Generate_for_publication

Arnaud

25/03/2022

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
graphics.off()
### Definitive script
#dev.off()
### Imports
library("usethis")
```

```
## Warning: package 'usethis' was built under R version 4.0.5
```

```
library("roxygen2")
```

```
## Warning: package 'roxygen2' was built under R version 4.0.5
```

```
library("devtools")
```

```
## Warning: package 'devtools' was built under R version 4.0.5
```

```
library(Rcpp)
```

```
## Warning: package 'Rcpp' was built under R version 4.0.5
```

```
library(RcppEigen)
library(methods)
library(lattice)
```

```
## Warning: package 'lattice' was built under R version 4.0.5
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union
```

```
library(purrr)
library(Matrix)
```

```
## Warning: package 'Matrix' was built under R version 4.0.5
```

```
library("viridis")
```

```
## Warning: package 'viridis' was built under R version 4.0.5
```

```
## Loading required package: viridisLite
```

```
library("optAM")
```

```
## Registered S3 methods overwritten by 'RcppArmadillo':
##   method               from
##   predict.fastLm       RcppEigen
##   print.fastLm         RcppEigen
##   summary.fastLm       RcppEigen
##   print.summary.fastLm RcppEigen
```

```
## Warning: replacing previous import 'RcppArmadillo::fastLmPure' by
## 'RcppEigen::fastLmPure' when loading 'optAM'
```

```
## Warning: replacing previous import 'RcppArmadillo::fastLm' by
## 'RcppEigen::fastLm' when loading 'optAM'
```

```
library(hrbrthemes)
```

```
## Warning: package 'hrbrthemes' was built under R version 4.0.5
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.
```

```
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
```

```
##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
library(grid)
library(raster)
```

```
## Warning: package 'raster' was built under R version 4.0.5
```

```
## Loading required package: sp
```

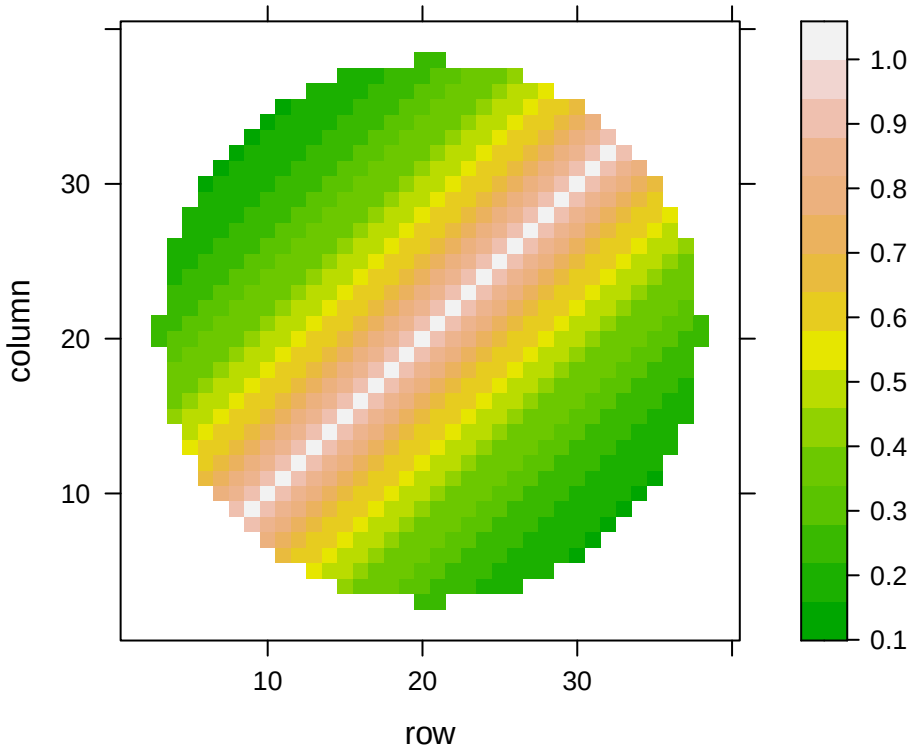
```
## Warning: package 'sp' was built under R version 4.0.5
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

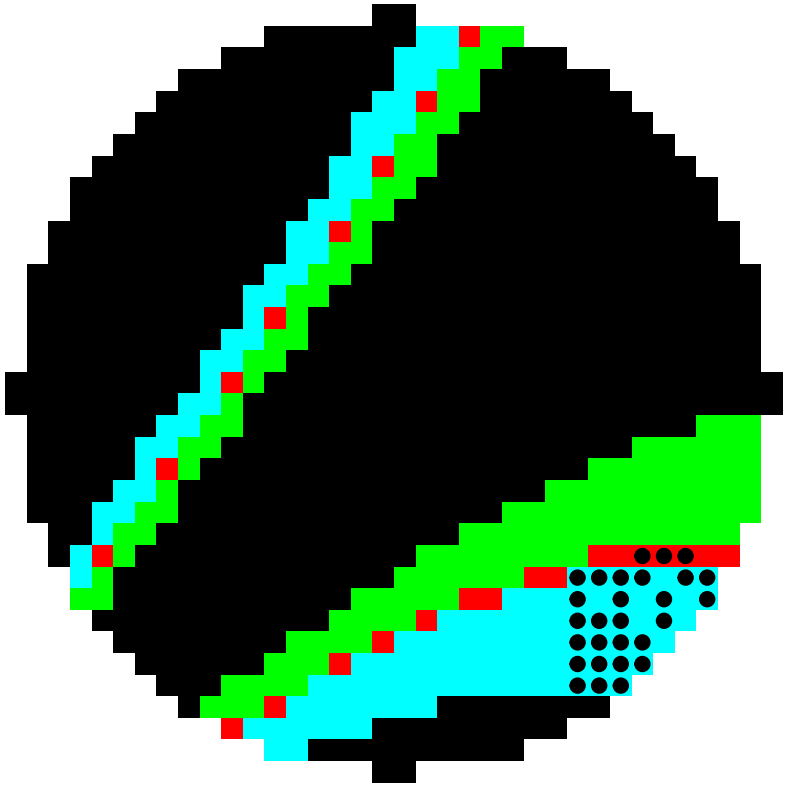
Including Plots

You can also embed plots, for example:




```
## Warning in .couldBeLonLat(x, warnings = warnings): CRS is NA. Assuming it is
## longitude/latitude
```

```
## Warning in if (class(cost) == "dtCMatrix") {: la condition a une longueur > 1 et
## seul le premier élément est utilisé
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
## [1] 1565
```

```
choix_ma = choix
choix_ma = choix_ma[(choix_ma[,1]!=-1),]

XY_ma = (choix[,c(1,2)]-1/2)/nrow
#points(XY_ma[,2], XY_ma[,1],col="white",pch=19)

possibilities = gsc
possibilities = possibilities +1

XY_ma2 = (possibilities[,c(1,2)]-1/2)/nrow
#points(XY_ma2[,2], XY_ma2[,1],col="brown",pch=19)


colonisation_matrices = cm

choix_ma_xy = cbind(choix_ma[,1],choix_ma[,2])
indices_choix_ma = matrix(0,length(choix_ma_xy[,1]))

for (i in (1:length(choix_ma_xy[,1]))){
  h = NA
  for (j in (1:length(possibilities[,1]))){
    if ((possibilities[j,1]==(choix_ma_xy[i,1]))&(possibilities[j,2]==(choix_ma_xy[i,2]))){
```

```
      h = possibilities[j,3]
      break
    }
  }
  if (!is.na(h)){
    indices_choix_ma[i] = possibilities[h,3]
  }

}

print(indices_choix_ma)
```

```
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]   91
##  [6,]   40
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]   71
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
## [21,]    0
## [22,]    0
## [23,]    0
## [24,]    0
## [25,]    0
## [26,]    0
## [27,]    0
## [28,]  285
## [29,]    0
## [30,]    0
## [31,]    0
## [32,]    0
## [33,]    7
## [34,]    0
## [35,]    0
## [36,]    0
## [37,]    0
```

```
XY_pres = which(pres==1,arr.ind = T)
XY_pres = cbind(XY_pres[,1],XY_pres[,2])
```

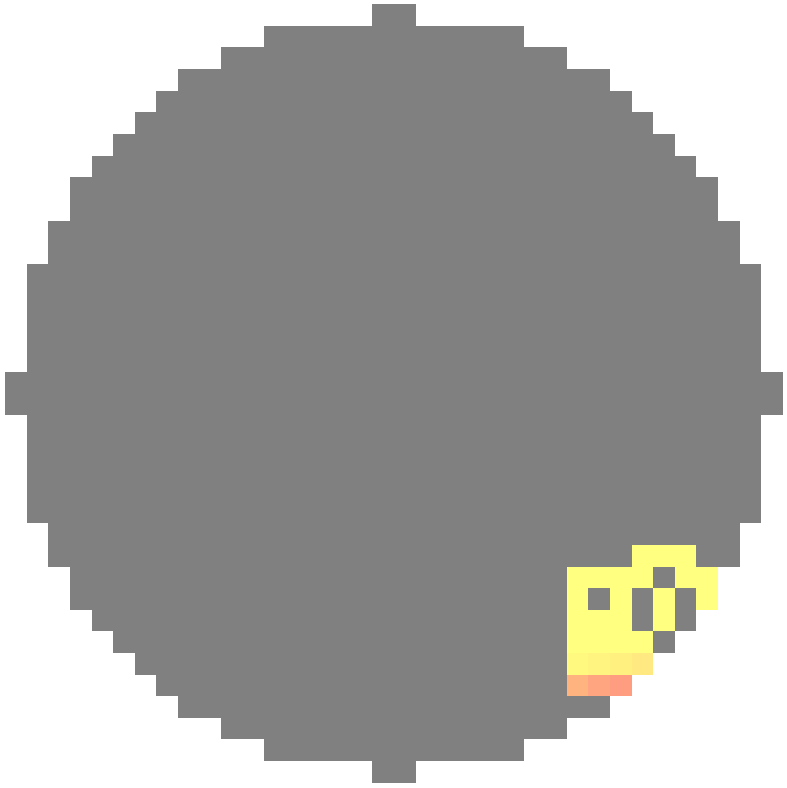
```
indices_pres = matrix(0,length(XY_pres[,1]))
#XY_pres = XY_pres[nrow:1,]
for (i in (1:length(XY_pres[,1]))){
  h = NA
  for (j in (1:length(possibilities[,1]))){
    if ((possibilities[j,1]==(nrow+1-XY_pres[i,1]))&(possibilities[j,2]==(XY_pres[i,2]))){
      h = possibilities[j,3]
      break
    }
  }
  if (!is.na(h)){
    indices_pres[i] = possibilities[h,3]
  }

}

print(indices_pres)
```

```
##       [,1]
##  [1,]  224
##  [2,]  223
##  [3,]  222
##  [4,]  221
##  [5,]  220
##  [6,]  219
##  [7,]  234
##  [8,]  232
##  [9,]  231
## [10,]  230
## [11,]  229
## [12,]  244
## [13,]  243
## [14,]  242
## [15,]  241
## [16,]  240
## [17,]  239
## [18,]  254
## [19,]  253
## [20,]  250
## [21,]  249
## [22,]  263
## [23,]  261
## [24,]  260
## [25,]  271
## [26,]  270
## [27,]  277
## [28,]  276
```

```
#groa = sparseMatrix(i = (a@i[1:length(a@p)-1]+1), j = a@p[1:length(a@p)-1]+1, x = a@x[1:length(a@p)-1]
coloni = as.matrix(cm[[1]])[,indices_pres]
colo2 = 0.99999999 - coloni
colo3 = round(1-apply(colo2, FUN= prod,2),2)
#colo3 = 1 - prod(colo2)
```

```
#####
##### Afficher la situation SAALEE (si aucune action légitime n'est entreprise)
#####

rvb_tensor = array(0.5,c(nrow,ncol,3))

for (i in 1:length(c(colo3))){
  if (colo3[i]>=0.2){
    #rvb_tensor[possibilities[indices_pres[i],1],possibilities[indices_pres[i],2],1]=0.5 + colo3[i]/2#R
    rvb_tensor[possibilities[indices_pres[i],1],possibilities[indices_pres[i],2],2]=0.5 + colo3[i]/2 #G
    #rvb_tensor[possibilities[indices_pres[i],1],possibilities[indices_pres[i],2],3]=0#B
  }

}

for (i in 1:length(XY_pres[,1])){
  rvb_tensor[nrow+1-XY_pres[i,1],XY_pres[i,2],1]=1 #R
  #rvb_tensor[nrow+1-XY_pres[i,1],XY_pres[i,2],2]=0 #G
  #rvb_tensor[nrow+1-XY_pres[i,1],XY_pres[i,2],3]=0 #B
}
# for (i in 1:length(XY_pres[,1])){
#   rvb_tensor[nrow-XY_pres[i,1]+1,XY_pres[i,2],1]=0 #R
#   rvb_tensor[nrow-XY_pres[i,1]+1,XY_pres[i,2],2]=1 #G
#   rvb_tensor[nrow-XY_pres[i,1]+1,XY_pres[i,2],3]=0 #B
# }


rvb_tensor[is.na(map)]=1
rvb_tensor2 = round(rvb_tensor*255)
raster_RGB = stack(raster(rvb_tensor2[,,1]),raster(rvb_tensor2[,,2]),raster(rvb_tensor2[,,3]))
plotRGB(flip(raster_RGB))
```

```
## Warning in .couldBeLonLat(x, warnings = warnings): CRS is NA. Assuming it is
## longitude/latitude
```



```
indices_pres = matrix(0,N_cycles,length(gsc[,1]))
for (i in 1:N_cycles){
  cm_loc = cm[[i]]
  for (j in 1:length(gsc[,1])){
    indices_pres[i,j] = sum(cm_loc[,j])
  }
}

opt_each_site = apply(indices_pres,FUN=max,2)

library("ramify")
```

```
## Warning: package 'ramify' was built under R version 4.0.5
```

```
##
## Attaching package: 'ramify'
```

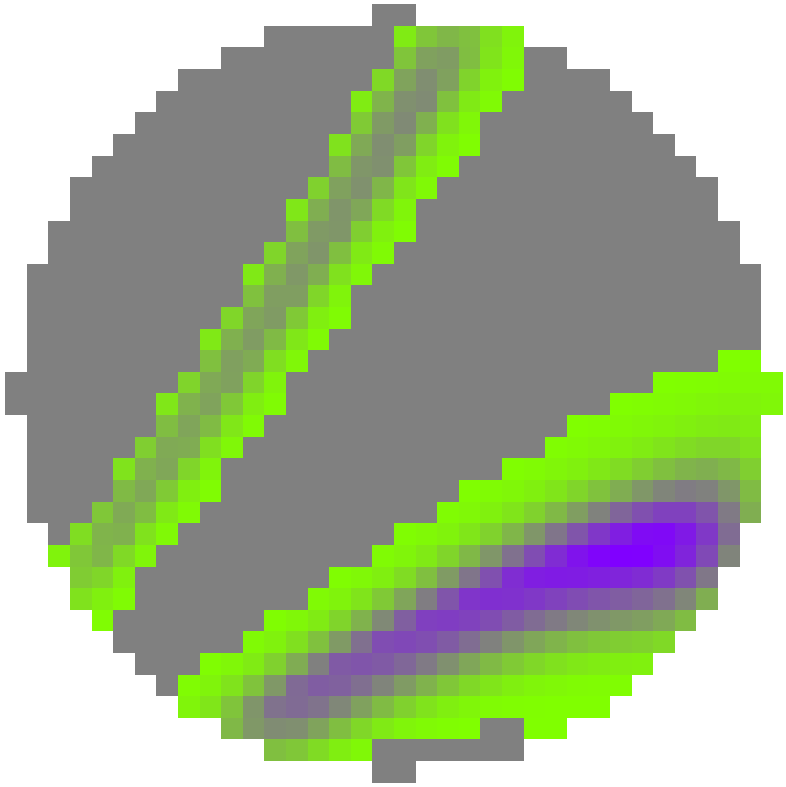
```
## The following objects are masked from 'package:Matrix':
##
##     tril, triu
```

```
## The following object is masked from 'package:purrr':
##
##     flatten
```

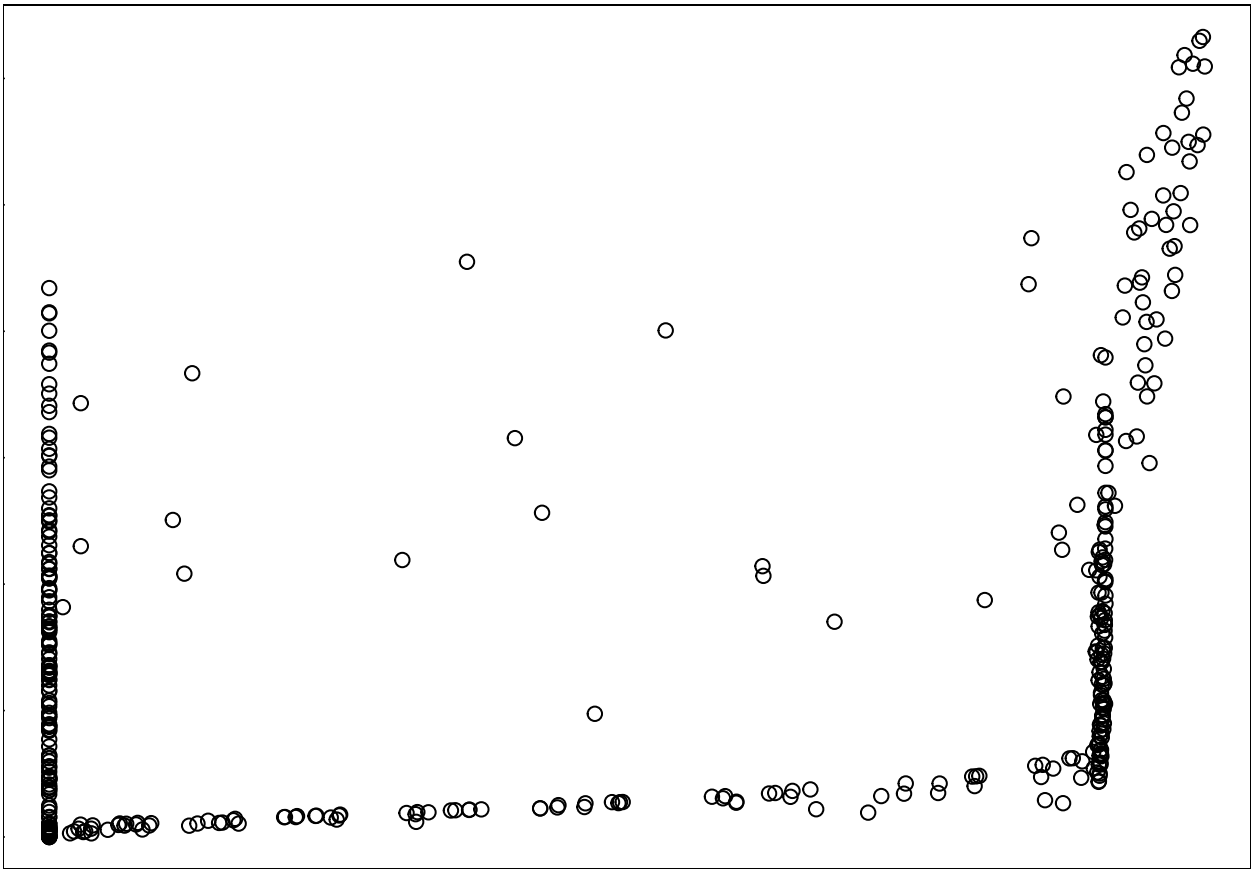
```
## The following object is masked from 'package:graphics':
##
##     clip
```

```r
#arg_opt_site = apply(indices_pres,FUN=argmax,2)


efficienty = opt_each_site / cost[gsc[,c(1,2)]+1]
eff = efficienty / max(efficienty)
#eff = (log(efficienty)-min(log(efficienty)))/(max(log(efficienty))-min(log(efficienty)))

eff_tensor = array(0.5,c(nrow,ncol,3))

for (i in 1:length(efficienty)){
  #rvb_tensor[nrow+1-XY_pres[i,1],XY_pres[i,2],1]=1 #R
  eff_tensor[gsc[i,1]+1,gsc[i,2]+1,2]=1-eff[i]  #G
  eff_tensor[gsc[i,1]+1,gsc[i,2]+1,3]=eff[i]  #B
}

# for (i in 1:length(XY_pres[,1])){
#   rvb_tensor[nrow-XY_pres[i,1]+1,XY_pres[i,2],1]=0 #R
#   rvb_tensor[nrow-XY_pres[i,1]+1,XY_pres[i,2],2]=1 #G
#   rvb_tensor[nrow-XY_pres[i,1]+1,XY_pres[i,2],3]=0 #B
# }

eff_tensor [is.na(map)]=1
eff_tensor2 = round(eff_tensor*255)
raster_RGB = stack(raster(eff_tensor2[,,1]),raster(eff_tensor2[,,2]),raster(eff_tensor2[,,3]))
plotRGB(flip(raster_RGB))
```

```
## Warning in .couldBeLonLat(x, warnings = warnings): CRS is NA. Assuming it is
## longitude/latitude
```

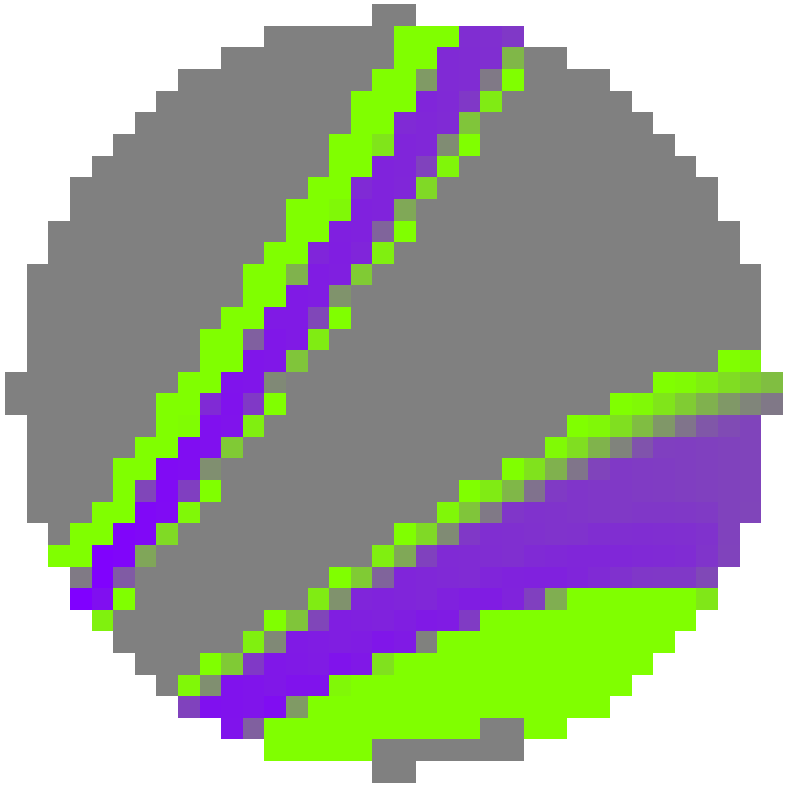


```
diag_cms = matrix(0,N_cycles,length(gsc[,1]))
for (i in 1:N_cycles){
  cm_loc = cm[[i]]
  diag_cms[i,] = diag(cm_loc)
}

maxi_diags = apply(diag_cms,FUN=max,2)

plot(maxi_diags,opt_each_site)
```



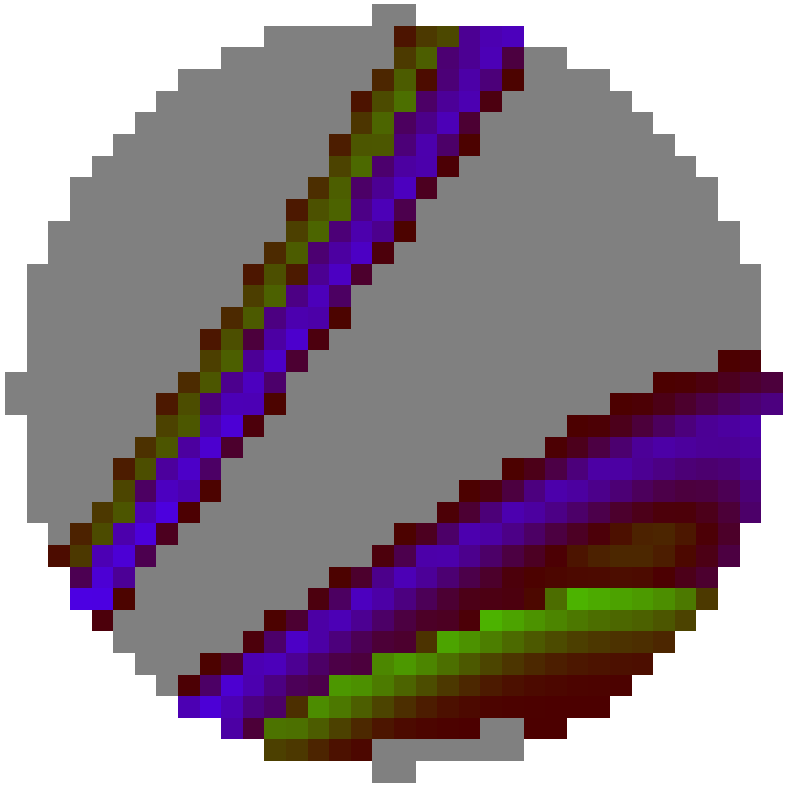
```
### ### ###
efficienty3 = maxi_diags / cost[gsc[,c(1,2)]+1]
eff3 = efficienty3 / max(efficienty3)
#eff = (log(efficienty)-min(log(efficienty)))/(max(log(efficienty))-min(log(efficienty)))

eff_tensor3 = array(0.5,c(nrow,ncol,3))

for (i in 1:length(efficienty3)){
  #rvb_tensor[nrow+1-XY_pres[i,1],XY_pres[i,2],1]=1 #R
  eff_tensor3[gsc[i,1]+1,gsc[i,2]+1,2]=1-eff3[i]  #G
  eff_tensor3[gsc[i,1]+1,gsc[i,2]+1,3]=eff3[i]  #B
}

eff_tensor3 [is.na(map)]=1
eff_tensor3 = round(eff_tensor3*255)
raster_RGB = stack(raster(eff_tensor3[,,1]),raster(eff_tensor3[,,2]),raster(eff_tensor3[,,3]))
plotRGB(flip(raster_RGB))
```

```
## Warning in .couldBeLonLat(x, warnings = warnings): CRS is NA. Assuming it is
## longitude/latitude
```



```
eff_tensor4 = array(0.5,c(nrow,ncol,3))
for (i in 1:length(efficienty3)){
  eff_tensor4[gsc[i,1]+1,gsc[i,2]+1,1]=0.3 #R
  eff_tensor4[gsc[i,1]+1,gsc[i,2]+1,2]=max(0,eff[i]-eff3[i])  #G
  eff_tensor4[gsc[i,1]+1,gsc[i,2]+1,3]=max(eff3[i]-eff[i],0)  #B
}

eff_tensor4 [is.na(map)]=1
eff_tensor4 = round(eff_tensor4*255)
raster_RGB = stack(raster(eff_tensor4[,,1]),raster(eff_tensor4[,,2]),raster(eff_tensor4[,,3]))
plotRGB(flip(raster_RGB))
```

```
## Warning in .couldBeLonLat(x, warnings = warnings): CRS is NA. Assuming it is
## longitude/latitude
```



```
#####
##### Afficher les endroits où on plante
#####
rvb_tensor3 = array(0.5,c(nrow,ncol,3))

for (i in 1:length(choix_ma[,1])){

  if (choix[i,3]!=0){

      loc_colo = as.matrix(cm[[choix[i,3]]])
      ind=indices_choix_ma[i]
      vect_colo = loc_colo[,ind]
      for (j in 1:length(vect_colo)){
        rvb_tensor3[possibilities[j,1],possibilities[j,2],1]=vect_colo[j]+rvb_tensor3[possibilities[j,1
        rvb_tensor3[possibilities[j,1],possibilities[j,2],2]=1 #G
        #rvb_tensor3[possibilities[j,1],possibilities[j,2],3]=0 #B
      }
  }



  #rvb_tensor3[possibilities[ind,1],possibilities[ind,2],1]=loc #R
}
# for (i in 1:length(choix_ma[,1])){
#   rvb_tensor3[choix_ma[i,1],choix_ma[i,2],1]=1 #R
#   rvb_tensor3[choix_ma[i,1],choix_ma[i,2],2]=0 #G
```

```
#   rvb_tensor3[choix_ma[i,1],choix_ma[i,2],3]=0 #B
# }

rvb_tensor3[is.na(map)]=1
rvb_tensor2 = round(rvb_tensor3*255)
raster_RGB = stack(raster(rvb_tensor2[,,1]),raster(rvb_tensor2[,,2]),raster(rvb_tensor2[,,3]))
plotRGB(flip(raster_RGB))
```

```
## Warning in .couldBeLonLat(x, warnings = warnings): CRS is NA. Assuming it is
## longitude/latitude
```

```
points(XY_ma[,2], XY_ma[,1],col="red",pch=8)
```