Bayesian SAE using Complex Survey Data
Lecture 9: Continuous spatial models

Geir-Arne Fuglstad
Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim

2018-04-22

- This tutorial goes through an example of fitting continuous spatial model.
- Download the 'Kenya' dataset folder from the website and all the R scripts under the folder 'Lecture9'.
- Throughout the slides, we assume the scripts and Kenya data folder are saved in "../codes/" and "../data/".

```
# install.packages("rmutil")
# install.packages("viridis")
source('../codes/Lecture9/continuous_functions.R')
```

```r
# Load data
load('../data/Kenya/simDataMulti.RData')
idxSurvey = 3
data = overSampDat$clustDat[[idxSurvey]]

# Organize data frame
data4inla = data.frame(y = data$died,
                       N = data$numChildren,
                       agegroup = rep(1, nrow(data)),
                       cluster = 1:nrow(data),
                       lon = data$lon,
                       lat = data$lat,
                       urban = as.factor(data$urban))
```

```r
# Read map of administrative regions
kenyaMap = readOGR(
  dsn = "../data/Kenya/Kenya_shape_file/",
  layer = "KEN_adm1")
## OGR data source with driver: ESRI Shapefile
## Source: "../data/Kenya/Kenya_shape_file/", layer: "KEN_a
## with 47 features
## It has 12 fields
```

```r
## Estimate model
# Formula
form.cov = ~ urban

# Random effects
rEffects = list(createSPDE_RE(data = data4inla,
                              idxName = "space",
                              max.edge = c(0.5, 1),
                              offset = c(1, 2),
                              prior.range = c(0.5, 0.05),
                              prior.sigma = c(3, 0.05),
                              regMap = kenyaMap,
                              weightedIntToZero = TRUE))
rEffects[[2]] = list(
formula = "f(iidCluster, model = 'iid', param = c(0.5, 5e-4))",
 A = 1,
effects = list(list(iidCluster = data4inla$cluster)))
```

```r
# Run discrete survival model
res = fitDiscreteSurvival(form.cov = form.cov,
                          data = data4inla,
                          rEffects = rEffects,
                          verbose = TRUE,
                          eb = FALSE)
# Extract inla results
res.inla = res$res
```

```
## Make grid for prediction Create spatial grid
nx = 300
ny = 300
nPred = nx * ny
x = seq(33, 43, length.out = nx)
xs = rep(x, each = ny)
y = seq(-5, 6, length.out = ny)
ys = rep(y, nx)
loc.pred = cbind(xs, ys)
xMat = matrix(xs, ncol = nx)
yMat = matrix(ys, ncol = ny)
```

```r
# Get stratification variables on prediction grid
provNum = locToProv(loc = loc.pred)
## OGR data source with driver: ESRI Shapefile
## Source: "../data/Kenya/Kenya_shape_file/", layer: "KEN_a
## with 47 features
## It has 12 fields
provFac = factor(x = provNum, levels = c(1, 2, 3, 4,
    5, 7, 8, 9))
urbanNum = getUrbanicity(loc = loc.pred, kenyaMap,
    2014)
urbanFac = factor(x = urbanNum, levels = c(1, 2))
regionUrban <- interaction(provFac, urbanFac)
regionUrban[regionUrban == "9.2"] = "9.1"
levels(regionUrban) = c(levels(regionUrban), "6.1",
    "6.2")
```

```r
# Get counties for prediction grid
gridP = data.frame(Longitude = as.vector(xMat),
                   Latitude = as.vector(yMat))
coordinates(gridP) = ~ Longitude + Latitude
proj4string(gridP) = proj4string(kenyaMap)
districtName = over(gridP, kenyaMap)[,5]
districtNum = as.numeric(districtName)
```

```
## Make predictions from posterior distribution
## Create prediction data frame Get stratification
## variables on prediction grid for correct year
provNum = locToProv(loc = loc.pred)
## OGR data source with driver: ESRI Shapefile
## Source: "../data/Kenya/Kenya_shape_file/", layer: "KEN_a
## with 47 features
## It has 12 fields
provFac = factor(x = provNum, levels = c(1, 2, 3, 4,
    5, 7, 8, 9))
uYear = 2014
urbanNum = getUrbanicity(loc = loc.pred, kenyaMap,
    uYear)
urbanFac = factor(x = urbanNum, levels = c(2, 1))
regionUrban <- interaction(provFac, urbanFac)
regionUrban[regionUrban == "9.2"] = "9.1"
levels(regionUrban) = c(levels(regionUrban), "6.1",
    "6.2")
```

```r
# Create prediction structure
numAG = length(unique(data4inla$agegroup))
data.pred = data.frame(
  cluster = rep(NA, length(xMat)*numAG),
 lon = rep(xs, each = numAG),
 lat = rep(ys, each = numAG),
 county = rep(districtName, each = numAG),
 countyIdx = rep(districtNum, each = numAG),
 urban = rep(urbanFac, each = numAG))
```

```r
## Set seed
inla.seed = 1585004955

## Sample from posterior
# Number of samples
nSamples = 1000

# Formula
form.cov.2 = form.cov

# Prediction random effects
rEffects.pred = list(createSPDE_RE(data = data.pred,
                        idxName = "space",
                        prevSPDE = rEffects[[1]]$dontUse,
                        constr = TRUE))
rEffects.pred[[2]] = list(
  formula = "f(iidCluster, model = 'iid')",
  A = 1,
  idx = data.pred$cluster)
```

```r
# Sample from posterior according to covariates in formula
# random effects in rEffects.1.pred
u5mr.samples.2 = predU5MR(res.DS = res,
           data.pred = data.pred,
           form.cov = form.cov.2,
           rEffects = rEffects.pred,
           time.in.ageGroup = time.in.ageGroup,
           nSamp = nSamples,
           debug = FALSE,
           inla.seed = inla.seed)

# Aggregation
u5mr.reg.2 = aggregateSamples(u5mr.samples.2$u5mr.samples,
  loc = loc.pred, districtNum,
  year = 2014, regMap = kenyaMap)
```

```r
# Post-process to only store median and standard deviations
u5Res = list(
  u5mr.median = matrix(0, nrow = 300, ncol = 300),
   u5mr.logit.mean = matrix(0, nrow = 300, ncol = 300),
   u5mr.sd      = matrix(0, nrow = 300, ncol = 300),
   u5mr.logit.sd = matrix(0, nrow = 300, ncol = 300))
```

```r
# u5mr
u5mr.median.2 = apply(X = u5mr.samples.2$u5mr.samples,
  MARGIN = 2, FUN = median)
u5mr.median.2 = matrix(u5mr.median.2, ncol = dim(xMat)[2])
u5Res$u5mr.median = u5mr.median.2

u5mr.logit.mean.2 = apply(X = logit(u5mr.samples.2$u5mr.samples)
u5mr.logit.mean.2 = matrix(u5mr.logit.mean.2,
  ncol = dim(xMat)[2])
u5Res$u5mr.logit.mean = u5mr.logit.mean.2

u5mr.sd.2 = apply(X = u5mr.samples.2$u5mr.samples,
  MARGIN = 2, FUN = sd)
u5mr.sd.2 = matrix(u5mr.sd.2, ncol = dim(xMat)[2])
u5Res$u5mr.sd = u5mr.sd.2
```

```
u5mr.logit.sd.2 = apply(X = logit(u5mr.samples.2$u5mr.samples),
 MARGIN = 2, FUN = sd)
u5mr.logit.sd.2 = matrix(u5mr.logit.sd.2,
 ncol = dim(xMat)[2])
u5Res$u5mr.logit.sd = u5mr.logit.sd.2

u5Res$u5mr.q025 = matrix(apply(X = u5mr.samples.2$u5mr.samples,
 MARGIN = 2, FUN = quantile, probs = c(0.025)),
ncol = dim(xMat)[2])
u5Res$u5mr.q975 = matrix(apply(X = u5mr.samples.2$u5mr.samples,
 MARGIN = 2, FUN = quantile, probs = c(0.975)),
ncol = dim(xMat)[2])
```

```r
# Do regional estimates
loc = cbind(as.vector(xMat), as.vector(yMat))
u5mr.reg.2 = aggregateSamples(u5mr.samples.2$u5mr.samples,
    loc = loc, districtNum, year = 2014, regMap = kenyaMap)
regMedian.2 = vector("numeric", length = 47)
regLogitMean.2 = vector("numeric", length = 47)
regSD.2 = vector("numeric", length = 47)
regLogitSD.2 = vector("numeric", length = 47)
for (i in 1:47) {
    regMedian.2[i] = median(u5mr.reg.2$regVal[[i]])
    regLogitMean.2[i] = mean(logit(u5mr.reg.2$regVal[[i]]))
    regSD.2[i] = sd(u5mr.reg.2$regVal[[i]])
    regLogitSD.2[i] = sd(logit(u5mr.reg.2$regVal[[i]]))
}
u5Res$u5mr.reg.median = regMedian.2
u5Res$u5mr.reg.logit.mean = regLogitMean.2
u5Res$u5mr.reg.sd = regSD.2
u5Res$u5mr.reg.logit.sd = regLogitSD.2
```

```r
# Store index into district
u5Res$region.idx = u5mr.reg.2$regIdx

# Create mask to remove stuff outside Kenya
gridP = data.frame(Longitude = as.vector(xMat),
                   Latitude = as.vector(yMat))
coordinates(gridP) = ~ Longitude + Latitude
proj4string(gridP) = proj4string(kenyaMap)
mask = as.numeric(over(gridP, kenyaMap)[,1])
mask[!is.na(mask)] = 1
mask = matrix(mask, ncol = 300)

# Remove stuff
u5Res$u5mr.median = u5Res$u5mr.median*mask
u5Res$u5mr.logit.mean = u5Res$u5mr.logit.mean*mask
u5Res$u5mr.sd = u5Res$u5mr.sd*mask
u5Res$u5mr.logit.sd = u5Res$u5mr.logit.sd*mask
```

```r
# Add grid locations
u5Res$xMat = xMat
u5Res$yMat = yMat

# Store region samples
u5Res$admin1.samples = u5mr.reg.2

# Store national samples
nNum = districtNum
nNum[!is.na(nNum)] = 1
u5Res$national.samples = aggregateSamples(
  u5mr.samples.2$u5mr.samples,
  loc = loc, nNum, year = 2014,
  regMap = kenyaMap)
```

```r
## Make plots Load extra functions for plotting
source("../codes/Lecture9/plotFuns.R")

# Load colormap
library(viridis)

## Table of parameter estimates
parTable = res.inla$summary.hyperpar[, 3:5]
parTable[3, ] = 1/sqrt(parTable[3, 3:1])
parTable = rbind(parTable, exp(res.inla$summary.fixed[2,
    3:5]))
rownames(parTable)[3:4] = c("Stddev for iidCluster",
    "urbanEffect")
```

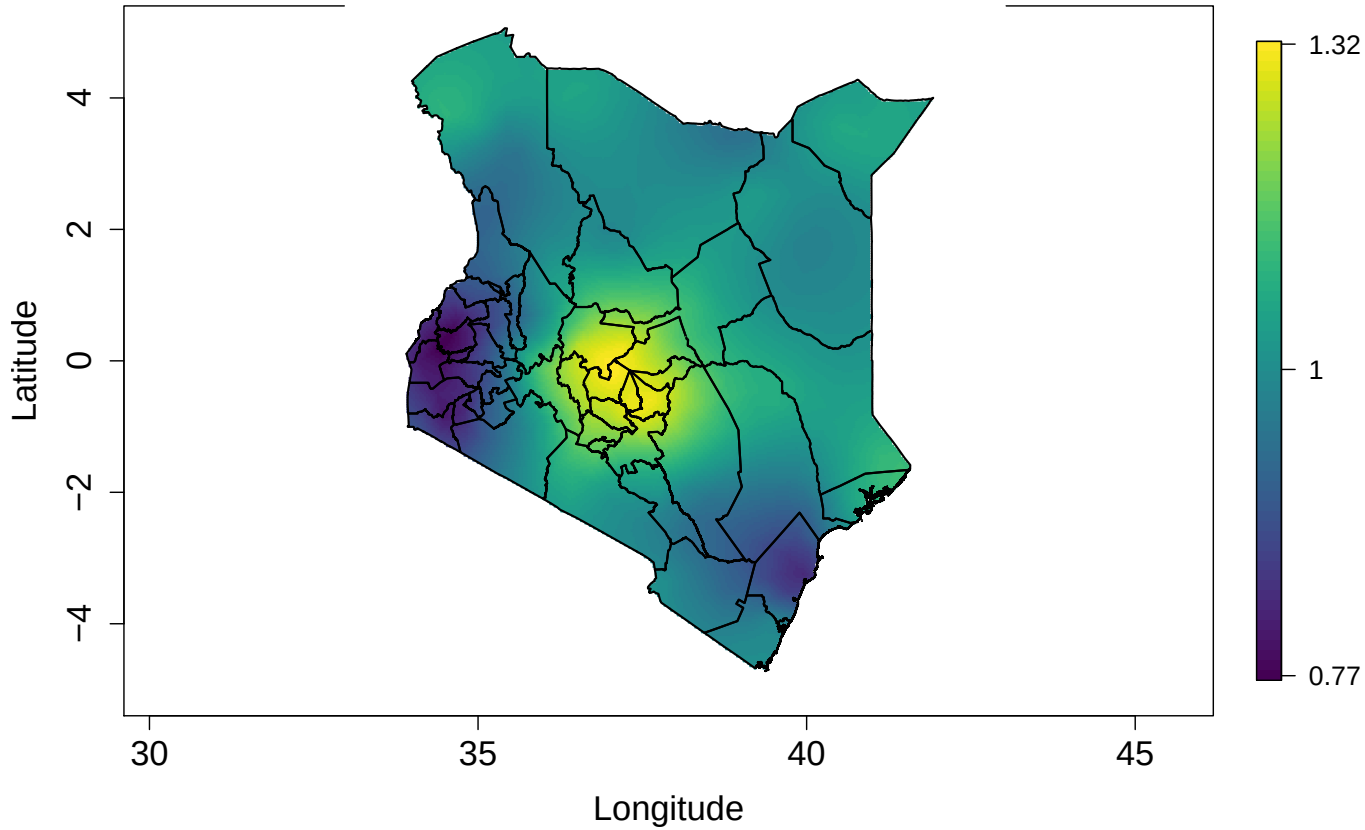
```
print(format(parTable, digits = 2, scientific = FALSE))
##                          0.025quant 0.5quant 0.975quant
## Range for space                0.889    2.432       7.51
## Stdev for space                0.105    0.184       0.31
## Stddev for iidCluster          0.016    0.044       0.23
## urbanEffect                    0.546    0.630       0.73
```

```
## Spatial effect
mesh = rEffects[[1]]$dontUse$mesh

medLim = c()
sdLim = c()
lVal = c()
uVal = c()
for (i in 1:1) {
    g.mean = inla.mesh.project(mesh, loc = cbind(xs,
        ys), res.inla$summary.random$space$mean[mesh$n *
        (i - 1) + 1:mesh$n]) * mask
    g.sd = inla.mesh.project(mesh, loc = cbind(xs,
        ys), res.inla$summary.random$space$sd[mesh$n *
        (i - 1) + 1:mesh$n]) * mask
    trans.median = exp(g.mean)
    trans.sd = sqrt((exp(g.sd^2) - 1) * exp(2 * g.mean +
        g.sd^2))
    medLim = c(medLim, range(trans.median, na.rm = TRUE))
    sdLim = c(sdLim, range(trans.sd, na.rm = TRUE))

    tmp = quantile(g.mean, probs = c(0.05, 0.95), na.rm = TRUE)
    tmp2 = exp(tmp)
    lVal = c(lVal, tmp2[1])
    uVal = c(uVal, tmp2[2])
}
print("5% and 95% pixel quantiles for spatial effect")
print(tmp2)
print(as.numeric(tmp2[2]/tmp2[1]))
```

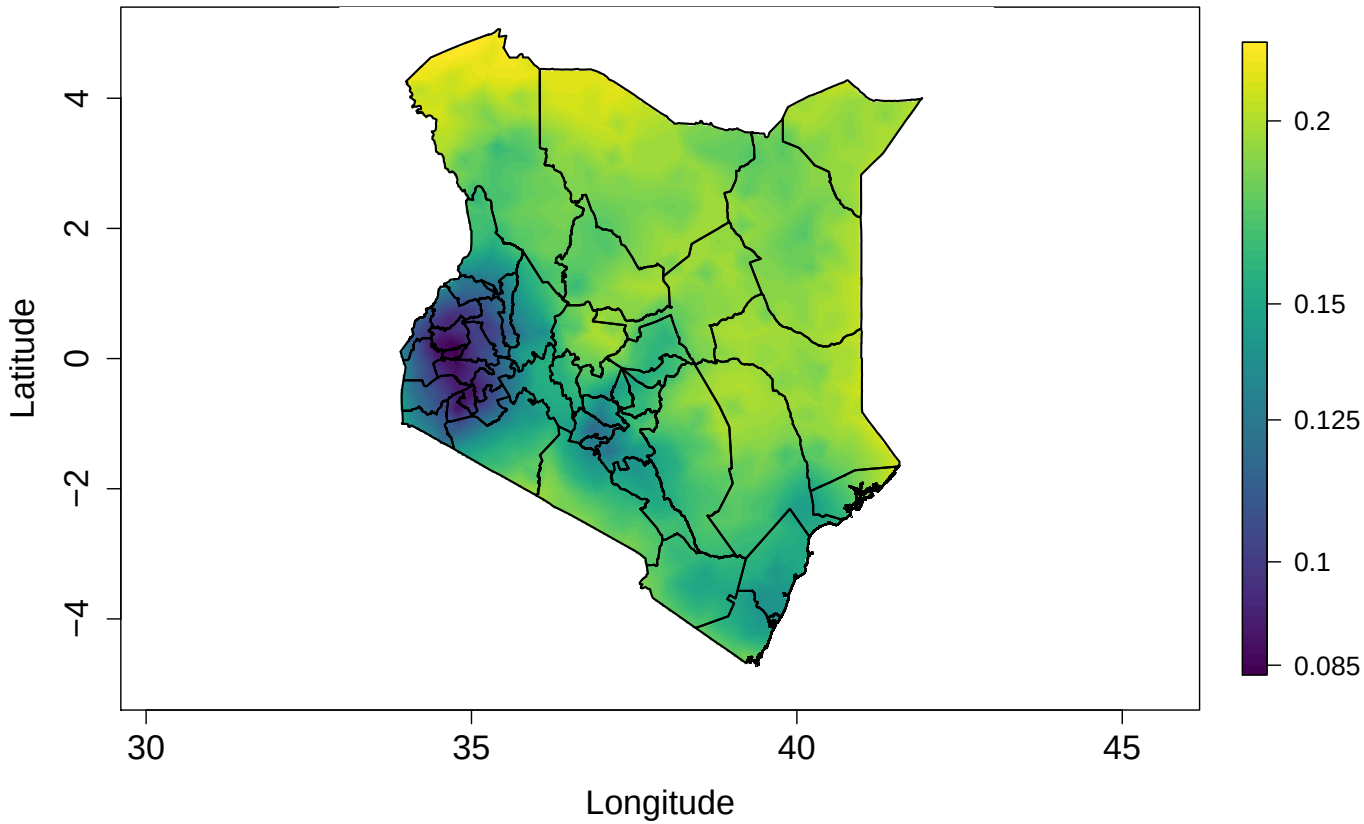
```
## [1] "5% and 95% pixel quantiles for spatial effect"
##        5%       95%
## 0.8588068 1.2096706
## [1] 1.408548
```

```r
# Plot spatial effect
medLim = range(medLim)
medBreaks = seq(medLim[1], 1, length.out = 33)
medBreaks = c(medBreaks[1:32], seq(1, medLim[2], len = 33))
sdLim = range(sdLim)
i <- 1
g.mean = inla.mesh.project(mesh, loc = cbind(xs, ys),
    res.inla$summary.random$space$mean[mesh$n * (i -
        1) + 1:mesh$n]) * mask
g.sd = inla.mesh.project(mesh, loc = cbind(xs, ys),
    res.inla$summary.random$space$sd[mesh$n * (i -
        1) + 1:mesh$n]) * mask

trans.median = exp(g.mean)
trans.sd = sqrt((exp(g.sd^2) - 1) * exp(2 * g.mean +
    g.sd^2))
```

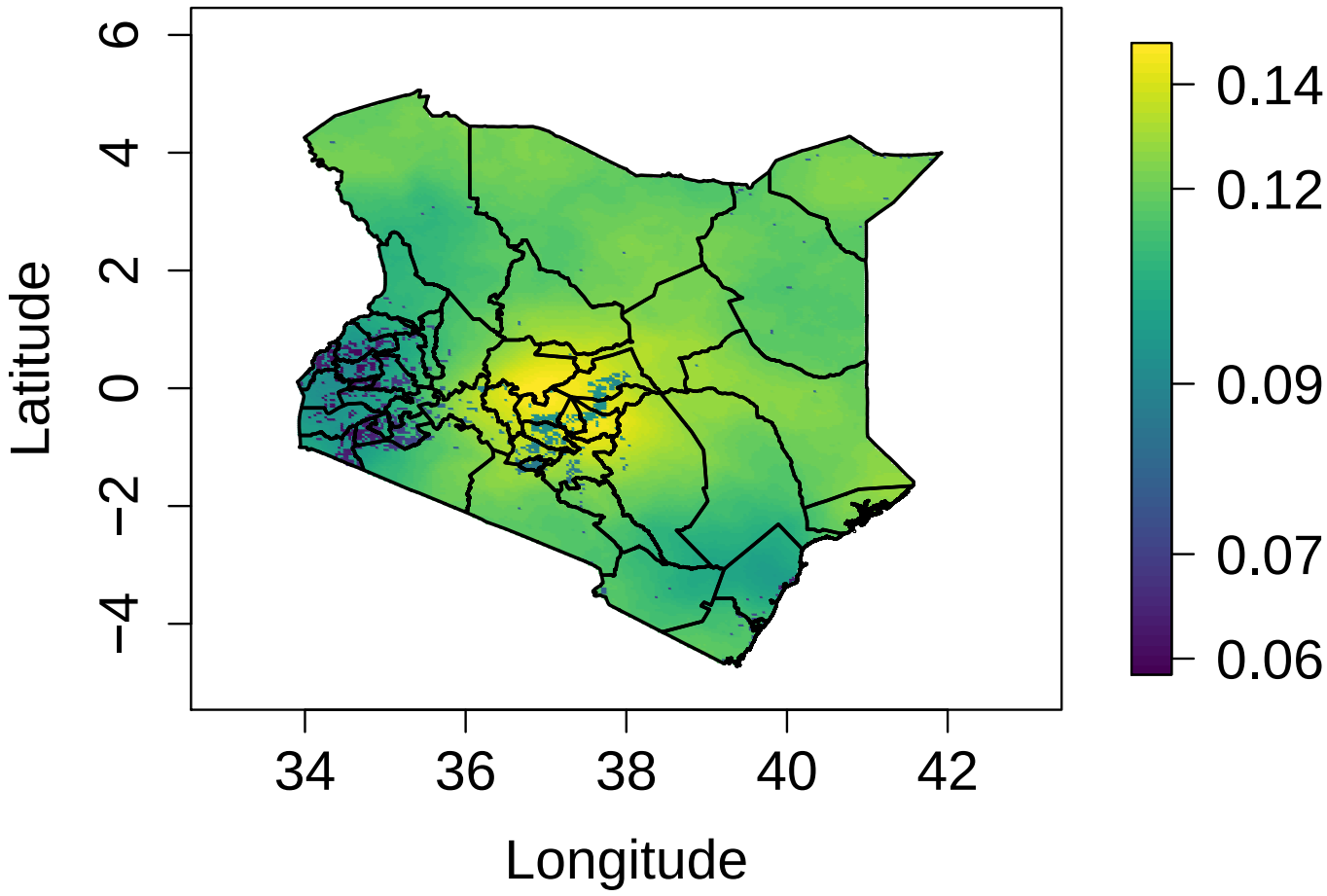
```
library(viridis)
ticks = c(round(medLim[1],2), 0.5, 1.0, 2.0, 3.0,
  round(medLim[2],2))
image.plot(xMat, yMat, matrix(log(trans.median), ncol = 300),
  xlab = "Longitude", ylab = "Latitude", cex.axis = 1.5,
  cex.lab = 1.5, axis.args = list(cex.axis = 1.2,
    at = log(ticks), labels = ticks),
  asp = 1, xlim = c(34, 41.8), ylim = c(-5, 5),
  zlim = log(medLim), col = viridis(64))
plot(kenyaMap, add = TRUE, lwd = 1.5)
```


Latitude
4
2
0
−2
−4
30
35
40
45
Longitude
1.32
1
0.77

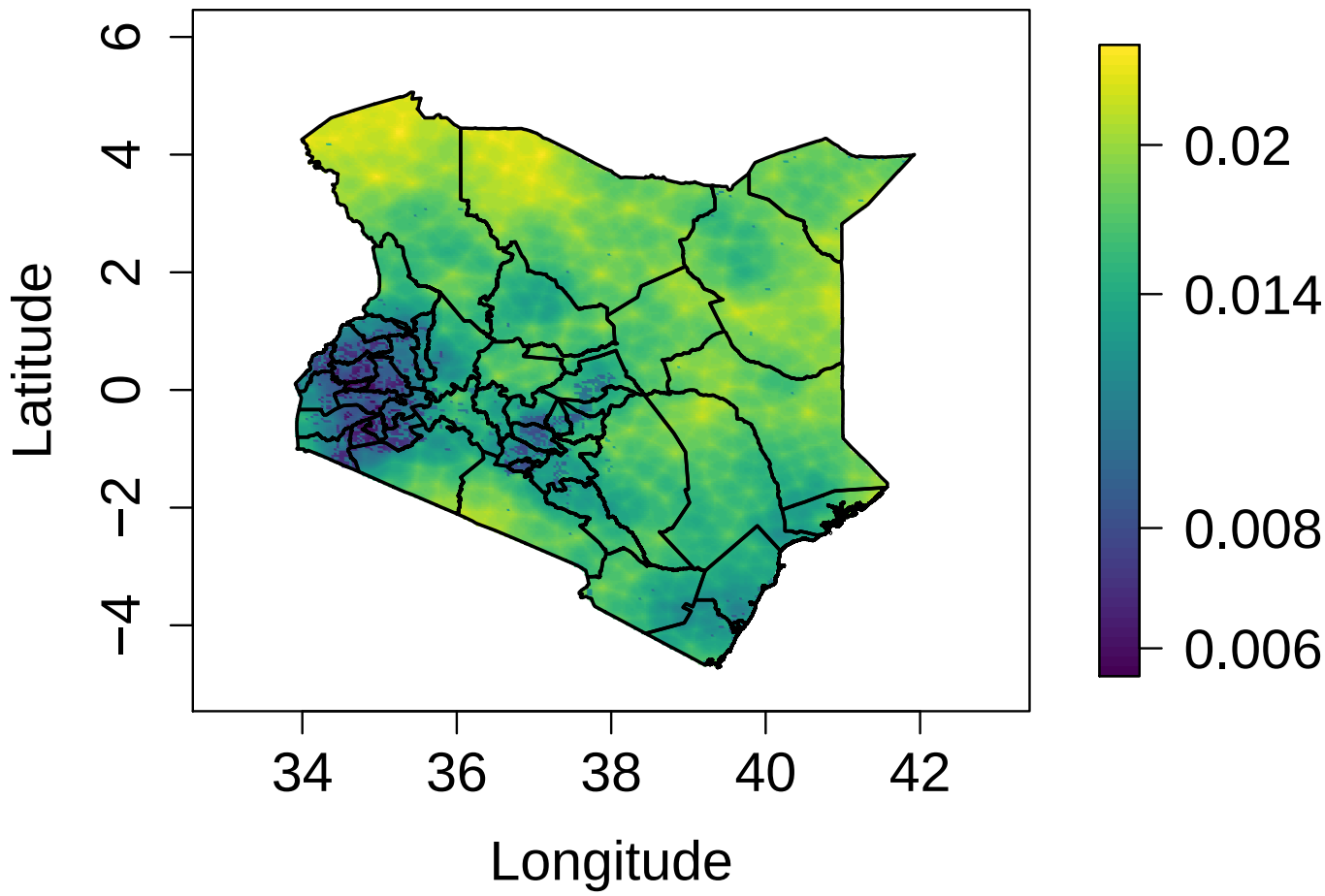
```
ticks = c(0.085, 0.10, 0.125, 0.15, 0.2)
image.plot(xMat, yMat, matrix(log(trans.sd), ncol = 300),
  xlab = "Longitude", ylab = "Latitude", cex.axis = 1.5,
  cex.lab = 1.5, axis.args = list(cex.axis = 1.2,
    at = log(ticks), labels = ticks),
  asp = 1, xlim = c(34, 41.8), ylim = c(-5, 5),
  zlim = log(sdLim), col = viridis(64))
plot(kenyaMap, add = TRUE, lwd = 1.5)
```


Latitude
4
2
0
−2
−4
30
35
40
45
Longitude
0.2
0.15
0.125
0.1
0.085

```r
## Predictions Pixel maps
medLim = range(u5Res$u5mr.median, na.rm = TRUE)
sdLim = range(u5Res$u5mr.sd, na.rm = TRUE)
ticks = c(0.06, 0.07, 0.09, 0.12, 0.14)
tSD = c(0.006, 0.008, 0.014, 0.02, 0.028)
image.plot(u5Res$xMat, u5Res$yMat, log(u5Res$u5mr.median),
  zlim = log(medLim), cex.axis = 1.5, cex.lab = 1.5,
  xlab = "Longitude", ylab = "Latitude",
  axis.args = list(cex.axis = 1.5, at = log(ticks),
    labels = ticks ), col = viridis(64))
plot(kenyaMap, add = TRUE, lwd = 1.5)
```


Latitude
6
4
2
0
−2
−4
34
36
38
40
42
Longitude
0.14
0.12
0.09
0.07
0.06

```
image.plot(u5Res$xMat, u5Res$yMat, log(u5Res$u5mr.sd),
  zlim = log(sdLim), cex.axis = 1.5, cex.lab = 1.5,
  xlab = "Longitude", ylab = "Latitude",
  axis.args = list(cex.axis = 1.5, at = log(tSD),
    labels =tSD ),  legend.mar = 6.1,
  col = viridis(64))
plot(kenyaMap, add = TRUE, lwd = 1.5)
```


Latitude
6
4
2
0
−2
−4
34
36
38
40
42
Longitude
0.02
0.014
0.008
0.006

```r
## Region estimates
medLim = range(u5Res$u5mr.reg.median, na.rm = TRUE)
sdLim = range(u5Res$u5mr.reg.sd, na.rm = TRUE)
perLim = range(u5Res$u5mr.reg.sd/u5Res$u5mr.reg.med*100,
  na.rm = TRUE)
perLim = c(3.8, 12)
breaks = c(7.6, 8, 9, 11, 12, 1e100)
# Plot each figure
print(range(u5Res$u5mr.reg.sd/u5Res$u5mr.reg.median))
ticks = c(0.073, 0.09, 0.11, 0.12, 0.13)
plotHatch(regPred = list(med = u5Res$u5mr.reg.median,
  sd = u5Res$u5mr.reg.sd),  zLim = medLim,
sdLim = perLim, breaks = breaks,
ticks = ticks, regMap = kenyaMap)
```

```
## [1] 0.0729967 0.1186839
```