LatticeKrigExample

Doug Nychka

2023-07-17

Introduction

This is a short example of using the LatticeKrig package to fit a large spatial data set and to compare timing with the standard method.

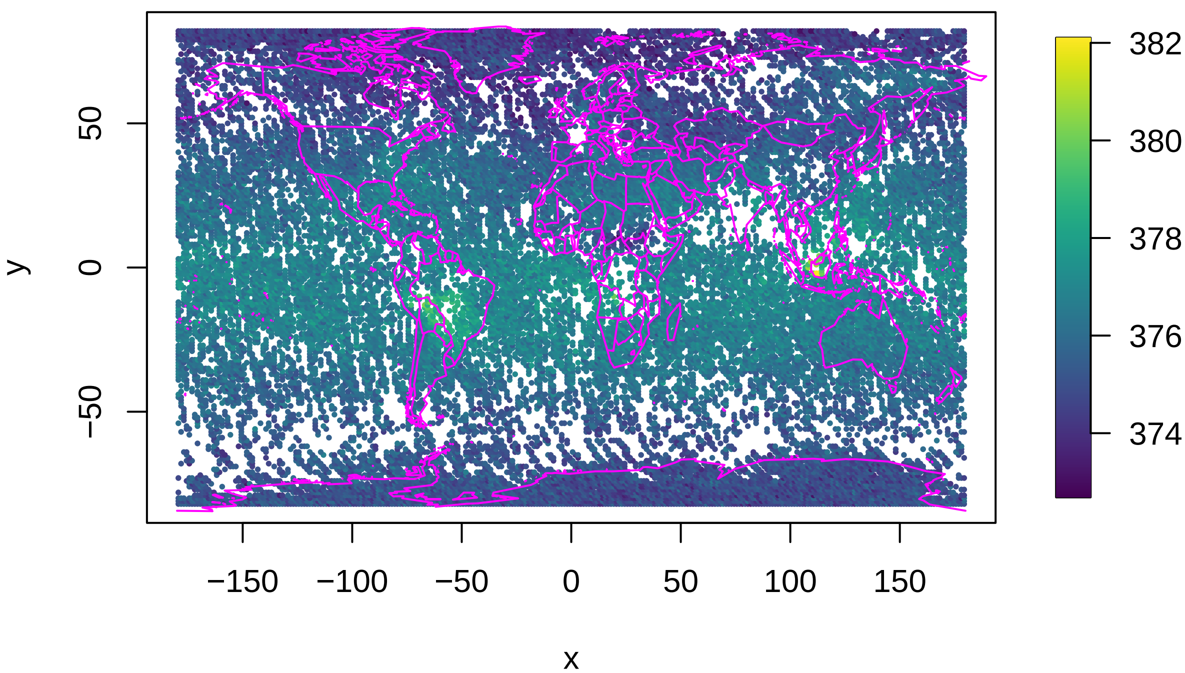
CO2 satelite data

Load the raw datra and plot. This is about 26K locations irregularly sampled

```
s<- CO2$lon.lat
z<- CO2$y
dim( s)
```

```
## [1] 26633     2
```

```
bubblePlot( s,z, highlight=FALSE, size=.4)
world( add=TRUE, col="magenta")
```



Subset using spatialProcess

Small subset that runs quickly using the standard spatial statistics model. A thin plate spline type covariance is used that does not require a range parameter

Subset of ~ 1900 locations over the US.

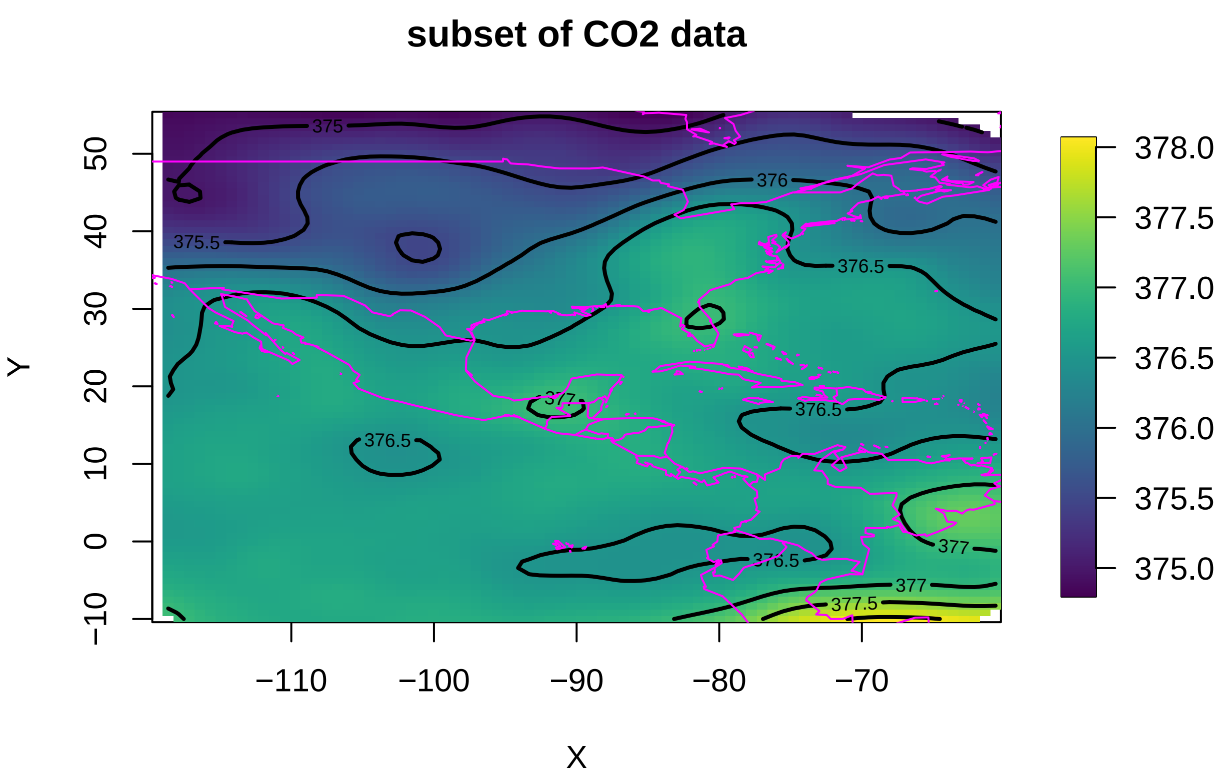
```
# 1884 locations
ind2<- which(
       s[,1]>= -120 & s[,1] <= -60 &
       s[,2]>=  -10 & s[,2] <=  55
        )

system.time(
  fit2<- spatialProcess(s[ind2,], z[ind2], cov.function="Tps.cov" )
)
```

```
##    user  system elapsed
##   6.076   0.729   6.807
```

For n=1884 get about 7 seconds a conservative lower bound for the full data set is at least 3 hours based on a timing of 35 seconds for n= 3800 and then scaling up to 26000 based on cubic order complexity.

The fitted surface for this subset

```
surface( fit2)
world( add=TRUE, col="magenta")
title("subset of CO2 data")
```



To get more control over the prediction one can specify the grid and

LatticeKrig fit

Use the default in the function but set the equivalent of the range parameter to be large. This approximates a thin plate spline model.

```r
system.time(
  fit4<- LatticeKrig(s, z, a.wght = 4.01 )
)
```

```
##    user  system elapsed
##  34.599   0.727  35.326
```

```r
# for ~27K locations get about  35 seconds
```

Summarize the fit. Note that the σ^2 and τ^2 can be interpreted in the usual way as process and nugget variances.

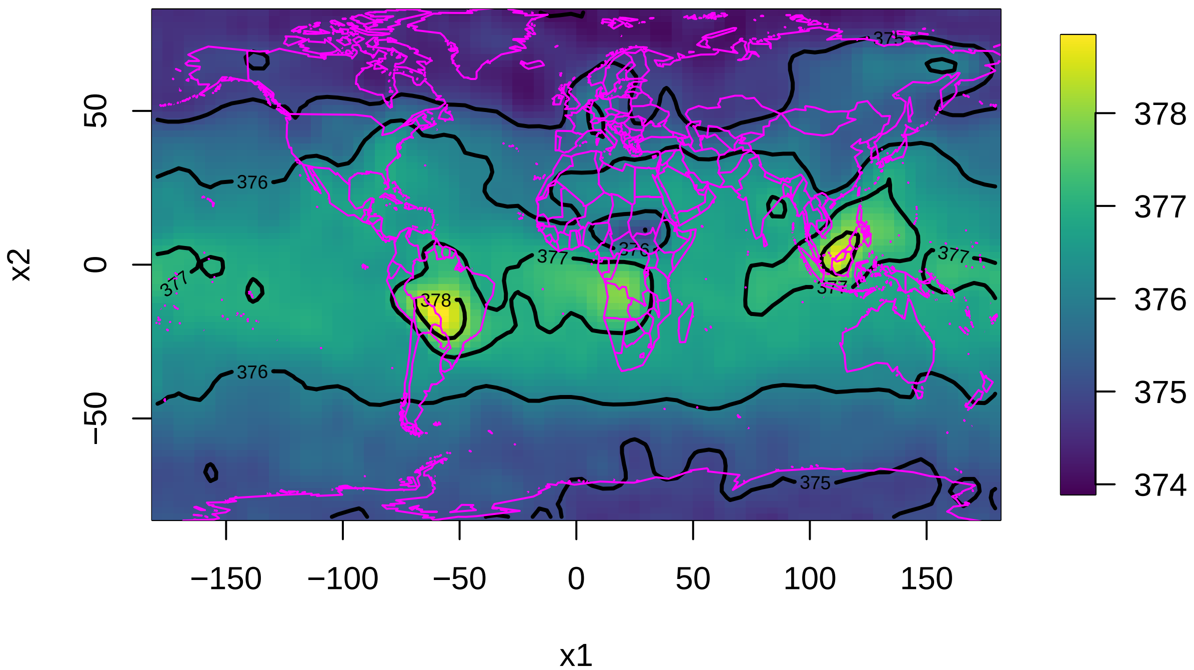
```r
# summary of fit
fit4
```

```
## Call:
## LatticeKrig(x = s, y = z, a.wght = 4.01)
##
##
##  Number of Observations:                     26633
##  Number of parameters in the fixed component 3
##   Effective degrees of freedom (EDF)         787.4
##     Standard Error of EDF estimate:          6.268
##  MLE sigma                                   0.5069
##  MLE rho                                     6.296
##  MLE lambda = sigma^2/rho                    0.04081
##
## Fixed part of model is a polynomial of degree 1 (m-1)
##
## Summary of estimated fixed model coefficients
##                Estimate  Std. Error      t value  Pr(>|t|)
## Intercept  3.749934e+02 1.144268973 327.71436617 0.0000000
## x1        -6.171181e-04 0.006787081  -0.09092541 0.9275525
## x2        -6.771584e-03 0.009510819  -0.71198746 0.4764788
##  Standard errors are  based on generalized LS
##  and for covariance parameters fixed at the estimated values
##
## Basis function :  Radial
## Basis function used:  WendlandFunction
## Distance metric:  Euclidean
##
## Lattice summary:
## 3  Level(s) 16050 basis functions with overlap of  2.5 (lattice units)
##
##  Level Lattice points Spacing
##      1           1242 10.2500
##      2           3483  5.1250
##      3          11325  2.5625
```

```
##
## Nonzero entries in Ridge regression matrix 9848348
```

Examine the global, fitted surface.

```
surface( fit4)
world( add=TRUE, col="magenta")
```



Now fit a more accurate model that uses basis functions defined foa spherical geometry. (See lattice figure in the lecture showing points generated from subdividing an isohedron.)

```
# same mulitresolution weights as the 2D  defaults
alpha<- fit4$LKinfo$alpha
LKinfo<- LKrigSetup( s, startingLevel=3 , nlevel=3,
                                           a.wght=1.05,
                            alpha=alpha,
                                           LKGeometry="LKSphere" )
```

List out the model specification

```
LKinfo
```

```
## Classes for this object are:  LKinfo LKSphere
## The second class usually will indicate the geometry
##      e.g.  2-d rectangle is  LKRectangle
##
## Some details on spatial autoregression flags:
## stationary:
## first order (by level):
## isotropic:  TRUE
##
```

```
## Ranges of locations in raw scale:
##          [,1] [,2]
## [1,] -179.375  -82
## [2,]  179.375   82
##
## Logical (collapseFixedEffect) if fixed effects will be pooled: FALSE
##
## Number of levels: 3
## delta scalings: 0.352 0.176 0.088
## with an overlap parameter of  2.5
## alpha:  0.7619048 0.1904762 0.04761905
##
## a.wght:  1.05 1.05 1.05
##
## Basis  type: Radial using  WendlandFunction  and GreatCircle  distance.
## Basis functions will be normalized
##
## Total number of basis functions  3285
##  Level Basis size
##      1        153
##      2        625
##      3       2507
##
## Lambda value:  NA
```

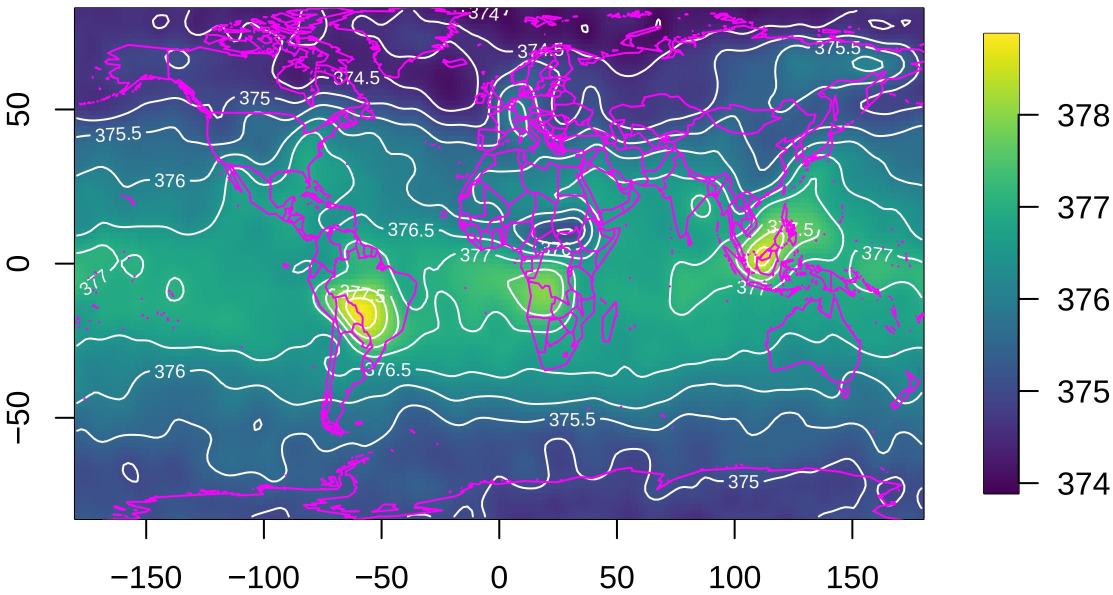
Now fit this model. It takes about 4 times longer because the spherical basis functions are not as efficient to work. For example the nodes are not on a regular grid.

```
system.time(
 fit5<- LatticeKrig(s, z, LKinfo=LKinfo)
)
```

```
##    user  system elapsed
##  22.653   1.955  24.612
```

Examine the surface as use this to show how to get the predited surface.

```
 fHat<- predictSurface( fit4, nx=200, ny=100)
 imagePlot( fHat, col=viridis( 256))
 contour( fHat, add=TRUE, col="white")
 world( add=TRUE, col="magenta")
```



On your own

1. How different are the fits using the usual 2D model and the spherical one?
2. Do you see any edge effect in fit4 because it does not join the right boundary with left one?