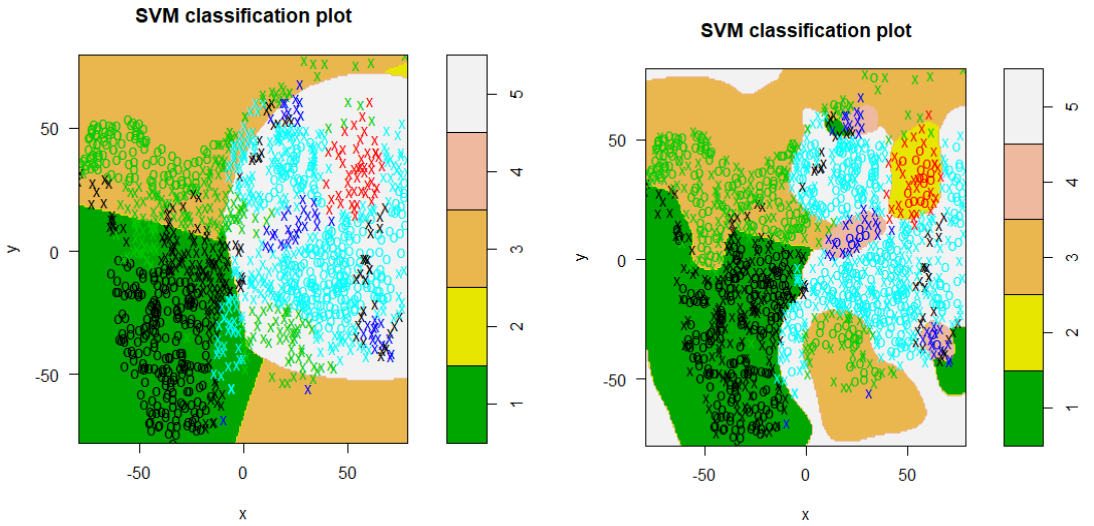
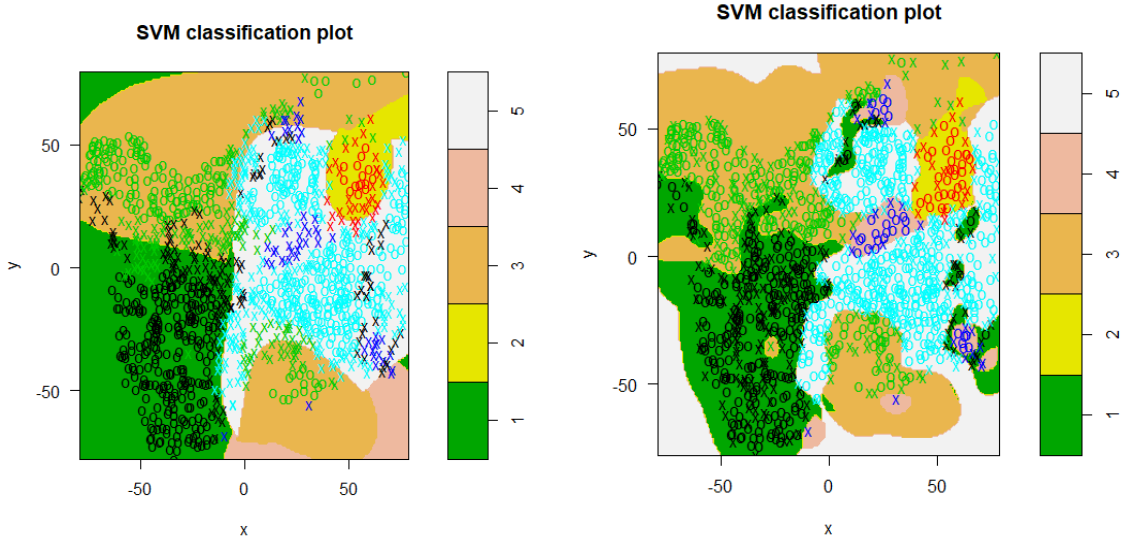
Problem 1

Solution:

a. Plot for the combination of (cost, gamma) = {(1, 0.1), (1,10), (100,0.1), (100,10)}



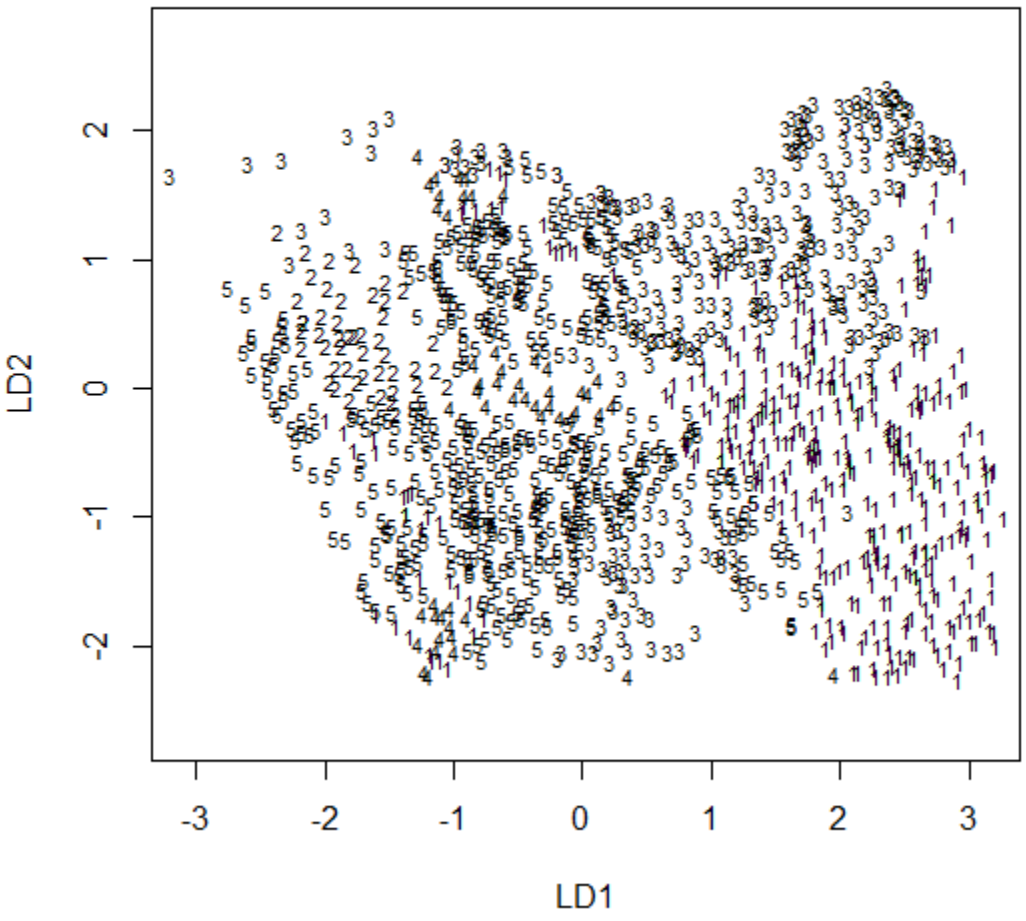


From these 4 plots we can see the choice of cost and gamma affect the character of separation boundaries between classes. For cost=100 we have found low bias and high variance. Low bias because we are penalizing the cost of misclassification a lot.
But cost=1 makes misclassification low. A small cost provides higher bias and lower variance.

Now, we know that gamma controls the shape of the "peaks". For gamma=0.1 we have found here a pointed bump in the higher dimensions, and for gamma=10 we have found here a softer, broader bump.

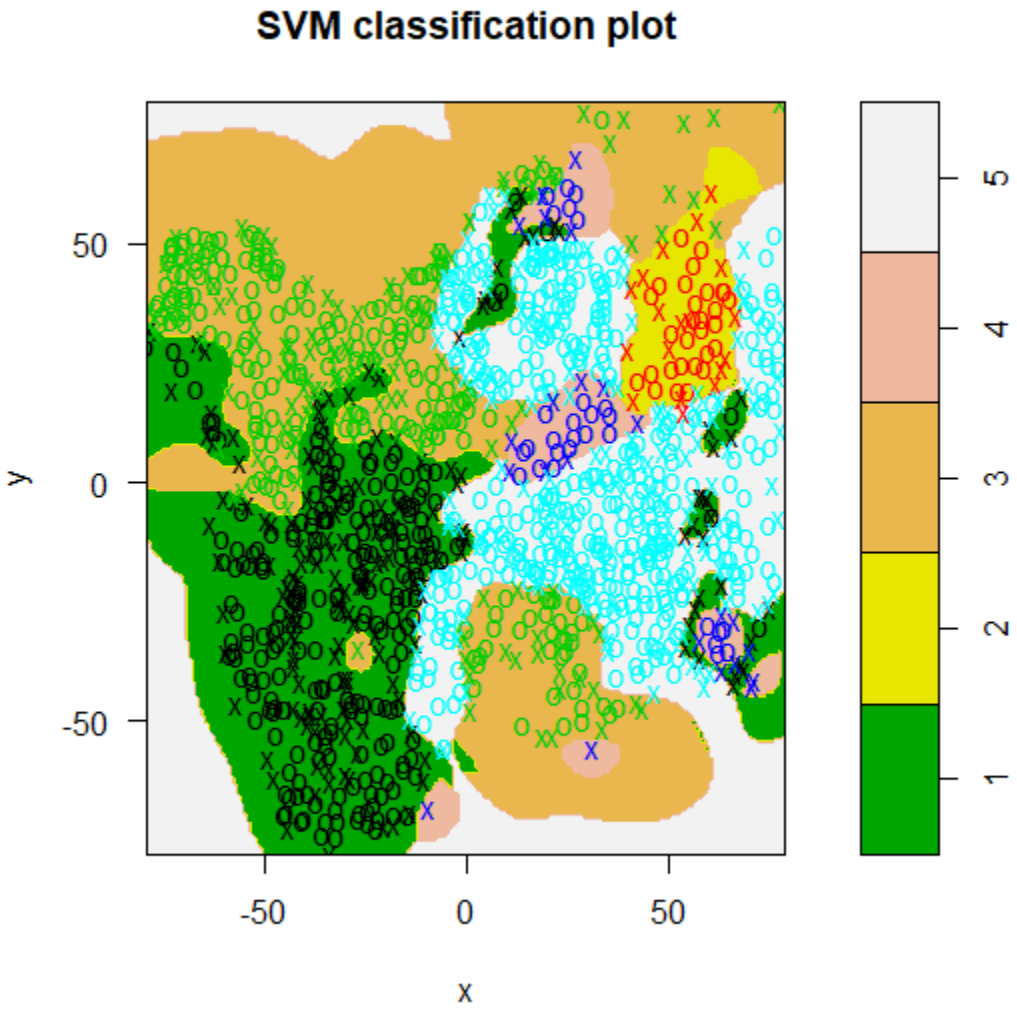
So, a small gamma will provide low bias and high variance while a large gamma will provide higher bias and low variance.

b. LDA won't be a good idea for this data set.



LDA assumes that data is Normally distributed and all groups are identically distributed. From this plot we can see they are not quite identically and normally distributed. So linearity condition will not work here. Data are showing nonlinear attribute here. So linear discriminant analysis will not work well.
Boundaries are not linearly dividing the data set rather they are curvature in shape. So, Here LDA is not a good distinction tool.

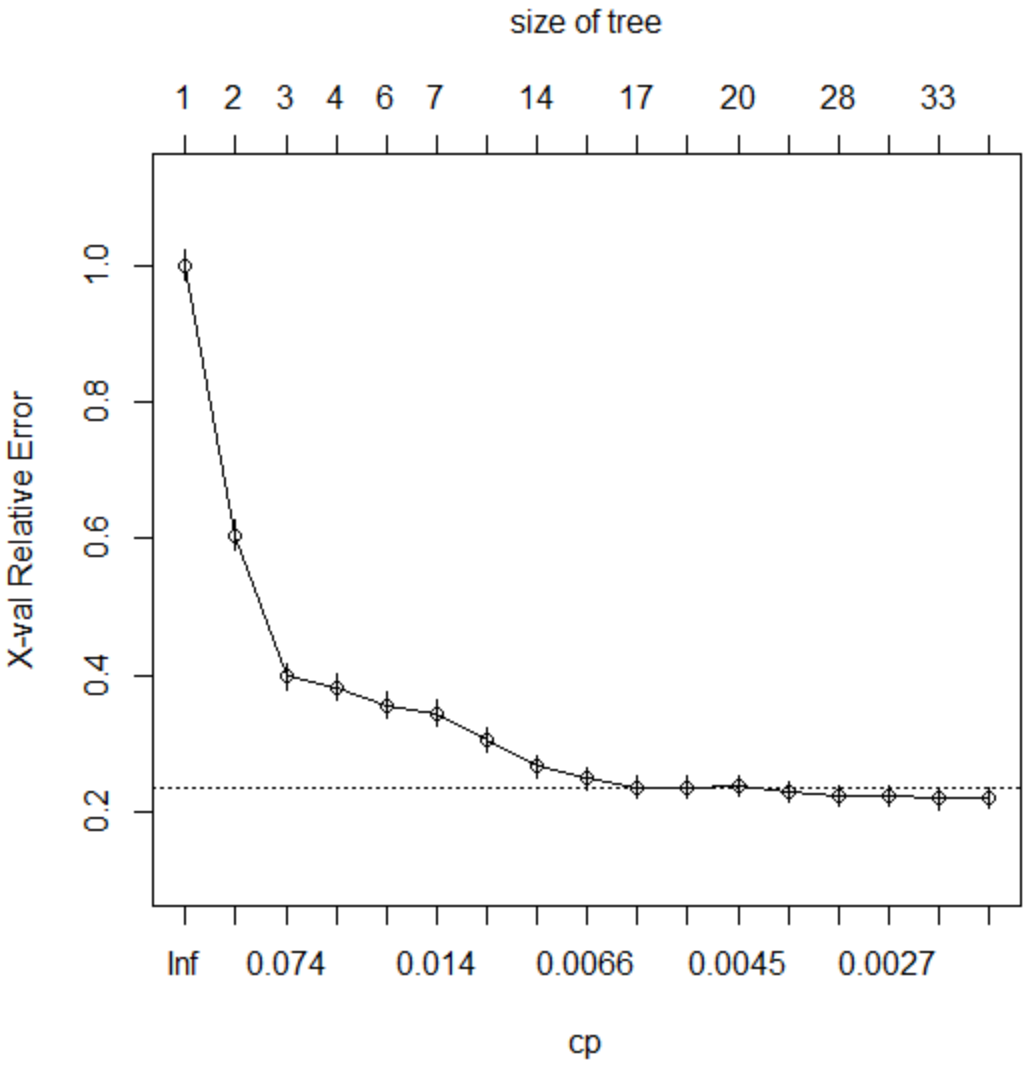
c. Classification analysis by using best tuning parameters we get



Resubstitution error rate is 0.009160305.
Cross validated error rate is 0.2061069
Here it is about 21%. So, in this case, there's not much to be gained by using SVM.

Classification of Trees

Finding best cp



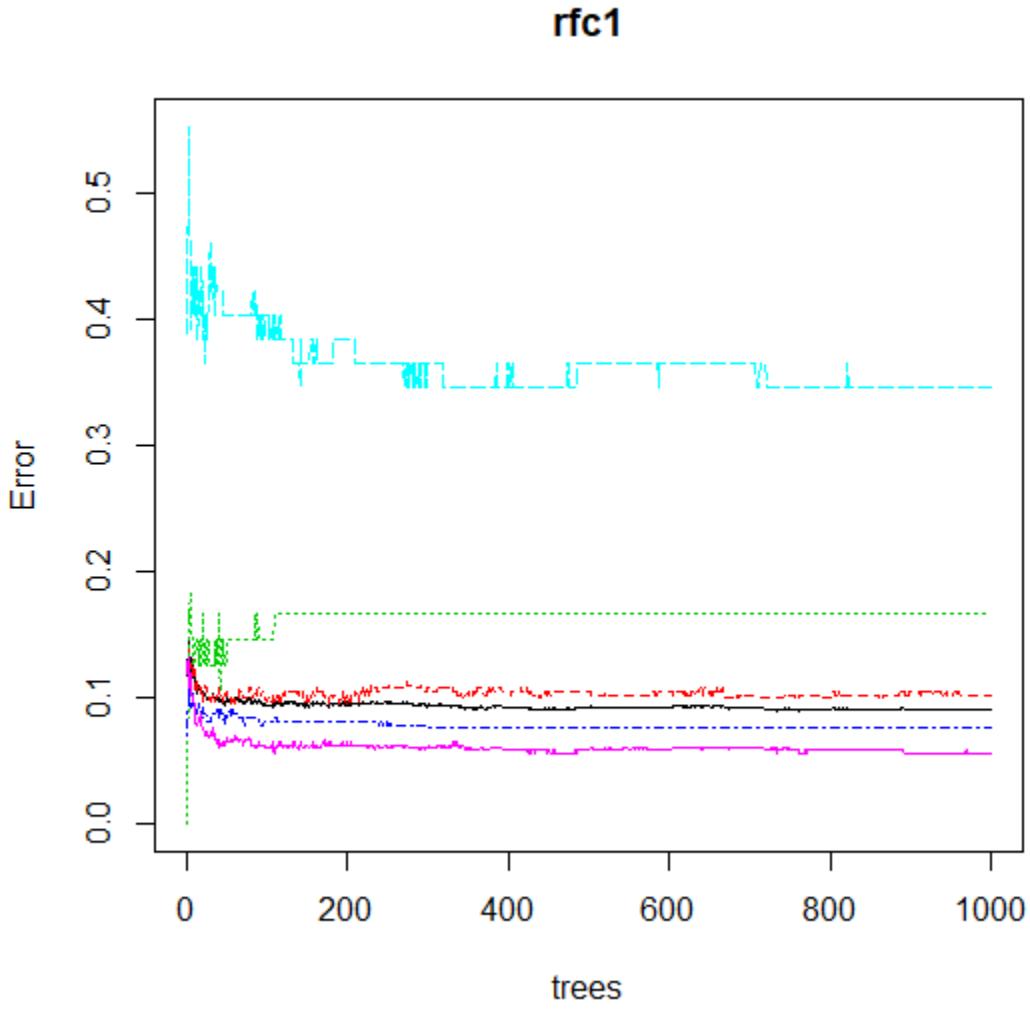
From the plot we have found cp = 0.0017
Now

After pruning Resubstitution success rate = 0.9160305
Resubstitution Error rate= 1-0.9160305 = 0.0839695
Note that we can calculate CV error rate from the output, multiplying the "xerror" rate by the "root node error rate"
Root node error rate=0.62977
xerror=0.21212
Cross validated error rate=root node error rate*xerror= 0.1335868

Random Forest

Classification Analysis

For Begging



Call:

randomForest(formula = Class ~ x + y, data = soil, mtry = 7, ntree = 1000)

Type of random forest: classification

Number of trees: 1000

No. of variables tried at each split: 2

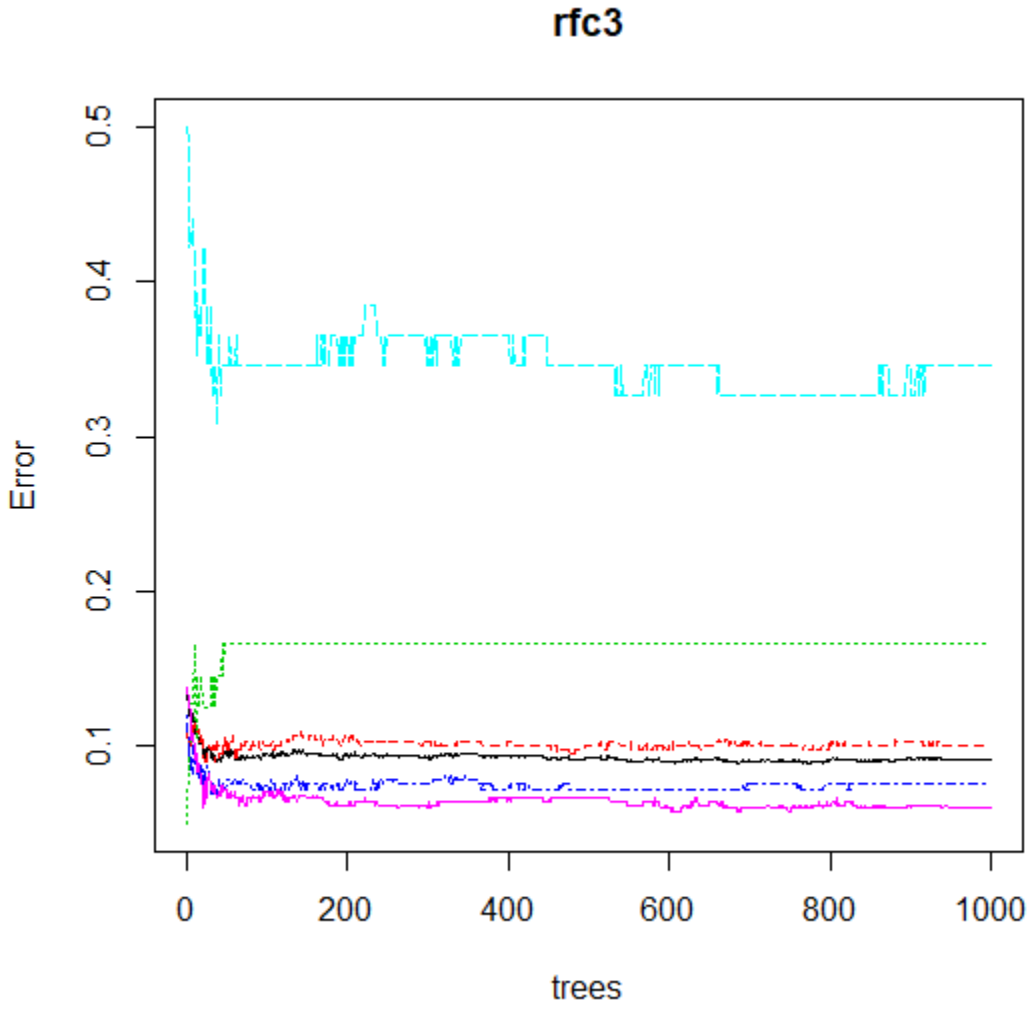
OOB estimate of error rate: 9.01%

Confusion matrix:

	1	2	3	4	5	class.error
1	352	0	15	5	20	0.10204082
2	0	40	3	0	5	0.16666667
3	9	4	308	3	9	0.07507508
4	8	0	4	34	6	0.34615385
5	13	3	5	6	458	0.05567010

Resubstitution error rate = 0.6687023

For True Random Forest



Call:
randomForest(formula = Class ~ x + y, data = soil, mtry = 3, ntree = 1000)
Type of random forest: classification
Number of trees: 1000
No. of variables tried at each split: 2

OOB estimate of error rate: 9.08%

Confusion matrix:

	1	2	3	4	5	class.error
1	353	0	15	5	19	0.09948980
2	0	40	3	0	5	0.16666667
3	9	4	308	3	9	0.07507508
4	9	0	4	34	5	0.34615385
5	14	3	6	6	456	0.05979381

Resubstitution error rate = 0.6694656
Cross validated error rate =0.09312977

For Boosting

Classification analysis

	var	rel.inf
x	x	58.61071
y	y	41.38929

Resubstitution success rate = 0.9648855
Resubstitution error rate = 0.03664122
Cross-validated error rate = 0.1206107

Methods	Resubstitution Error Rate	Cross-validated Error Rate
SVM	0.009160305	0.2061069
Tress	0.0839695	0.1335868
Random Forest	0.6694656	0.09312977
Boosting	0.03664122	0.1206107

R-code

```
##Packages
install.packages("tree")
install.packages("randomForest")
install.packages("gbm")
install.packages("rpart")
install.packages("MASS")
require(tree)
require(randomForest)
require(gbm)
require(rpart)
require(MASS)
##########a
install.packages("e1071")
library(e1071)
soil <- read.csv('Soil_types.csv') # soils dataset from Kanevsky
head(soil)
soil$Class <- factor(soil$Type)
   # this is important, otherwise "SVM" will do regression
x <- soil[,1]
y <- soil[,2]
kolors <- topo.colors(5) # c("red", "blue", "magenta", "green", "grey")
plot(x,y, col="white")
for (i in 1:5){
        subs <- (soil$Type == i)
        points(x[subs], y[subs], col= kolors[i], pch=16)
}
```

```
soil.svm1 <- svm(Class ~ x + y, data = soil, kernel = "linear")


soil.svm1 <- svm(Class ~ x + y, data = soil, kernel = "radial", cost = 1, gamma = 0.1)
plot(soil.svm1, soil, grid=200, color.palette = terrain.colors, y ~ x)


soil.svm2 <- svm(Class ~ x + y, data = soil, kernel = "radial", cost = 1, gamma = 10)
plot(soil.svm2, soil, grid=200, color.palette = terrain.colors, y ~ x)


soil.svm3 <- svm(Class ~ x + y, data = soil, kernel = "radial", cost = 100, gamma = 0.1)
plot(soil.svm3, soil, grid=200, color.palette = terrain.colors, y ~ x)


soil.svm4 <- svm(Class ~ x + y, data = soil, kernel = "radial", cost = 100, gamma = 10)
plot(soil.svm4, soil, grid=200, color.palette = terrain.colors, y ~ x)



##########b

attach(soil)
soil.lda=lda(Class ~ x + y, data = soil)
summary(soil.lda)
 plot(soil.lda)
###############c
soil.tune = tune(svm, Class ~ x + y, data = soil, kernel = "radial", ranges = list(cost = c(1,100),
gamma = c(0.1,10)) )
 summary(soil.tune)
##best
soil.svm4 <- svm(Class ~ x + y, data = soil, kernel = "radial", cost = 100, gamma = 10)
plot(soil.svm4, soil, grid=200, color.palette = terrain.colors, y ~ x)
##resubstitution error rate
 pred = predict(soil.svm4, data = soil, decision.values = TRUE)
 (t1 = table(pred, soil$Class))
 (miss.rate1 = 1 - sum(diag(t1))/sum(t1))
####Cross validation Error rate
V = 10     # number of splits = "folds" in CV
  n = length(soil[,1])
     # to avoid programming complications, cut the size to be divisible by V
           n0 = n %/% V
           n1 = n0 * V


  Vsample = sample(1:n)    # this generates a permutation of length n
  tot.succ = 0      # total number of correctly classified cases
```

```
  for (i in 1:V){
    subset1 = ((i-1)*n0+1):(i*n0)
          test = Vsample[subset1]
          svm.train = svm(Class ~ x + y, data = soil,subset=-test,mtry=3)

          tab1 = table(soil$Class[test], predict(svm.train, soil[test,]))
    tot.succ = tot.succ + sum(diag(tab1))
  }
  (CV.miss.rate = 1 - tot.succ/n)
# about 21%. So, in this case, there's not much to be gained by using SVM.

###Classification of Trees
set.seed(456)
soil.rp = rpart(Class ~ x + y, data = soil,cp=0.001, parms=list(split="gini"))
                        # cp is the complexity parameter

soil.rp
plot(soil.rp, uniform = T, branch = 0.4, main = "soil unpruned, cp=0.001")
 text(soil.rp, all=T, use.n=T, pretty=0, fancy=T, fwidth=0.4, fheight=0.3)
                # adds text labels to the tree plot

  pred = predict(soil.rp, type="class")
  (t1 = table(soil$Class, pred))          # resubstitution success rate
  (resub = sum(diag(t1)/nrow(soil)))     # about 0.92
plotcp(soil.rp)
 printcp(soil.rp)


 cp3 =0.0017

 soil.rp3 = prune(soil.rp, cp= cp3)    # pruning
  plot(soil.rp3, uniform = T, branch = 0.4, margin = 0.03, main = paste("Pima pruned with cp=",
cp3))
    text(soil.rp3, all=T, use.n=T, pretty=0, cex = 0.8, fancy=T, fwidth=0.3, fheight=0.3)
plotcp(soil.rp3)
printcp(soil.rp3)

pred3 = predict(soil.rp3, type="class")
  (t3 = table(soil$Class, pred3))          # resubstitution success rate
  (resub1 = sum(diag(t3)/nrow(soil)))

##cross validation
 # note we can calculate CV error rate from the output, multiplying the "xerror" rate by the "root
node error rate"
rootnodeerrorrate=0.62977
xerror=0.21212
```

```
(CVerrorrate=rootnodeerrorrate*xerror)
#####Error rate is about 13%

#######Random Forest
rfc1 = randomForest(Class~x+y, data = soil, mtry = 7, ntree = 1000)    #  this is bagging
 print(rfc1)
 plot(rfc1)

 d1 = predict(rfc1,type="prob")
   head(d1)
 pred2 = (d1[,2] > 0.3)
 (t2 = table(soil$Class,pred2))
 n = sum(t2)
 (err2 = (n - sum(diag(t2)))/n) ###resubstitution error rate

   rfc3 = randomForest(Class~x+y, data = soil, mtry = 3, ntree = 1000)    #  this is "true" Random
Forest
 print(rfc3)
 plot(rfc3)

 d3 = predict(rfc3,type="prob")
   head(d3)
 pred3 = (d3[,2] > 0.4)
 (t3 = table(soil$Class,pred3))
 n = sum(t3)
 (err3 = (n - sum(diag(t3)))/n)

  ##cross validation
library(randomForest)

n <- dim(soil)[1]
V <- 10      # number of splits = "folds" in CV
# to avoid programming complications, cut the size to be divisible by V
n0 <- n %/% V
n1 <- n0*V
Vsample <- sample(1:n)    # this generates a permutation of length n
tot.succ <- 0      # total number of correctly classified cases
for (i in 1:V){
subset1 <- ((i-1)*n0+1):(i*n0)
test <- Vsample[subset1]
rf.train <- randomForest(Class~x+y, data = soil, subset = -test, mtry = 2)
tab1 <- table(soil$Class[test], predict(rf.train,soil[test,]))
tot.succ <- tot.succ + sum(diag(tab1))
}
(CV.miss.rate <- 1 - tot.succ/n1)
####  0.09847328
```

```
####Boosting
library(gbm)

 boos1 = gbm(Class ~ x+y ,data = soil , distribution ="multinomial", n.trees = 1000,shrinkage =
0.01, interaction.depth = 4)
          # distribution ="multinomial" for multiple outcomes
           # use     distribution ="gaussian"    for regression, distribution = "bernoulli"  for 0/1
outcomes
 summary(boos1)
pred = predict(boos1, data = soil, n.trees = 1000, type="response")
n1=length(soil$Class)
classY1 = rep(0,n)
 for (i in 1:n){
  classY1[i] = which.max(pred[i,,])
 }

(t0 = table(soil$Class, classY1))
  (re.succ = sum(diag(t0)/n1)) #resubstitution success rate
(reer.rate <- 1 - re.succ)
###Cross validation
n = nrow(soil)
 n2 = floor(n/2)
 train = sample(1:n, n-n2, replace = F)

 boo1t = gbm(Class ~ x+y ,data = soil[train,] , distribution ="multinomial", n.trees = 500,
shrinkage = 0.01, interaction.depth = 15)


 pred1 = predict(boo1t, newdata = soil[-train,], n.trees = 500, type="response")
   pred1[1:5,,]
 classY = rep(0,n2)
 for (i in 1:n2){
   classY[i] = which.max(pred1[i,,])
 }

 (t4 = table(soil[-train,]$Class, classY))
  (test.succ = sum(diag(t4)/n2))



 (CV.miss.rate <- 1 - test.succ)
```

Problem 2

a. Comparative table of training MSE and 10-fold cross-validated MSE

Methods	Training MSE	10-fold cross-validated MSE
S.O. Polynomial	0.07404312	0.02472353
Trees	0.0938181	0.09384734
Random Forest	0.0862197	0.005407757
SVM	0.08695172	0.004039536

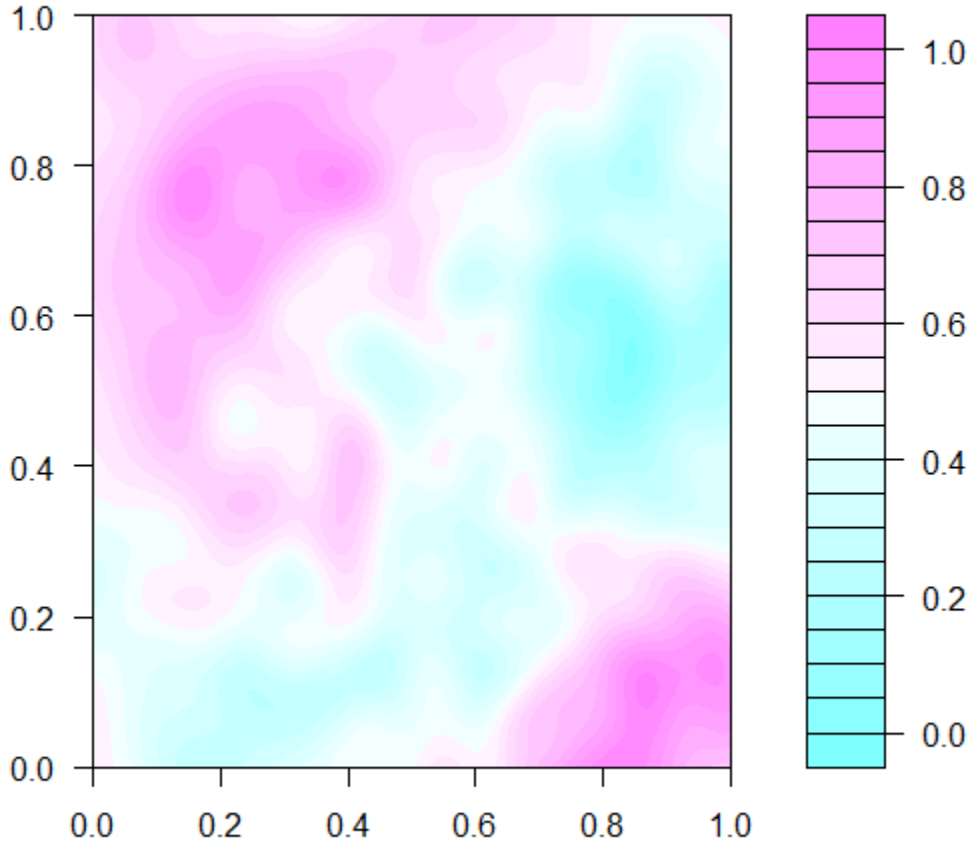
R-code

```
porosity=read.csv('Porosity.csv')

#####a
head(porosity)
attach(porosity)
n = nrow(porosity)
 n2 = floor(n/2)
 train = sample(1:n, n-n2, replace = F)
lm1=lm(por~x1+x2+I(x1^2)+I(x2^2)+x1*x2 , data=porosity[train,])
summary(lm1)
pred=predict(lm1)
MSE=sum(((por-pred)^2))/length(por)
MSE
### 0.07404312
##10fold-MSE
n <- dim(porosity)[1]
V <- 10      # number of splits = "folds" in CV
# to avoid programming complications, cut the size to be divisible by V
n0 <- n %/% V
n1 <- n0*V
Vsample <- sample(1:n)    # this generates a permutation of length n
Ers <- 0      # total number of correctly classified cases
for (i in 1:V){
subset1 <- ((i-1)*n0+1):(i*n0)
test <- Vsample[subset1]
lm2=lm(por~x1+x2+I(x1^2)+I(x2^2)+x1*x2 , data=porosity, subset = -test)
pred1=predict(lm2,porosity[test,])
error= porosity[test,]$por - pred1

Ers=Ers+sum((error^2))/n1
}
(MSEcv=Ers)
####0.02472353
####tress
finding good tuning parameter
por.rp = rpart(por~x1+x2, data = porosity)
plotcp(por.rp)
printcp(por.rp)
```

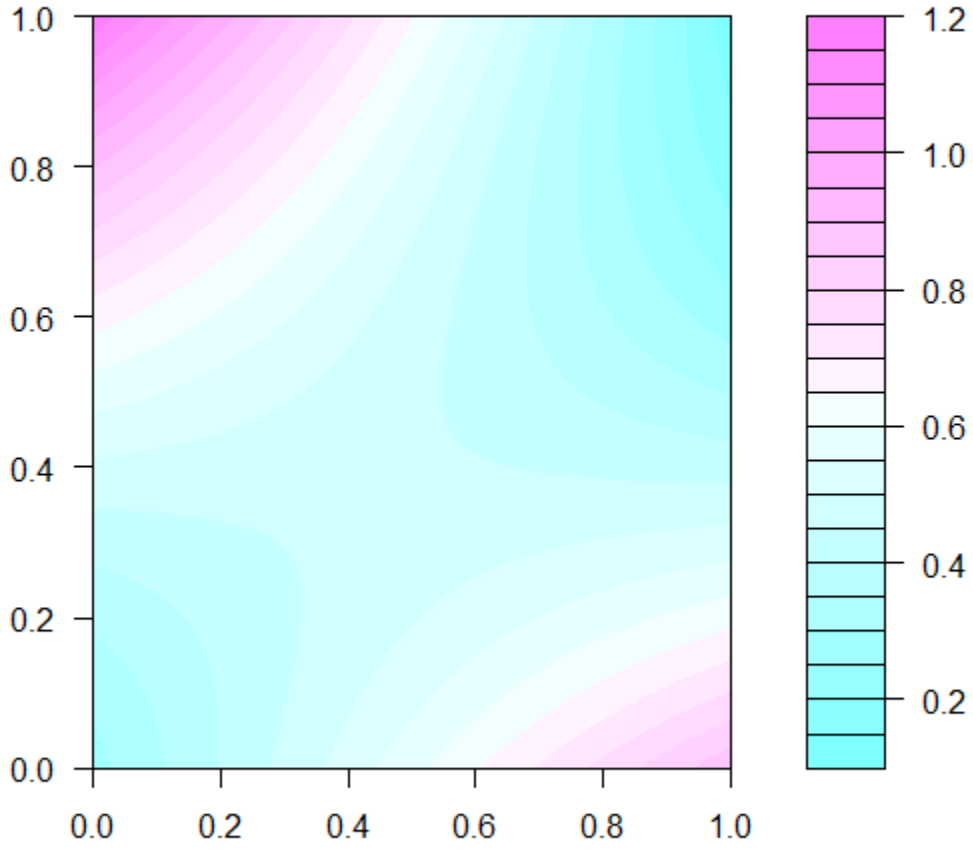
```
##0.01


n = nrow(porosity)
 n2 = floor(n/2)
 train = sample(1:n, n-n2, replace = F)
por.rp1 = rpart(por~x1+x2 , data=porosity[train,], cp=0.01)
summary(por.rp1)
predt=predict(por.rp1)
MSEt=sum(((por-predt)^2))/length(por)
MSEt
### 0.0938181
####CVmse
n <- dim(porosity)[1]
V <- 10      # number of splits = "folds" in CV
# to avoid programming complications, cut the size to be divisible by V
n0 <- n %/% V
n1 <- n0*V
Vsample <- sample(1:n)    # this generates a permutation of length n
Erst <- 0      # total number of correctly classified cases
for (i in 1:V){
subset1 <- ((i-1)*n0+1):(i*n0)
test <- Vsample[subset1]
por.rpc = rpart(por~x1+x2 , data=porosity, subset = -test ,cp=0.01)
predt=predict(por.rpc,porosity[test,])
errort= porosity[test,]$por - pred1
Erst=Erst+sum((errort^2))/n1
}
(MSEcvt=Erst)
###0.09384734

####Random Forest
n = nrow(porosity)
 n2 = floor(n/2)
 train = sample(1:n, n-n2, replace = F)
rfc = randomForest(por~x1+x2, data = porosity[train,], mtry = 3, ntree = 1000)
predr=predict(rfc)
MSEr=sum(((por-predr)^2))/length(por)
MSEr
###0.0862197
####CVmse
n <- dim(porosity)[1]
V <- 10      # number of splits = "folds" in CV
# to avoid programming complications, cut the size to be divisible by V
n0 <- n %/% V
n1 <- n0*V
Vsample <- sample(1:n)    # this generates a permutation of length n
```

```
Ersr <- 0      # total number of correctly classified cases
for (i in 1:V){
subset1 <- ((i-1)*n0+1):(i*n0)
test <- Vsample[subset1]
rfcc = randomForest(por~x1+x2, data = porosity ,subset=-test, mtry = 2, ntree = 1000)
predrc=predict(rfcc,porosity[test,])
errorr= porosity[test,]$por - predrc
Ersr=Ersr+sum((errorr^2))/n1
}
(MSEcvr=Ersr)
#####0.005407757
###SVM
n = nrow(porosity)
 n2 = floor(n/2)
 train = sample(1:n, n-n2, replace = F)
por.svm2 <- svm(por ~ x1 + x2, data = porosity[train,], kernel = "radial", cost = 100, gamma =
10)
predsvm=predict(por.svm2)
MSEs=sum(((por-predsvm)^2))/length(por)
MSEs
#####0.08695172
####CVmse
n <- dim(porosity)[1]
V <- 10      # number of splits = "folds" in CV
# to avoid programming complications, cut the size to be divisible by V
n0 <- n %/% V
n1 <- n0*V
Vsample <- sample(1:n)    # this generates a permutation of length n
Erss <- 0      # total number of correctly classified cases
for (i in 1:V){
subset1 <- ((i-1)*n0+1):(i*n0)
test <- Vsample[subset1]
por.svms <- svm(por ~ x1 + x2, data = porosity,subset=-test, kernel = "radial", cost = 100,
gamma = 10)
predrc=predict(por.svms,porosity[test,])
errors= porosity[test,]$por - predrc
Erss=Erss+sum((errors^2))/n1
}
(MSEcvs=Erss)

###0.004039536
```

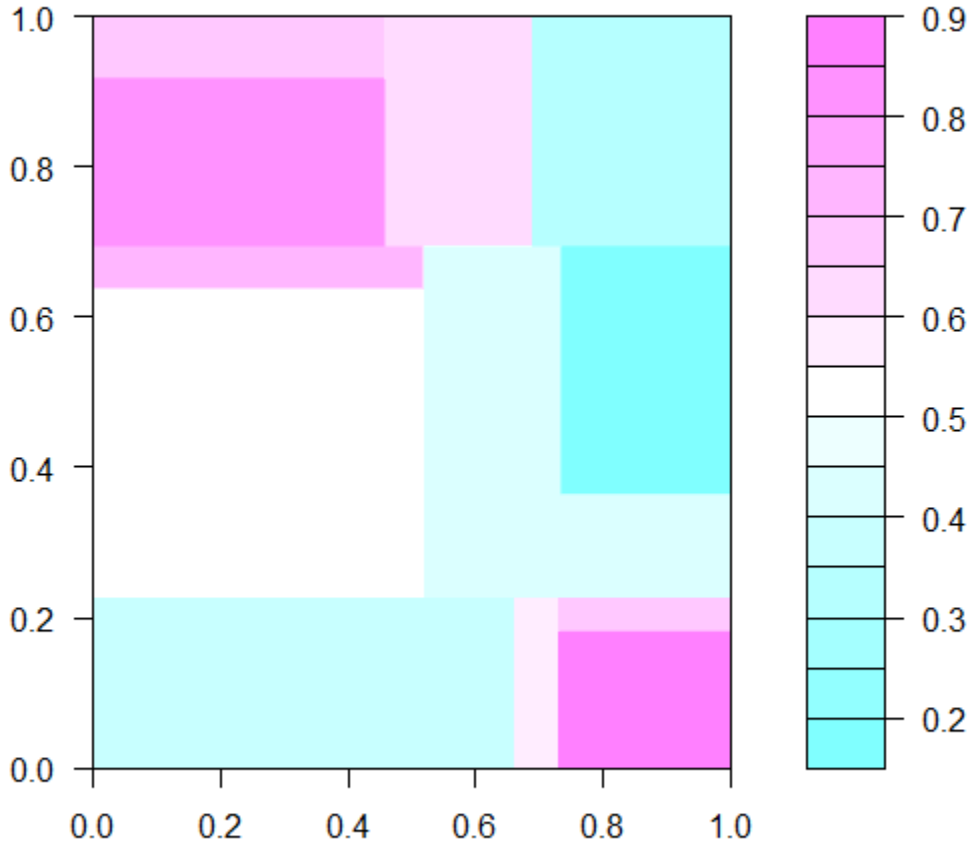
b. Contour plot for SVM



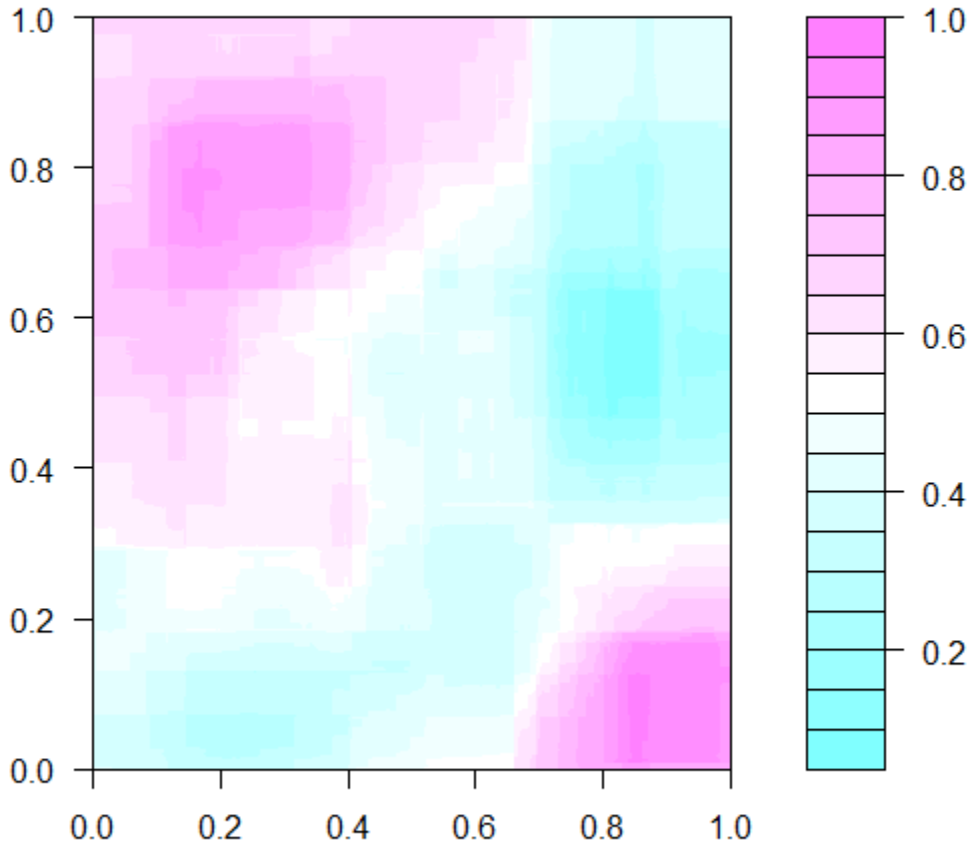
Contour plot for second order polynomial regression



Contour Plot for tress



Contour plot for Random Forest



R-code

```
###b
library(e1071O)
svmp <- svm(por ~ x1+x2, data = porosity, cost = 100, gamma = 10)
xgrid <- seq(0,1,0.002)
ygrid <- seq(0,1,0.002)
nx <- length(xgrid)
ny <- length(ygrid)
xygrid1 <- xgrid %x% rep(1,ny)
xygrid2 <- rep(1,nx) %x% ygrid
newxy <- data.frame(x1 = xygrid1, x2 = xygrid2)
z <- predict(svmp, newdata = newxy)
z <- matrix(z, ny, nx)
contour( xgrid, ygrid, t(z), labcex = 0.9)
filled.contour(xgrid,ygrid,t(z))

###Poly
lmc=lm(por~x1+x2+I(x1^2)+I(x2^2)+x1*x2 , data=porosity)
xgrid <- seq(0,1,0.002)
ygrid <- seq(0,1,0.002)
nx <- length(xgrid)
ny <- length(ygrid)
xygrid1 <- xgrid %x% rep(1,ny)
xygrid2 <- rep(1,nx) %x% ygrid
newxy <- data.frame(x1 = xygrid1, x2 = xygrid2)
z <- predict(lmc, newdata = newxy)
z <- matrix(z, ny, nx)
contour( xgrid, ygrid, t(z), labcex = 0.9)
filled.contour(xgrid,ygrid,t(z))
```

```
##Tress
por.rpc = rpart(por~x1+x2, data = porosity)
xgrid1 <- seq(0,1,0.002)
ygrid1 <- seq(0,1,0.002)
nx1 <- length(xgrid)
ny1 <- length(ygrid)
xygrid1a <- xgrid1 %x% rep(1,ny1)
xygrid2b <- rep(1,nx1) %x% ygrid1
newxy1 <- data.frame(x1 = xygrid1a, x2 = xygrid2b)
z1 <- predict(por.rpc, newdata = newxy1)
z1 <- matrix(z1, ny1, nx1)
contour( xgrid1, ygrid1, t(z1), labcex = 0.9)
filled.contour(xgrid1,ygrid1,t(z1))

###Random Forest

rfcc = randomForest(por~x1+x2, data = porosity, mtry = 2, ntree = 1000)
xgrid <- seq(0,1,0.002)
ygrid <- seq(0,1,0.002)
nx <- length(xgrid)
ny <- length(ygrid)
xygrid1 <- xgrid %x% rep(1,ny)
xygrid2 <- rep(1,nx) %x% ygrid
newxy <- data.frame(x1 = xygrid1, x2 = xygrid2)
z <- predict(rfcc, newdata = newxy)
z <- matrix(z, ny, nx)
contour( xgrid, ygrid, t(z), labcex = 0.9)
filled.contour(xgrid,ygrid,t(z))
```