```
#set working directory: this should be modified by user
setwd("P:/Subjects/SpatialStats/Project")

#some of the R packages used to deal with spatial data
library(sp)
library(BayesX)
library(R2BayesX)

#*******************************************************************
#as a first simple step to spatial data analysis
#state based potato yield of Germany is plotted
#*******************************************************************

#state based map of Germany taken from GADM project
#data source: http://gadm.org/
map1 = readRDS("DEU_adm1.rds")

#state based potato yield of Germany taken from
#the Regional Database Germany
#data source: https://www.regionalstatistik.de/
data1 = read.table(file = "yield.csv", header = T,
                  sep = ",",dec = ".",na.strings = ".")

#{
#due to lack of experience the first map is plotted in a labor
#intensive way, but it is still useful for showing some ways
#which can be used for advanced user modifications. In fact,
#it does better in drawing regions without data.

#potato yields are rescaled between 0 and 1
#which will be used for coloring the map
data1$col=(data1$Kartoffeln-min(data1$Kartoffeln,na.rm = TRUE))/
  (max(data1$Kartoffeln,na.rm = TRUE)
   -min(data1$Kartoffeln,na.rm = TRUE))

#map1 has more than 10 definitions for German states(see map1@data)
#the definition which is compatible with data1 is filtered
map1Filt=map1["CCA_1"]

#state codes are converted from string to number
#again for compatibility with data1
map1Filt$CCA_1=as.numeric(map1Filt$CCA_1)

#rescaled potato yields are used to create color codes
#for each German state according to their order in map
regCol=rep("",length(map1Filt$CCA_1))
for(i in 1:length(map1Filt$CCA_1)){
  entry=data1$col[which(data1$StateNo==map1Filt$CCA_1[i])]
  if(is.na(entry)){
    regCol[i]="red"
  }else{
    regCol[i]=gray(entry)
  }
}

#potato yield of German states is plotted
#potato yield increases from black to white color
#data is not provided for red areas
plot(map1Filt,col = regCol, main="Potato yield of Germany",
     border = 'darkgrey')
#}

#{
#in this map again potato yield of German states is plotted
#but this time ready R functions are used

#map1 is saved as "boundary" object
map1Bnd=sp2bnd(spObject = map1,
               regionNames = as.character(as.numeric(map1$CCA_1)))
```

```
#potato yield values are standardized
data1$zscore=(data1$Kartoffeln-mean(data1$Kartoffeln,na.rm = TRUE))/
  sd(data1$Kartoffeln,na.rm = TRUE)

#potato yield of German states is plotted
drawmap(data=data1[which(!is.na(data1$zscore)),],
        map=map1Bnd,main="Potato yield of Germany",
        regionvar=1,plotvar=5)
#}

#*****************************************************************
#potato yield estimation of German districts
#for which official data is not provided
#*****************************************************************

#map of German administrative districts is loaded
#this map is provided in R package "R2BayesX"
data("GermanyBnd")

#district based potato yield of Germany taken from
#the Regional Database Germany
#data source: https://www.regionalstatistik.de/
data2 = read.table(file = "yield2.csv", header = T,
                   sep = ",",dec = ".",na.strings = c(".","-"))

#districts with "NA" value are deleted
data2=data2[-which(is.na(data2$kartoffeln)),]

#districts with "0" value are deleted
data2=data2[-which(data2$kartoffeln<1),]

#state names in data2 are updated after the deletions above
data2$stateName=factor(data2$stateName)

#potato yield values are standardized
data2$zscore=(data2$kartoffeln-mean(data2$kartoffeln))/
  sd(data2$kartoffeln)

#potato yield of German districts is plotted before estimation
R2BayesX::plotmap(map = GermanyBnd, x = data2, c.select = 4,
                  legend = TRUE, missing = TRUE,
                  names = FALSE, values = TRUE,
                  cex.names = 0.5, cex.values = 0.5,
                  digits = 2,
                  main="Potato yield of Germany")

#map is converted from boundary to graph via neigborhood structure
#and this graph is a Markov random field
GermanyGraph=bnd2gra(GermanyBnd)

#check the graph; it can be used as a penalty matrix
GermanyGraph[1:10,1:10]

#codes for 439 districts are taken from graph
districtCode=colnames(GermanyGraph)

#{
#Ruegen is the only district which consists solely of islands
#the matrix in the solution becomes singular (non-invertable)
#a neighbor should be defined for Ruegen

#equals to zero, meaning that no neighbor
GermanyGraph[362,362]

#Ruegen with code 13061 has not any neighbors
districtCode[362]

#Nordvorpommern with code 13057 will be the neighbor
which(districtCode=="13057")
districtCode[358]
```

```r
#penalty matrix is adjusted for new neighbors
GermanyGraph[362,362]=1
GermanyGraph[362,358]=-1
GermanyGraph[358,362]=-1
GermanyGraph[358,358]=GermanyGraph[358,358]+1
#}

#{
#properties of penalty matrix are checked below

#penalty matrix is symetric
sum(GermanyGraph==t(GermanyGraph)) == 439*439

#row sums and column sums are zero
sum(rowSums(GermanyGraph))
sum(colSums(GermanyGraph))
#}

#{
#28 districts have different codes in data2 and GermanyBnd
#these 28 districts are deleted from data2
for(i in nrow(data2):1){
  if(!any(data2$stateCode[i]==districtCode)){
    data2=data2[-i,]
  }
}

#state names in data2 are updated after the deletions above
data2$stateName=factor(data2$stateName)
#}


#n is the number of observed yields in data2
n = nrow(data2)

#d is the number of German districts
d = length(districtCode)

#design matrix Z is created
Z = matrix(0,n,d)

#Z is computed considering districts of each observation
for(i in 1:d) {
  Z[which(data2$stateCode==districtCode[i]),i] = 1
}

#design matrix Z is checked for errors
sum(rowSums(Z))
sum(colSums(Z))


#{
#model is estimated with penalized least square criterion
#several values of smoothing parameter lambda are used

lambda = 1
gamma1 = as.vector(solve(t(Z)%*%Z+lambda*GermanyGraph)%*%
                     t(Z)%*%data2$zscore)
lambda = 3
gamma3 = as.vector(solve(t(Z)%*%Z+lambda*GermanyGraph)%*%
                     t(Z)%*%data2$zscore)
lambda = 5
gamma5 = as.vector(solve(t(Z)%*%Z+lambda*GermanyGraph)%*%
                     t(Z)%*%data2$zscore)
#}

#{
#estimated spatial effect for each district is plotted
```

```
#all the estimates and district codes are combined for plotting
result = data.frame(districtCode,gamma1,gamma3,gamma5)

R2BayesX::plotmap(map = GermanyBnd, x = result, c.select = 2,
                  legend = TRUE, missing = TRUE,
                  names = FALSE, values = TRUE,
                  cex.names = 0.5, cex.values = 0.5,
                  digits = 2,
                  main="Yields smoothed at lambda=1")

R2BayesX::plotmap(map = GermanyBnd, x = result, c.select = 3,
                  legend = TRUE, missing = TRUE,
                  names = FALSE, values = TRUE,
                  cex.names = 0.5, cex.values = 0.5,
                  digits = 2,
                  main="Yields smoothed at lambda=3")

R2BayesX::plotmap(map = GermanyBnd, x = result, c.select = 4,
                  legend = TRUE, missing = TRUE,
                  names = FALSE, values = TRUE,
                  cex.names = 0.5, cex.values = 0.5,
                  digits = 2,
                  main="Yields smoothed at lambda=5")
#}


#residuals from the estimated models are saved in data2Res
data2Res=data2
data2Res$gamma1=0
data2Res$gamma3=0
data2Res$gamma5=0
index=0
for(i in 1:n){
  index=which(result$districtCode==data2Res$stateCode[i])
  data2Res$gamma1[i]=result$gamma1[index]
  data2Res$gamma3[i]=result$gamma3[index]
  data2Res$gamma5[i]=result$gamma5[index]
}
data2Res$res1=data2Res$zscore - data2Res$gamma1
data2Res$res3=data2Res$zscore - data2Res$gamma3
data2Res$res5=data2Res$zscore - data2Res$gamma5

#{
#below assumptions are tested

#residuals reveal homoscedastic and linear behavior
for(i in 1:3){
  plot(data2Res[,(i+4)],data2Res[,(i+7)],
       main=colnames(data2Res)[i+7],
       xlab="fitted values", ylab="residuals")
  abline(a = 0,b = 0)
}

#residuals are normally disributed
#histogram of residuals with normal curve
for(i in 1:3){
  x=data2Res[,(i+7)]
  h=hist(x, breaks=20, col="red",
         main=paste("Histogram of ",colnames(data2Res)[i+7]),
         xlab = colnames(data2Res)[i+7])
  xfit=seq(min(x),max(x),length=40)
  yfit=dnorm(xfit,mean=mean(x),sd=sd(x))
  yfit=yfit*diff(h$mids[1:2])*length(x)
  lines(xfit, yfit, col="blue", lwd=2)
  rm(x,h,xfit,yfit)
}

#q-q plot of residuals
for(i in 1:3){
  qqnorm(data2Res[,(i+7)],
```

```r
         main = paste("Q-Q plot of ",colnames(data2Res)[i+7]))
  qqline(data2Res[,(i+7)])
}

#kernel density of residuals
for(i in 1:3){
  x=data2Res[,(i+7)]
  plot(density(x),lwd=2,
       main = paste("Kernel Density of ",colnames(data2Res)[i+7]))
  xfit=seq(min(x),max(x),length=40)
  yfit=dnorm(xfit,mean=mean(x),sd=sd(x))
  lines(xfit, yfit, col="blue", lty=2,lwd=2)
  rm(x,xfit,yfit)
}
#}

#residual variance increases as lambda inreases
var(data2Res$res1)
var(data2Res$res3)
var(data2Res$res5)

#observed yield values are not normally distributed
x=data2$zscore
h=hist(x, breaks=20, col="red",main="Histogram of observed yield")
xfit=seq(min(x),max(x),length=40)
yfit=dnorm(xfit,mean=mean(x),sd=sd(x))
yfit=yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
rm(x,h,xfit,yfit)
```