Homework 3

Jing Leng

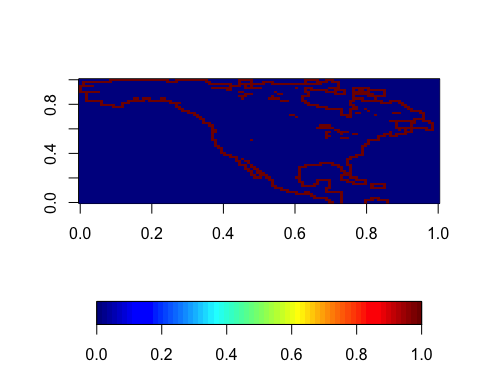
October 6, 2014

## 1

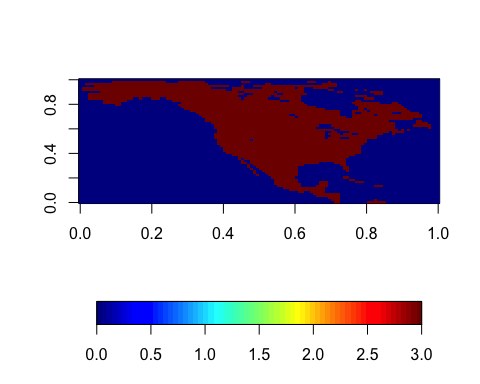
1. the locations of LAI 0 are waters. The location of LAI 0 and at least one neiboring location with non-zero LAI is on the boundary.
2. Use breadth first search

## Loading required package: spam  
## Loading required package: grid  
## Spam version 1.0-1 (2014-09-09) 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

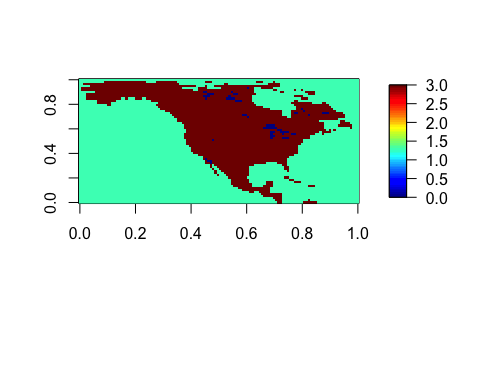
LAIsp <- LAI[,,7]  
bound <- array(0, c(120, 60))  
for (i in 1:120) {  
 for (j in 1:60) {  
 if (LAIsp[i, j] == 0 && sum(LAIsp[(i-1):min((i+1), 120), (j-1):min((j+1), 60)]) > 0 ) {  
 bound[i, j] <- 1   
 }  
 }  
}  
  
image.plot(bound, horizontal = T)



L = (LAI[,,7] > 0) \* 3  
image.plot(L, horizontal = T)



n = dim(L)[1]  
m = dim(L)[2]  
  
unexplored = array(0, c(120, 60))  
crt\_r = c(1) #row index  
crt\_c = c(1) #column index  
unexplored[1,1] = 1  
  
step\_r = c(-1, 1, 0, 0) #searching direction in row  
step\_c = c(0, 0, -1, 1) #searching direction in column  
  
while (length(crt\_r) != 0){  
 nbh\_r = nbh\_c = c()  
 for (i in 1:length(crt\_r)){  
 L[crt\_r[i],crt\_c[i]] = 1.3 #grean ocean:)  
 for (j in 1:4){ #search for 4 directions  
 next\_r = min(max(crt\_r[i]+step\_r[j], 1), n) #row index of the new point  
 next\_c = min(max(crt\_c[i]+step\_c[j], 1), m) #column index of the new point  
 if(L[next\_r, next\_c] == 0 & unexplored[next\_r, next\_c] == 0){  
 nbh\_r = c(nbh\_r, next\_r)  
 nbh\_c = c(nbh\_c, next\_c)  
 unexplored[next\_r, next\_c] = 1  
 }  
 }  
 }  
 crt\_r = nbh\_r   
 crt\_c = nbh\_c  
}  
image.plot(L) #plot the root points in this search phase



## 2

Use bubble sort. Single out the largest item and move it to the rightmost.

mysort <- function (x, y, z) {  
 if (x > y) {  
 t = x   
 x = y  
 y = t  
 }  
 if (y > z) {  
 t = y  
 y = z  
 z = t  
 if (x > y) {  
 t = x   
 x = y  
 y = t  
 }  
 }  
 return (c(x, y, z))  
}  
  
mysort(1,5,2)

## [1] 1 2 5

mysort2 <- function (x) {  
   
 for (j in 1:(length(x) -1) ) {  
   
 for (i in 1:(length(x) - j)) {  
 if (x[i] > x[i+1]) {  
 t = x[i]  
 x[i] = x[i+1]   
 x[i+1] = t  
 }  
 }  
 }  
 return (x)  
}   
  
mysort2(c(10,6,3,12,5,3,5,6,9))

## [1] 3 3 5 5 6 6 9 10 12

mysort3 <- function (s, d) {  
 s <- mysort2(s)  
 if (d == 1) {  
 return (s)  
 }  
 if (d == 0) {  
 for (i in 1:(length(s)/2)) {  
 t = s[i]  
 s[i] = s[length(s) - i +1]  
 s[length(s) - i + 1] = t  
 }  
 return (s)  
 }  
   
   
}  
mysort3(c(14, 12, 19, 3, 9, 21, 19), 0)

## [1] 21 19 19 14 12 9 3

mysort3(c(14, 12, 19, 3, 9, 21, 19), 1)

## [1] 3 9 12 14 19 19 21