Recovering_rf_mixture

The objective of these simulations are to examine whether mixture parameters are recoverable with spatial data. We'll start by simulating some data on a grid from two gaussian random fields. We will initially assume that the fields have the same shape/scale parameters of the gaussian covariance function, but the variances are different – this allows one field to be normal, and the other fat-tailed.

```
library(cluster)
library(mvtnorm)
```

```
grid = as.matrix(expand.grid("lon" = seq(5,15,1), "lat" = seq(5,15,1)))
nLocs = dim(grid)[1]
```

Approximating data with spatial random field

We could use RandomField, or other packages to do this. Initially we'll just simulate data using the estimation model. Use pam() to choose a number of knots on the grid. We'll specify 20 initially. We'll also jitter them slightly so knot locations don't fall exactly on stations.

```
nKnots = 20
knots = jitter(pam(grid,nKnots)$medoids)
```

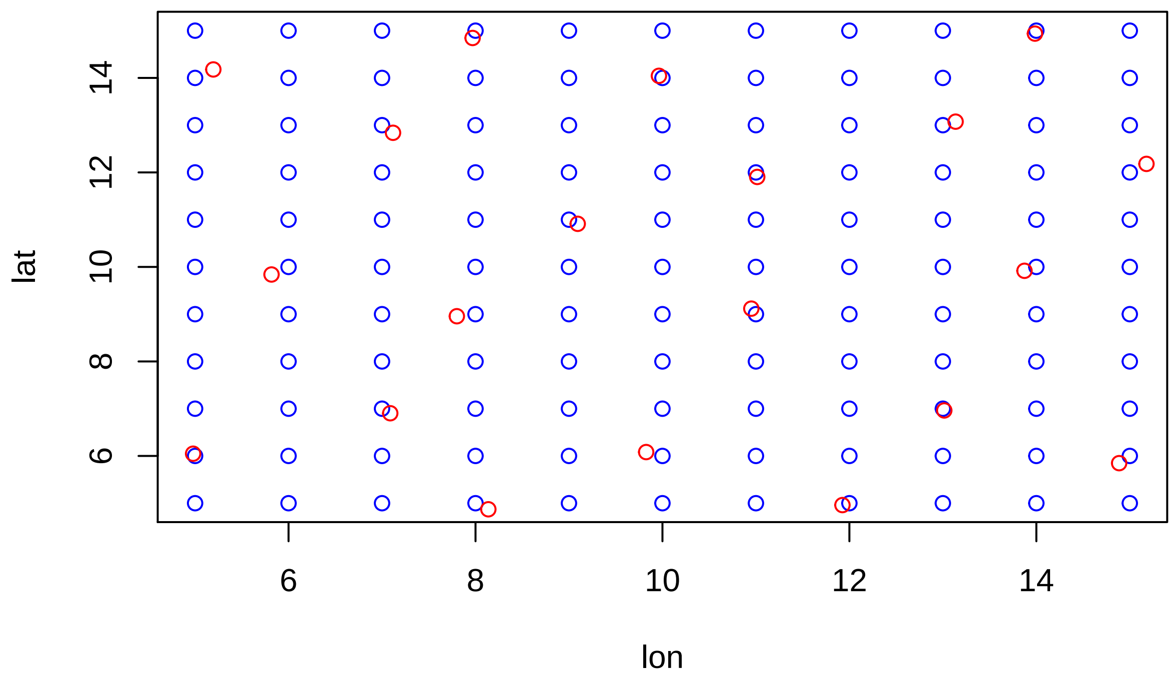

Figure 1: Simulated grid (blue) and knots for random effects (red)

Approximating data with spatial random field

Initially, we'll assume all data is observed in the same year, so there'll just be a single random effect field for normal years, and a single effect for catastrophic / abnormal years.

```
# distance matrix of knots
distKnots = as.matrix(dist(knots))
distKnots2 = distKnots^2 # squared distances
# note: shape parameter scaled to distance matrix
gp_scale = 0.001
sigma.norm = 0.01
sigma.fat = 0.02
corKnots = exp(-gp_scale*distKnots2)
Sigma.normal = corKnots * sigma.norm * sigma.norm
Sigma.fat = corKnots * sigma.fat * sigma.fat
```

Calculate distance matrix from stations to knots, and covariance matrices,

```
# Calculate distance from knots to grid
distAll = as.matrix(dist(rbind(grid, knots)))^2
dist21 = t(distAll[1:nKnots, -c(1:nKnots)])

Sigma21.normal = exp(-gp_scale*dist21) * sigma.norm * sigma.norm
Sigma21.fat = exp(-gp_scale*dist21) * sigma.fat * sigma.fat
```

Generate single MVN field for each of the normal and 'fat-tailed' distributions.

```
re.norm = rmvnorm(1, mean = rep(0, nKnots), Sigma.normal)
re.fat = rmvnorm(1, mean = rep(0, nKnots), Sigma.fat)
```

Project to the station / grid locations, and combine the random fields using some proportion of 'fat-tailed' events. Initially, we'll use 0.05.

```
# take inverse
invSigmaKnots.norm = solve(Sigma.normal)
invSigmaKnots.fat = solve(Sigma.fat)

# Project
proj.norm = Sigma21.normal %*% invSigmaKnots.norm %*% matrix(re.norm,ncol=1)
proj.fat = Sigma21.fat %*% invSigmaKnots.fat %*% matrix(re.fat,ncol=1)

# Combine to single RF
pfat = 0.05
proj = (1-pfat)*proj.norm + pfat*proj.fat
```

Next, we'll simulate 3 kinds of data from the spatial mixture: gaussian, binomial, and poisson counts.

```
nPoints = 100
indices = sample(seq(1,nLocs), size=nPoints, replace=T)

# assume no observation error, so observed/simulated gaussian data are same
x = grid[indices,]
y.gaussian = proj[indices,1]

y.binomial = rbinom(nPoints, size=1, prob=plogis(proj[indices,1]))

y.poisson = rpois(nPoints, exp(proj[indices,1]))
```