Climate data and spatial data methods

Doug Nychka[1]

July 23, 2015

[1] Thanks to Will and Dorit

Background

- Look at the climate of Colorado using some graphics
- Learn how to fit surfaces to data

Loading packages and data

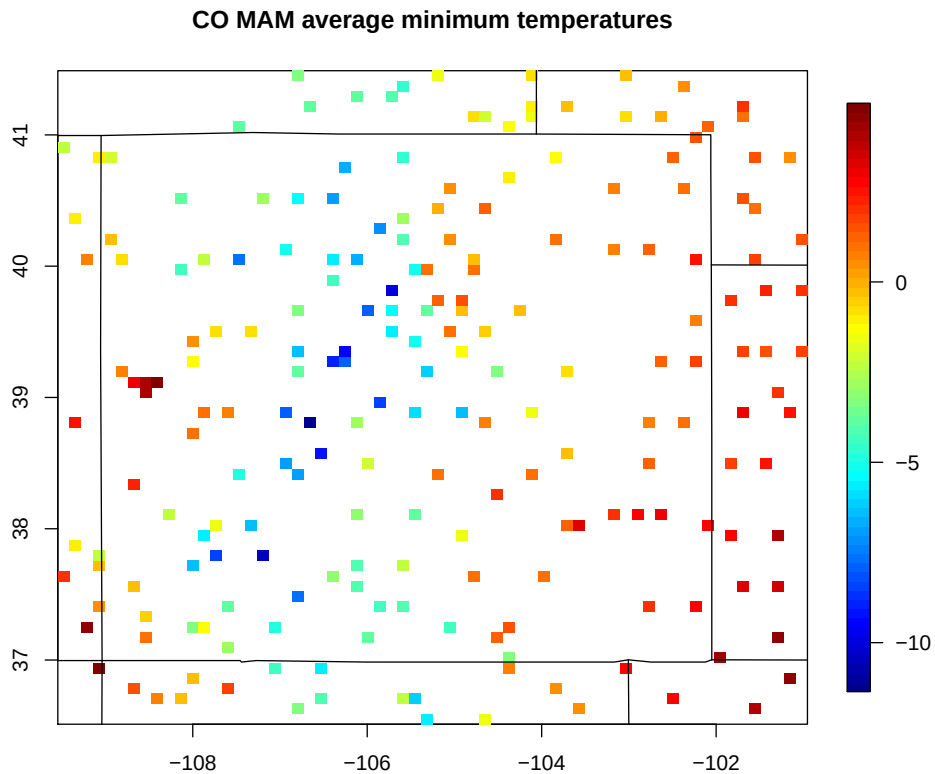
```
# some setup
library( fields)

## Loading required package:  spam
## Loading required package:  grid
## Spam version 1.0-1 (2014-09-09) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g.  'help( chol.spam)'.
##
## Attaching package:  'spam'
##
## The following objects are masked from 'package:base':
##
##    backsolve, forwardsolve
##
## Loading required package:  maps
```

Setting up the data for easy typing

```
data(COmonthlyMet)
ls()
 x<- CO.loc
 y<- CO.tmin.MAM.climate
 elev<- CO.elev
 good<- !is.na( y)
 x<- x[good,]
 y<- y[good]
 elev<- elev[good]
# interpolate elevations to a useful a grid (will use these later)
 NGRID <- 50
 COGrid<- fields.x.to.grid( x, nx=NGRID, ny=NGRID)
 names(COGrid)<- c("x","y")
 data( RMelevation) # elevations for the ROcky mountain area.
 elevGrid<- interp.surface.grid( RMelevation, COGrid )
```

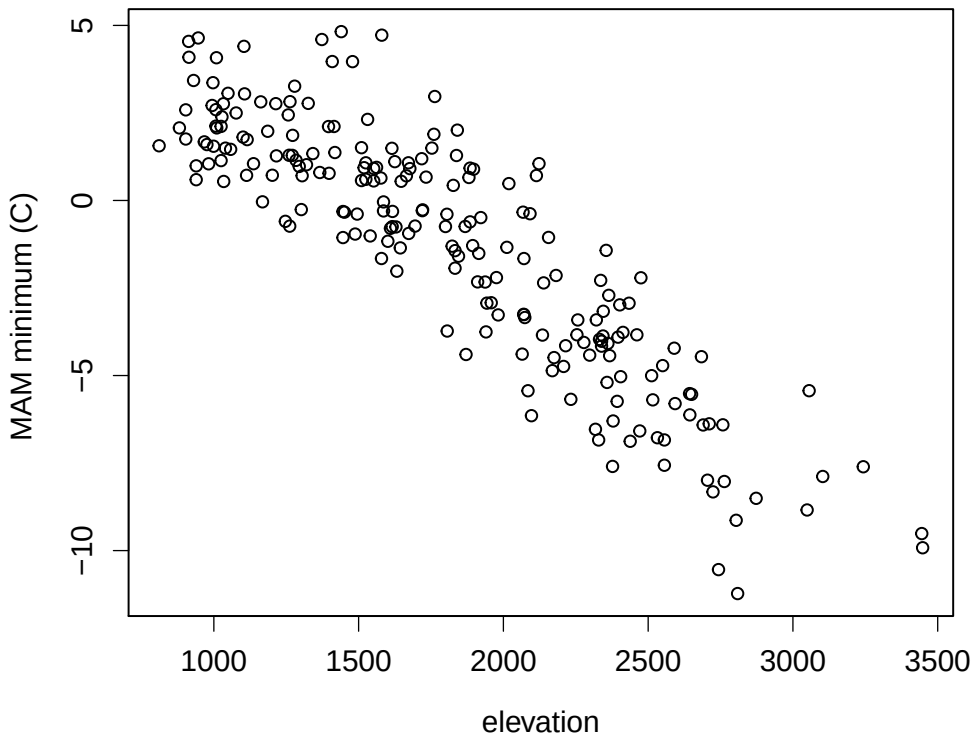
Colorado data (lon,lat)

```
quilt.plot( x, y)
US( add=TRUE)
title("CO MAM average minimum temperatures")
```



Colorado data – elevation

```
fields.style()
plot( elev, y, xlab="elevation", ylab="MAM minimum (C) ")
```



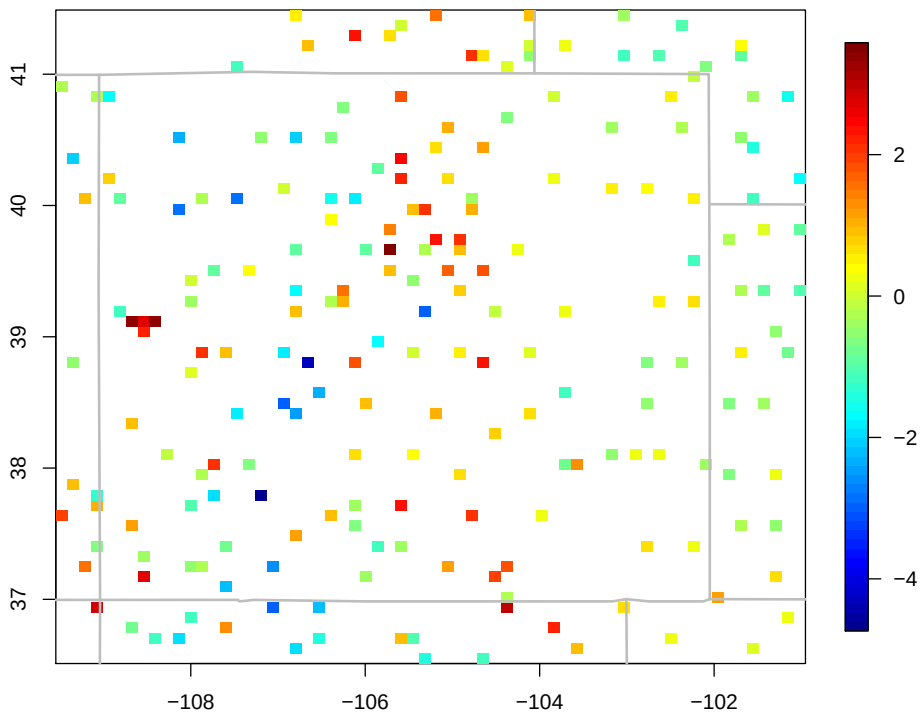
lm fit first

```
 X<- cbind( x, elev)
 lmObj<- lm( y ~ lon + lat + elev, data=X )
 summary( lmObj)

##
## Call:
## lm(formula = y ~ lon + lat + elev, data = X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6706 -0.8541 -0.1308  0.9088  3.5183
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.9372343  5.9584509   1.332    0.184
## lon         -0.1982295  0.0499149  -3.971 9.83e-05 ***
## lat         -0.4951096  0.0691068  -7.164 1.32e-11 ***
## elev        -0.0059590  0.0002029 -29.374  < 2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.412 on 209 degrees of freedom
## Multiple R-squared:  0.8415, Adjusted R-squared:  0.8393
## F-statistic:   370 on 3 and 209 DF,  p-value: < 2.2e-16
```

Check residuals

Look for any spatial structure having removed the linear effects of lon,lat and elevation.

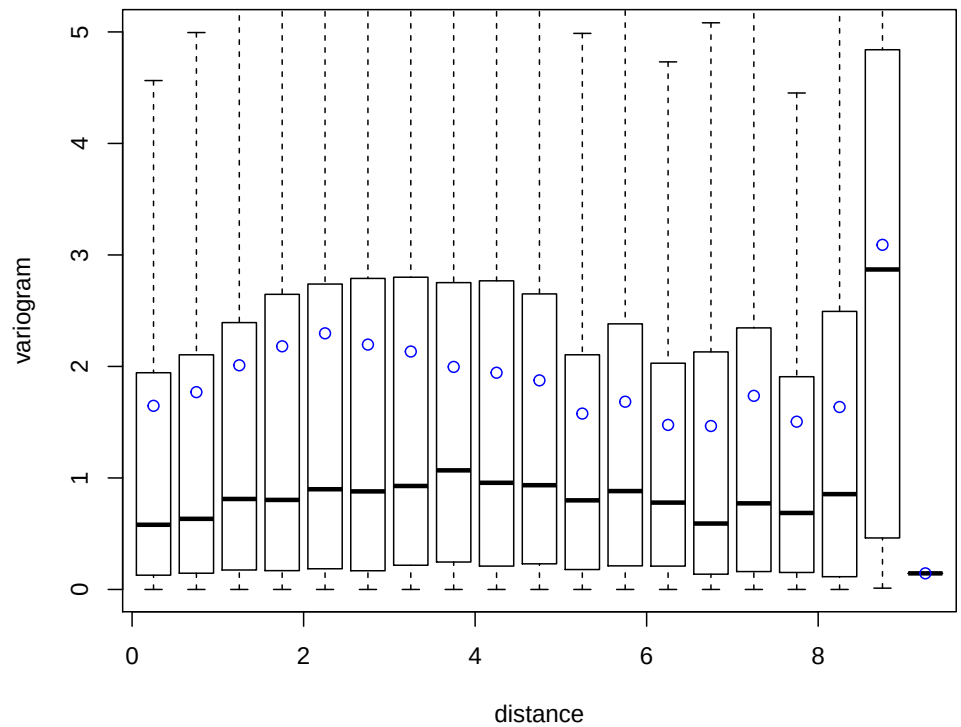
```
quilt.plot( x, lmObj$residuals)
US( add=TRUE, col="grey", lwd=2)
```



Variogram of residuals

Look for any spatial structure using a variogram

```
 look<- vgram( x, lmObj$residuals, N=30)
 bplot.xy( look$d, look$vgram, breaks= look$breaks,
           outline=FALSE, ylim=c(0,5),
           xlab="distance", ylab="variogram")
points( look$centers, look$stats[2,], col="blue")
```



This is also a general example how to find estimate a variogram in fields.

Complete fit of surface

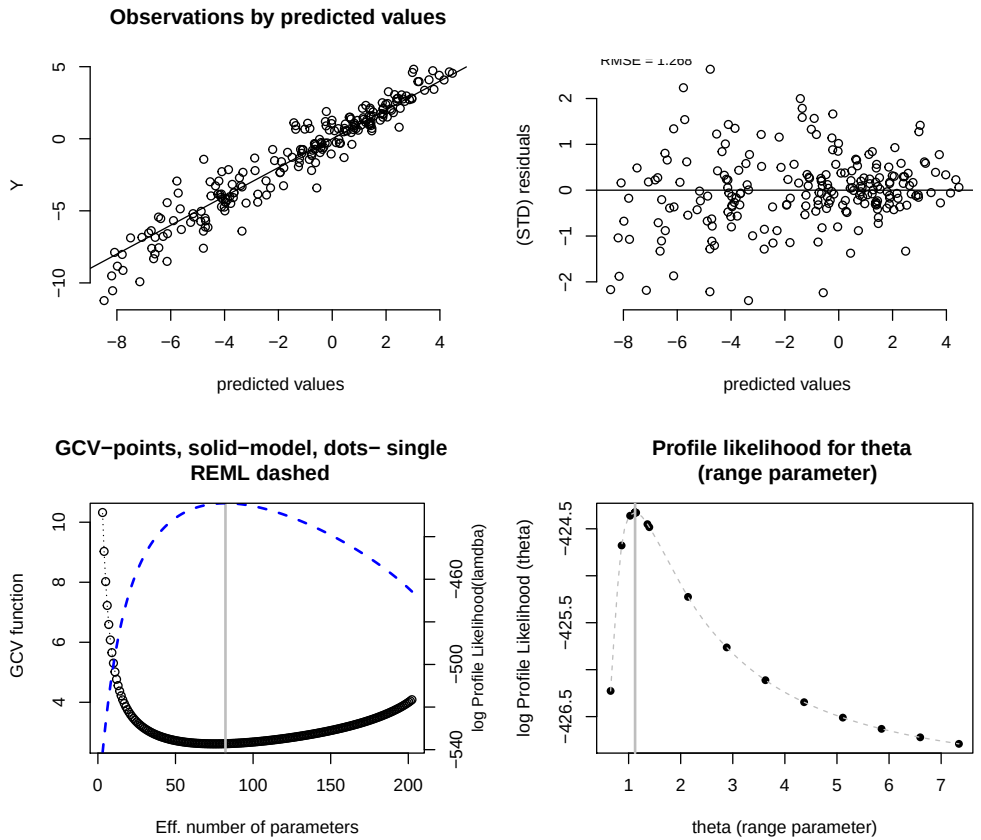
`spatialProcess`

:Easy to use but it makes lots of choices for you!
Assumes a Matern with smoothness 1.0.

```
fit1<- spatialProcess( x,y)
print( fit1)

## Call:
## spatialProcess(x = x, y = y)
##
##  Number of Observations:                 213
##  Number of parameters in the null space 3
##  Parameters for fixed spatial drift     3
##  Model degrees of freedom:               82.2
##  Residual degrees of freedom:            130.8
##  GCV estimate for sigma:                 1.268
##  MLE for sigma:                          1.259
##  MLE for rho:                            10.52
##  lambda                                  0.15
##  User supplied rho                       NA
##  User supplied sigma^2                   NA
## Summary of estimates:
##                lambda      trA      GCV     shat
## GCV        0.1869168 75.50444 2.614782 1.299189
## GCV.model         NA       NA       NA       NA
## GCV.one    0.1869168 75.50444 2.614782 1.299189
```

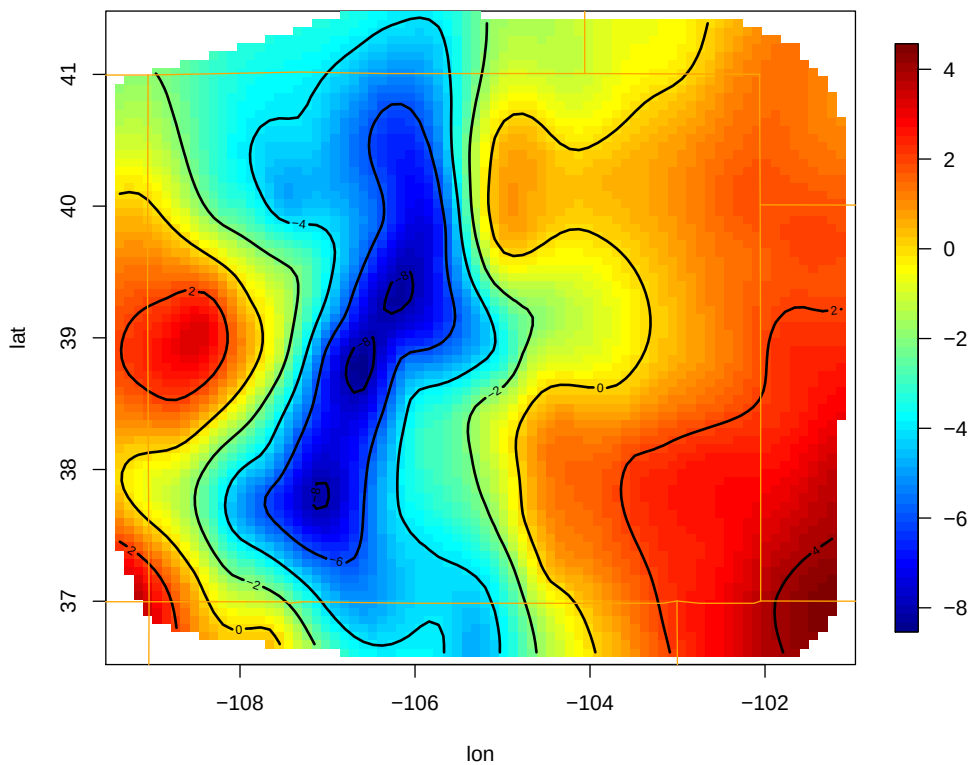
Some diagnostic plots

```
set.panel( 2,2)

## plot window will lay out plots in a 2 by 2 matrix

plot( fit1)
```



take a look at the surface

```
surface( fit1)
US(add= TRUE)
```



Smooth prediction because elevations not included

Adding elevation

Just need to add in the elevation covariate to the model. Note: lon and lat are automatically included by default.
But increase the number of range parameters searched to (from 10 to 20).

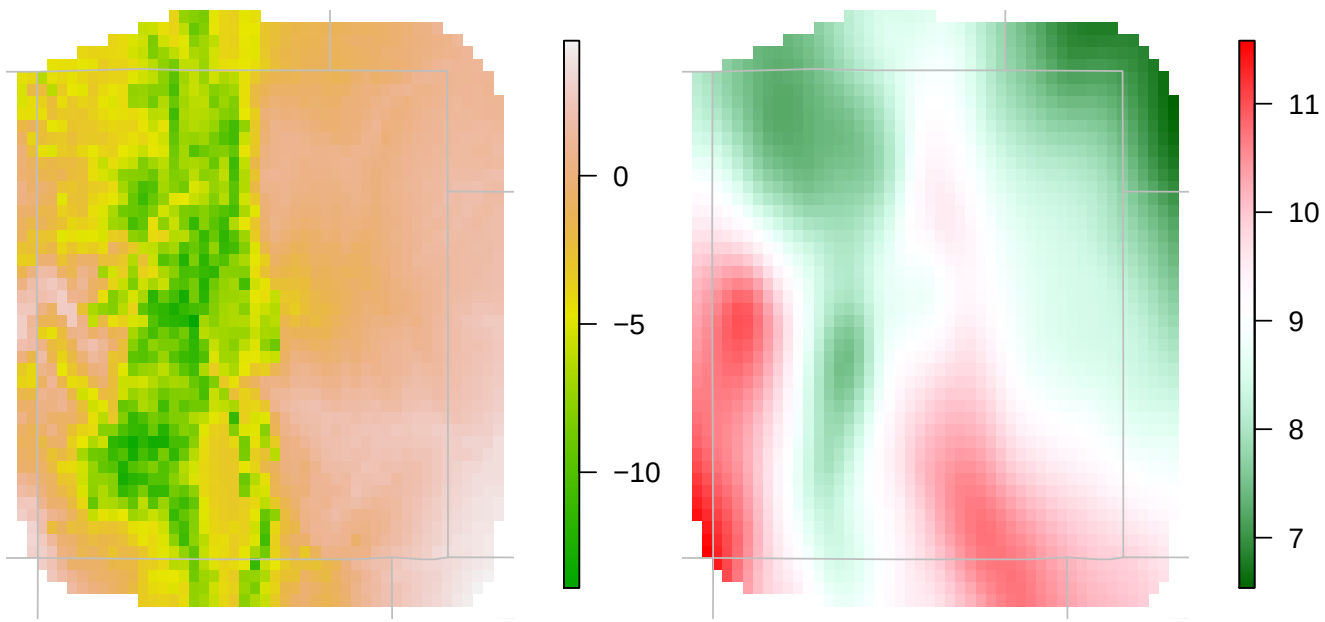
```
fit1E<- spatialProcess( x,y, Z= elev, ngrid=20)
```

Evaluate surface on same grid as elevations.

```
surFull<- predictSurface( fit1E, grid.list= COGrid,
                          ZGrid= elevGrid)
surSmooth<-predictSurface( fit1E, grid.list= COGrid, drop.Z=TRUE)
```

Full estimate and smooth component

```
 set.panel( 1,2)

## plot window will lay out plots in a 1 by 2 matrix

par( mar=c(1,1,1,4))
image.plot( surFull, col=terrain.colors(256),
            axes=FALSE,xlab="", ylab="")
US(add=TRUE, col="grey")
image.plot( surSmooth, col=two.colors(256),
            axes=FALSE,xlab="", ylab="")
US(add=TRUE, col="grey")
```



Ten member ensemble for uncertanty

```
# unroll the grids in locations and vectors
 COGridPoints<- make.surface.grid( COGrid)
 COGridElevations<- c(elevGrid$z)
 dim( COGRidPoints)
 SEout<- sim.Krig( fit1E,
              xp=COGridPoints,  Z= COGridElevations, M= 10, drop.Z=TRUE)
 dim( SEout)
```

Look at ensemble members

as.surface is a handy function to convert the values in vector form back to a grid. It uses the information attached to the grid points to do this.

```
set.panel( 3,3)
  ctab<- rainbow( 256)
  image.plot( surSmooth, zlim=c(3.5,14.5), col=ctab)
  US(add=TRUE)
  par( mar=c(3,3,1,1))
title("Best estimate", adj=0)
  for( k in 1:8){
      image( as.surface( COGridPoints, SEout[k,]),
                zlim =c(3.5,14.5), axes=FALSE, col=ctab)
       US(add=TRUE)
      title( paste(k), adj=0)
  }
```

estimate and 8 members