Fun Cross Validation Test on a Spatial Dataset

Antony Sikorski

First, a little setup to get our spatial data. We are looking at the ozone measurement data set that is available in the fields package:

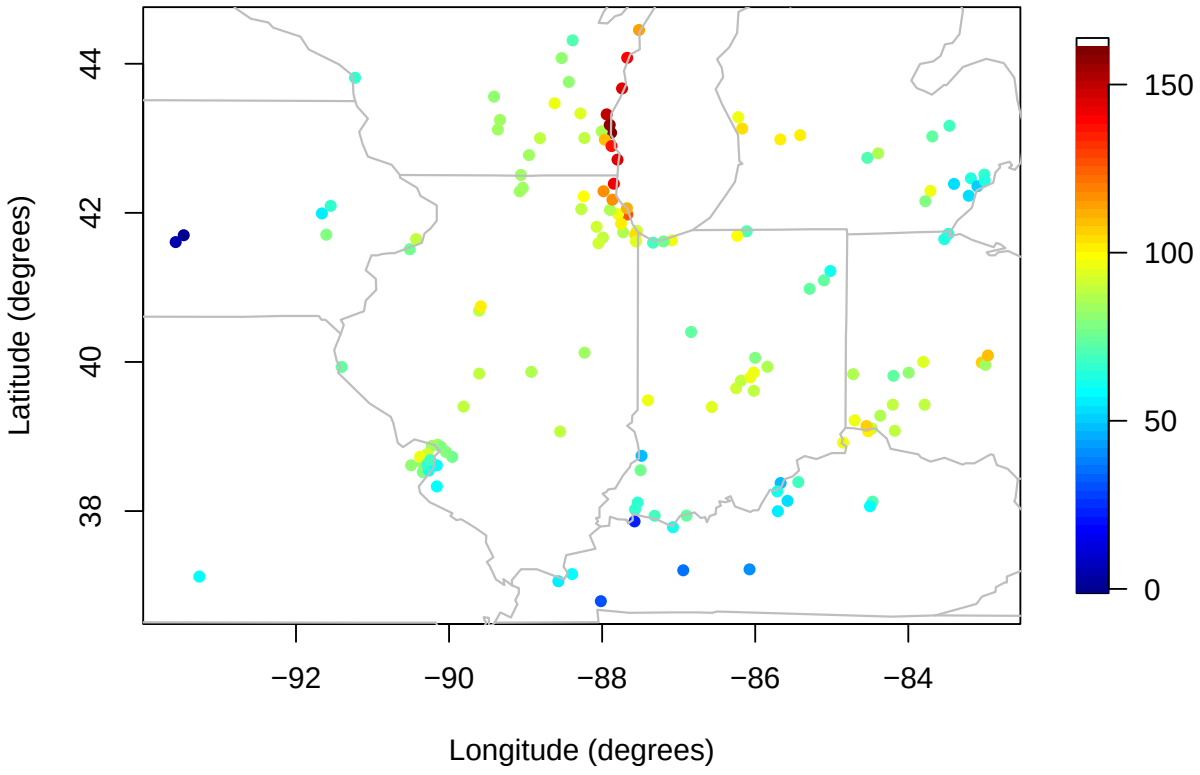
```
suppressMessages(library( fields))
suppressMessages(library(ggplot2))
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
#removing NA's
data(ozone2)
s<- ozone2$lon.lat
z<- ozone2$y[16,]
good<- !is.na( z)
s<- s[good,]
z<- z[good]
```

Quick visualization of our dataset (ozone measurements at long/lat locations in the US):

```
bubblePlot(s,z, col=c(tim.colors(), alpha = .7), highlight=FALSE,
           xlab="Longitude (degrees)", ylab="Latitude (degrees)", size=0.9)
US( add=TRUE, col="grey", lwd = 1.2)
```



Now we use fields to do a spatial process fit (the function spatialProcess makes this very easy, the important part is understanding the mathematics behind it.. most importantly Kriging and MLE theory). We assume that the Matern covariance kernel is appropriate for this process. Let's take a look at the massive amount of work that fields did:

```
fit <- spatialProcess(s,z, cov.args = list(Covariance = "Matern", smoothness = 1))
fit
```

```
## CALL:
## spatialProcess(x = s, y = z, cov.args = list(Covariance = "Matern",
##     smoothness = 1))
##
##  SUMMARY OF MODEL FIT:
##
##  Number of Observations:                     147
##  Degree of polynomial in fixed part:         1
##  Total number of parameters in fixed part:   3
##  sigma Process stan. dev:                    22.36
##  tau  Nugget stan. dev:                      9.472
##  lambda   tau^2/sigma^2:                     0.1795
##  aRange parameter (in units of distance):    0.6999
##  Approx.  degrees of freedom for curve       64.31
##     Standard Error of df estimate:           1.977
##  log Likelihood:                             -610.8674172117
##  log Likelihood REML:                        -616.926928166435
##
##  ESTIMATED COEFFICIENTS FOR FIXED PART:
##
##    estimate      SE pValue
## d1  181.100 193.200 0.3485
## d2    2.957   1.817 0.1037
## d3    3.730   2.434 0.1255
##
##  COVARIANCE MODEL: stationary.cov
##   Covariance function:  Matern
##    Non-default covariance arguments and their values
## Covariance :
## [1] "Matern"
## smoothness :
## [1] 1
## aRange :
## [1] 0.6998657
## onlyUpper :
## [1] FALSE
## distMat :
## [1] NA
## Nonzero entries in covariance matrix  21609
##
## SUMMARY FROM Max. Likelihood ESTIMATION:
## Parameters found from optim:
##    lambda    aRange
## 0.1795135 0.6998657
## Approx. confidence intervals for MLE(s)
##          lower95%  upper95%
```

```
## lambda 0.09420382 0.3420784
## aRange 0.40298679 1.2154543
##
##  Note: MLEs for  tau and sigma found analytically from lambda
##
## Summary from estimation:
## lnProfileLike.FULL lnProfileREML.FULL        lnLike.FULL        lnREML.FULL
##       -610.8674172       -616.9269282                 NA                 NA
##              lambda                tau             sigma2             aRange
##           0.1795135          9.4720350        499.7922647          0.6998657
##              eff.df                GCV
##          64.3099528        155.9547263
```

We now do a 90/10 split, and investigate our RMSE of the testing data. We use a 10 fold cross-validation strategy to analyze the differences in the standard vs. the model based estimates for the RMSE.

```
#10 fold CV for RMSE
N<- nrow( s)
about10Percent<- round(.1*N)
len <- 10
trials <- seq(1:len)

#will be appending to these lists
regRMSEs <- c()
modelRMSEs <- c()

for (i in 1:len){

  #felt clever for the seed iteration even though it's so simple
  set.seed(777 + 13*i)

  #shuffling data indices
  IShuffle <- sample( 1:N, N, replace=FALSE)
  I <- IShuffle[1: about10Percent ]

  #90/10 split into training and testing using the shuffled indices
  trainLoc <- s[-I,]
  trainData <- z[-I]
  testLoc <- s[I,]
  testData <- z[I]

  #Out of sample RMSE
  tempFit <- spatialProcess(trainLoc, trainData, smoothness = 1)
  tempPred <- predict(tempFit, testLoc)
  regRMSE <- sqrt(mean((tempPred - testData)^2))
  regRMSEs <- append(regRMSEs, regRMSE)

  #Model Based out of sample RMSE
  SE10 <- predictSE( tempFit,testLoc )
  tauHat <- tempFit$summary["tau"]
  modelRMSE <- sqrt( mean( SE10^2 + tauHat^2))
  modelRMSEs <- append(modelRMSEs, modelRMSE)
}
```

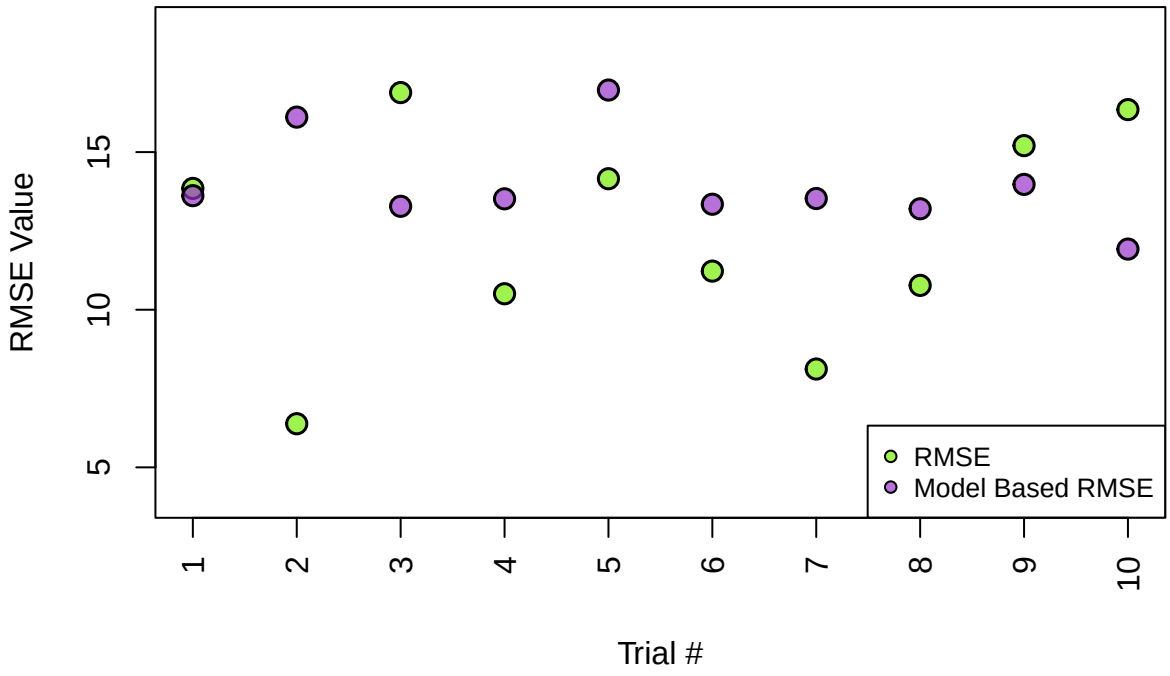
Now we can plot our results to compare the two:

```
#Plotting results
plot(trials, regRMSEs, pch = 21, bg = alpha("chartreuse2", 0.7),
     lwd = 1.5,
     cex = 1.4,
     xlab = "Trial #",
     ylab = "RMSE Value",
     ylim = c(4,19),
     xaxt = "n")

points(trials, modelRMSEs,
       pch = 21,
       bg = alpha("darkorchid", 0.7),
       lwd = 1.5,
       cex = 1.4)

legend("bottomright",
       legend = c("RMSE", "Model Based RMSE"),
       pch = 21,
       pt.bg = c(alpha("chartreuse2", 0.7), alpha("darkorchid", 0.7)),
       cex = 0.8)

axis(1, at = trials, las = 2)
```



```
#Standard deviations of the RMSEs
sd(modelRMSEs)
```

```
## [1] 1.480498
```

```
sd(regRMSEs)
```

```
## [1] 3.510026
```

It is clear that the model based prediction RMSE's have much less variability (more reliability), which is exactly what we expected. The model based predictions are effectively predicting an average surface, while the regular ones are making point-wise/single predictions. Naturally, the set of average surface predictions will have a lower root mean squared error.