Project 3 - Spatial

Amir Ahmed

11 4 2020

```
library(ggplot2)
library(geoR)
```

```
## ----------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.8-1 (built on 2020-02-08) is now loaded
## ----------------------------------------------------------
```

```
library(fields)
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

```
## Loading required package: grid
```

```
## Spam version 2.5-1 (2019-12-12) 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
```

```
## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

```
library(akima)
library(ggpubr)
```

```
## Loading required package: magrittr
```

```
set.seed(123)

par()
```

```
## $xlog
## [1] FALSE
##
## $ylog
```

```
## [1] FALSE
##
## $adj
## [1] 0.5
##
## $ann
## [1] TRUE
##
## $ask
## [1] FALSE
##
## $bg
## [1] "transparent"
##
## $bty
## [1] "o"
##
## $cex
## [1] 1
##
## $cex.axis
## [1] 1
##
## $cex.lab
## [1] 1
##
## $cex.main
## [1] 1.2
##
## $cex.sub
## [1] 1
##
## $cin
## [1] 0.15 0.20
##
## $col
## [1] "black"
##
## $col.axis
## [1] "black"
##
## $col.lab
## [1] "black"
##
## $col.main
## [1] "black"
##
## $col.sub
## [1] "black"
##
## $cra
## [1] 10.8 14.4
##
## $crt
```

```
## [1] 0
##
## $csi
## [1] 0.2
##
## $cxy
## [1] 0.02851711 0.07518797
##
## $din
## [1] 6.5 4.5
##
## $err
## [1] 0
##
## $family
## [1] ""
##
## $fg
## [1] "black"
##
## $fig
## [1] 0 1 0 1
##
## $fin
## [1] 6.5 4.5
##
## $font
## [1] 1
##
## $font.axis
## [1] 1
##
## $font.lab
## [1] 1
##
## $font.main
## [1] 2
##
## $font.sub
## [1] 1
##
## $lab
## [1] 5 5 7
##
## $las
## [1] 0
##
## $lend
## [1] "round"
##
## $lheight
## [1] 1
##
## $ljoin
```

```
## [1] "round"
##
## $lmitre
## [1] 10
##
## $lty
## [1] "solid"
##
## $lwd
## [1] 1
##
## $mai
## [1] 1.02 0.82 0.82 0.42
##
## $mar
## [1] 5.1 4.1 4.1 2.1
##
## $mex
## [1] 1
##
## $mfcol
## [1] 1 1
##
## $mfg
## [1] 1 1 1 1
##
## $mfrow
## [1] 1 1
##
## $mgp
## [1] 3 1 0
##
## $mkh
## [1] 0.001
##
## $new
## [1] FALSE
##
## $oma
## [1] 0 0 0 0
##
## $omd
## [1] 0 1 0 1
##
## $omi
## [1] 0 0 0 0
##
## $page
## [1] TRUE
##
## $pch
## [1] 1
##
## $pin
```

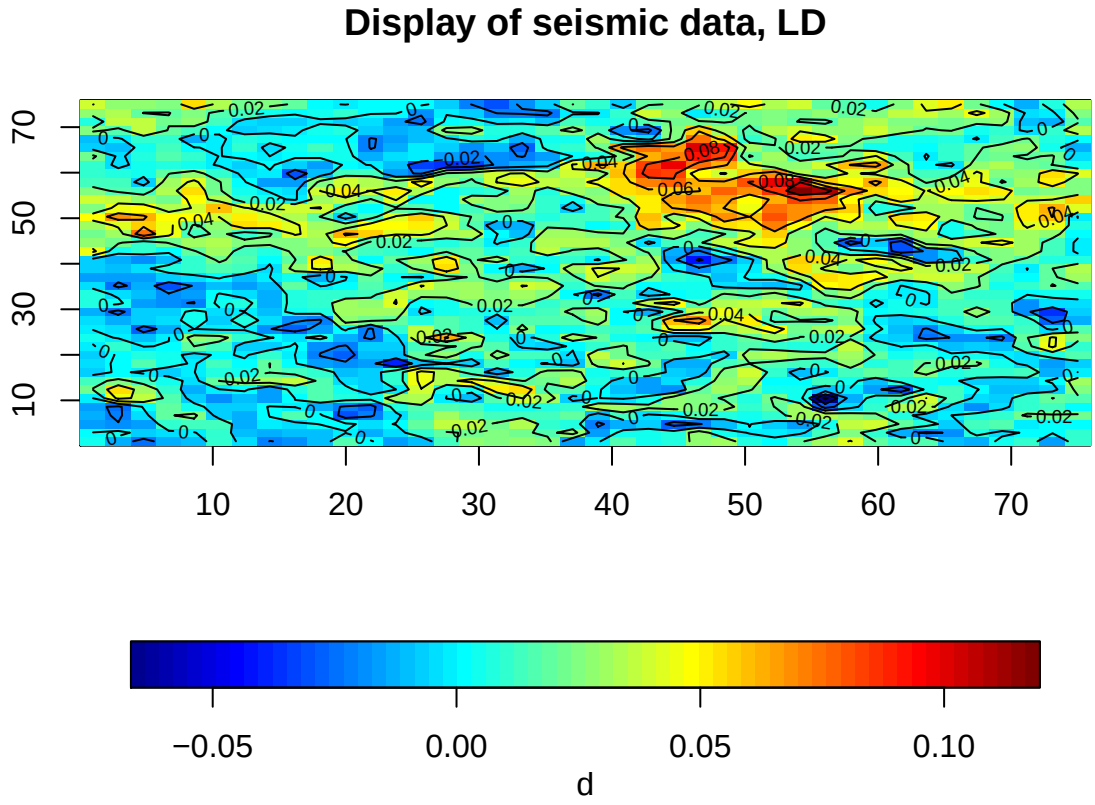
```
## [1] 5.26 2.66
##
## $plt
## [1] 0.1261538 0.9353846 0.2266667 0.8177778
##
## $ps
## [1] 12
##
## $pty
## [1] "m"
##
## $smo
## [1] 1
##
## $srt
## [1] 0
##
## $tck
## [1] NA
##
## $tcl
## [1] -0.5
##
## $usr
## [1] 0 1 0 1
##
## $xaxp
## [1] 0 1 5
##
## $xaxs
## [1] "r"
##
## $xaxt
## [1] "s"
##
## $xpd
## [1] FALSE
##
## $yaxp
## [1] 0 1 5
##
## $yaxs
## [1] "r"
##
## $yaxt
## [1] "s"
##
## $ylbias
## [1] 0.2
```

```
opar <- par()
```

```
seismic <- read.csv("seismic.dat", header = F)
seismic <- as.data.frame(seismic)
names(seismic) <- c("d")
```

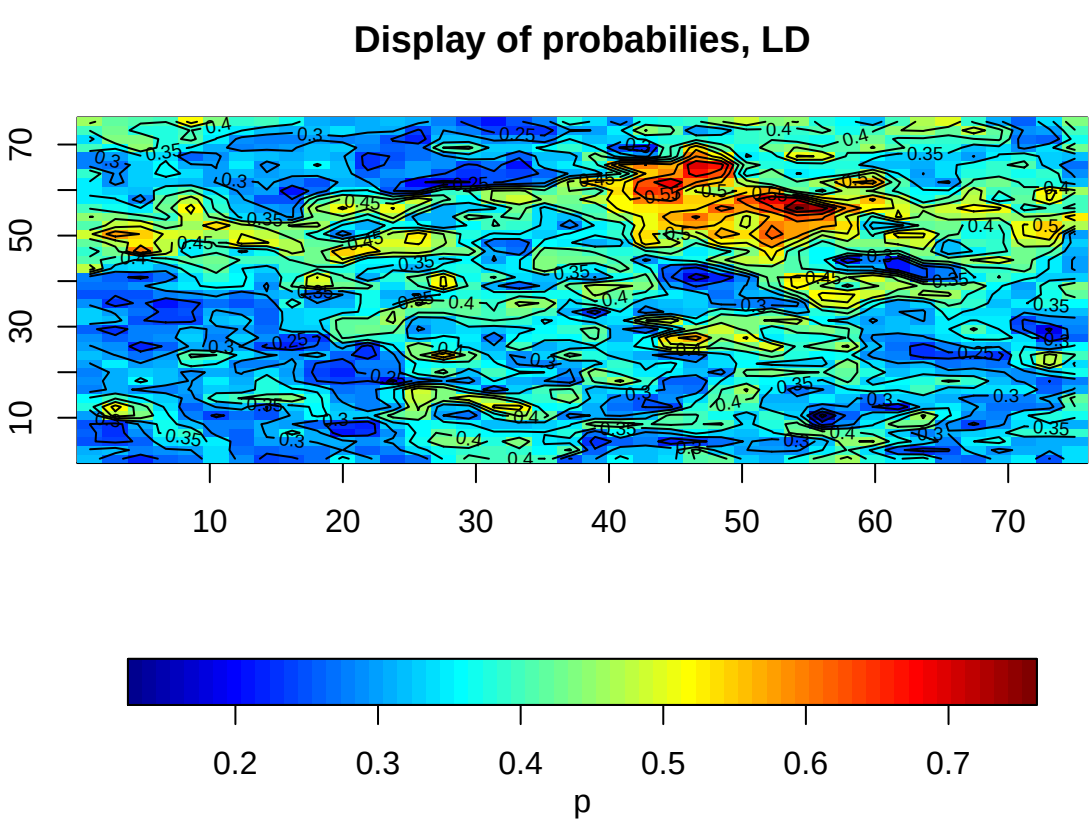
```r
seismic$x <- 0:(nrow(seismic)-1) %/% 75 + 1
seismic$y <- 0:(nrow(seismic)-1)%% 75 + 1
complit <- read.csv("complit.dat", sep = " ")
```

```r
# Figure 1
topo.li <- interp(seismic$x, seismic$y, seismic$d)
image.plot(topo.li, main = "Display of seismic data, LD", horizontal = T, legend.lab = "d")
contour(topo.li,add=T)
```




```r
par(mfrow=c(1,1))
```

```r
#1b
# Calculating pi-s from data.
pi <- dnorm(seismic$d, mean = 0.08, sd = 0.06)/(dnorm(seismic$d, mean = 0.02, sd = 0.06) + dnorm(seismi
topo.li <- interp(seismic$x, seismic$y, pi)
image.plot(topo.li, main = "Display of probabilies, LD", horizontal = T, legend.lab = "p")
contour(topo.li,add=T)
```




```
par(mfrow=c(1,1))
```

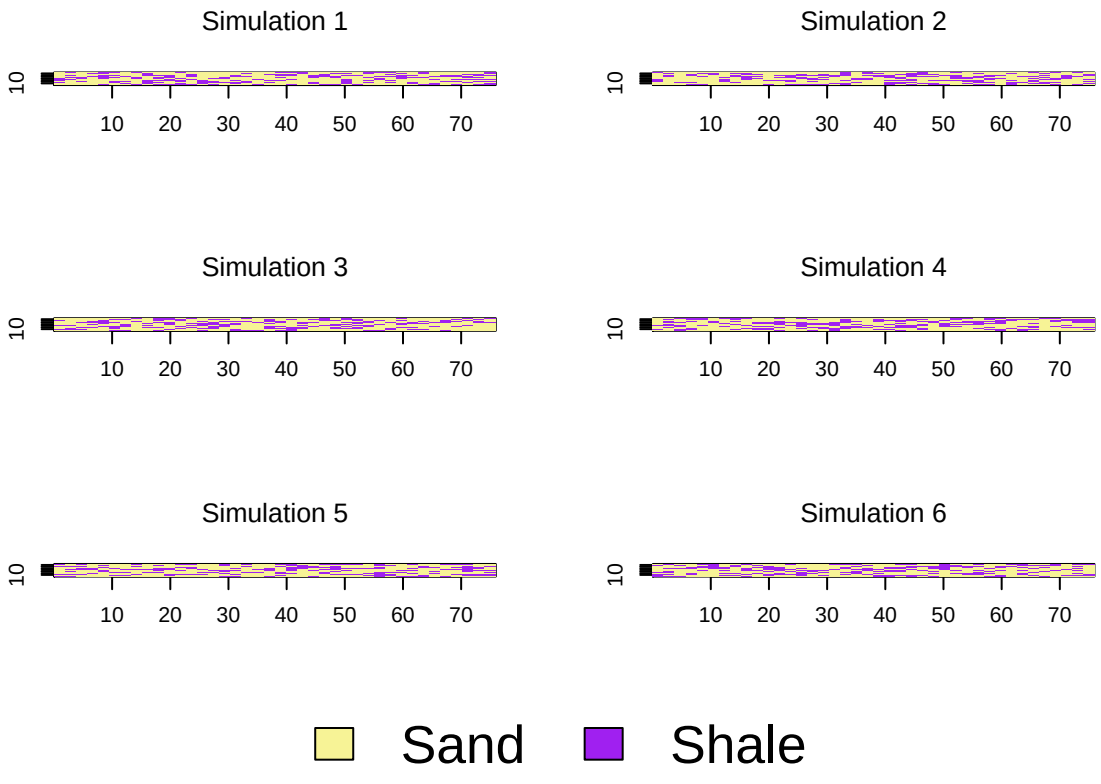
```
par(oma = c(4, 1, 1, 1))

simulate.unif <- function(){
  sapply(pi, function(p) rbinom(n = 1, size = 1, prob = p))
}

par(mfrow=c(3,2))
set.seed(1)
for(i in 1:6){
  sim.res <- simulate.unif()
  topo.li <- interp(seismic$x, seismic$y, sim.res)
  image(topo.li, nlevel = 2, col = c("#F7F396", "purple"), cex = 10)
  title(main = paste0("Simulation ", i), font.main = 12)

}
```

```
## Warning in plot.window(...): "nlevel" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
```

```
## Warning in box(...): "nlevel" is not a graphical parameter
## Warning in title(...): "nlevel" is not a graphical parameter
## Warning in plot.window(...): "nlevel" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
## Warning in box(...): "nlevel" is not a graphical parameter
## Warning in title(...): "nlevel" is not a graphical parameter
## Warning in plot.window(...): "nlevel" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
## Warning in box(...): "nlevel" is not a graphical parameter
## Warning in title(...): "nlevel" is not a graphical parameter
## Warning in plot.window(...): "nlevel" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
## Warning in box(...): "nlevel" is not a graphical parameter
## Warning in title(...): "nlevel" is not a graphical parameter
## Warning in plot.window(...): "nlevel" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
## Warning in box(...): "nlevel" is not a graphical parameter
## Warning in title(...): "nlevel" is not a graphical parameter
## Warning in plot.window(...): "nlevel" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in box(...): "nlevel" is not a graphical parameter

## Warning in title(...): "nlevel" is not a graphical parameter
```

```
par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0), new = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("bottom", c("Sand", "Shale"), xpd = TRUE, horiz = TRUE, inset = c(0,
    0), bty = "n", fill = c("#F7F396", "purple"), cex = 2.5)
```



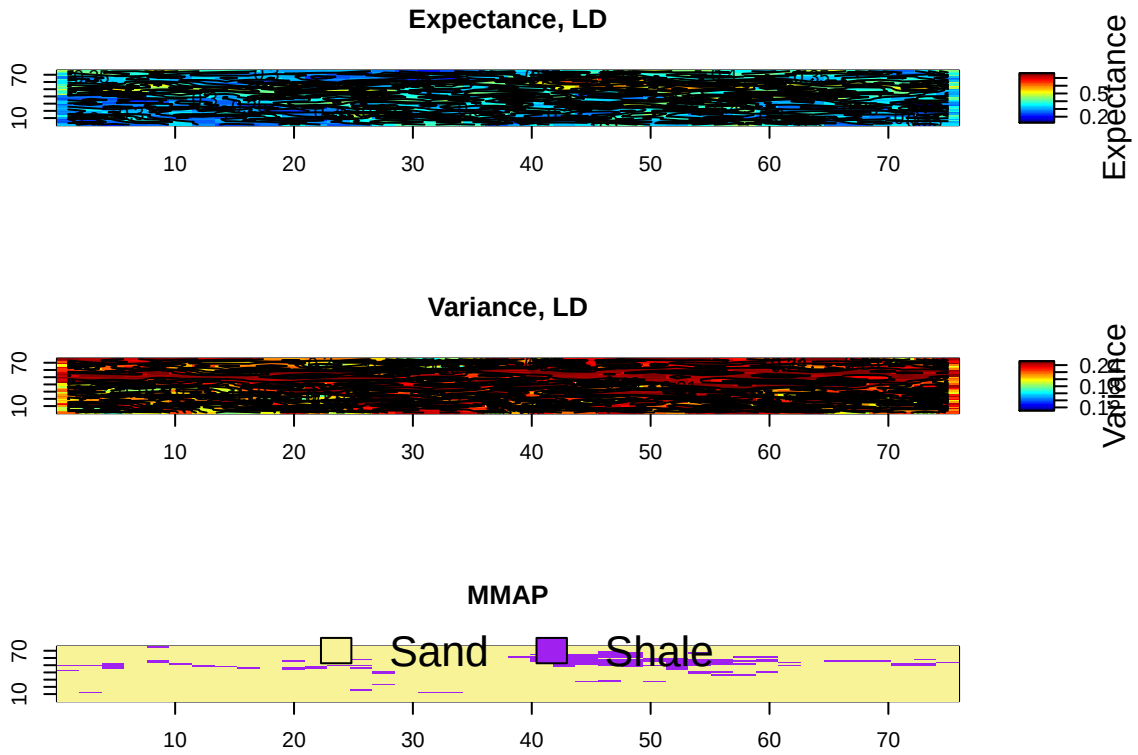

```
par(mfrow=c(1,1))
```

```
# IB MMAP expectance and variance.
ex <- pi
var <- pi*(1-pi)
MMAP <- pi >= 0.5
MMAP <- as.numeric(MMAP)

par(mfrow=c(3,1))
topo.li <- interp(seismic$x, seismic$y, ex)
image.plot(topo.li, main = "Expectance, LD", horizontal = F, legend.lab = "Expectance")
contour(topo.li,add=T)
```

```
topo.li <- interp(seismic$x, seismic$y, var)
image.plot(topo.li, main = "Variance, LD", horizontal = F, legend.lab = "Variance")
contour(topo.li,add=T)


topo.li <- interp(seismic$x, seismic$y, MMAP)
image(topo.li, main = "MMAP", nlevel = 2, col = c("#F7F396", "purple"))
```

```
## Warning in plot.window(...): "nlevel" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in box(...): "nlevel" is not a graphical parameter

## Warning in title(...): "nlevel" is not a graphical parameter
```

```
legend("bottom", c("Sand", "Shale"), xpd = TRUE, horiz = TRUE, inset = c(0,
    0), bty = "n", fill = c("#F7F396", "purple"), cex = 2)
```




```
par(mfrow=c(1,1))
```

```
# c)
plot.map <- function(df, title = "Dc"){
```

```
  data <- c()
  for(i in 1:nrow(df)){
    for(j in 1:ncol(df)){
     data <- rbind(data, c(i, j, df[i,j]))
    }
  }
  data <- as.data.frame(data)
  colnames(data) <- c("x", "y", "l")
  topo.li <- interp(data$x, data$y, data$l)
  image(topo.li, main = title, nlevel = 2, col = c("#F7F396", "purple"))
}
par(oma = c(4, 1, 1, 1))
par(mfrow=c(1,1))
plot.map(complit)
```

```
## Warning in plot.window(...): "nlevel" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
```

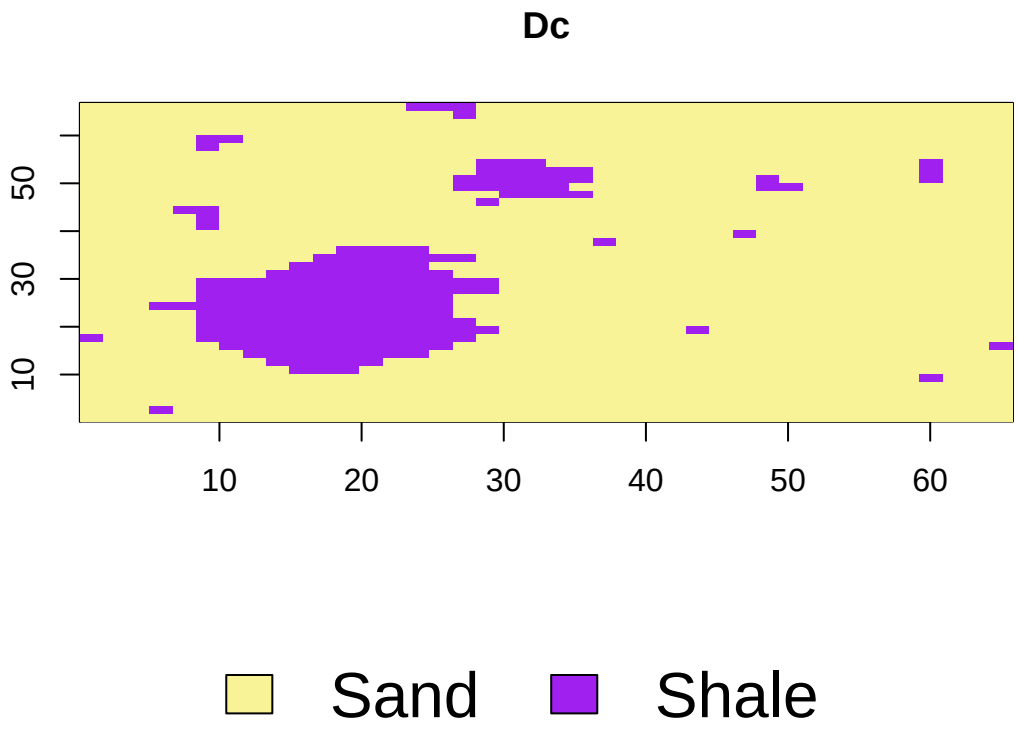
```
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
```

```
## Warning in box(...): "nlevel" is not a graphical parameter
```

```
## Warning in title(...): "nlevel" is not a graphical parameter
```

```
par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0), new = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("bottom", c("Sand", "Shale"), xpd = TRUE, horiz = TRUE, inset = c(0,
    0), bty = "n", fill = c("#F7F396", "purple"), cex = 2)
```



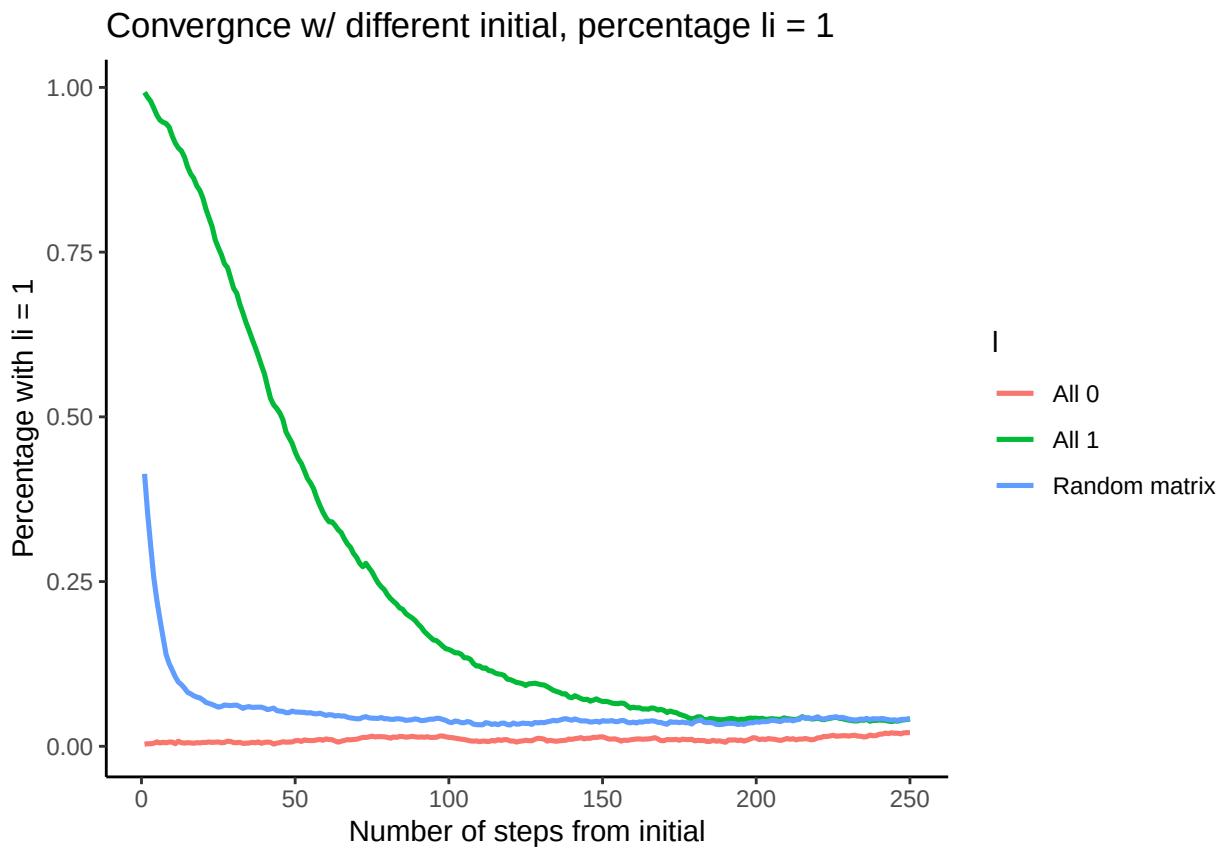

```
# Pseudo likelihood
mmpl <- function(d, beta){
  res <- 0
  for(i in 1:nrow(d)){
    for(j in 1:ncol(d)){
      li <- d[i, j]
      ns <- c()
      if(i != 0){
        ns <- c(ns, d[i-1,j])
      }

      if(i != nrow(d)){
        ns <- c(ns, d[i+1,j])
      }


      if(j != 0){
        ns <- c(ns, d[i,j-1])
      }

      if(j != ncol(d)){
        ns <- c(ns, d[i,j+1])
      }

      ns <- unlist(ns)
      lj.eq.to.li <- sapply(ns, function(lj) li == lj)
```

```
      lj.eq.to.0 <- sapply(ns, function(lj) 0 == lj)
      lj.eq.to.1 <- sapply(ns, function(lj) 1 == lj)

      res <- res + sum(lj.eq.to.li)*log(beta) - log(beta^(sum(lj.eq.to.0))+beta^(sum(lj.eq.to.1)))
    }
  }
  res
}

mmpl.vec <- function(d, bs){
  sapply(bs, function(b) mmpl(d = d, beta = b))
}

# Finding best estiamte using optim
opt.res <- optim(fn = function(bs) mmpl.vec(complit, bs),
      par = c(3),
      lower = c(1),
      upper = c(Inf),
      control=list(fnscale=-1),
      method = "L-BFGS-B")
```

```
# Set start beta parameter
beta <- opt.res$par
# Calcukate probability of li being 1
pl.li.1.given.ns <- function(ns, di) {
  lj.eq.to.0 <- sapply(ns, function(lj) 0 == lj)
  lj.eq.to.1 <- sapply(ns, function(lj) 1 == lj)

  dnorm(di, mean = 0.08, sd = 0.06) *
    beta^(sum(lj.eq.to.1))/
    (beta^(sum(lj.eq.to.1))*dnorm(di, mean = 0.08, sd = 0.06)+
       beta^(sum(lj.eq.to.0))*dnorm(di, mean = 0.02, sd = 0.06))
}

## Gibbssampler
# Sample from a cell
single.step.cell <- function(i, j, l, d){
  di <- d[i, j]
  li <- l[i, j]


  ns <- c()
  if(i != 0){
    ns <- c(ns, l[i-1,j])
  }

  if(i != nrow(l)){
    ns <- c(ns, l[i+1,j])
  }


  if(j != 0){
    ns <- c(ns, l[i,j-1])
  }
```

```
  if(j != ncol(l)){
    ns <- c(ns, l[i,j+1])
  }

  # Chance translate to 1
  p <- pl.li.1.given.ns(ns, di)
  li.next <- rbinom(n = 1, size = 1, prob = p)

  l[i, j] <- li.next
  return(l)
}

# Sweep a single time
single.step <- function(l, d){
  for(i in 1:nrow(l)){
    for(j in 1:ncol(l)){
      i <- sample(nrow(l), size = 1)
      j <- sample(ncol(l), size = 1)
      l <- single.step.cell(i, j, l, d)
    }
  }

  return(l)
}

d <- matrix(NA, nrow = 75, ncol = 75)

for(i in 1:nrow(seismic)){
  d[seismic[i,]$x,seismic[i,]$y] <- seismic[i,]$d
}

m <- 250
# Sweep m times
gibbs.sampler <- function(l, m, d){
  ls <- c()
  for(q in 1:m){
    print(q)
    l <- single.step(l, d)
    ls <- c(ls, sum(l))
  }
  list(l = l, ls = ls)
}



# Run m steps with different initial values
# ALl 1
 # l <- matrix(1, nrow = 75, ncol = 75)
 # res1 <- gibbs.sampler(l, m, d)
 #
  # Store number of ls in each step, in data frame
# res <- res1$ls
# all.res <- as.data.frame(res)
```

```r
# all.res$x <- 1:m
# all.res$l <- "All 1"
# # All 0
# l <- matrix(0, nrow = 75, ncol = 75)
# res0 <- gibbs.sampler(l, m, d)
# # Add number of ls in each step, in data frame
# res <- res0$ls
# temp.res <- as.data.frame(res)
# temp.res$x <- 1:m
# temp.res$l <- "All 0"
# all.res <- rbind(all.res, temp.res)
# # Random with 0.5 probabilit
# set.seed(123)
# l <- matrix(rbinom(prob = 0.5, n = 75*75, size = 1), nrow = 75, ncol = 75)
# res.rand <- gibbs.sampler(l, m, d)
# # Add number of ls in each step, in data frame
# res <- res.rand$ls
# temp.res <- as.data.frame(res)
# temp.res$x <- 1:m
# temp.res$l <- "Random matrix"
# all.res <- rbind(all.res, temp.res)
# Save results
#save(all.res, file = "gibbsinitials.RData")
# Load results, uncomment above if you like to watch paint dry
load(file = "gibbsinitials.RData")

all.res$l <- as.factor(all.res$l)
all.res$res <- all.res$res/(75*75)
ggplot(all.res) + geom_line(aes(x = x, y = res, color = l), size = 0.9) + theme_classic() +
  ylab("Percentage with li = 1") +
  xlab("Number of steps from initial") +
  ggtitle("Convergnce w/ different initial, percentage li = 1")
```




```r
# Starting with random 50-50 seems to have converged fastes, so set burnin to 50, and generate some mor

# Sweep m times and store
m <- 2500
storing.gibbs.sampler <- function(l, m, d){
  ls <- c()
  stored.ls <- list()
  for(q in 1:m){
    print(q)
    l <- single.step(l, d)
    ls <- c(ls, sum(l))
    stored.ls <- c(stored.ls, list(l))
  }
  list(l = l, ls = ls, stored.ls = stored.ls)
}
set.seed(123)
l <- matrix(rbinom(prob = 0.5, n = 75*75, size = 1), nrow = 75, ncol = 75)

# Only run this if you have serious amount of time
#res.rand <- storing.gibbs.sampler(l, m, d)
#save(res.rand, file = "res_rand.RData")
load("res_rand.RData")
# set burning
burnin <- 50

# Find distance to last
```

```r
par(oma = c(4, 1, 1, 1))
par(mfrow=c(1,2), mar = c(1, 1, 1, 1))
plot.map(res.rand$stored.ls[[1]], "Gibbs: Initial iteration")
```

```
## Warning in plot.window(...): "nlevel" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
```

```
## Warning in box(...): "nlevel" is not a graphical parameter
```

```
## Warning in title(...): "nlevel" is not a graphical parameter
```

```r
plot.map(res.rand$stored.ls[[2500]], "Gibbs: Last iteration")
```

```
## Warning in plot.window(...): "nlevel" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter
```

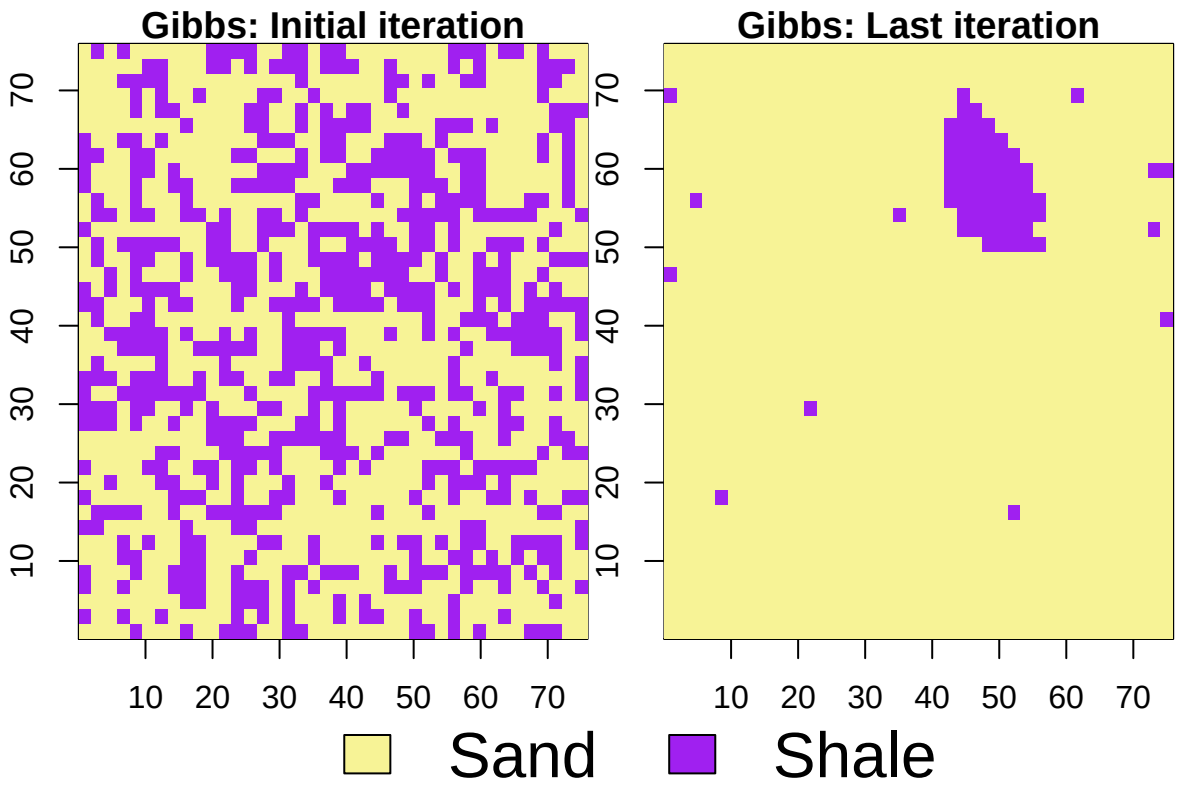
```
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter
```

```
## Warning in box(...): "nlevel" is not a graphical parameter
```

```
## Warning in title(...): "nlevel" is not a graphical parameter
```

```r
par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0), new = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("bottom", c("Sand", "Shale"), xpd = TRUE, horiz = TRUE, inset = c(0,
    0), bty = "n", fill = c("#F7F396", "purple"), cex = 2)
```




```
par(opar)
```

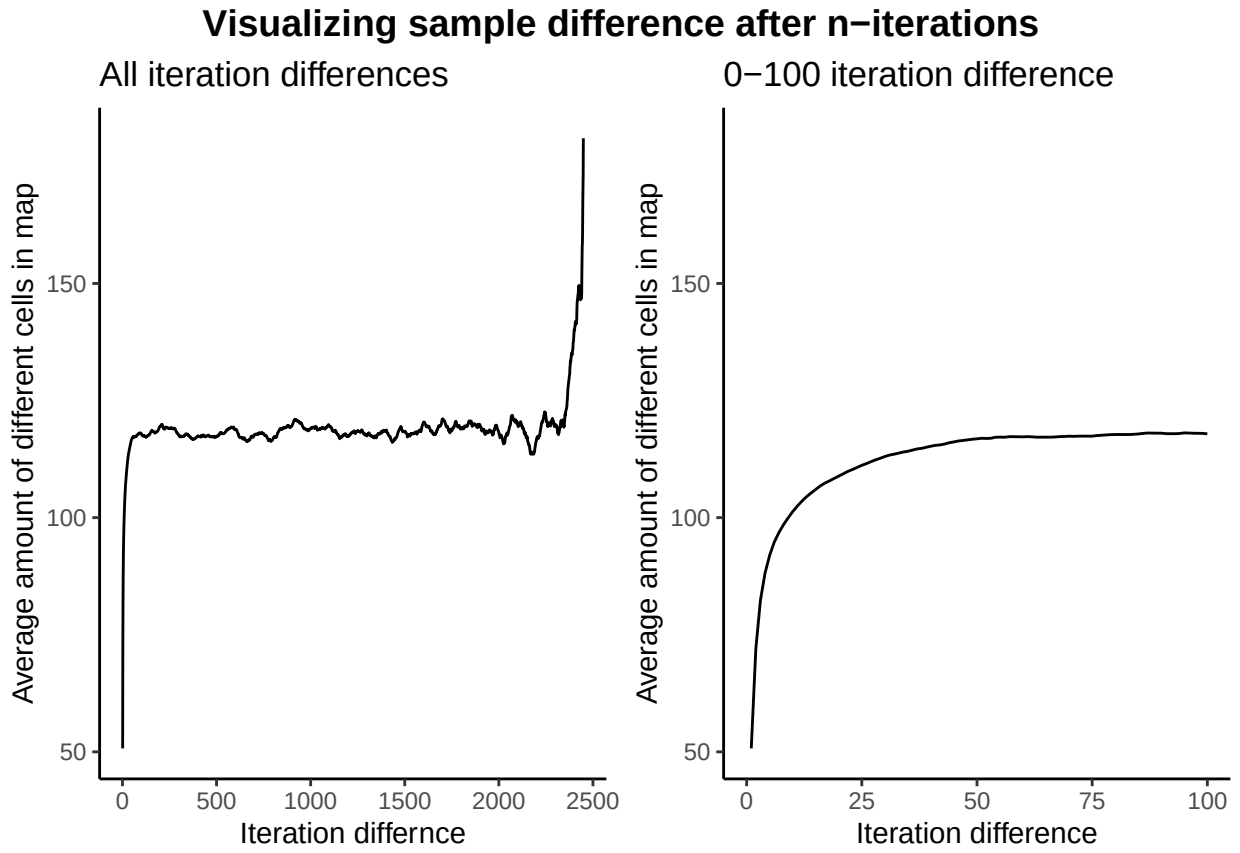
```
## Warning in par(opar): graphical parameter "cin" cannot be set
## Warning in par(opar): graphical parameter "cra" cannot be set
## Warning in par(opar): graphical parameter "csi" cannot be set
## Warning in par(opar): graphical parameter "cxy" cannot be set
## Warning in par(opar): graphical parameter "din" cannot be set
## Warning in par(opar): graphical parameter "page" cannot be set
```

```
#Remove burnin:
res <- res.rand$stored.ls[-(1:50)]

# Dont run this, it is slow
# different.df <- c()
# for(i in 1:length(res)){
#   print(i)
#   for(j in 1:length(res)){
#     if(i == j){
#       next()
#     }
#
#     num.different <- res[[i]] - res[[j]]
#     num.different <- abs(num.different)
#     num.different <- sum(num.different)
```

```
#     different.df <- rbind(different.df, c(num.different, abs(i-j)))
#   }
#
#   if(i %% 10 == 0){
#     write.table(different.df, "myDF.csv", sep = ",", col.names = !file.exists("myDF.csv"), append = T
#     different.df <- c()
#   }
#
# }
#
# df <- read.csv("myDF.csv")
# library(dplyr)
# head(df)
# colnames(df) <- c("index", "num.diff", "dist")
# df <- df %>% group_by(dist) %>% summarise(avg.diff = mean(num.diff))
# save(df, file = "df.RData")
load("df.RData")


p1 <- ggplot(df)+ geom_line(aes(x = dist, y = avg.diff)) + theme_classic() + xlab("Iteration differnce"
p2 <- ggplot(df)+ geom_line(aes(x = dist, y = avg.diff)) + xlim(c(0,100)) + theme_classic()+ xlab("Itera
p <- ggarrange(p1, p2, ncol = 2, nrow = 1)
```

```
## Warning: Removed 2349 row(s) containing missing values (geom_path).
```

```
annotate_figure(p,
                top = text_grob("Visualizing sample difference after n-iterations", color = "black", fa
```

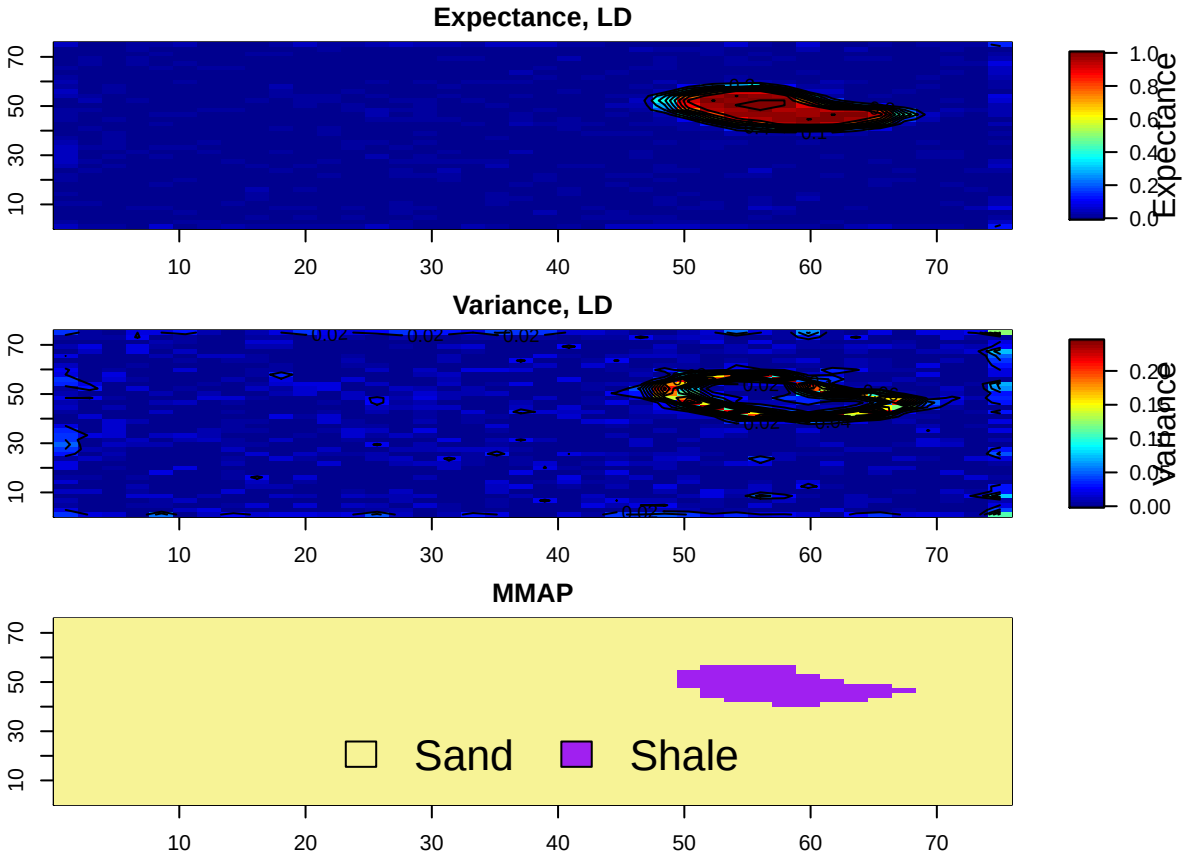



```r
# Keep every 50th  for independence
res <- res[(1:length(res))[1:length(res) %% 50 == 0]]

# number of independent samples:
m <- length(res)
P <- Reduce("+", res)
P <- P/m
P <- as.vector(P) # row then columns

df <- as.data.frame(P)
names(df) <- c("p")
df$x <- 0:(nrow(seismic)-1) %/% 75 + 1
df$y <- 0:(nrow(seismic)-1)%% 75 + 1
df$var <- df$p*(1-df$p)
df$mmap <- df$p >= 0.5
df$mmap <- as.numeric(df$mmap)

par(mfrow=c(3,1), mar = c(2,2,2,2))
topo.li <- interp(df$x, df$y, df$p)
image.plot(topo.li, main = "Expectance, LD", horizontal = F, legend.lab = "Expectance")
contour(topo.li,add=T)

topo.li <- interp(seismic$x, seismic$y, df$var)
image.plot(topo.li, main = "Variance, LD", horizontal = F, legend.lab = "Variance")
contour(topo.li,add=T)
```

```
topo.li <- interp(seismic$x, seismic$y, df$mmap)
image(topo.li, main = "MMAP", nlevel = 2, col = c("#F7F396", "purple"))
```

```
## Warning in plot.window(...): "nlevel" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "nlevel" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "nlevel" is not a
## graphical parameter

## Warning in box(...): "nlevel" is not a graphical parameter

## Warning in title(...): "nlevel" is not a graphical parameter
```

```
legend("bottom", c("Sand", "Shale"), xpd = TRUE, horiz = TRUE, inset = c(0,
    0), bty = "n", fill = c("#F7F396", "purple"), cex = 2)
```




```
par(mfrow=c(1,1))
```