

  and the posterior predictive distribution,

$$\pi(z(\mathbf{x})|\mathbf{y}) = \int_{\lambda,\boldsymbol{\rho}} \pi(z(\mathbf{x})|\lambda, \boldsymbol{\rho}, \mathbf{y})\pi(\lambda, \boldsymbol{\rho}|\mathbf{y}) d\lambda d\boldsymbol{\rho}.$$

- Lets start with the posterior...

Bayesian GP Regression: The Posterior

- Sampling from $\pi(\lambda, \boldsymbol{\rho}|\mathbf{y})$ in closed form is not possible.
- However, we can get samples from $\pi(\lambda|\boldsymbol{\rho}, \mathbf{y})$ in closed form due to our conjugate prior for the precision.
- And we can get approximate samples from $\pi(\rho_i|\lambda, \boldsymbol{\rho}_{-i}, \mathbf{y})$ using the Metropolis-Hastings algorithm.
- This will define a Metropolis-within-Gibbs sampler, which will give us an algorithm that returns approximate draws from our posterior.

Bayesian GP Regression: Precision Parameter

- One can show† the full conditional for the precision is

$$\pi(\lambda|\boldsymbol{\rho}, \mathbf{y}) \sim \text{Gamma}\left(a + \frac{n}{2}, b + \frac{1}{2}\mathbf{y}^T\mathbf{R}^{-1}\mathbf{y}\right).$$

† an often used term in the scientific vernacular for "you should show".

Bayesian GP Regression: Correlation Parameter

- For the correlation parameters, we will need a proposal distribution, $q(\rho_j \to \rho_j')$.
- A uniform proposal with width δ_j has seemingly worked well for me,

$$q(\rho_j \to \rho_j') = \text{Uniform}(\rho_j - \frac{\delta_j}{2}, \rho_j + \frac{\delta_j}{2}).$$

  - δ will be an important tuning parameter in order to get our algorithm to work well.
- In order to calculate the Metropolis-Hastings acceptance probability for ρ_i', we need the part of the posterior that is relevant, namely,

$$\pi(\rho_i|\lambda, \boldsymbol{\rho}_{-i}, \mathbf{y}) \propto \frac{\lambda/2}{det(\mathbf{R})^{1/2}} exp\left(-\frac{\lambda}{2}\mathbf{y}^T\mathbf{R}^{-1}\mathbf{y}\right)\rho_j^{a-1}(1-\rho_j)^{b-1}$$

Metropolis-within-Gibbs Sampler

- So at iteration t, our posterior sampling algorithm will look like the following:

1. Set $\boldsymbol{\rho}^{(t)} = \boldsymbol{\rho}^{(t-1)}$.
2. For $j = 1, \ldots, d$ draw $\rho_j^{(t)}$:
   - Draw $\rho_j' \sim \text{Uniform}\left(\rho_j^{(t)} - \frac{\delta}{2}, \rho_j^{(t)} + \frac{\delta}{2}\right)$
   - †Calculate
   $$\alpha = min\left(1, \frac{\pi(\rho_i'|\lambda, \boldsymbol{\rho}_{-i}^{(t)}, \mathbf{y})}{\pi(\rho_i^{(t)}|\lambda, \boldsymbol{\rho}_{-i}^{(t)}, \mathbf{y})}\right).$$
   - Draw $u \sim \text{Uniform}(0, 1)$
   - If $u < \alpha$ then accept $\rho_i^{(t)} = \rho_i'$.
3. Draw $\lambda^{(t)} \sim \text{Gamma}\left(a + \frac{n}{2} + \frac{1}{2}\mathbf{y}^T\mathbf{R}^{-1}(\boldsymbol{\rho}^{(t)})\mathbf{y}\right)$
4. Repeat steps (1)-(3) until convergence.

† The ratio involving q does not appear here since the q we are using is a symmetric proposal and so cancels.

Metropolis-within-Gibbs Sampler

- How to choose the δ_j's?
  - Trial and error.
  - Adaptive techniques to target a desired acceptance rate of 44%- we will see shortly.

Example

```
regress<-function(y,l.dez,N,pi,mh,last=1000,adapt=TRUE)
{
  n=length(y)
  k=length(l.dez)
  draw.lambdaz=rep(NA,N)
  draw.rhoz=matrix(NA,nrow=N,ncol=k)
  rr=rep(mh$rr,k)
  # initial guesses
  draw.lambdaz[1]=1/var(y)
  draw.rhoz[1,]=rep(.5,k)
  #accept/reject ratio trackers
  accept.rhoz=rep(0,k)

  lastadapt=0
  In=diag(n)
  one=rep(1,n)
  rhoz.new=rep(NA,k)
```

Example

```
for(i in 2:N)
{
  # Draw the correlation parameters (Metropolis-Hastings
  draw.rhoz[i,]=draw.rhoz[i-1,]
  for(j in 1:k)
  {
    rhoz.new=draw.rhoz[i,]
    rhoz.new[j]=runif(1,draw.rhoz[i-1,j]-rr[j],
                      draw.rhoz[i-1,j]+rr[j])
    a=min(0,logp(y,draw.lambdaz[i-1],rhoz.new,l.dez,pi)
      -logp(y,draw.lambdaz[i-1],draw.rhoz[i,],l.dez,pi))
    if(log(runif(1))<a)
    {
      draw.rhoz[i,j]=rhoz.new[j]
      accept.rhoz[j]=accept.rhoz[j]+1
    }
  }
```

Example

```
# Draw the marginal precision of the data (Gibbs step)
Rz=rhogeodacecormat(l.dez,draw.rhoz[i,])$R
Rz=Rz+In*1e-14  #cheat
cholRz=chol(Rz)
Rzinv=chol2inv(cholRz)
draw.lambdaz[i]=rgamma(1,pi$az+n/2,pi$bz+
                       0.5*t(y)%*%Rzinv%*%y)
```

Example

```
# Adaptive MCMC:
# adapt the proposal step every N/20 iterations,
# but only for the first 50% of the iterations
if(adapt && i%%(N/20)==0 && i<(N*.5+1))
{
  rate.rhoz=accept.rhoz/(i-lastadapt)
  cat("Adapting rates from ",rate.rhoz,"\n");
  for(j in 1:k)
    if(rate.rhoz[j]>.49 || rate.rhoz[j]<.39)
        rr[j]=rr[j]*rate.rhoz[j]/.44
  lastadapt=i
  accept.rhoz=rep(0,k)
}
```

Example

```
source("dace.sim.r")

# Generate response:
set.seed(88)
n=5; k=1; rhotrue=0.2; lambdatrue=1
design=as.matrix(runif(n))
l1=list(m1=outer(design[,1],design[,1],"-"))
l.dez=list(l1=l1)
R=rhogeodacecormat(l.dez,c(rhotrue))$R
L=t(chol(R))
u=rnorm(nrow(R))
z=L%*%u

# Our observed data:
y=z
```

Example

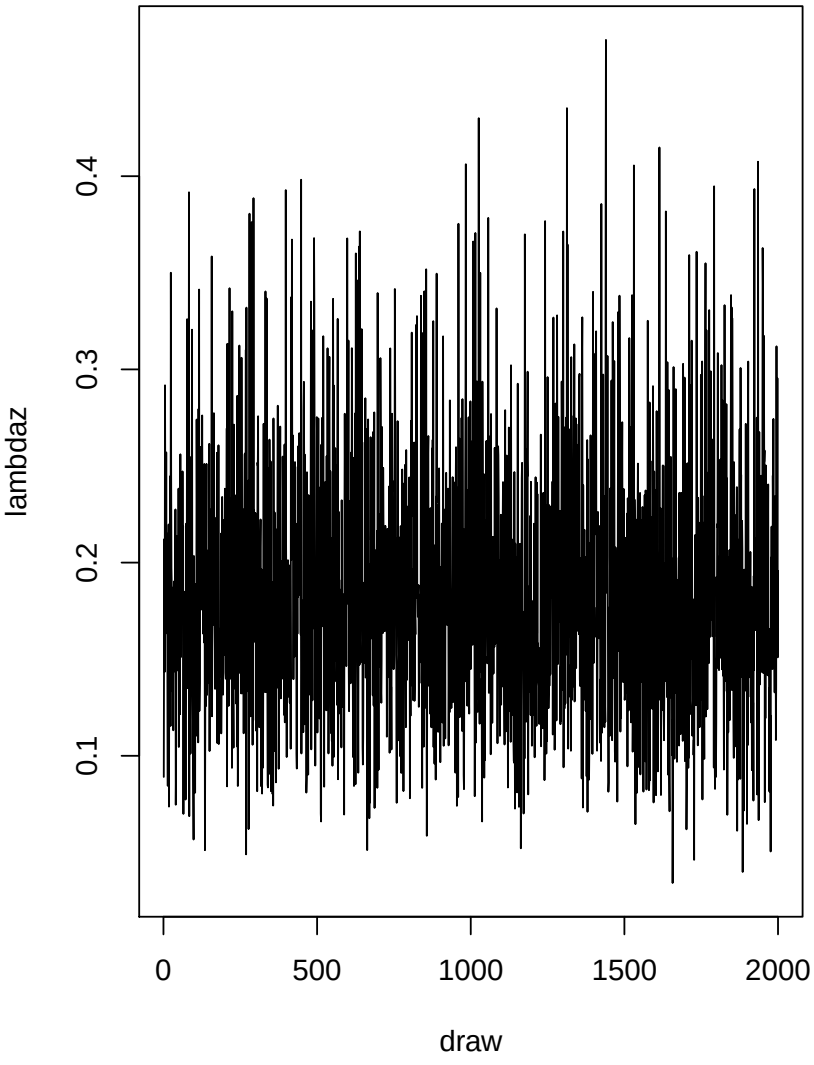
```
# Now fit the GP model
source("regression.r")
pi=list(az=5,bz=5,rhoa=rep(1,k),rhob=rep(5,k))
mh=list(rr=1e-5)
fit=regress(y,l.dez,5000,pi,mh,last=2000,adapt=FALSE)
```

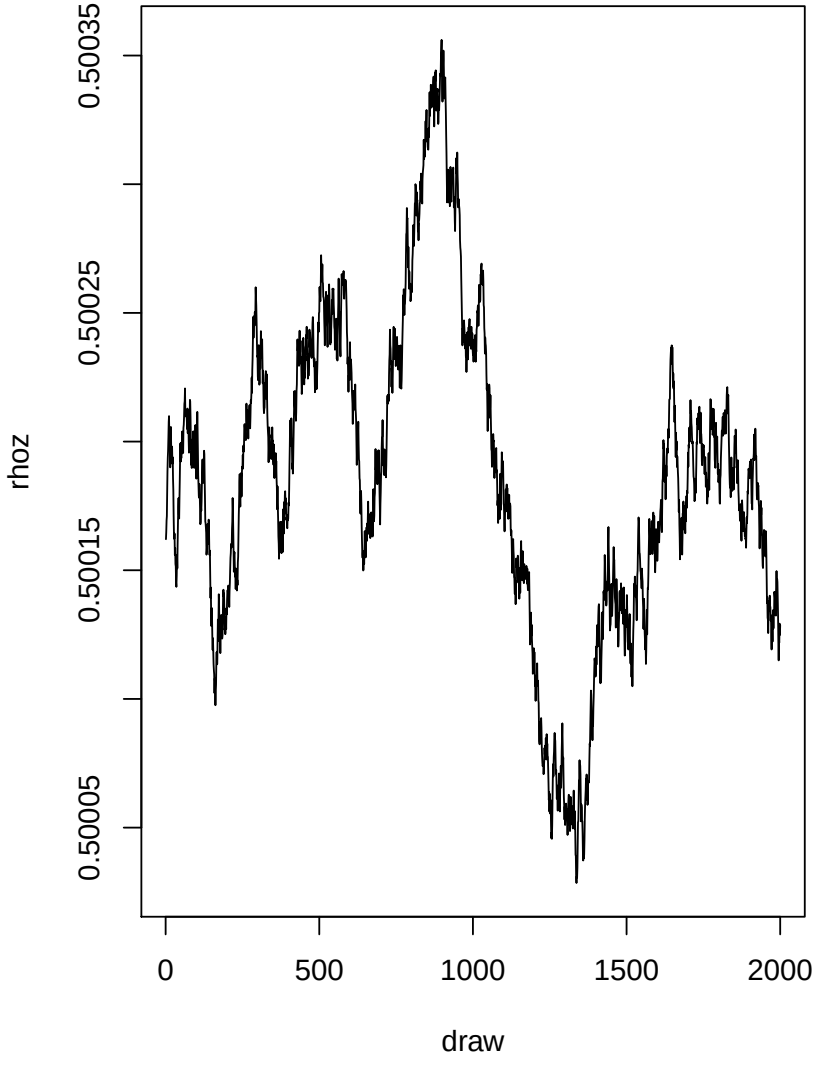
```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  1999  samples from the posterior will be repo
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  ----------------------------------------------------
##
##
## 0.04  percent complete
0.06  percent complete
0.08  percent complete
```

Example

```
par(mfrow=c(1,2))
plot(fit$lambdaz,type='l',xlab="draw",
     ylab="lambdaz")
abline(h=lambdatrue)
plot(fit$rhoz,type='l',xlab="draw",
     ylab="rhoz")
abline(h=rhotrue)
```

Example





MH acceptance rate for ρ: 0.9996

Example

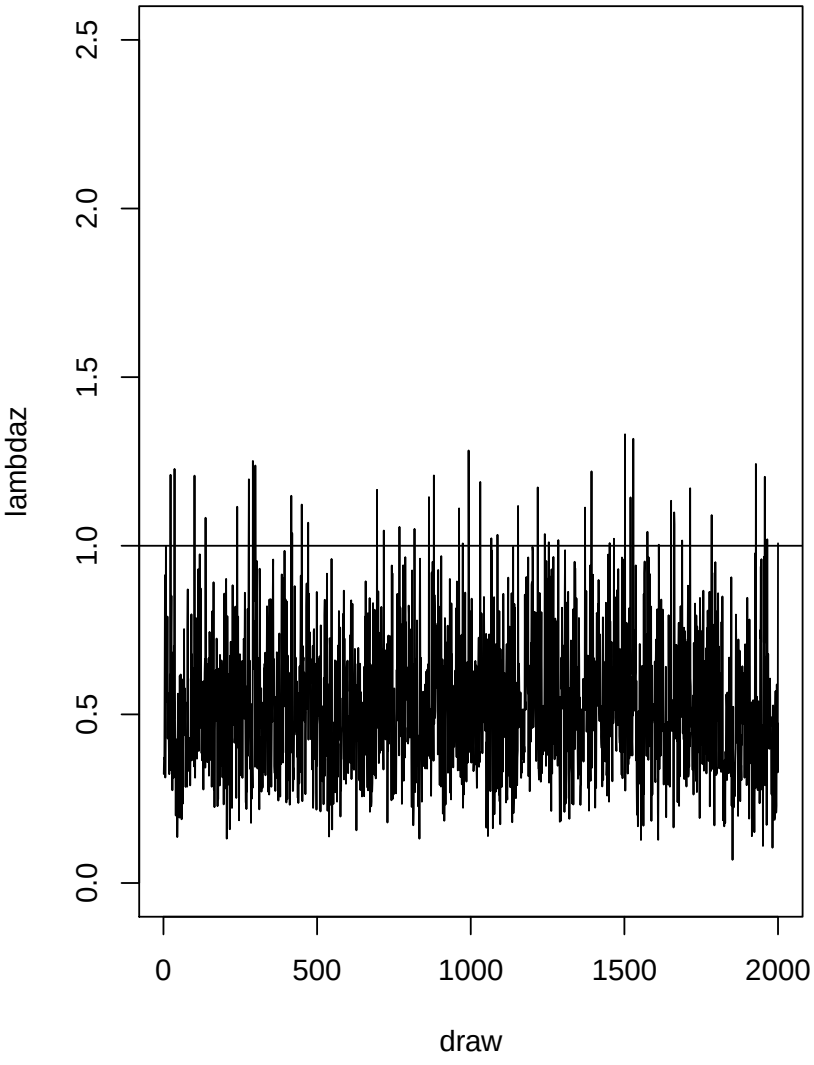
```
# Try different value for proposal distribution
mh=list(rr=0.9)
fit=regress(y,l.dez,5000,pi,mh,last=2000,adapt=FALSE)
```

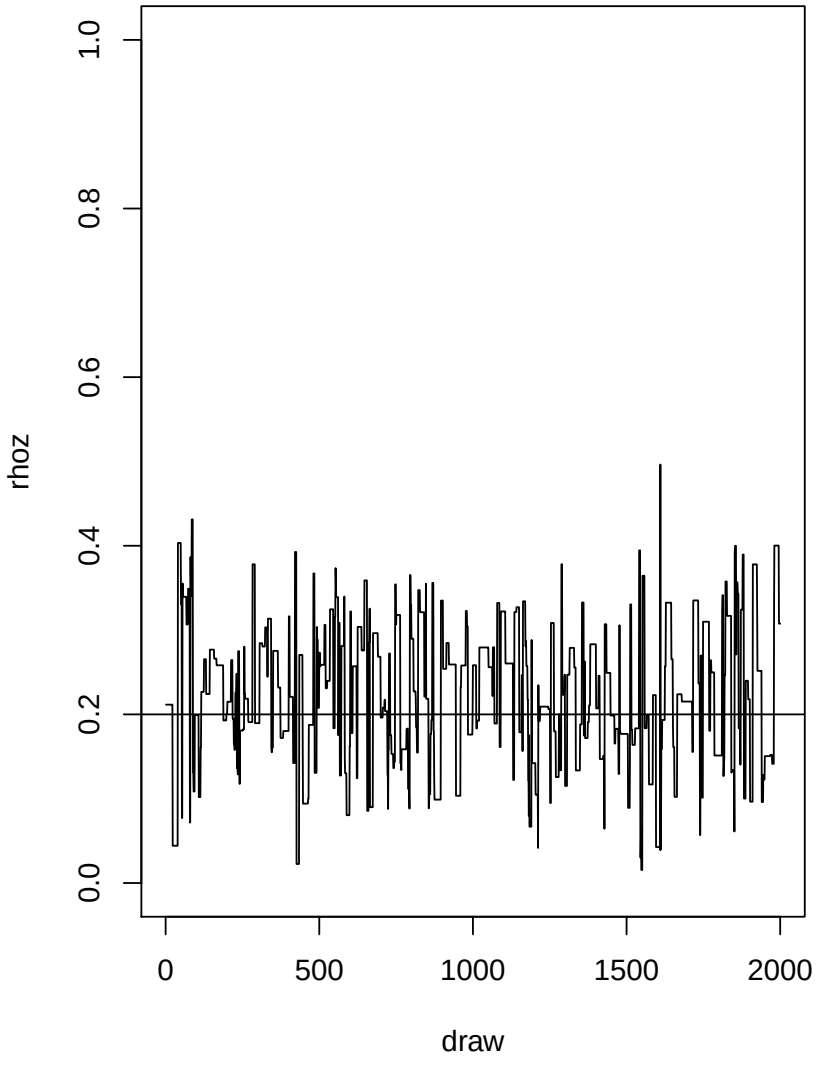
```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  1999  samples from the posterior will be repo
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  ----------------------------------------------------
##
##
## 0.04  percent complete
0.06  percent complete
0.08  percent complete
0.1  percent complete
0.12  percent complete
```

Example

```
par(mfrow=c(1,2))
plot(fit$lambdaz,type='l',xlab="draw",ylab="lambdaz",
     ylim=c(0,2.5))
abline(h=lambdatrue)
plot(fit$rhoz,type='l',xlab="draw",ylab="rhoz",
     ylim=c(0,1))
abline(h=rhotrue)
```

Example





MH acceptance rate for ρ: 0.1418

Example

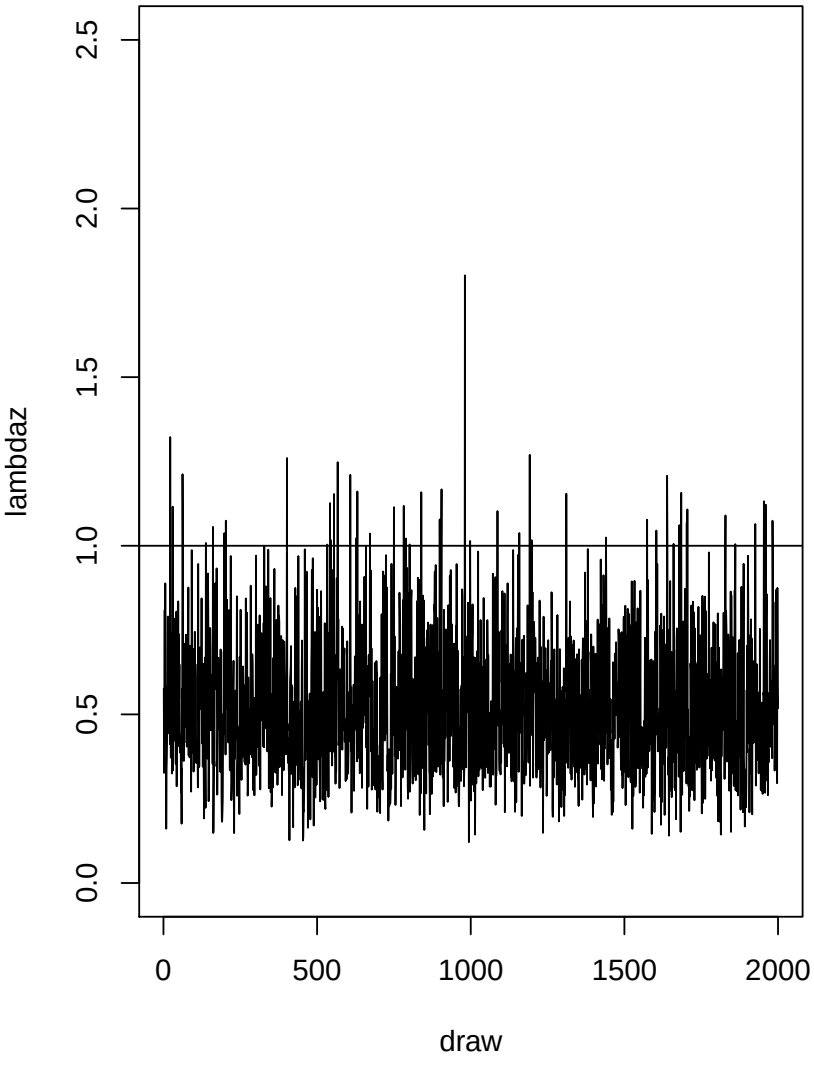
```
# Try yet another value for proposal distribution
mh=list(rr=0.27)
fit=regress(y,l.dez,5000,pi,mh,last=2000,adapt=FALSE)
```

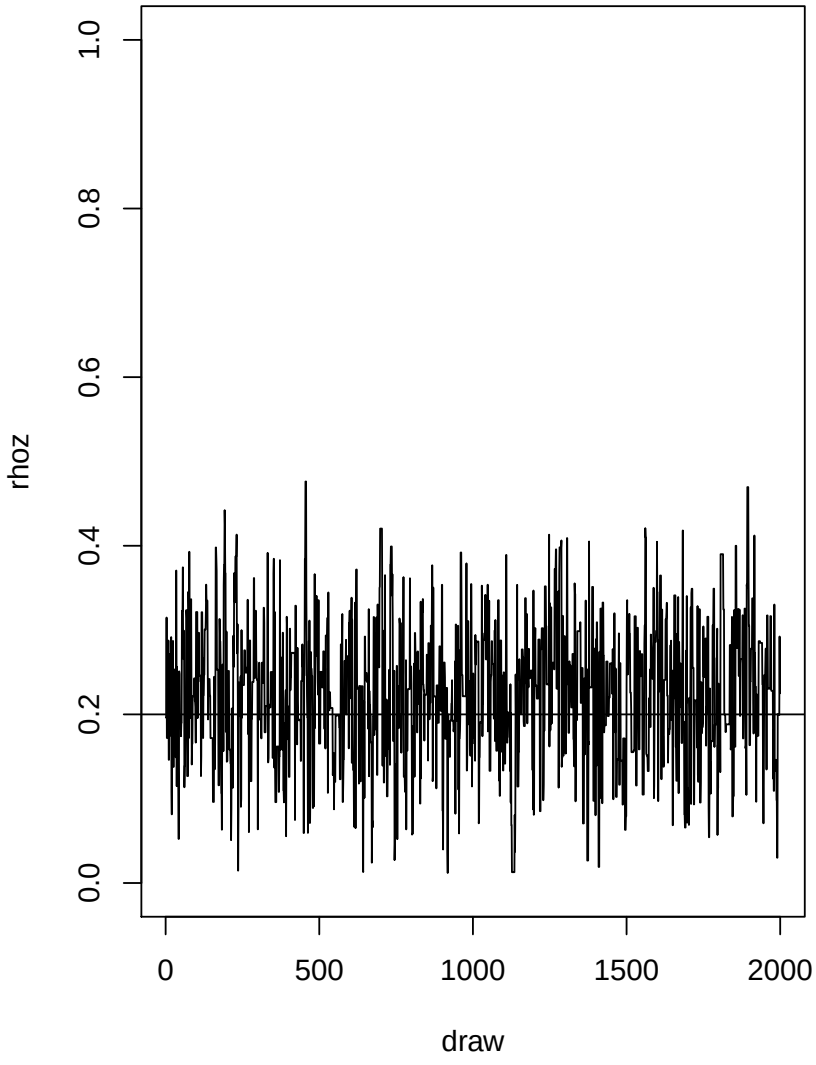
```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  1999  samples from the posterior will be repo
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  ----------------------------------------------------
##
##
## 0.04  percent complete
0.06  percent complete
0.08  percent complete
0.1  percent complete
0.12  percent complete
```

Example

```
par(mfrow=c(1,2))
plot(fit$lambdaz,type='l',xlab="draw",ylab="lambdaz",
     ylim=c(0,2.5))
abline(h=lambdatrue)
plot(fit$rhoz,type='l',xlab="draw",ylab="rhoz",
     ylim=c(0,1))
abline(h=rhotrue)
```

Example





MH acceptance rate for ρ: 0.4438

Adaptive MCMC

- Tuning the proposal distribution by hand can be difficult and time-consuming, *especially* when we have many parameters being updated by Metropolis-Hastings steps.
- When the proposal distribution has large variance (is "wide") then the sampler can explore large areas of the parameter space
  - but much of this space does not have high probability under the posterior
  - so proposals will be rejected
  - which means we rarely accept proposals
- When the proposal distribution has small variance (is "narrow") then the sampler can explore in a direction that is improves the chances of being accepted
  - so proposals are almost always accepted
  - but it is very slow to explore the parameter space and reach the region of high posterior probability
  - and samples will be highly correlated even if it does reach the right region of parameter space

Adaptive MCMC

- Adaptive MCMC seeks to automatically tune the proposal distribution (here, δ) to get good "mixing" of the Markov Chain.
- If r is the proportion of accepted proposals after m iterations, update δ as

$$\delta' = \frac{r}{0.44}\delta.$$

- This seeks to adjust δ so that the acceptance rate will be closer to an idealized rate of 0.44, or 44%.
- The 44% target is a theoretical result† for a 1-dimensional parameter. For a higher-dimensional parameter, the theoretical target decreases (e.g. $\sim$23% for dimensions ≥ 5).

† For a Normal random walk proposal distribution and a Normal target density, the optimal proposal width is one that gives an acceptance rate of 44% for a 1-dimensional parameter.

Adaptive MCMC

- This automatic adaptive trick is very helpful, but it's usage can be easily abused†.
- For instance, one cannot adapt at every iteration. Rather adapt every once in awhile, the update the acceptance rate.
- Once it looks like we have reached a good acceptance rate, stop adapting the proposal. All the draws up to this point will be thrown out.
- Now that we have a good δ, run the chain forward until burn-in and then save as many posterior draws as we desire.

† See Jeff Rosenthal's "adapt" example of adaptive MCMC gone wrong at `http://www.probability.ca/jeff/java/`.

Example

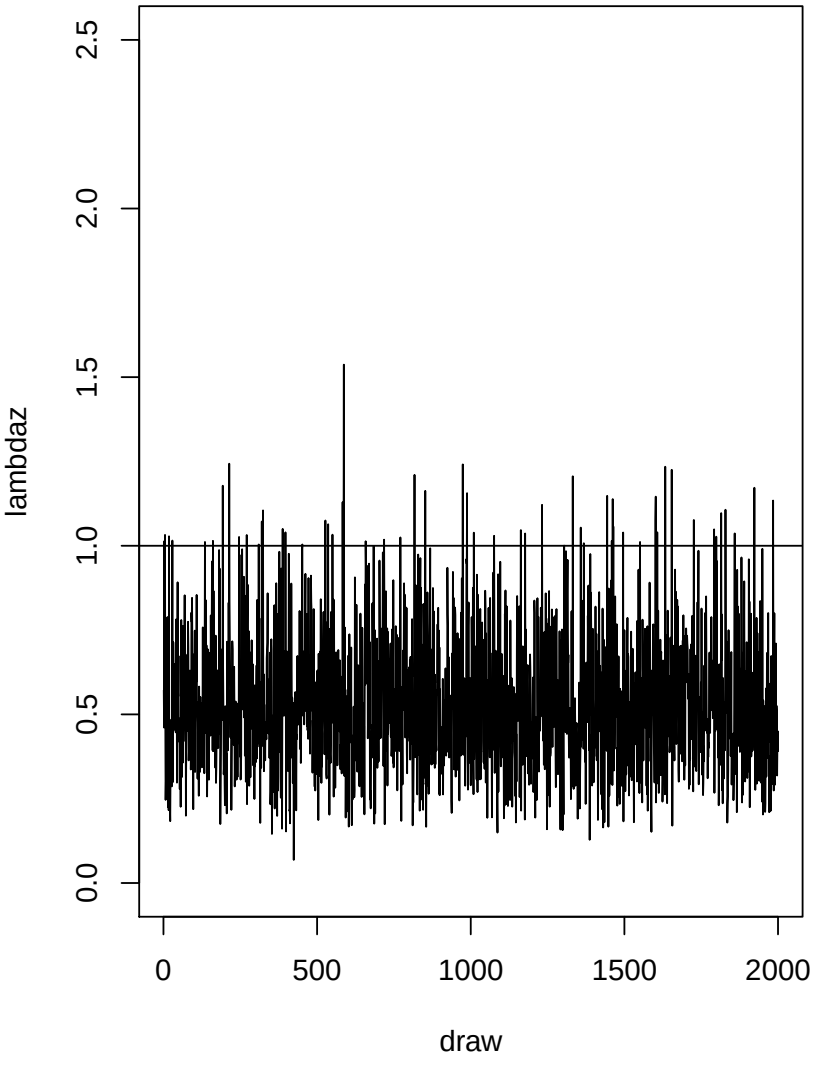
```
# Try adaptive method
mh=list(rr=.05)
fit=regress(y,1.dez,5000,pi,mh,last=2000,adapt=TRUE)
```

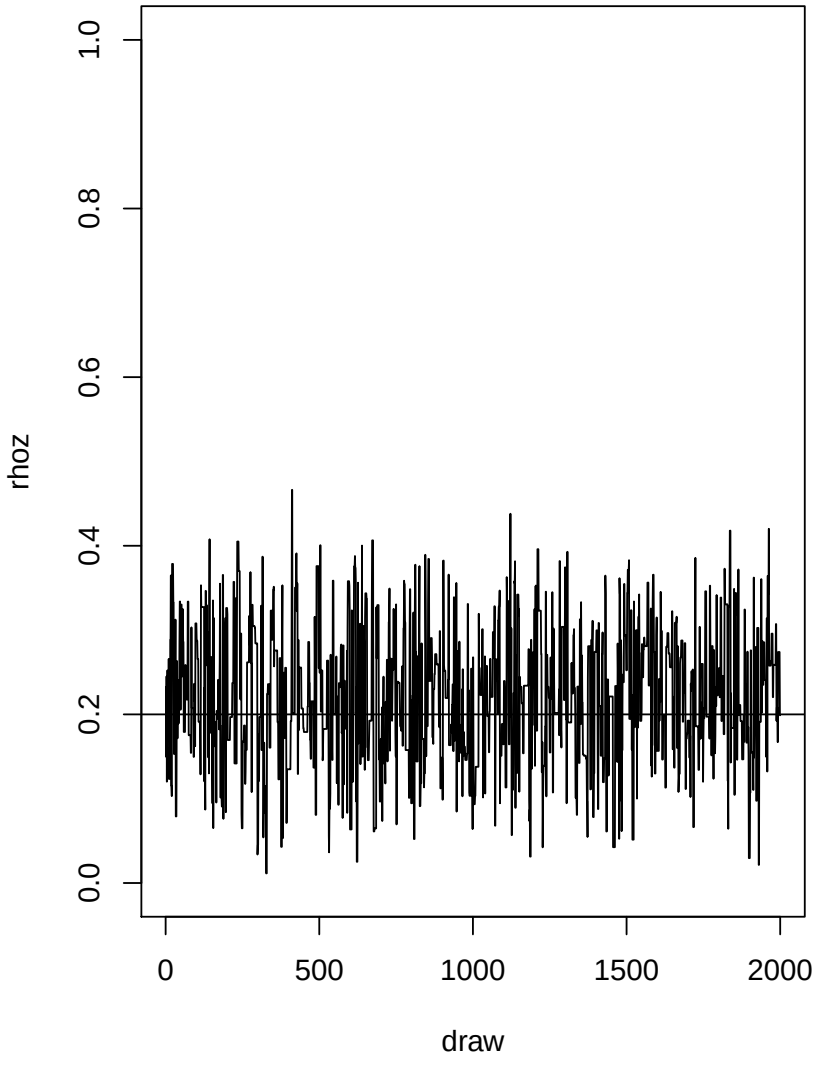
```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  1999  samples from the posterior will be repo
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  ----------------------------------------------------
##
##
## 0.04  percent complete
0.06  percent complete
0.08  percent complete
0.1  percent complete
0.12  percent complete
```

Example

```
par(mfrow=c(1,2))
plot(fit$lambdaz,type='l',xlab="draw",ylab="lambdaz",
     ylim=c(0,2.5))
abline(h=lambdatrue)
plot(fit$rhoz,type='l',xlab="draw",ylab="rhoz",
     ylim=c(0,1))
abline(h=rhotrue)
```

Example





MH acceptance rate for ρ: 0.428

Using our Posterior Samples

- Once we have draws from our posterior distribution for some parameter(s), say $\theta_1, \ldots, \theta_N$, we can use these to approximate statistics of interest.
- For example, say we are interested in an estimator of some function $t(\theta)$. The ergodic average is

$$\bar{t}_N = \frac{1}{N} \sum_{i=1}^{N} t(\theta_i).$$

If the Markov Chain is ergodic and $E[t(\theta)] < \infty]$ then $\bar{t}_N \to E[t(\theta)]$ a.s. as $N \to \infty$.

  - this is the Markov Chain equivalent of the Strong Law of Large Numbers.

Using our Posterior Samples

- Some typical examples of $t(\theta)$'s:
  - $t(\theta) = \theta \Rightarrow \frac{1}{N}\sum \theta_i \rightarrow E[\theta]$
  - $t(\theta) = \mathcal{I}(\theta \leq u) \Rightarrow \frac{1}{N}\sum \mathcal{I}(\theta_i \leq u) \rightarrow P(\theta \leq u)$.
  - A $100(1-\alpha)\%$ credible interval for $t(\theta)$ is $[\mathbf{t}_{\alpha N/2}, \mathbf{t}_{N(1-\alpha/2)}]$ where $\mathbf{t} = (t(\theta_{\sigma(1)}), \ldots, t(\theta_{\sigma(N)}))$ where $\theta_{\sigma(1)} < \theta_{\sigma(2)} < \cdots < \theta_{=}\sigma(N)$ (easy to get using R's `quantile` function).
  - etc.

Bayesian GP Predictions

- Given our samples from the posterior distribution, $\pi(\lambda, \boldsymbol{\rho}|\mathbf{y})$, how do we form predictions?
- Would like to use the posterior predictive distribution,

$$\pi(z(\mathbf{x})|\mathbf{y}) = \int_{\lambda,\boldsymbol{\rho}} \pi(z(\mathbf{x})|\lambda, \boldsymbol{\rho}, \mathbf{y})\pi(\lambda, \boldsymbol{\rho}|\mathbf{y}) d\lambda d\boldsymbol{\rho}$$

  - we know it will be hopeless to do this integration in closed-form since we don't even have $\pi(\lambda, \boldsymbol{\rho}|\mathbf{y})$ in closed form.
  - so we approximate it numerically:

$$\pi(z(\mathbf{x})|\mathbf{y}) \approx \sum_{i=1}^{N} \pi(z(\mathbf{x})|\lambda^{(i)}, \boldsymbol{\rho}^{(i)})$$

  where N is the number of samples saved from the posterior distribution during our MCMC algorithm.

Bayesian GP Predictions

- The quantity $\sum_{i=1}^{N} \pi(z(\mathbf{x})|\lambda^{(i)}, \boldsymbol{\rho}^{(i)})$ is rarely of interest in and of itself. Rather we are interested in predictive realizations of our process.
- What if we are interested in only the mean? Recall $E[z(\mathbf{x})|\lambda, \boldsymbol{\rho}, \mathbf{y}] = \mathbf{r}^T \mathbf{R}^{-1} \mathbf{y}$, the mean of the predictive distribution $\pi(z(\mathbf{x})|\lambda, \boldsymbol{\rho}, \mathbf{y})$.
- Then we can get the posterior mean as

$$\begin{aligned} E[z(\mathbf{x})|\mathbf{y}] &= \int_{\lambda,\boldsymbol{\rho}} z(\mathbf{x}) \pi(z(\mathbf{x})|\lambda, \boldsymbol{\rho}, \mathbf{y}) \pi(\lambda, \boldsymbol{\rho}|\mathbf{y}) \\ &\approx \sum_{i=1}^{N} \mathbf{r}(\lambda_i, \boldsymbol{\rho}_i)^T \mathbf{R}^{-1}(\lambda_i, \boldsymbol{\rho}_i) \mathbf{y} \end{aligned}$$

- This is equivalent to our simpler examples where $t(z(\mathbf{x}) = E[z(\mathbf{x})]$. But what about the uncertainties?

Bayesian GP Posterior Predictive Distribution

- To get the *full* posterior predictive distribution, we simply take $t = z(\mathbf{x})$. In other words, calculate

$$z_i(\mathbf{x}) \sim \pi(z(\mathbf{x})|\lambda_i, \boldsymbol{\rho}_i, \mathbf{y})$$

  for $i = 1, \ldots, N$
- The resulting realizations $z_1, \ldots, z_N$ are (approximate) draws from the posterior predictive distribution, $\pi(z(\mathbf{x})|\mathbf{y})$.
- We can calculate the posterior predictive mean as

$$E[z(\mathbf{x})|\mathbf{y}] \approx \frac{1}{N} \sum_{i=1}^{N} z_i(\mathbf{x}).$$

- Similarly, we can calculate pointwise quantiles or standard deviations to arrive at credible intervals of the posterior predictive distribution (the uncertainties we want!)
- One could also calculate other functions, $t(\cdot)$, of the z_i's.

Example

```
predict<-function(l.v,fit,eps=1e-10)
{
  n=length(fit$y)  # num observations
  m=nrow(l.v[[1]])-n  # num prediction points
  N=length(fit$lambdaz)

  draw.preds=matrix(0,nrow=N,ncol=m)
```

Example

```
for(i in 1:N) {
  Rall=rhogeodacecormat(l.v,fit$rhoz[i,])$R
  # Extract the sub-matrices we need
  Ryy=Rall[1:n,1:n]
  Rgg=Rall[(n+1):(n+m),(n+1):(n+m)]
  Rgy=Rall[(n+1):(n+m),1:n]
  Ryy.inv=chol2inv(chol(Ryy))

  # Mean of conditional distribution:
  m.cond=Rgy%*%Ryy.inv%*%y
  # Covariance of conditional distribution:
  E.cond=fit$lambdaz[i]^(-1)*(Rgg-Rgy%*%Ryy.inv%*%t(Rgy))

  # Let's generate a realization!
  L=t(chol(E.cond+diag(ng)*eps))
  u=rnorm(ng)
  draw.preds[i,]=m.cond + L%*%u
```

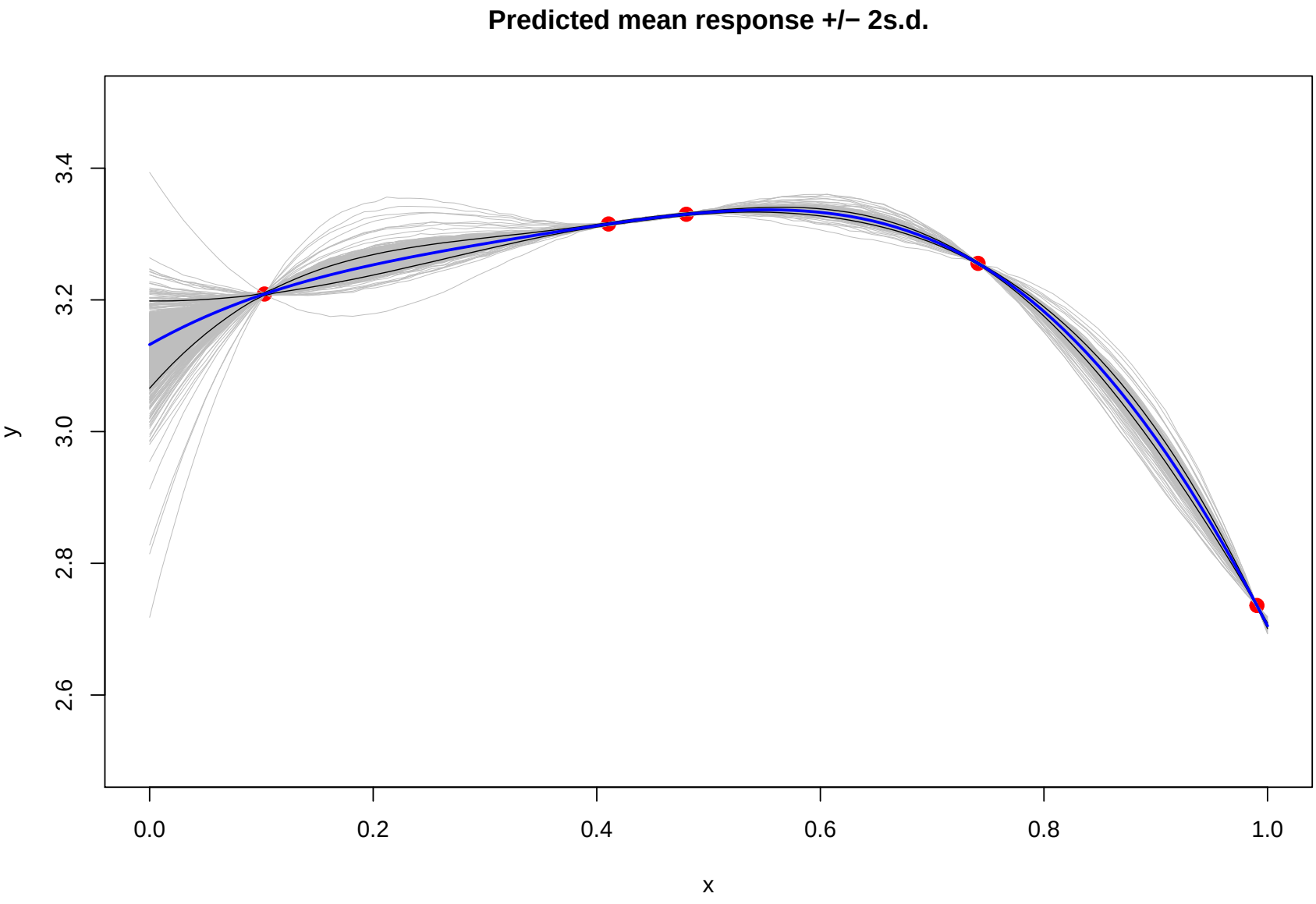
Example

```
grid=as.matrix(seq(0,1,length=100))
design.all=rbind(design,grid)
l1=list(m1=outer(design.all[,1],design.all[,1],"-"))
l.v=list(l1=l1)
fitp=predict(l.v,fit)
```

Example

```
plot(design,y,pch=20,col="red",cex=2,xlim=c(0,1),
    ylim=c(2.5,3.5),xlab="x",
    main="Predicted mean response +/- 2s.d.")
for(i in 1:nrow(fitp$preds))
  lines(grid,fitp$preds[i,],col="grey",lwd=0.25)
mean=apply(fitp$preds,2,mean)
sd=apply(fitp$preds,2,sd)
lines(grid,mean-1.96*sd,lwd=0.75,col="black")
lines(grid,mean+1.96*sd,lwd=0.75,col="black")
lines(grid,mean,lwd=2,col="blue")
```

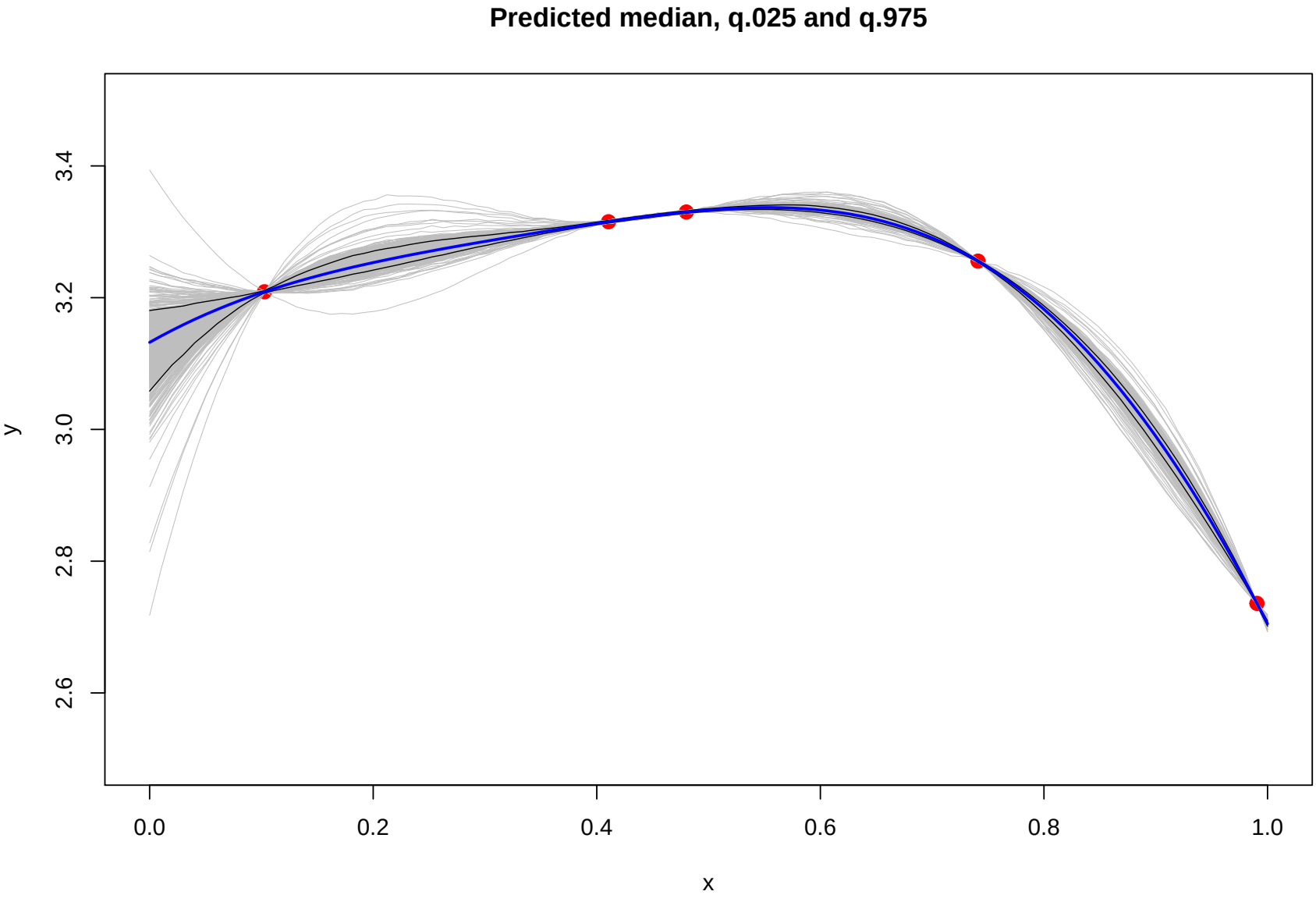
Example



Example

```
plot(design,y,pch=20,col="red",cex=2,xlim=c(0,1),ylim=c(2.5
    xlab="x",main="Predicted median, q.025 and q.975")
for(i in 1:nrow(fitp$preds))
  lines(grid,fitp$preds[i,],col="grey",lwd=0.25)
med=apply(fitp$preds,2,quantile,0.5)
q.025=apply(fitp$preds,2,quantile,0.025)
q.975=apply(fitp$preds,2,quantile,0.975)
lines(grid,q.025,lwd=0.75,col="black")
lines(grid,q.975,lwd=0.75,col="black")
lines(grid,mean,lwd=2,col="blue")
```

Example


Predicted median, q.025 and q.975
y
3.4
3.2
3.0
2.8
2.6
0.0
0.2
0.4
0.6
0.8
1.0
x

Calibrating the Priors

- There are many papers† discussing the problem of how one should setup prior distributions. I will take a simple approach for the current problem at hand.
- Strictly speaking, one should calibrate priors before seeing the data.
- In practice, often a weakly informative summary statistic of the data may be helpful in calibrating key aspects of the prior distribution.
    - for instance, the sample mean or sample variance of the data, and relating those to the appropriate moments of the prior.

† J. Oakley: *Eliciting Gaussian process priors for complex computer codes*, Journal of the Royal Statistical Society: Series D (The Statistician) vol.51, pp.81–97 (2002).

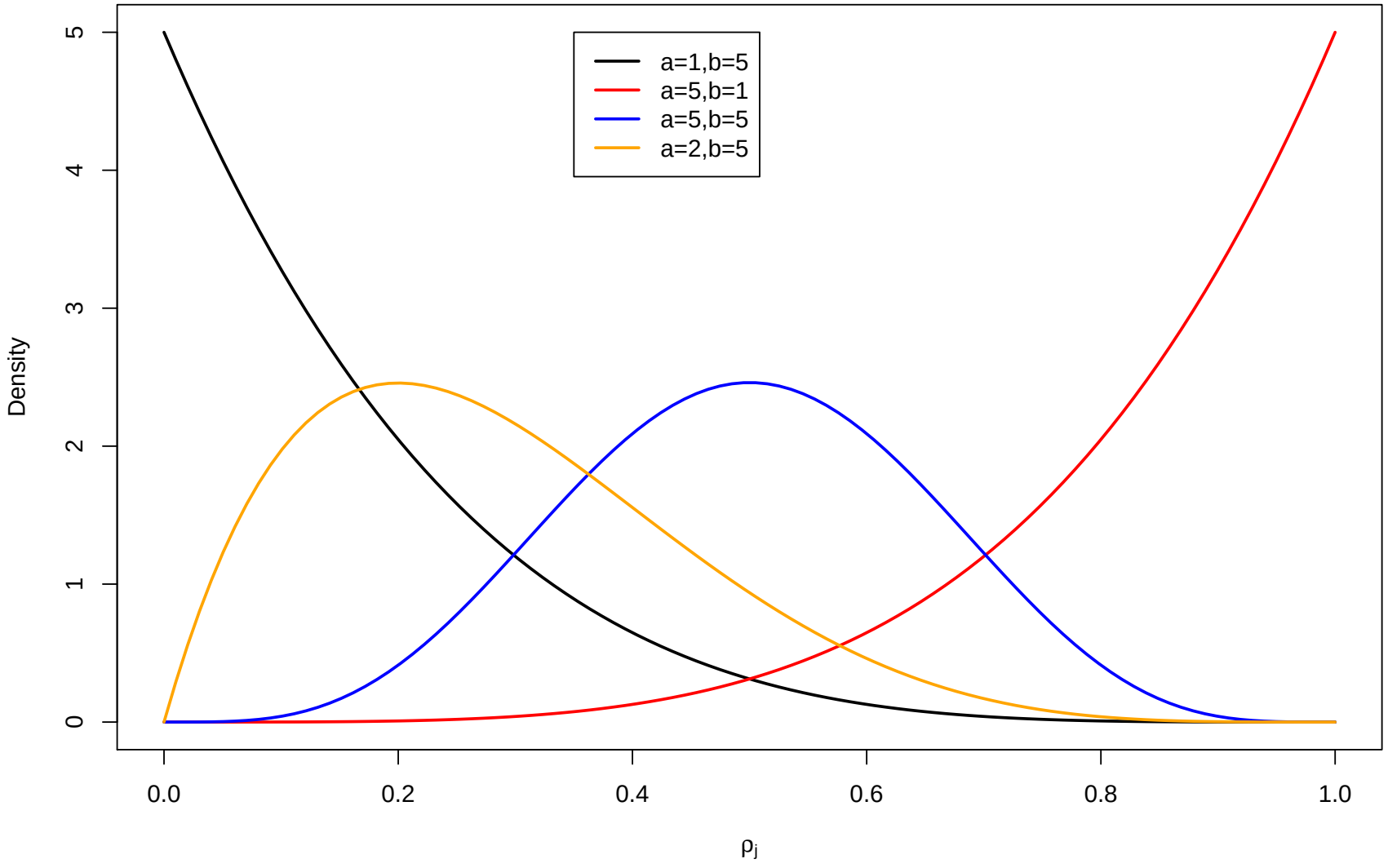
Calibrating the Priors

- For the correlation prior, $\pi(\boldsymbol{\rho}) = \prod_{j=1}^{d} \pi(\rho_j)$ we had used a Beta(α, β) prior,

$$\pi(\rho_j) \propto \rho_j^{\alpha-1}(1-\rho_j)^{\beta-1}.$$

- This distribution has mean $E[\rho_j] = \frac{\alpha}{\alpha+\beta}$
- For the GP model, I typically place more prior mass on a smooth function when working with simulators, and let the data drive the posterior away from smoothness.
- Lets look at a few possibilities.

Calibrating the Priors



Calibrating the Priors

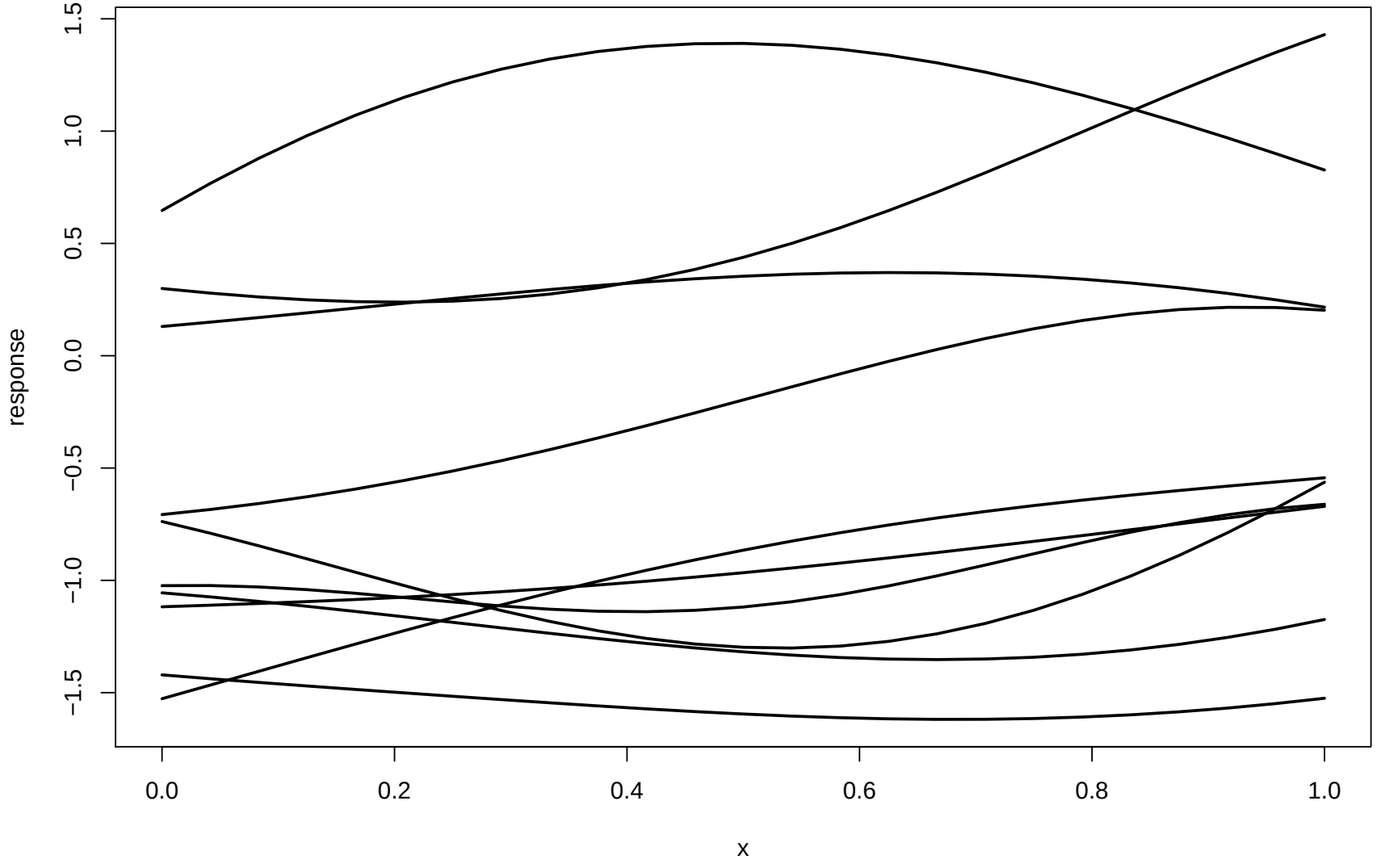
- So a reasonable strategy is to set the mean to some a priori level of smoothness, and calibrate the shape.
- For instance, one might look at a few priors, generate unconditional realizations from the GP and show them to your scientific collaborator to get feedback on what properties they empirically expect the response to exhibit.
- For the correlation parameter, Beta$(5, 5)$ (blue) or Beta$(2, 5)$ (orange) have been reasonably good starting points in my experience.

Example: $\pi(\rho) \sim$ Beta$(5, 5)$

```
set.seed(88)
a=5; b=5
m=10
n=25
draws=matrix(0,nrow=m,ncol=n)
x=seq(0,1,length=n)
X=abs(outer(x,x,"-"))

for(i in 1:m) {
  rho=rbeta(1,a,b)
  R=rho^(X^2)
  L=t(chol(R+diag(n)*.Machine$double.eps*100))
  mu=0
  Z=rnorm(n,mean=0,sd=1)
  draws[i,]=L%*%Z+mu
}
plot(x,draws[1,],xlim=c(0,1),ylim=range(draws),type='l',lwd
```

Example: $\pi(\rho) \sim \text{Beta}(5, 5)$

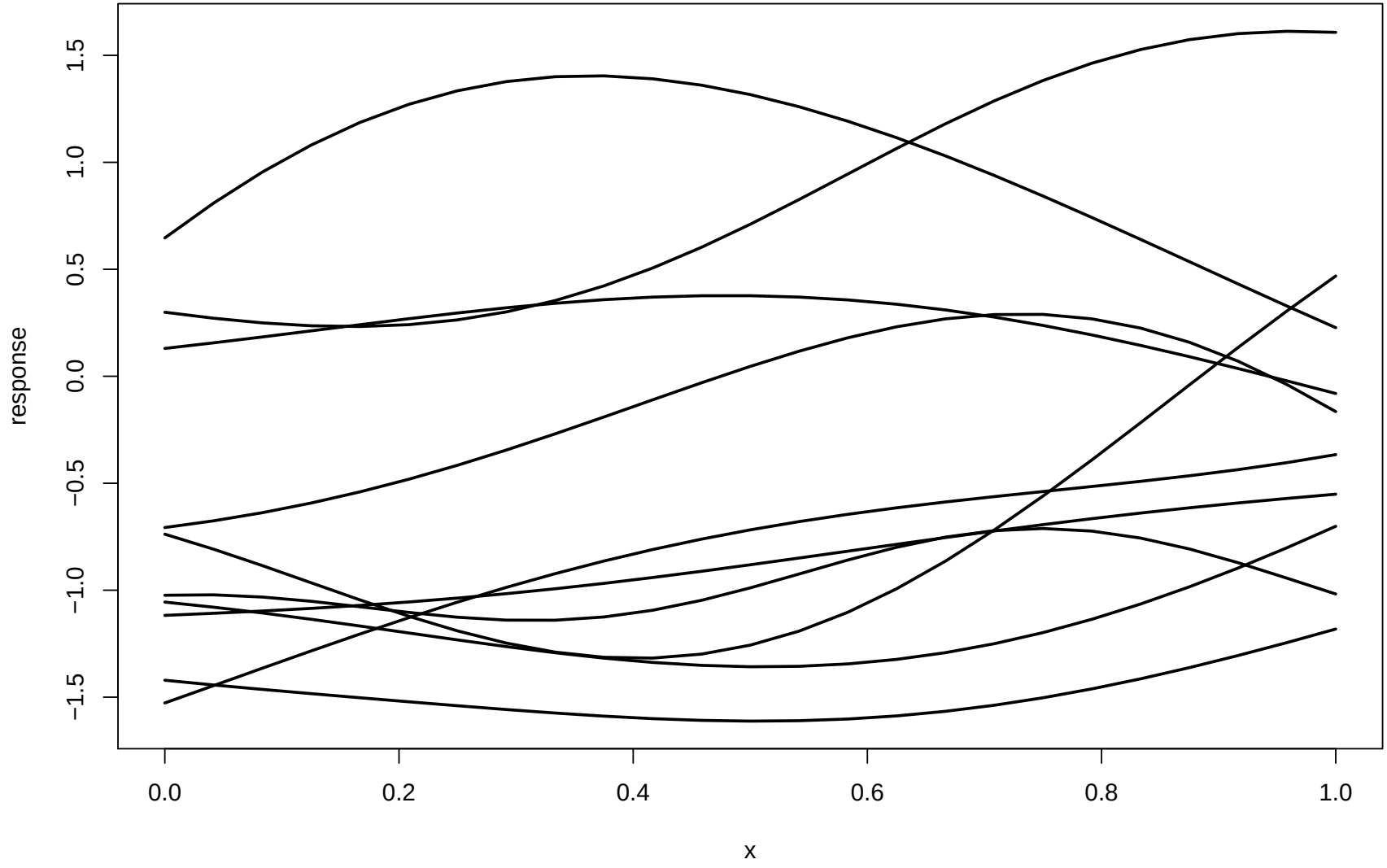


Example: $\pi(\rho) \sim$ Beta$(2, 5)$

```
set.seed(88)
a=2; b=5
m=10
n=25
draws=matrix(0,nrow=m,ncol=n)
x=seq(0,1,length=n)
X=abs(outer(x,x,"-"))

for(i in 1:m) {
  rho=rbeta(1,a,b)
  R=rho^(X^2)
  L=t(chol(R+diag(n)*.Machine$double.eps*100))
  mu=0
  Z=rnorm(n,mean=0,sd=1)
  draws[i,]=L%*%Z+mu
}
plot(x,draws[1,],xlim=c(0,1),ylim=range(draws),type='l',lwd
```

Example: $\pi(\rho) \sim \textbf{Beta}(2, 5)$

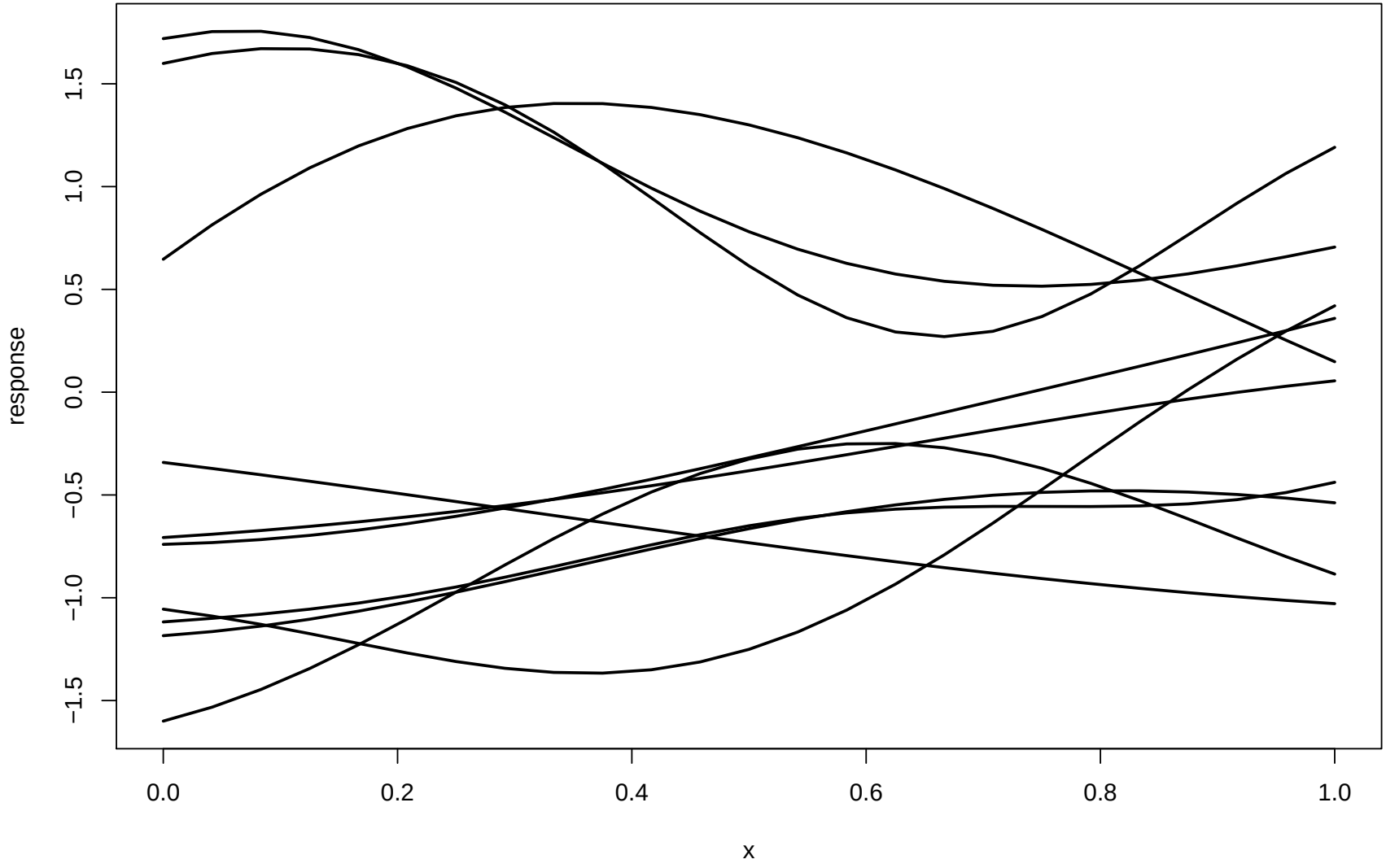


Example: $\pi(\rho) \sim$ Beta$(1, 5)$

```
set.seed(88)
a=1; b=5
m=10
n=25
draws=matrix(0,nrow=m,ncol=n)
x=seq(0,1,length=n)
X=abs(outer(x,x,"-"))

for(i in 1:m) {
  rho=rbeta(1,a,b)
  R=rho^(X^2)
  L=t(chol(R+diag(n)*.Machine$double.eps*100))
  mu=0
  Z=rnorm(n,mean=0,sd=1)
  draws[i,]=L%*%Z+mu
}
plot(x,draws[1,],xlim=c(0,1),ylim=range(draws),type='l',lwd
```

Example: $\pi(\rho) \sim \textbf{Beta}(1, 5)$



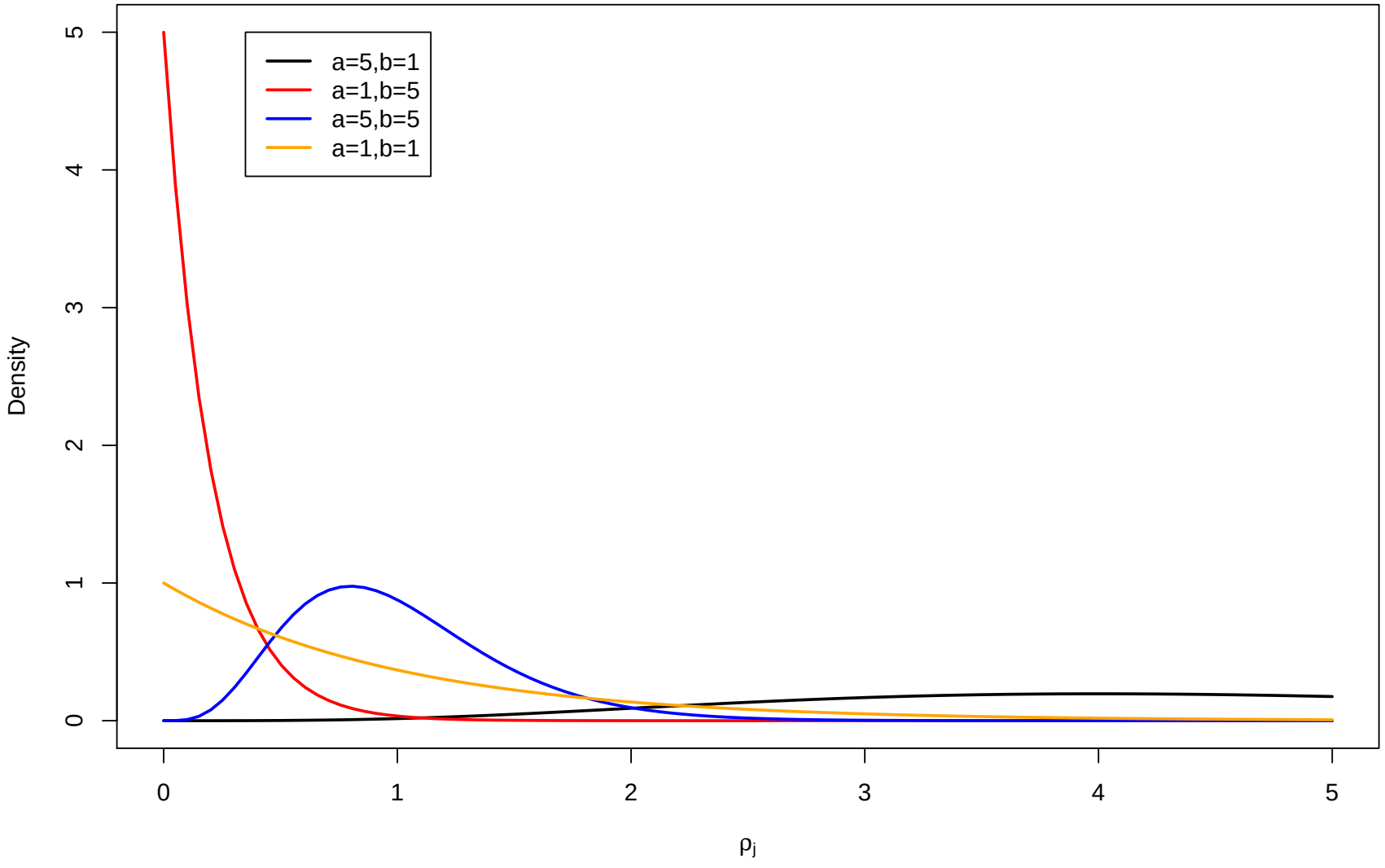
Calibrating the Priors

- For the precision prior, $\pi(\lambda)$ we had used a Gamma(a, b) prior,

$$\pi(\lambda) \propto \lambda^{a-1} exp(-b\lambda)$$

  with shape parameter $a > 0$ and rate paramter $b > 0$.
- This distribution has mean $E[\lambda] = \frac{a}{b}$ and variance $Var(\lambda) = \frac{a}{b^2}$.
- For the GP model, I typically calibrate the mean to the inverse of the sample variance.
- Lets look at a few possibilities.

Calibrating the Priors



Calibrating the Priors

- If the variance of the data is around 1, then Gamma$(5, 5)$ is a reasonable starting point.
- One might again look at a few priors, generate unconditional realizations from the GP and show them to your scientific collaborator to get feedback on what properties they empirically expect the response to exhibit.
    - or look at the sample variances of these draws.

Example: $\pi(\lambda) \sim$ Gamma$(5, 5)$

```
set.seed(88)
a=5; b=5
m=100
n=25
rho=0.2
draws=rep(0,m)
x=seq(0,1,length=n)
X=abs(outer(x,x,"-"))

for(i in 1:m) {
  R=rho^(X^2)
  L=t(chol(R+diag(n)*.Machine$double.eps*100))
  mu=0
  Z=rnorm(n,mean=0,sd=1)
  Y=sqrt(1/rgamma(1,shape=a,rate=b))*L%*%Z+mu
  Rinv=chol2inv(t(L))
  draws[i]=1/n*t(Y)%*%Rinv%*%Y
```

Example: $\pi(\lambda) \sim \textbf{Gamma}(5, 5)$