Gaussian Process Regression and Emulation

STAT8810, Fall 2017

M.T. Pratola

September 4, 2017

Today

Further thoughts on (frequentist) model fitting;
Compact Support Covariances

Previously: Fitting the GP to data

- Maximum Likelihood approach:

Previously: Fitting the GP to data

- Maximum Likelihood approach:
  - $\arg\max_{\boldsymbol{\theta}} \ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta})$ where $\ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta}) = -\frac{n}{2} log(\hat{\sigma}^2) - \frac{1}{2} log(|\mathbf{R}|) - \frac{n}{2}.$

Previously: Fitting the GP to data

- Maximum Likelihood approach:
  - $\arg\max_{\boldsymbol{\theta}} \ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta})$ where $\ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta}) = -\frac{n}{2} log(\hat{\sigma}^2) - \frac{1}{2} log(|\mathbf{R}|) - \frac{n}{2}$.
  - and $\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta})$ were the usual suspects

Previously: Fitting the GP to data

- Maximum Likelihood approach:
  - $\arg\max_{\boldsymbol{\theta}} \ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta})$ where $\ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta}) = -\frac{n}{2} log(\hat{\sigma}^2) - \frac{1}{2} log(|\mathbf{R}|) - \frac{n}{2}$.
  - and $\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta})$ were the usual suspects
- Restricted/Residual Maximum Likelihood (REML), Penalized MLE.

Previously: Fitting the GP to data

- Maximum Likelihood approach:
  - $\arg\max_{\boldsymbol{\theta}} \ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta})$ where $\ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta}), \boldsymbol{\theta}) = -\frac{n}{2} log(\hat{\sigma}^2) - \frac{1}{2} log(|\mathbf{R}|) - \frac{n}{2}$.
  - and $\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \hat{\sigma}^2(\boldsymbol{\theta})$ were the usual suspects
- Restricted/Residual Maximum Likelihood (REML), Penalized MLE.
- Newton-Raphson, Conjugate Gradient, Nelder-Mead, ...

Maximizing the log-Likelihood

- We have 2 basic problems

Maximizing the log-Likelihood

- We have 2 basic problems

1. The likelihood function is often not well behaved. As motivated last class, this can be the result of identifiability problems when estimating parameters from effectively a single realization of our process.

Maximizing the log-Likelihood

- We have 2 basic problems

1. The likelihood function is often not well behaved. As motivated last class, this can be the result of identifiability problems when estimating parameters from effectively a single realization of our process.
    - even when the infill asymptotics say the parameters should be identifiable, in practice we have a finite (discretized) sample from our unknown function

Example in 2D with Gaussian Correlation

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

logl<-function(rho,y,design.distmat,alpha=2,conditioning=0)
{
    n=length(y)
    R=rhogeodacecormat(design.distmat,rho,alpha)$R
    cR=chol(R+diag(n)*conditioning)
    Rinv=chol2inv(cR)

    s2.hat=(1/n)*t(y)%*%Rinv%*%y
    logdetR.div2=sum(log(diag(cR)))
    l=-n/2*log(s2.hat)-logdetR.div2-n/2

    return(l)
}
```

Example in 2D with Gaussian Correlation

```
library(rgl); rgl.clear()
set.seed(66)

n=25
x1=runif(n)
x2=runif(n)
X=as.matrix(cbind(x1,x2))
l1=list(m1=abs(outer(X[,1],X[,1],"-")))
l2=list(m2=abs(outer(X[,2],X[,2],"-")))
l.dez=list(l1=l1,l2=l2)

rho=c(0.9,0.1)
R=rhogeodacecormat(l.dez,rho)$R

L=t(chol(R+diag(n)*.Machine$double.eps*100))
Z=rnorm(n)
Y=L%*%Z
```

Example in 2D with Gaussian Correlation

```
rho1=seq(.001,1,length=20)
rho2=seq(.001,1,length=20)
rho.grid=as.matrix(expand.grid(rho1,rho2))
l.vals=rep(0,nrow(rho.grid))

for(i in 1:nrow(rho.grid))
    l.vals[i]=logl(rho.grid[i,],Y,l.dez,
                   conditioning=.Machine$double.eps*1000)

persp3d(matrix(l.vals,20,20),col="grey",xlab="rho1",
        ylab="rho2",zlab="logl",xlim=range(rho1),
        ylim=range(rho2))
```

Example in 2D with Gaussian Correlation

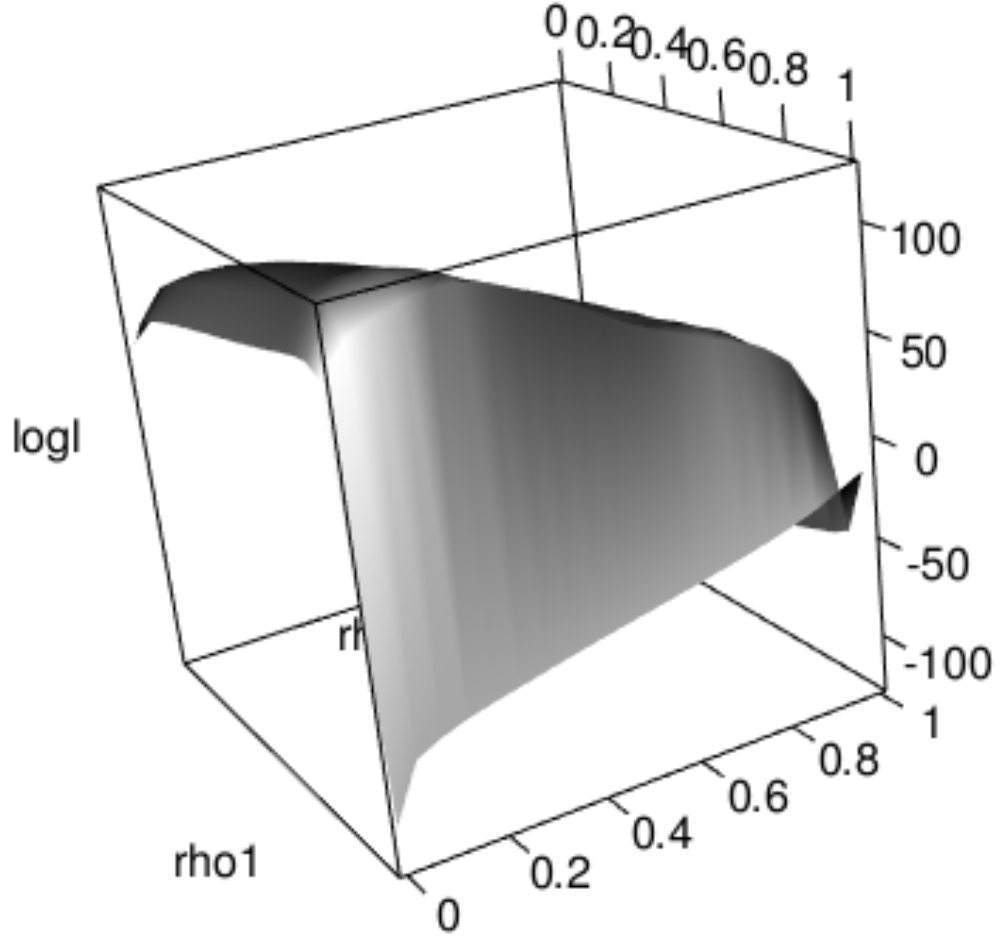


Figure 1: log-likelihood function

Example in 2D with Gaussian Correlation

```
set.seed(66)

n=15
x1=runif(n)
x2=runif(n)
X=as.matrix(cbind(x1,x2))
l1=list(m1=abs(outer(X[,1],X[,1],"-")))
l2=list(m2=abs(outer(X[,2],X[,2],"-")))
l.dez=list(l1=l1,l2=l2)

rho=c(0.9,0.1)
R=rhogeodacecormat(l.dez,rho)$R

L=t(chol(R+diag(n)*.Machine$double.eps*100))
Z=rnorm(n)
Y=L%*%Z
```

Example in 2D with Gaussian Correlation

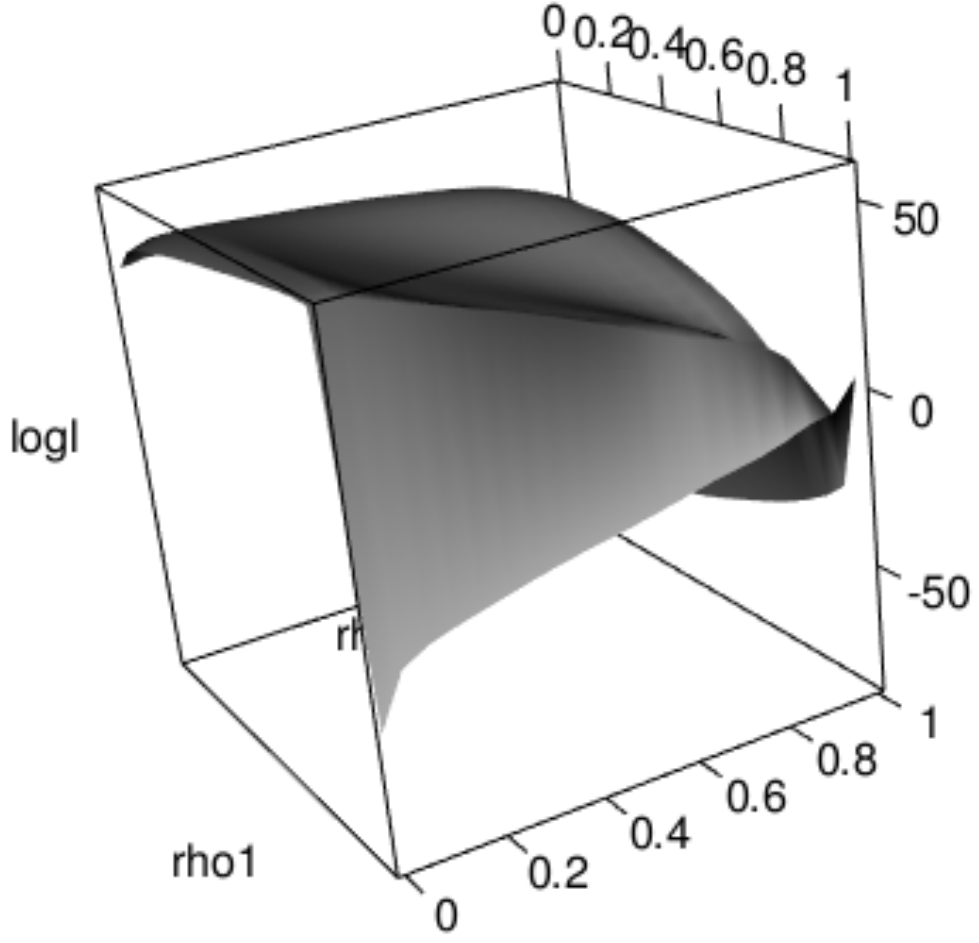


Figure 2: log-likelihood function

Example in 2D with Gaussian Correlation

```
set.seed(66)

n=8
x1=runif(n)
x2=runif(n)
X=as.matrix(cbind(x1,x2))
l1=list(m1=abs(outer(X[,1],X[,1],"-")))
l2=list(m2=abs(outer(X[,2],X[,2],"-")))
l.dez=list(l1=l1,l2=l2)

rho=c(0.9,0.1)
R=rhogeodacecormat(l.dez,rho)$R

L=t(chol(R+diag(n)*.Machine$double.eps*100))
Z=rnorm(n)
Y=L%*%Z
```

Example in 2D with Gaussian Correlation

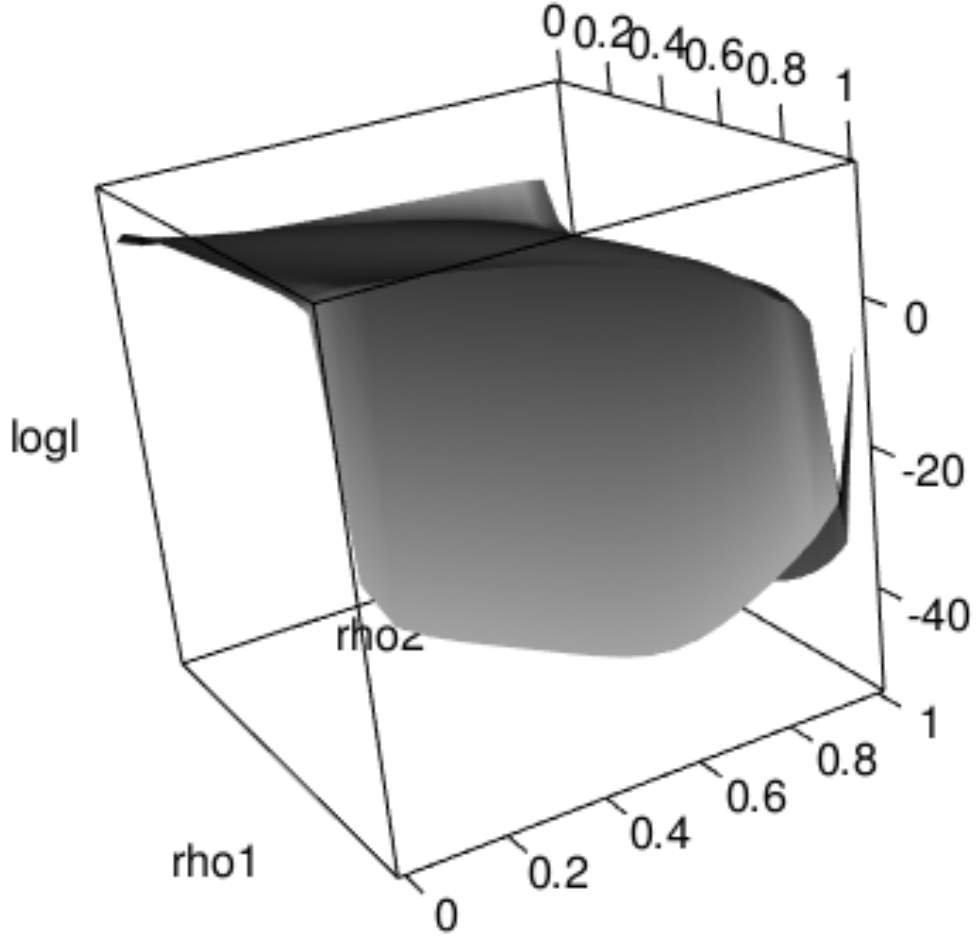


Figure 3: log-likelihood function

Same example with penalized likelihood

- Recall we can take a penalized likelihood of the form

$$\ell(\boldsymbol{\theta}) - n \sum_{k=1}^{d} p_\lambda(\rho_k)$$

```
l.vals=rep(0,nrow(rho.grid))
lambda=0.1

for(i in 1:nrow(rho.grid))
    l.vals[i]=logl(rho.grid[i,],Y,l.dez,conditioning=.Machi
```

Same example with penalized likelihood

- Recall we can take a penalized likelihood of the form

$$\ell(\boldsymbol{\theta}) - \lambda n \sum_{k=1}^{d} p_\lambda(\rho_k)$$

- Say we use $p_\lambda(\rho_k) = -\rho_k^2(1-\rho_k)^2$, then this prefer some moderate degree of smoothness.

```
l.vals=rep(0,nrow(rho.grid))
lambda=0.1

for(i in 1:nrow(rho.grid))
    l.vals[i]=logl(rho.grid[i,],Y,l.dez,conditioning=.Machi
```

Same example with penalized likelihood

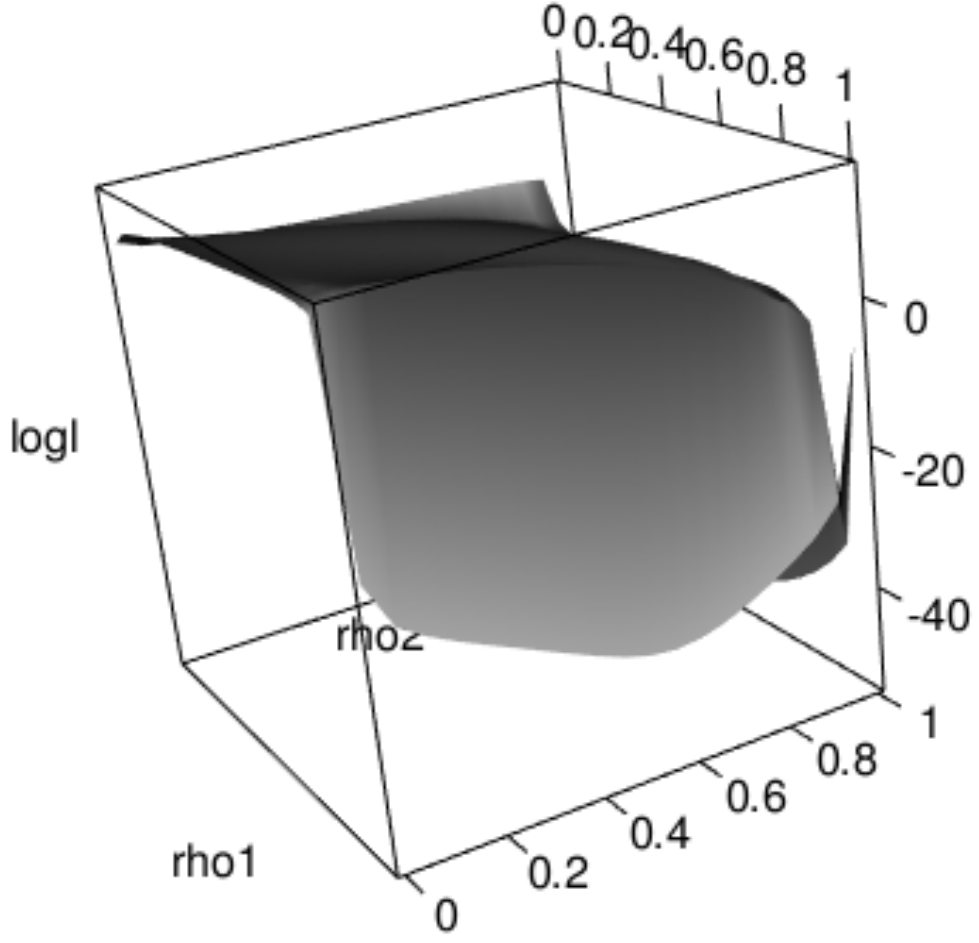


Figure 4: Penalized with lambda=0.1

Same example with penalized likelihood

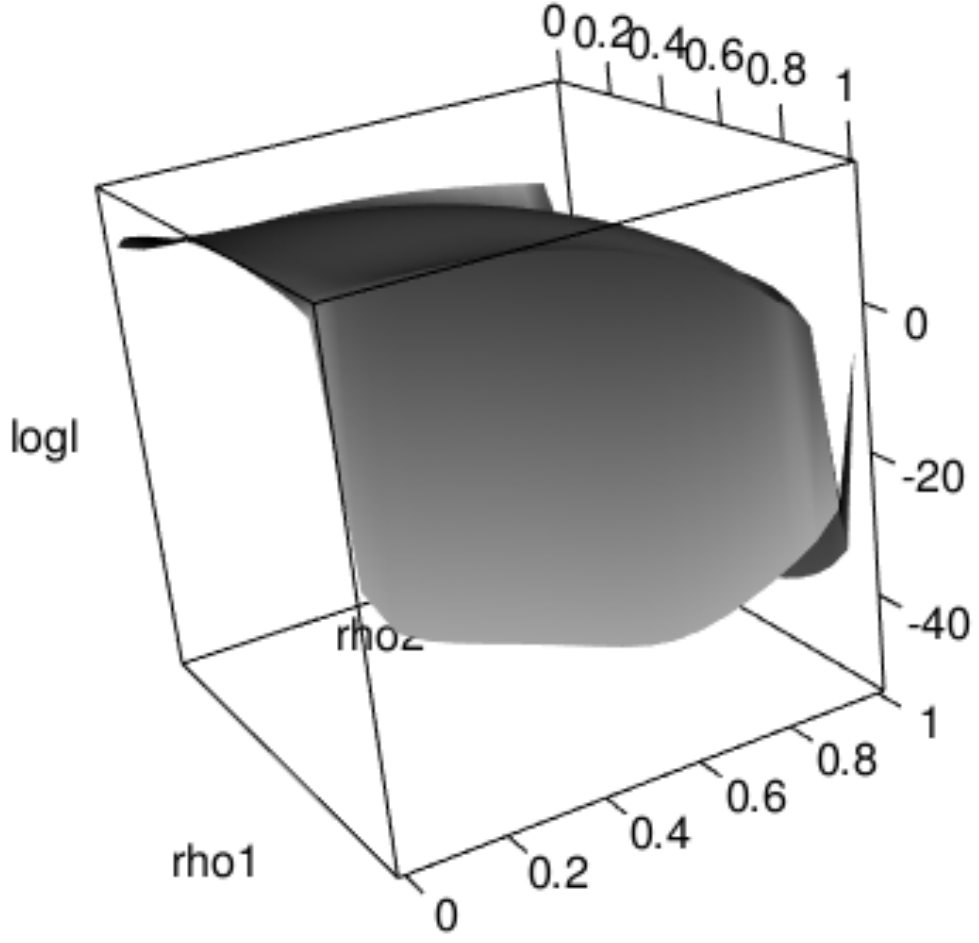


Figure 5: Penalized with lambda=5

Same example with penalized likelihood

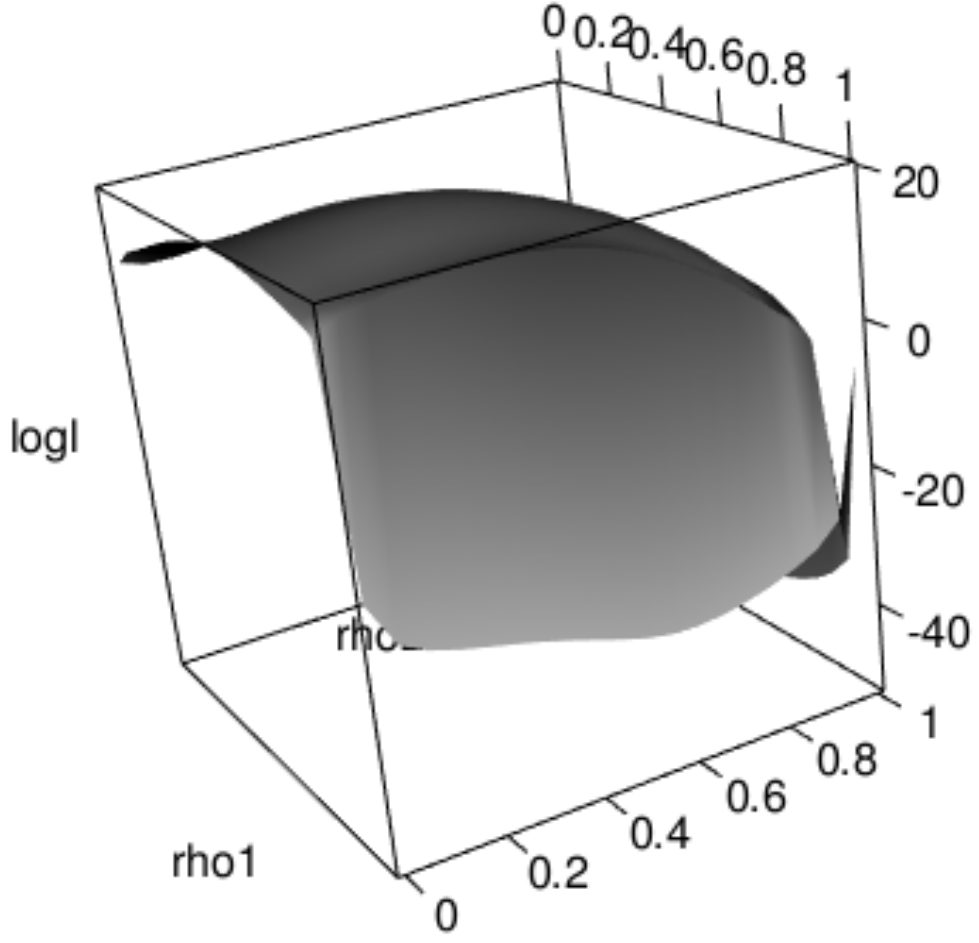


Figure 6: Penalized with lambda=10

Maximizing the log-Likelihood

- We have 2 basic problems

Maximizing the log-Likelihood

- We have 2 basic problems

2. Computational constraints. If our discretized sample of our function contains n datapoints, we need to compute the inverse of an $n \times n$ correlation matrix, an $\mathcal{O}(n^3)$ operation - i.e. very slow.

Maximizing the log-Likelihood

- We have 2 basic problems

2. Computational constraints. If our discretized sample of our function contains n datapoints, we need to compute the inverse of an $n \times n$ correlation matrix, an $\mathcal{O}(n^3)$ operation - i.e. very slow.
    - besides the computational constraints, the memory constraints also can become problematic quickly as they grow like $\mathcal{O}(n^2)$.

Old-school Optimization

- Suppose we want to maximize a non-linear function of a single variable, $\ell(\theta)$, or of many variables, $\ell(\boldsymbol{\theta})$.

Old-school Optimization

- Suppose we want to maximize a non-linear function of a single variable, $\ell(\theta)$, or of many variables, $\ell(\boldsymbol{\theta})$.
- Equivalent to finding $\ell'(\theta) = 0$ and second derivative test.

Old-school Optimization

- Suppose we want to maximize a non-linear function of a single variable, $\ell(\theta)$, or of many variables, $\ell(\boldsymbol{\theta})$.
- Equivalent to finding $\ell'(\theta) = 0$ and second derivative test.
- Most or all of the usual approaches I am going to mention are things you already know. My goal is not to explain them in overt detail (although I do provide references). My purpose for mentioning these will become clearer later on...

Newton-Raphson Method

- In this approach our function is the score function $S(\boldsymbol{\theta})$ and we want $S(\boldsymbol{\theta}) = 0$ where the score is the gradient of the log-likelihood wrt $\boldsymbol{\theta}$, $S(\boldsymbol{\theta}) = \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

Newton-Raphson Method

- In this approach our function is the score function $S(\boldsymbol{\theta})$ and we want $S(\boldsymbol{\theta}) = 0$ where the score is the gradient of the log-likelihood wrt $\boldsymbol{\theta}$, $S(\boldsymbol{\theta}) = \frac{\partial \ell(\boldsymbol{\theta})}{\boldsymbol{\theta}}$.
- Approximate the score by a linear Taylor series expansion about a particular $\boldsymbol{\theta}^{(k)}$:

$$S(\boldsymbol{\theta}) \approx S(\boldsymbol{\theta}^{(k)}) - \mathbf{H}(\boldsymbol{\theta}^{(k)})(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)})$$

where $\mathbf{H}(\cdot)$ is the Hessian matrix,

$$[\mathbf{H}(\boldsymbol{\theta})]_{ij} = \frac{\partial^2 \ell}{\partial \theta_i \partial \theta_j}.$$

Newton-Raphson Method

- Setting equal to zero and rearranging,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

Newton-Raphson Method

- Setting equal to zero and rearranging,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

- This motivates the NR iterations $k = 1, 2, \ldots$

Newton-Raphson Method

- Setting equal to zero and rearranging,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

- This motivates the NR iterations $k = 1, 2, \ldots$
- If ℓ is concave and unimodal, $\boldsymbol{\theta}^{(k)}, k = 1, 2, \ldots$ converges to the MLE.

Newton-Raphson Method

- Setting equal to zero and rearranging,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

- This motivates the NR iterations $k = 1, 2, \ldots$
- If ℓ is concave and unimodal, $\boldsymbol{\theta}^{(k)}, k = 1, 2, \ldots$ converges to the MLE.
- When not concave NR is not guaranteed to converge from an arbitrary starting value.

Newton-Raphson Method

- Setting equal to zero and rearranging,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

- This motivates the NR iterations $k = 1, 2, \ldots$
- If ℓ is concave and unimodal, $\boldsymbol{\theta}^{(k)}, k = 1, 2, \ldots$ converges to the MLE.
- When not concave NR is not guaranteed to converge from an arbitrary starting value.
- Expensive when lots of parameters because of $\mathbf{H}^{-1}$. Computation of $\mathbf{H}^{-1}$ also expensive in GP models because of $\mathbf{R}^{-1}$.

Newton-Raphson Method

- Modified NR: replace $\mathbf{H}(\cdot)$ by it's expected value, the Fisher Information matrix evaluated at $\boldsymbol{\theta}^{(k)}$:

$$[I(\boldsymbol{\theta})]_{ij} = -E\left[\left(\frac{\partial \ell(\boldsymbol{\theta})}{\theta_i}\right)\left(\frac{\partial \ell(\boldsymbol{\theta})}{\theta_j}\right)\middle|\boldsymbol{\theta}\right].$$

giving

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{I}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

Newton-Raphson Method

- Modified NR: replace $\mathbf{H}(\cdot)$ by it's expected value, the Fisher Information matrix evaluated at $\boldsymbol{\theta}^{(k)}$:

$$[I(\boldsymbol{\theta})]_{ij} = -E\left[\left(\frac{\partial \ell(\boldsymbol{\theta})}{\theta_i}\right)\left(\frac{\partial \ell(\boldsymbol{\theta})}{\theta_j}\right)\middle|\boldsymbol{\theta}\right].$$

giving

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{I}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

- For i.i.d. data, replace $\mathbf{I}(\cdot)$ with the empirical information matrix,

$$\mathcal{I}(\boldsymbol{\theta}^{(k)}) = \frac{1}{n}\sum_{i=1}^{n} S\left(y_i\middle|\boldsymbol{\theta}^{(k)}\right) S\left(y_i\middle|\boldsymbol{\theta}^{(k)}\right)^T$$

Newton-Raphson Method

- Modified NR: replace $\mathbf{H}(\cdot)$ by it's expected value, the Fisher Information matrix evaluated at $\boldsymbol{\theta}^{(k)}$:

$$[I(\boldsymbol{\theta})]_{ij} = -E\left[\left(\frac{\partial \ell(\boldsymbol{\theta})}{\theta_i}\right)\left(\frac{\partial \ell(\boldsymbol{\theta})}{\theta_j}\right)\middle|\boldsymbol{\theta}\right].$$

  giving

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{I}^{-1}(\boldsymbol{\theta}^{(k)})S(\boldsymbol{\theta}^{(k)})$$

- For i.i.d. data, replace $\mathbf{I}(\cdot)$ with the empirical information matrix,

$$\mathcal{I}(\boldsymbol{\theta}^{(k)}) = \frac{1}{n}\sum_{i=1}^{n} S\left(y_i\middle|\boldsymbol{\theta}^{(k)}\right) S\left(y_i\middle|\boldsymbol{\theta}^{(k)}\right)^T$$

- But we don't have i.i.d. data...

Nelder-Mead†

- Uses squishy-stretchy triangles (2D) or simplex (higher-D) to traverse the maximization surface.

† Nelder and Mead: *A Simplex Method for Function Minimization*, The Computer Journal, vol. 7, pp.308–313 (1965). Citation count is 25604 as of September 2017(!)

Nelder-Mead†

- Uses squishy-stretchy triangles (2D) or simplex (higher-D) to traverse the maximization surface.
- Derivative-free method (good) but cannot directly handle constraints (less good).

† Nelder and Mead: *A Simplex Method for Function Minimization*, The Computer Journal, vol. 7, pp.308–313 (1965). Citation count is 25604 as of September 2017(!)

Nelder-Mead†

- Uses squishy-stretchy triangles (2D) or simplex (higher-D) to traverse the maximization surface.
- Derivative-free method (good) but cannot directly handle constraints (less good).
- For a function of d variables, define the current simplex in d-dimensional space by the $d + 1$ points $\theta_0, \ldots, \theta_d$.

† Nelder and Mead: *A Simplex Method for Function Minimization*, The Computer Journal, vol. 7, pp.308–313 (1965). Citation count is 25604 as of September 2017(!)

Nelder-Mead†

- Uses squishy-stretchy triangles (2D) or simplex (higher-D) to traverse the maximization surface.
- Derivative-free method (good) but cannot directly handle constraints (less good).
- For a function of d variables, define the current simplex in d-dimensional space by the $d+1$ points $\theta_0, \ldots, \theta_d$.
- Write $\ell_i = \ell(\theta_i)$ and $\ell_h = max_i\ \ell_i$ and $\ell_l = min_i\ \ell_i$.

† Nelder and Mead: *A Simplex Method for Function Minimization*, The Computer Journal, vol. 7, pp.308–313 (1965). Citation count is 25604 as of September 2017(!)

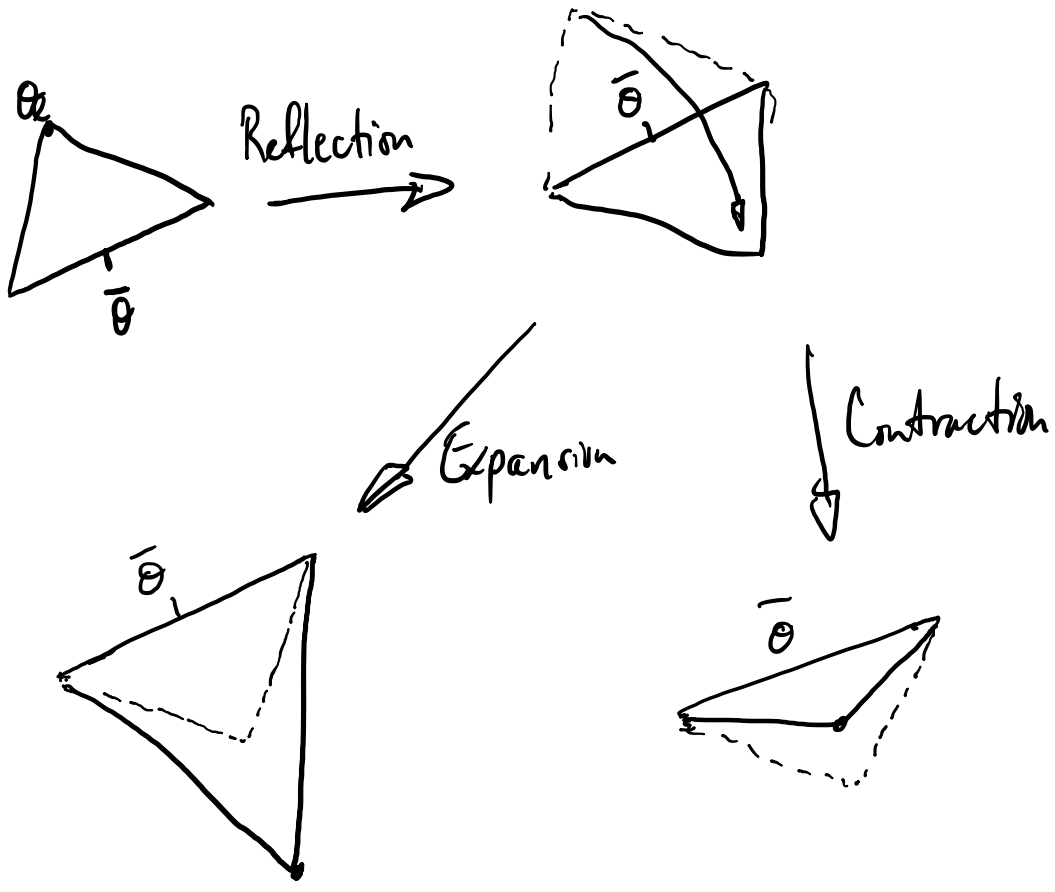
Nelder-Mead†

- Uses squishy-stretchy triangles (2D) or simplex (higher-D) to traverse the maximization surface.
- Derivative-free method (good) but cannot directly handle constraints (less good).
- For a function of d variables, define the current simplex in d-dimensional space by the $d+1$ points $\theta_0, \ldots, \theta_d$.
- Write $\ell_i = \ell(\theta_i)$ and $\ell_h = max_i\ \ell_i$ and $\ell_l = min_i\ \ell_i$.
- Define the centroid of the points with $i \neq l$ as $\bar{\theta}$ and at each stage replace θ_l by a new point via three operations: reflection, contraction and expansion.

† Nelder and Mead: *A Simplex Method for Function Minimization*, The Computer Journal, vol. 7, pp.308–313 (1965). Citation count is 25604 as of September 2017(!)

Nelder-Mead

Reflection, contraction and expansion:



Nelder-Mead

- Idea: keep ℓ_{max} within the simplex and keep trying to collapse the simplex on this point.

Nelder-Mead

- Idea: keep ℓ_{max} within the simplex and keep trying to collapse the simplex on this point.
- cool animation

Some Additional References

Nash and Varadhan: *Unifying Optimization Algorithms to Aid Software System Users: `optimx` for `R`*, Journal of Statistical Software, vol. 43, pp. 1–14 (2011)

- a wrapper library for general minimization of non-linear smooth functions of n parameters possibly with box constraints

  - similar syntax to `R`'s built-in `optim()` function.

Some Additional References

Nash and Varadhan: *Unifying Optimization Algorithms to Aid Software System Users: `optimx` for `R`*, Journal of Statistical Software, vol. 43, pp. 1–14 (2011)

- a wrapper library for general minimization of non-linear smooth functions of n parameters possibly with box constraints

- similar syntax to `R`'s built-in `optim()` function.
- allows running many optimization algorithms in one call and provides a comparative summary of the methods results.

Some Additional References

Nash and Varadhan: *Unifying Optimization Algorithms to Aid Software System Users: `optimx` for `R`*, Journal of Statistical Software, vol. 43, pp. 1–14 (2011)

- a wrapper library for general minimization of non-linear smooth functions of n parameters possibly with box constraints

- similar syntax to `R`'s built-in `optim()` function.
- allows running many optimization algorithms in one call and provides a comparative summary of the methods results.
- can also set it up to apply methods sequentially using `follow.on=TRUE`. E.g. start with Nelder-Mead and follow-up with a gradient-based method to refine estimate.

Prediction and Uncertainty Quantification

- Once we have our fitted model we can:

Prediction and Uncertainty Quantification

- Once we have our fitted model we can:
  - construct our prediction/emulation of our function;

Prediction and Uncertainty Quantification

- Once we have our fitted model we can:
  - construct our prediction/emulation of our function;
  - quantify the uncertainties in our prediction/emulation;

Prediction and Uncertainty Quantification

- Once we have our fitted model we can:
  - construct our prediction/emulation of our function;
  - quantify the uncertainties in our prediction/emulation;
  - do other things, such as sensitivity analysis (more on this later).

1D Example - generate our data

```
set.seed(77)

n=4
x1=seq(0.1,0.9,length=n)+runif(n,-.05,.05)
l1=list(m1=abs(outer(x1,x1,"-")))
l.dez=list(l1=l1)

rho=c(0.001)
R=rhogeodacecormat(l.dez,rho)$R

L=t(chol(R+diag(n)*.Machine$double.eps*100))
Z=rnorm(n)
Y=L%*%Z
```

1D Example - estimate via MLE

```
rho.hat=optimize(logl,interval=c(0,1),Y,l.dez,lower=0,
        upper=0.9,maximum=TRUE)$maximum
rho.hat
```

```
## [1] 0.003514553
```

```
R=rhogeodacecormat(l.dez,rho.hat,2)$R
cR=chol(R+diag(n)*.Machine$double.eps*0)
Rinv=chol2inv(cR)
s2.hat=(1/n)*t(Y)%*%Rinv%*%Y
s2.hat
```

```
##              [,1]
## [1,] 0.7797767
```

1D Example - predict and pointwise uncertainty intervals

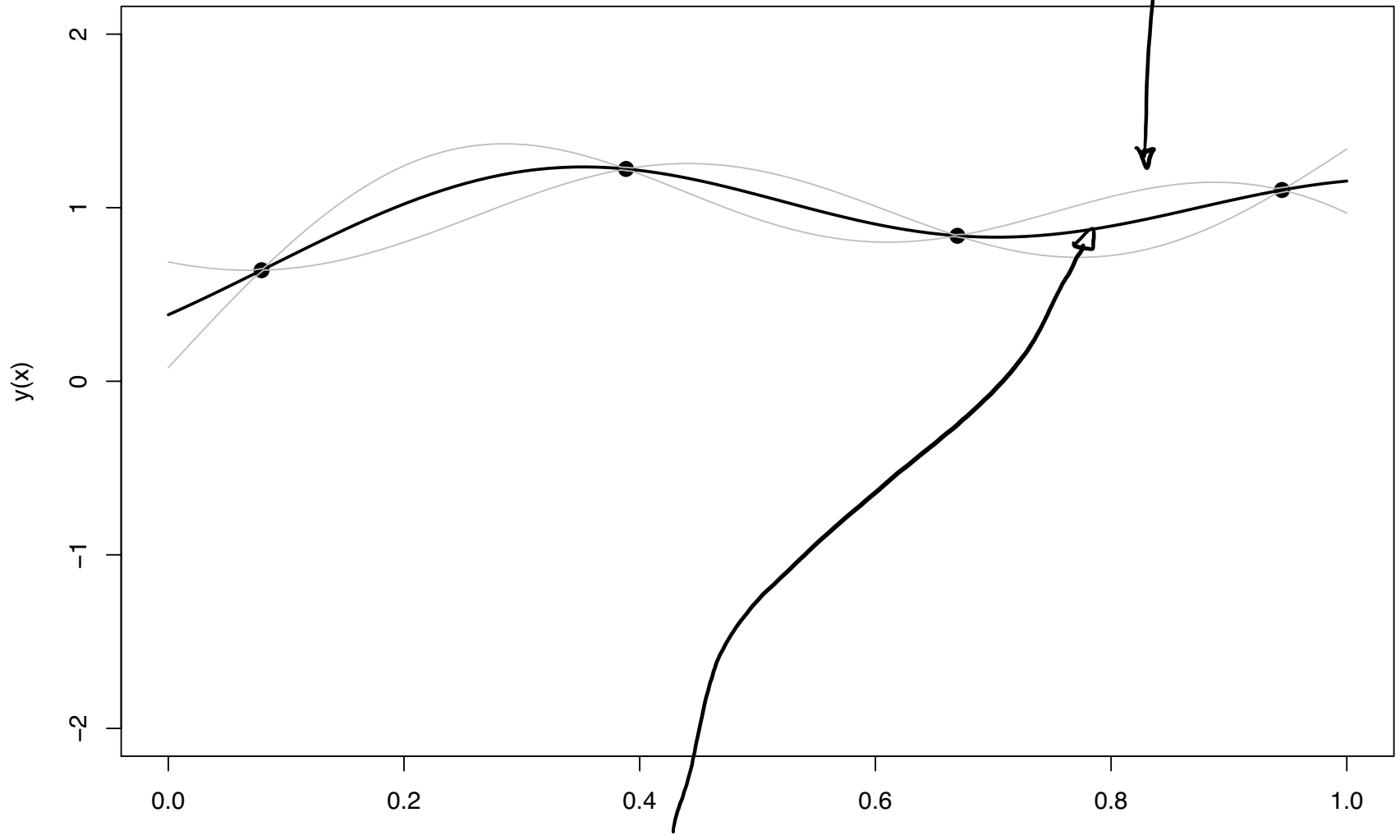
```
ngrid=100
pred.grid=seq(0,1,length=100)
X=c(pred.grid,x1)
l1=list(m1=abs(outer(X,X,"-")))
l.dez=list(l1=l1)

Rall=rhogeodacecormat(l.dez,rho.hat)$R
R0=Rall[1:ngrid,(ngrid+1):(ngrid+n)]
rm(Rall)

yhat=R0%*%Rinv%*%Y
s2hat=s2.hat*diag(1-R0%*%Rinv%*%t(R0))
```

1D Example - plot

```
plot(x1,Y,pch=20,cex=2,xlim=c(0,1),ylim=c(-2,2),
     xlab="x",ylab="y(x)")
lines(pred.grid,yhat,lwd=2,col="black")
lines(pred.grid,yhat-1.96*sqrt(s2hat),col="grey")
lines(pred.grid,yhat+1.96*sqrt(s2hat),col="grey")
```

$\hat{\sigma}^2(f) = \hat{\sigma}^2 \left(1 - r^T R^{-1} r + (f - F^T R^{-1} r)^T F^T R^{-1} R (f - F^T R^{-1} r)\right)$ when $\mu = 0$: $\hat{\sigma}^2 (1 - r^T R^{-1} r)$

1D Example - plot



$\hat{y}(x) = f^T \hat{\beta} + r^T R^{-1} (y - F\hat{\beta})$ when $\mu = 0$: $\hat{y}(x) = {}^x r^T R^{-1} y$

2D Example - generate our data

```
set.seed(99)

n=10
x1=runif(n)
x2=runif(n)
X=cbind(x1,x2)
l1=list(m1=abs(outer(X[,1],X[,1],"-")))
l2=list(m2=abs(outer(X[,2],X[,2],"-")))
l.dez=list(l1=l1,l2=l2)

rho=c(0.001,0.5)
R=rhogeodacecormat(l.dez,rho)$R

L=t(chol(R+diag(n)*.Machine$double.eps*100))
Z=rnorm(n)
Y=L%*%Z
```

2D Example - estimate via MLE

```
rho.hat=optim(c(0.5,0.5),logl,gr=NULL,lower=0,
        upper=0.9,method="L-BFGS-B",
        control=list(fnscale=-1),Y,l.dez)$par
rho.hat
```

```
## [1] 0.3434825 0.0526240
```

```
R=rhogeodacecormat(l.dez,rho.hat,2)$R
cR=chol(R+diag(n)*.Machine$double.eps*0)
Rinv=chol2inv(cR)
s2.hat=(1/n)*t(Y)%*%Rinv%*%Y
s2.hat
```

```
##          [,1]
## [1,] 4.150426
```

2D Example - predict and pointwise uncertainty intervals

```
ngrid=10
pred.grid=as.matrix(expand.grid(seq(0,1,length=ngrid),
                    seq(0,1,length=ngrid)))
Xall=rbind(pred.grid,X)
l1=list(m1=abs(outer(Xall[,1],Xall[,1],"-")))
l2=list(m2=abs(outer(Xall[,2],Xall[,2],"-")))
l.dez=list(l1=l1,l2=l2)

Rall=rhogeodacecormat(l.dez,rho.hat)$R
R0=Rall[1:(ngrid^2),(ngrid^2+1):(ngrid^2+n)]
rm(Rall)

yhat=R0%*%Rinv%*%Y
se.pred=sqrt(s2.hat*diag(1-R0%*%Rinv%*%t(R0)))
```

2D Example - plot

```
rgl.clear()
plot3d(X[,1],X[,2],Y,type='s',radius=0.075,col="red",
       xlim=c(0,1),ylim=c(0,1),zlim=c(-3,3))
persp3d(seq(0,1,length=ngrid),seq(0,1,length=ngrid),
        yhat,col="blue",add=TRUE)
persp3d(seq(0,1,length=ngrid),seq(0,1,length=ngrid),
       yhat-1.96*se.pred,col="grey",add=TRUE)
persp3d(seq(0,1,length=ngrid),seq(0,1,length=ngrid),
       yhat+1.96*se.pred,col="grey",add=TRUE)
rgl.snapshot("2dexample.png")
```

2D Example - plot

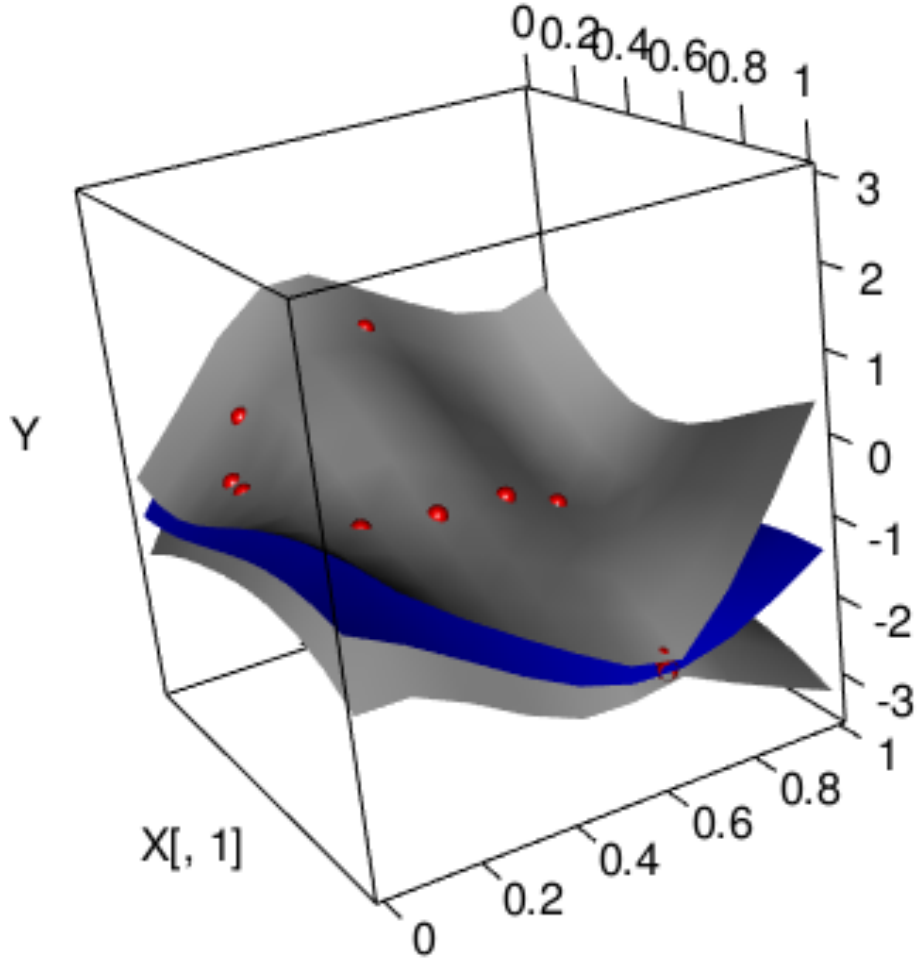
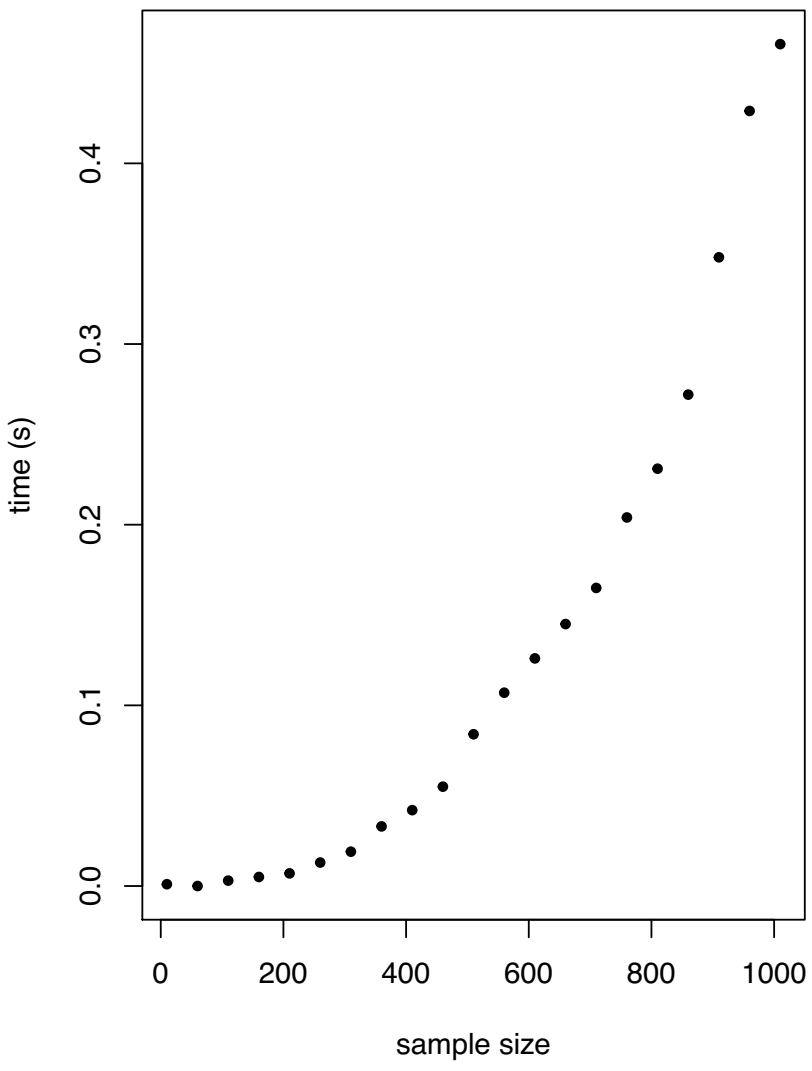


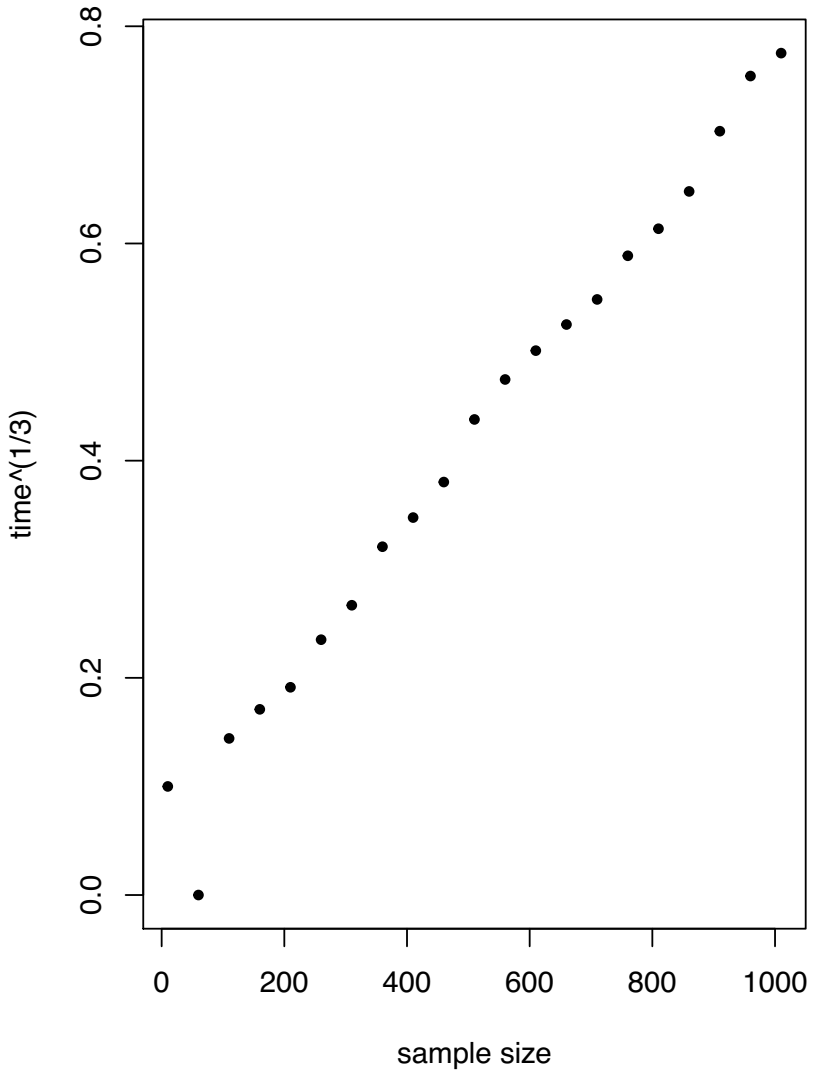
Figure 7: Fitted Response and 95% uncertainty interval

Dealing with Computational Limitations

```
N=seq(10,1010,by=50)
times=rep(0,length(N))
for(i in 1:length(N))
{
    n=N[i]; x1=runif(n); x2=runif(n); X=cbind(x1,x2)
    l1=list(m1=abs(outer(X[,1],X[,1],"-")))
    l2=list(m2=abs(outer(X[,2],X[,2],"-")))
    l.dez=list(l1=l1,l2=l2)

    rho=c(0.01,0.01)
    elapt=system.time({
    R=rhogeodacecormat(l.dez,rho)$R;
    Ri=chol2inv(chol(R+diag(n)*.Machine$double.eps*100));
    rm(R); rm(Ri)
    })
    times[i]=elapt[[1]]
}
```

Dealing with Computational Limitations





Dealing with Computational Limitations

- The computation problem has motivated the direction for much research in this area

Dealing with Computational Limitations

- The computation problem has motivated the direction for much research in this area
  - well, we still want good predictive performance and uncertainty quantification – otherwise, what is the point?

Dealing with Computational Limitations

- The computation problem has motivated the direction for much research in this area
  - well, we still want good predictive performance and uncertainty quantification – otherwise, what is the point?
  - however, the stationarity assumption of the GP model is overly burdensome in many practical contexts, so perhaps one can hope to allevaite two issues at once.

Dealing with Computational Limitations

- The computation problem has motivated the direction for much research in this area
  - well, we still want good predictive performance and uncertainty quantification – otherwise, what is the point?
  - however, the stationarity assumption of the GP model is overly burdensome in many practical contexts, so perhaps one can hope to allevaite two issues at once.
- A popular approach has been to *sparsify* (introduce 0's into) the correlation matrix $\mathbf{R}$. This can lead to significant computational reductions by using so-called sparse matrix algebra packages.

Dealing with Computational Limitations

- The computation problem has motivated the direction for much research in this area
  - well, we still want good predictive performance and uncertainty quantification – otherwise, what is the point?
  - however, the stationarity assumption of the GP model is overly burdensome in many practical contexts, so perhaps one can hope to allevaite two issues at once.
- A popular approach has been to *sparsify* (introduce 0's into) the correlation matrix $\mathbf{R}$. This can lead to significant computational reductions by using so-called sparse matrix algebra packages.
- What we need are correlation functions that decayed to 0 exactly at a finite range from the origin, like the cubic correlation function. The are called *compact correlation functions*.

Compact Correlation Functions

- Applications of compact covariances to statisitcal uncertainty quantification trace back to Furrer et al. and Kaufman et al. †.

Kaufman, Bingham, Habib, Heitmann and Frieman: *Efficient Emulators of Computer Experiments Using Compactly Supported Correlation Functions, with an Application to Cosmology*, The Annals of Applied Statistics, vol. 5, pp. 2470–2492 (2011).

Furrer, Genton and Nychka: *Covariance tapering for interpolation of large spatial datasets*, Journal of Computational and Graphical Statistics, vol. 15, pp. 502–523 (2006).

Compact Correlation Functions

- Applications of compact covariances to statisitcal uncertainty quantification trace back to Furrer et al. and Kaufman et al. †.
- A key contribution from this string of research has been the `R` package `fields` by Furrer et al.

Kaufman, Bingham, Habib, Heitmann and Frieman: *Efficient Emulators of Computer Experiments Using Compactly Supported Correlation Functions, with an Application to Cosmology*, The Annals of Applied Statistics, vol. 5, pp. 2470–2492 (2011).

Furrer, Genton and Nychka: *Covariance tapering for interpolation of large spatial datasets*, Journal of Computational and Graphical Statistics, vol. 15, pp. 502–523 (2006).

Compact Correlation Functions

- Applications of compact covariances to statisitcal uncertainty quantification trace back to Furrer et al. and Kaufman et al. †.
- A key contribution from this string of research has been the R package `fields` by Furrer et al.
  - this package combines compact covariance capabilities with efficient sparse matrix algebra.

Kaufman, Bingham, Habib, Heitmann and Frieman: *Efficient Emulators of Computer Experiments Using Compactly Supported Correlation Functions, with an Application to Cosmology*, The Annals of Applied Statistics, vol. 5, pp. 2470–2492 (2011).

Furrer, Genton and Nychka: *Covariance tapering for interpolation of large spatial datasets*, Journal of Computational and Graphical Statistics, vol. 15, pp. 502–523 (2006).

Compact Correlation Functions

- Applications of compact covariances to statisitcal uncertainty quantification trace back to Furrer et al. and Kaufman et al. †.
- A key contribution from this string of research has been the R package `fields` by Furrer et al.
  - this package combines compact covariance capabilities with efficient sparse matrix algebra.
  - however, it's focus is on 2D and 3D problems.

Kaufman, Bingham, Habib, Heitmann and Frieman: *Efficient Emulators of Computer Experiments Using Compactly Supported Correlation Functions, with an Application to Cosmology*, The Annals of Applied Statistics, vol. 5, pp. 2470–2492 (2011).

Furrer, Genton and Nychka: *Covariance tapering for interpolation of large spatial datasets*, Journal of Computational and Graphical Statistics, vol. 15, pp. 502–523 (2006).

Compact Correlation Functions

- In reality, these statistical results are built on key contributions from mathematics in the study of positive semi-definite functions with compact support.

Gneiting: *Radial Positive Definite Functions Generated by Euclid's Hat*, Journal of Multivariate Analysis, vol. 69, pp. 88–119 (1999).

Gneiting: Strictly and non-strictly positive definite functions on spheres_, Bernoulli, vol. 19, pp. 1327—1349 (2013).

Bevilacqua, Faouzi, Furrer and Porcu: *Estimation and Prediction using generalized Wendland Covariance Functions under fixed domain asymptotics*, arXiv preprint arXiv:1607.06921 (2017).

Compact Correlation Functions

- In reality, these statistical results are built on key contributions from mathematics in the study of positive semi-definite functions with compact support.
- Early contributions were the Askey class of functions which later developed into the Wendland functions that are positive semi-definite in 2D and 3D.

Gneiting: *Radial Positive Definite Functions Generated by Euclid's Hat*, Journal of Multivariate Analysis, vol. 69, pp. 88–119 (1999).

Gneiting: Strictly and non-strictly positive definite functions on spheres_, Bernoulli, vol. 19, pp. 1327—1349 (2013).

Bevilacqua, Faouzi, Furrer and Porcu: *Estimation and Prediction using generalized Wendland Covariance Functions under fixed domain asymptotics*, arXiv preprint arXiv:1607.06921 (2017).

Compact Correlation Functions

- In reality, these statistical results are built on key contributions from mathematics in the study of positive semi-definite functions with compact support.
- Early contributions were the Askey class of functions which later developed into the Wendland functions that are positive semi-definite in 2D and 3D.
- Finally this culminated in the generalized Wendland functions, which support arbitrary finite dimensions and allow one to parameterize the degree of differentiability much like the Matern.

Gneiting: *Radial Positive Definite Functions Generated by Euclid's Hat*, Journal of Multivariate Analysis, vol. 69, pp. 88–119 (1999).

Gneiting: Strictly and non-strictly positive definite functions on spheres_, Bernoulli, vol. 19, pp. 1327—1349 (2013).

Bevilacqua, Faouzi, Furrer and Porcu: *Estimation and Prediction using generalized Wendland Covariance Functions under fixed domain asymptotics*, arXiv preprint arXiv:1607.06921 (2017).

The Askey Function

- The Askey function is said to belong to the class of positive semi-definite functions (a continuous mapping from $\mathbb{R}^d \to \mathbb{R}$) with compact support on θ,

$$\mathcal{A}(h) = \left(1 - \frac{h}{\theta}\right)_+^{\mu} = \begin{cases} \left(1 - \frac{h}{\theta}\right)^{\mu}, 0 \leq h < \theta \\ 0, h \geq \theta \end{cases},$$

if and only if $\mu \geq (d+1)/2$ and where h is the usual Euclidean norm.

The Wendland Function

- Furrer et al. introduce the Wendland$_1$ and Wendland$_2$ functions which give valid covariances in 3D with different degrees of differentiability:

$$\text{Wendland}_1: \quad \left(1-\tfrac{h}{\theta}\right)_+^4 \left(1+4\tfrac{h}{\theta}\right)$$

$$\text{Wendland}_2: \quad \left(1-\tfrac{h}{\theta}\right)_+^6 \left(1+6\tfrac{h}{\theta}+\tfrac{35h^2}{3\theta^2}\right)$$

The Wendland Function

- Furrer et al. introduce the Wendland$_1$ and Wendland$_2$ functions which give valid covariances in 3D with different degrees of differentiability:

$$\text{Wendland}_1: \quad \left(1-\tfrac{h}{\theta}\right)_+^4 \left(1+4\tfrac{h}{\theta}\right)$$
$$\text{Wendland}_2: \quad \left(1-\tfrac{h}{\theta}\right)_+^6 \left(1+6\tfrac{h}{\theta}+\tfrac{35h^2}{3\theta^2}\right)$$

- Their approach was to pair these with the Matern to form a "tapered" covariance, defined as the product:

$$R_{\text{tap}}(h) = R_{\text{Matern}}(h) R_{\text{Wendland}}(h)$$

The Wendland Function

- Furrer et al. were able to show that in some important ways† the tapered correlation function gave behaviour that was largely indistinguishable from just using the Matern itself.

† Asymptotically equivalent MSPE (in the in-fill sense). See Stein: *Interpolation of Spatial Data: Some Theory for Kriging* (1999) for further details.

The Wendland Function

- Furrer et al. were able to show that in some important ways† the tapered correlation function gave behaviour that was largely indistinguishable from just using the Matern itself.
  - in particular, they recommend pairing Wendland_1 with the Matern with $\nu \leq 1.5$ and Wendland_2 with Matern with $\nu \leq 2.5$.

† Asymptotically equivalent MSPE (in the in-fill sense). See Stein: *Interpolation of Spatial Data: Some Theory for Kriging* (1999) for further details.

The Wendland Function

- Furrer et al. were able to show that in some important ways† the tapered correlation function gave behaviour that was largely indistinguishable from just using the Matern itself.
  - in particular, they recommend pairing Wendland_1 with the Matern with $\nu \leq 1.5$ and Wendland_2 with Matern with $\nu \leq 2.5$.
- The idea was to, in some sense, borrow properties of both the Matern and the compactness of the Wendland in forming the tapered correlation function.

† Asymptotically equivalent MSPE (in the in-fill sense). See Stein: *Interpolation of Spatial Data: Some Theory for Kriging* (1999) for further details.

The Generalized Wendland Function

- Takes as parameters the input dimension and the the degree of differentiability desired.

The Generalized Wendland Function

- Takes as parameters the input dimension and the the degree of differentiability desired.
- Has been shown to be of the form $\mathcal{A}(h)P_k(h)$ where $\mathcal{A}(h)$ is the Askey function and $P_k(h)$ is a polynomial of order k in h.

The Generalized Wendland Function

- Takes as parameters the input dimension and the the degree of differentiability desired.
- Has been shown to be of the form $\mathcal{A}(h)P_k(h)$ where $\mathcal{A}(h)$ is the Askey function and $P_k(h)$ is a polynomial of order k in h.
- The general form when k is not a positive integer is written in terms of an integral (Bevilacqua et al.):

$$R(h) = \begin{cases} \frac{1}{B(2k,\mu+1)} \int_h^1 u(u^2 - h^2)^{k-1}(1-u)^{\mu} du, \ 0 \leq h < 1 \\ 0, \ h \geq 1 \end{cases}$$

where $B(\cdot)$ is the Beta function.

The Generalized Wendland Function

- Takes as parameters the input dimension and the the degree of differentiability desired.
- Has been shown to be of the form $\mathcal{A}(h)P_k(h)$ where $\mathcal{A}(h)$ is the Askey function and $P_k(h)$ is a polynomial of order k in h.
- The general form when k is not a positive integer is written in terms of an integral (Bevilacqua et al.):

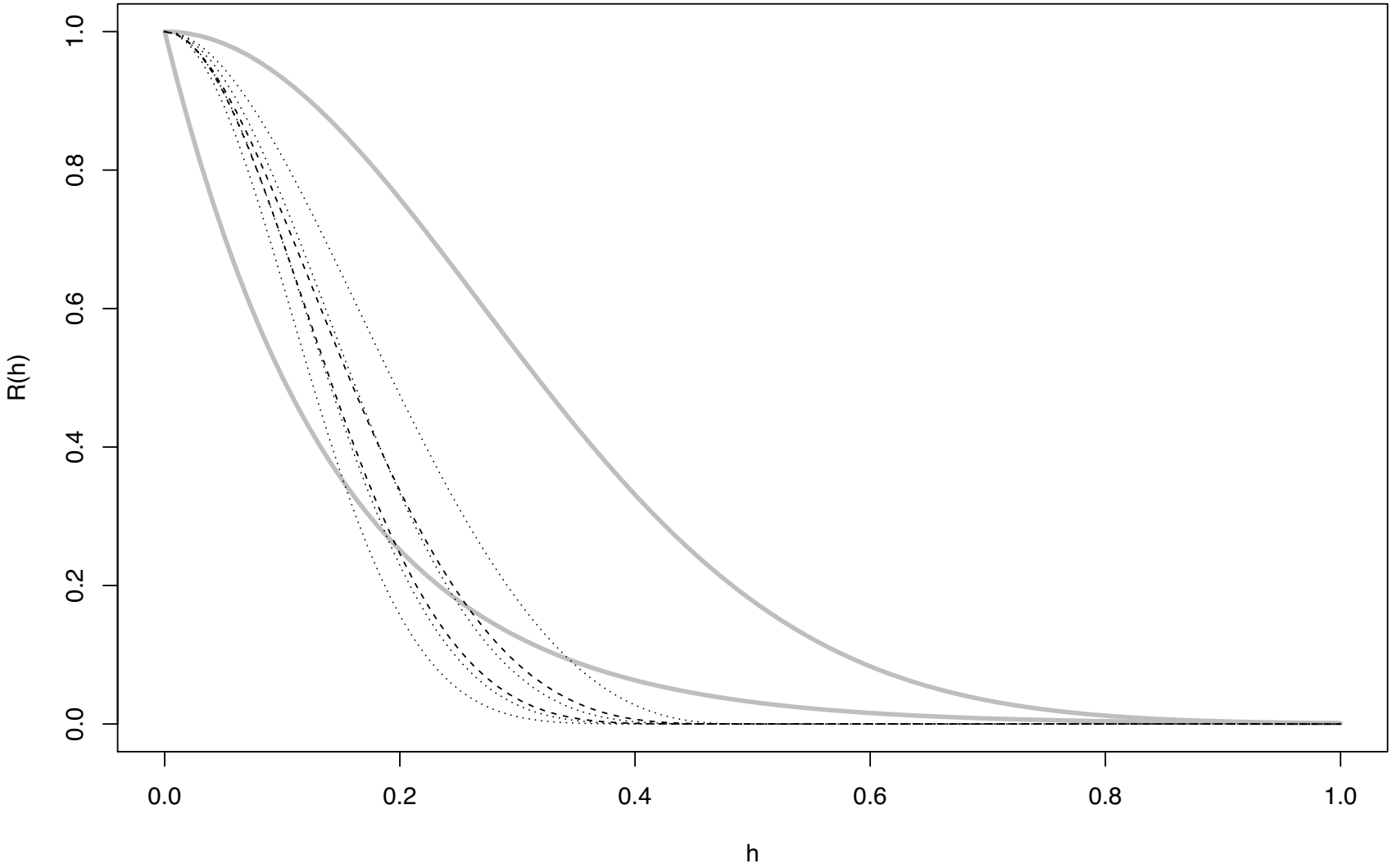
$$R(h) = \begin{cases} \frac{1}{B(2k,\mu+1)} \int_h^1 u(u^2 - h^2)^{k-1}(1-u)^{\mu} du, \ 0 \leq h < 1 \\ 0, \ h \geq 1 \end{cases}$$

  where $B(\cdot)$ is the Beta function.
- Instead of tapering, we might view this as a more direct way of specifying the compactness and differentiability properties one seeks.

Wendland and friends

```
x=seq(0,1,length=100)
l1=list(m1=abs(outer(x,x,"-")))
l.dez=list(l1=l1)
h=sqrt((0-x)^2)

R.gauss=rhogeodacecormat(l.dez,rho=0.001,alpha=2)$R
R.exp=rhogeodacecormat(l.dez,rho=0.001,alpha=1)$R
R.wend1=wendland1(l.dez,0.5)$R
R.wend2=wendland2(l.dez,0.5)$R
R.gw1=generalized.wendland(l.dez,0.5,1)$R
R.gw2=generalized.wendland(l.dez,0.5,2)$R
R.gw3=generalized.wendland(l.dez,0.5,3)$R
R.gw4=generalized.wendland(l.dez,0.5,4)$R
```

Wendland and friends



Grey=Gaussian and Exponential; Dashed=Wendland 1,2;
Dotted=Generalized Wendland with k=1,2,3,4

"Usual" GP model

Method	n	Computing
"Standard"	100-1,000	CPU
Franey et al (2012)	4,064	CPU+GPU
Paciorek et al (2013)	67,275	CPU+GPU Cluster

Approximate GP

Method	Approximation	n	Computing
Kaufman et al (2012)	Comp. Cov.	20,000	CPU
Eidsvik et al (2014)	Comp. Lik.	173,405	GPU
Gramacy & Apley (2014)	Local Approx. GP	millions	GPU

Parallel Bayesian Additive Regression Trees

Method	Approximation	n	Computing
Pratola et al (2014)	none	9 million+	CPU cluster
Scott et al (submitted)	p BARTs on n/p	more?	CPU cluster