

Why R?

Dr. Erin Hodgess

Western Governors Univ

June 11, 2020

Using R with High Performance Tools on a Windows Laptop

Overview

- Introduction
- Ordinary Kriging for Spatio-Temporal Data: rmpiFort
- Theoretical Background
- Empirical Background
- The Irish Wind Data Study
- The Cuba Temperature Data Study
- A Discussion of the Computer Setup: Open Accelerators (OpenACC), the Message Passing Interface (MPI), and CUDA Fortran
- Conclusion

Introduction

- Several years ago, I was fortunate enough to work on supercomputers for geostatistics research on ordinary kriging.
- I worked with spatial data and was able to achieve excellent speedup.
- I wondered if laptops would work for those who didn't have access to supercomputers.

Introduction

- Ordinary Kriging on Spatio-Temporal Data Sets
- Spatio-Temporal Data Extended the Models
- The existing R process was slow, and did not allow for large data sets

Ordinary Kriging for Spatio-Temporal Data: rmpiFort

- I constructed a R package with high performance tools on Windows
- I ran comparisons on a MacBook Air and a Windows laptop
- I obtained surprisingly good results

Theoretical Background

- I have a spatio-temporal vector:

$$\mathbf{z} = z(s_1, t_1), z(s_2, t_2), \dots, z(s_n, t_n)$$

where $z(s_i, t_i)$ are the locations and times of the response variable.

- First Order Stationarity

$$E[\mathbf{z}] = \boldsymbol{\mu}$$

Theoretical Background

- Second Order Stationarity: Covariance for the S-T model

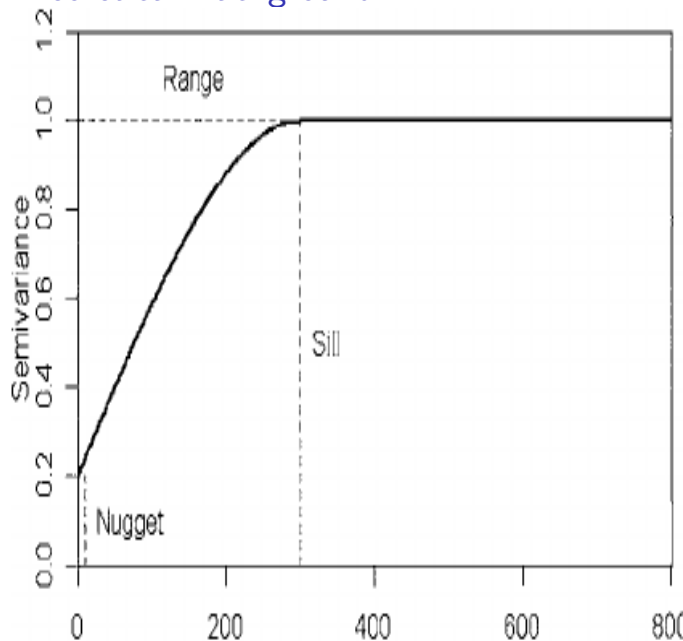
$$\mathbf{C}_{st} = \text{Cov}(z(s, t), z(s + h, t + u))$$

$$\mathbf{C}_{st} = \text{Cov}(z(0, 0), z(h, u))$$

Theoretical Background

- In spatial and spatio-temporal processes, there is a semivariogram which aids in selecting an appropriate model.
- The semivariogram is constructed by using point pairs.
- There would be $\frac{n(n-1)}{2}$ point pairs.
- We construct bins and a range for distance and direction.
- This semivariogram is used to build the Covariance matrix..

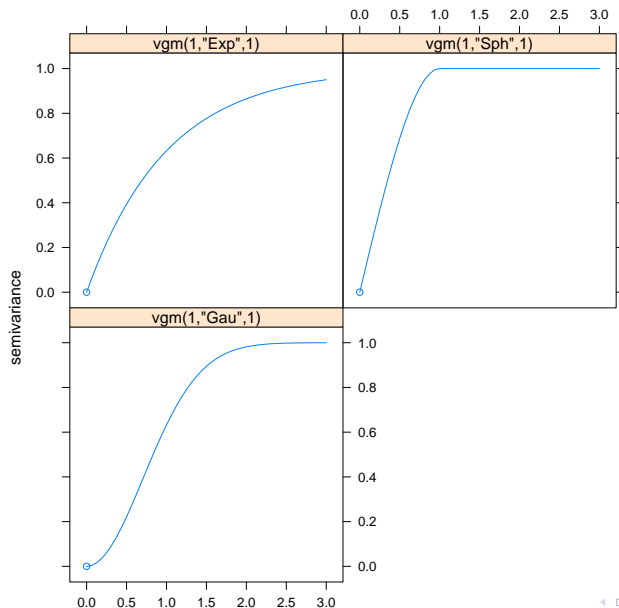
Theoretical Background



Theoretical Background

- Authorized models are fit to the spatial, temporal, and joint aspects of the data set individually.
- These are fit via an automatic process using the minimum mean error sum of squares.
- The potential variogram models are Exponential, Spherical, and Gaussian.

Theoretical Background



Theoretical Background

- I have c as the partial sill, h as the distance, a as the range, and nug as the nugget.
- Exponential

$$\gamma(h) = c[1 - e^{(-\frac{h}{a})}] + nug$$

- Gaussian

$$\gamma(h) = c[1 - e^{(\frac{h}{a})^2}] + nug$$

- Spherical

$$\gamma(h) = \begin{cases} c[\frac{3h}{2a} - \frac{1}{2}(\frac{h}{a})^3] + nug & \text{for } h < a \\ c + nug & \text{for } h \geq a \end{cases}$$

Theoretical Background

- For S-T sets:
- Structure for the Separable model

$$\gamma(h, u)_{\text{sep}} = c\gamma_s(h)\gamma_t(u)$$

- Structure for the Product-sum model

$$\begin{aligned}\gamma(h, u)_{\text{ps}} &= (k \cdot c_t + 1)\gamma_s(h) + (k \cdot c_s + 1)\gamma_t(u) - k\gamma_s(h)\gamma_t(u), \\ c_{st} &= kc_sc_t + c_s + c_t\end{aligned}$$

- Structure for the Sum-metric model

$$\gamma(h, u)_{\text{sm}} = \gamma_s(h) + \gamma_t(u) + \gamma_{\text{joint}}\left(\sqrt{h^2 + (\kappa \cdot u)^2}\right),$$

where κ is an anisotropy parameter.

Empirical Background

- I fit the sample space-time variograms.
- Then I fit 3 space-time models based on the space-time variograms and the individual model results.
- I fit separable, product-sum, and sum-metric models.
- For the new method, I always keep the sum-metric model.
- It is the most appropriate for real world data.
- For the Windows and the Mac old version, I select the best of the 3 models.

Empirical Background

- I obtain the estimates for the sill, nugget, range and the anisotropy parameter.
- I can now calculate the mean, the covariance matrix, and the predicted values.

Empirical Background

- I produce the overall sample mean:

$$\hat{\mu} = (1' \cdot C_{sm}^{-1} \cdot 1)^{-1} \cdot (1' \cdot C_{sm}^{-1} \cdot z)$$

where C_{sm} is

$$C_{sm} = C_s(h) + C_t(u) + C_{joint} \left(\sqrt{h^2 + (\kappa \cdot u)^2} \right).$$

- For a one point forecast, I obtain:

$$\hat{z} = \hat{\mu} + c_0 C_{sm}^{-1} (z - \hat{\mu} 1).$$

This is easily generalized for multiple points in space and time.

The Irish Wind Data Study

- I looked at the Irish Wind data, as discussed in Pebesma(2004), and which originally appeared in Haslett and Raftery (1989).
- The data are daily wind speeds at 12 stations around Ireland.
- The data run from January 1961 - December 1978.
- I began with 3 months, 1 year, and in one year increments until December 1969.

The Irish Wind Data Study

- I ran the existing process on a 2 core MacBook Air, and a 6 core Windows laptop.
- I ran the new process on the 6 core Windows laptop in which R had been compiled with the OpenBLAS package.
- I re-ran our study within the last two weeks with R Version 4.0.0.
- The output is a spatio-temporal grid with interpolated values.

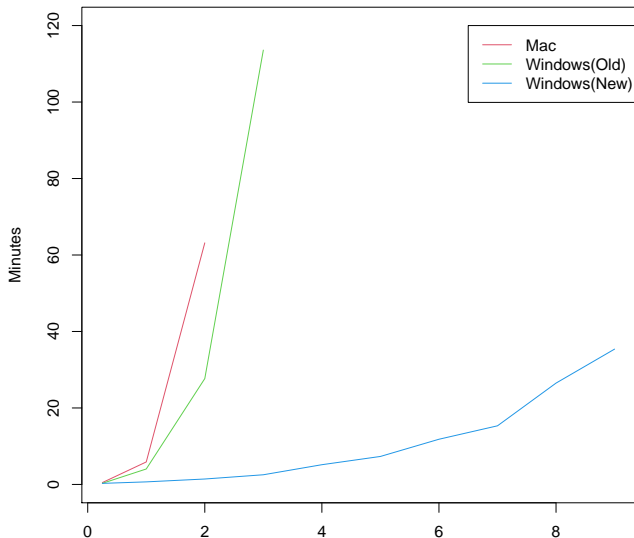
The Irish Wind Data Study

Duration	Mac	Windows(Old)	Windows(New)
3 months	0.48	0.29	0.28
1 year	5.88	4.04	0.67
2 years	63.17	27.68	1.42
3 years		113.59	2.53
4 years			5.16
5 years			7.32
6 years			11.82
7 years			15.33
8 years			26.50
9 years			35.39

Time in Minutes

The Irish Wind Data Study

Results from the Irish Wind Data Set



The Cuba Temperature Data Study

- I looked at 4 locations in Cuba from 2015-2016.
- The new procedure was run on these to produce a map
- The process took 1.37 minutes.
- There was a noticeable measure of anisotropy, from east to west.

A Discussion of the Computer Setup

- Why Fortran? Why not C or C++?
- Initial work on the supercomputer was with C and the rmpi package.
- The speedup was negligible.
- I experimented with Fortran, and the process began to show great promise in terms of time.

A Discussion of the Computer Setup

- The laptop must contain an NVIDIA graphics card.
- I download the Portland Group Inc. (PGI) Community Edition compiler.
- I must also download the latest version of CUDA drivers, from the NVIDIA website.
- I download Microsoft MPI (Message Passing Interface) as well.
- Finally, I obtain the free version of Visual Studio (2019).

A Discussion of the Computer Setup

- I developed an R package for this new process.
- Everything can be done with our free software.
- I begin with Avraham Adler's website on building OpenBLAS.
- Then I compile R with this OpenBLAS.

A Discussion of the Computer Setup

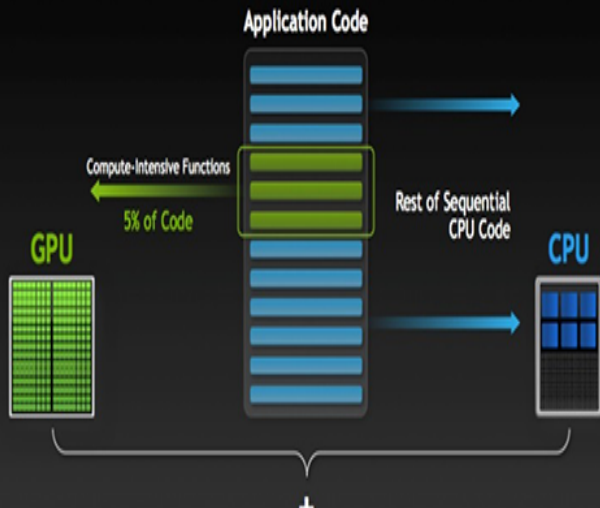
- Typically, when compiling an R package with foreign programs in the src directory, the Makevars file is constructed in that src directory,
- However, when using the PGI compiler, the Makevars does not work.
- For compilation of the R package, I use the Makeconf file, which is located in the R-4.0.0/etc/x64 directory.
- The compiler and the flags are updated in the Makeconf file to reflect the PGI compiler.
- If producing a single program with OpenACC or MPI, we can use the PGI compiler from the command line.

A Discussion of the Computer Setup

- How does OpenACC work?
- This is done via Fortran with OpenACC directives.
- The OpenACC moves the data from the host (CPU) to the GPU device for quick processing.

A Discussion of the Computer Setup

How GPU Acceleration Works



A Discussion of the Computer Setup

```
!$acc loop seq
do i=1,n1
  do j=1,n1
    mat1(i,j) = (((x(i)-x(j))**2 + (y(i)-y(j))**2))**0.5
  enddo
enddo
!$acc end loop
```

A Discussion of the Computer Setup: OpenACC Example

```
subroutine ac1(n,y,xt)
!DEC$ ATTRIBUTES DLLEXPORT :: ac1
use cudafor
implicit none
integer :: n, y(n),i
real :: xt,xa,xb
xa=0.0;xb=0.0
call cpu_time(xa)
!$acc loop seq
    do i=1,n
        y(i) = i + 2
    enddo
!$acc end loop
call cpu_time(xb)
xt = xb-xa
end subroutine ac1
```

A Discussion of the Computer Setup: OpenACC Example

```
> dyn.load("cudarama.dll")
> n <- 1024^2
> y <- numeric(length=n)
> xt1 <- .Fortran("ac1",n=as.integer(n),y=as.integer(0),
xt=as.single(0.0))
> xt1$xt
[1] 0.002
attr(,"Csingle")
[1] TRUE
```

A Discussion of the Computer Setup

- I utilize our estimates for sill, nugget, etc.
- I calculate the spatial, temporal, and joint matrices and combine them.
- I use our OpenACC subroutines to produce the matrices.
- I actually return the matrices as an extended vectors.

A Discussion of the Computer Setup

- I use MPI to initialize our next step in the procedure.
- The pbdDMAT package uses MPI, and will split our matrices into submatrices based on the process rank.
- This package creates a special structure for large matrices.

A Discussion of the Computer Setup

- How does MPI work? (in a nutshell)
- There is an initialization of the processes.
- There is a verification of the “world”; that is, the number of processes.
- The program checks for the rank of the process. This runs from 0 to $(n - 1)$.
- The calculations are made, based on the rank.
- There is a finalization call.

A Discussion of the Computer Setup

```
        subroutine finit(ierr)
!DEC$ ATTRIBUTES DLLEXPORT :: finit
implicit none
include 'mpif.h'

integer :: ierr

call MPI_INIT(ierr)

end subroutine finit
```

A Discussion of the Computer Setup

```
init1 <- function(ierror=0)
return(.Fortran("finit",as.integer(ierror)))
```

A Discussion of the Computer Setup: MPI Example

```
mpitest2a <- function(ierror=0,comm=1,size=1,rank=0)
library(rmpiFort)
dyn.load("sum1.dll")
y <- 0
nSteps <- 2048
  x <- 0
  integral <- 0
  xlow <- 0
  xupp <- 1
  dx <- (xupp - xlow)/nSteps
  nb <- 5
zz <- init1(ierror=ierror)
sz1 <- univ1(comm=comm,size=size)[[2]]
rk1 <- unlist(rank1(comm=comm,rank=rank)[[2]])
```

```
iMin <- (rk1*nSteps)/sz1  
iMax <- ((rk1+1)*nSteps)/sz1
```

```
for(i in iMin:(iMax-1))  
  x <- xlow + dx*(i+0.5)  
  integral <- integral + (x^nb)*dx
```

```
total_int <- unlist(.Fortran("sum1",x=as.single(integral,  
rank=as.integer(rk1),  
y=as.single(y))[[3]]))
```

```
if(rk1==0)  
  print("final")  
  print(total_int)
```

```
za <- fin1(ierror=ierror)
```

```
subroutine sum1(x,rank,y)
!DEC$ ATTRIBUTES DLLEXPORT :: sum1
implicit none
include 'mpif.h'
```

```
integer rank,size,ierror,tag,status(MPI_STATUS_SIZE),i,np
real :: x
real, intent(inout) :: y
```

```
call MPI_REDUCE(x,y,1,MPI_REAL,MPI_SUM,0,MPI_COMM_WORLD
```

```
end subroutine sum1
```

```
> mpiexec -np 4 Rscript nextmpi.in >mpi2a.out  
>cat mpi2a.out
```

```
[1] "final"
```

```
[1] 0.1666666
```

```
attr(,"Csingle")
```

```
[1] TRUE
```

A Discussion of the Computer Setup

- The main function in our process is inverting the final covariance matrix.
- The pbdDMAT package makes this possible.
- In the final segment of our study, I am inverting a 39420×39420 matrix on a laptop.
- This happens, along with all of the other calculations, in 35 minutes.

A Discussion of the Computer Setup

- A Quick Note regarding CUDA Fortran: Does it work with R?
- It does!
- However, it involves construction of numbers of threads and blocks, which can be quite complicated.
- OpenACC can accomplish many similar computations and is easier to work with.

Conclusion

- The speedup was unexpectedly good.
- The data size was also better than I thought.
- I would like to try this on a laptop with more cores and more memory.

Thank you!

- Questions, please?