R Notebook

```
froot <- "~/Desktop/paulien/smsp/1705smsp/out3/"
froot <- "D:/sjh/stringmol/smsp/1705smsp/out3/"
```

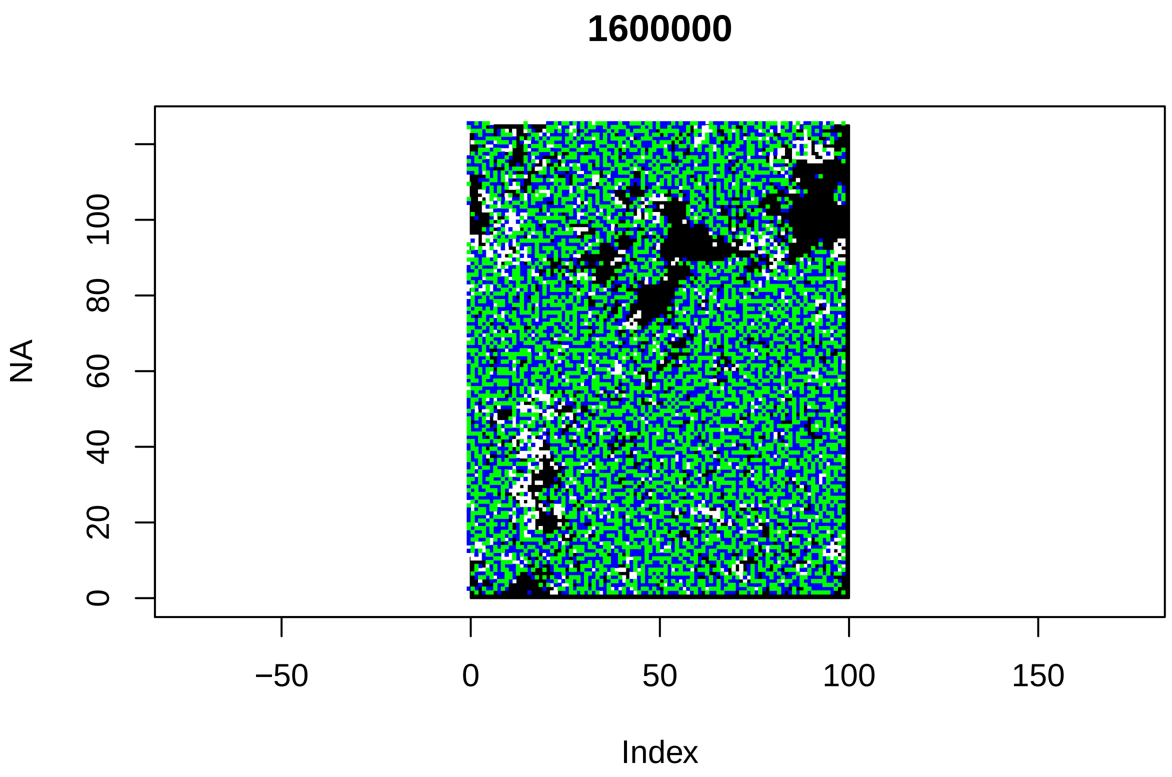
Let's load the final conf file:

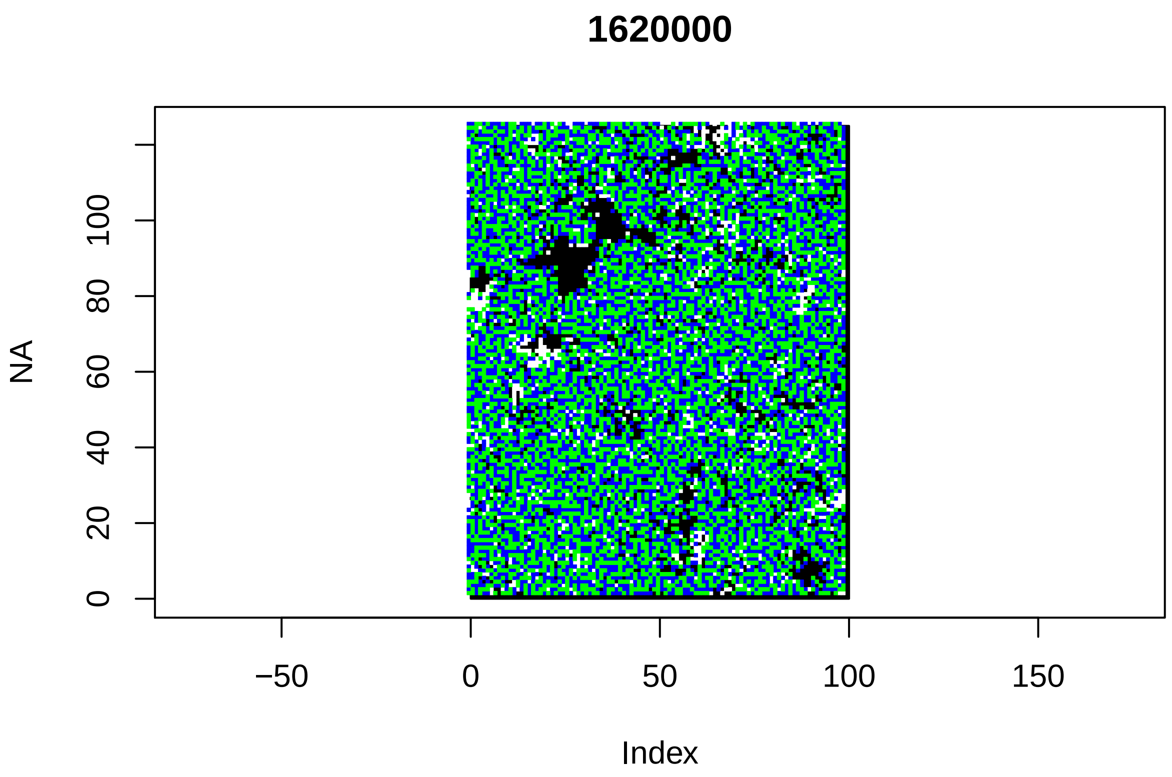
```
  cf <- read.table(sprintf("%sout1_1500000.conf",froot),as.is = T, fill = T, sep=" ",comment.char="",st
```

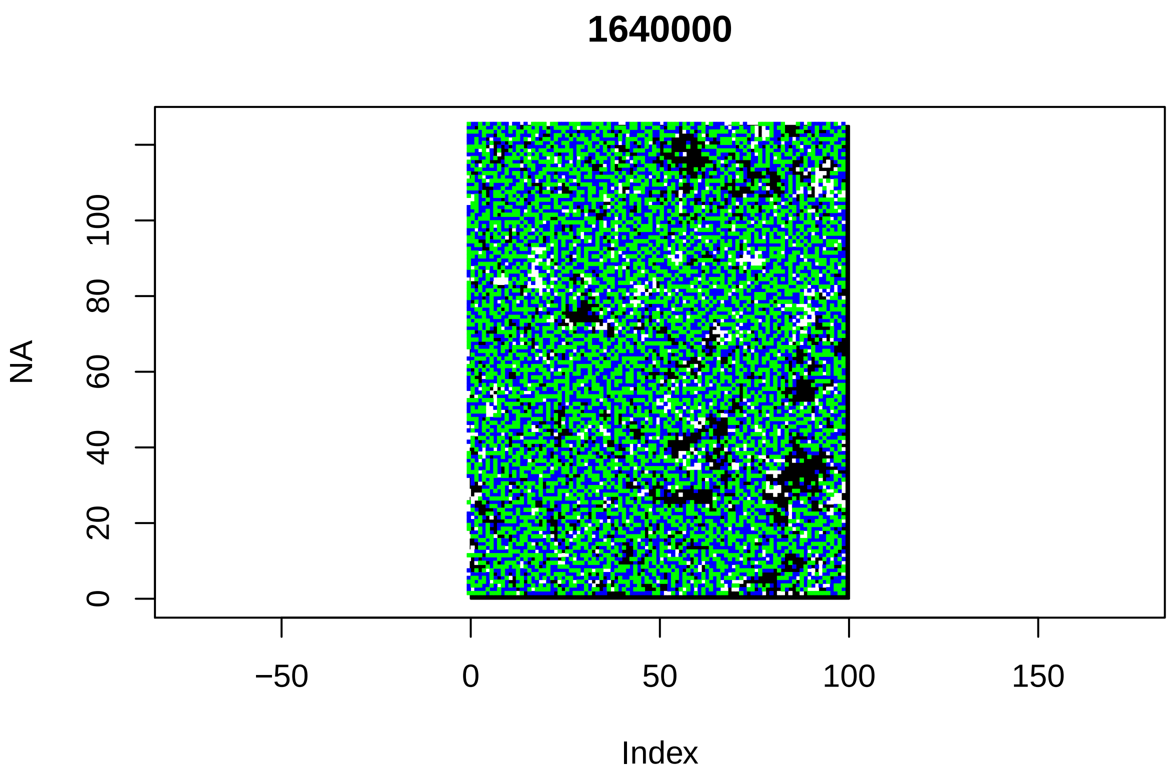
Now we need to find the most numerous spp - let's write a function to do that:

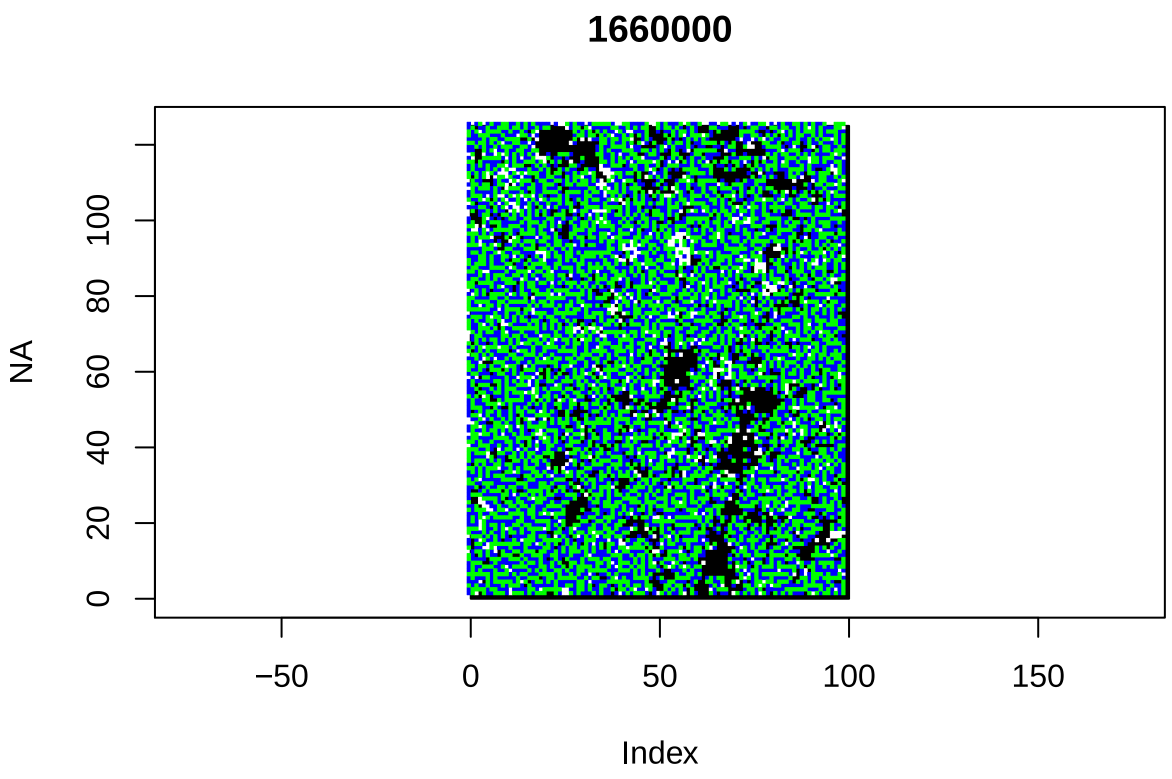
```
get_pos_plot <- function(cf,t,box=NA){

  ub <- cf[cf$V1=="GRIDPOS",]
  ub$V4 <- as.numeric(ub$V4)
  ub$V3 <- as.numeric(ub$V3)

  aa <- cf[cf$V3=="grid:",]
  aa$V4 <- as.numeric(aa$V4)
  aa$V5 <- as.numeric(aa$V5)

  ap <- cf[cf$V4=="grid:",]
  bpy <- cf[(which(cf$V4=="grid:"))+1,]
  ap <- data.frame(ap,yy=bpy$V1,stringsAsFactors = F)
  ap$V5 <- as.numeric(ap$V5)
  ap$yy <- as.numeric(ap$yy)

  plot(NA,xlim=c(0,100),ylim=c(0,125),asp=1,main=sprintf("%d",t))
  rect(xleft = 0,xright = 100,ybottom = 0,ytop = 125,col="black")

  xl <- ub$V4-1
  rect(xleft=xl,xright = ub$V4,   ybottom=125-ub$V3,ytop=125-ub$V3+1,col="white",border = "white",lwd=0)
  rect(xleft=aa$V5-1,xright=aa$V5,ybottom=125-aa$V4,ytop=125-aa$V4+1,col="blue",border="blue",lwd=0)
  rect(xleft=ap$yy-1,xright=ap$yy,ybottom=125-ap$V5,ytop=125-ap$V5+1,col="green",border="green",lwd=0)

  if(!is.na(box)){
    segments(x0=box[1],y0=box[3],y1=box[4],col="red")
    segments(x0=box[2],y0=box[3],y1=box[4],col="red")

    segments(x0=box[1],x1=box[2],y0=box[3],col="red")
    segments(x0=box[1],x1=box[2],y0=box[4],col="red")

    hits <- ub[(ub$V4>box[1] & ub$V4<box[2] &  (125-ub$V3) > box[3] & (125-ub$V3) < box[4]),]
    if(nrow(hits)>0){
    #  message("Found %d hits\n",nrow(hits))
      #for(hh in 1:nrow(hits)){
        #message(sprintf("Found ub at %d %d (y,x) order\n",hits$V3,hits$V4))
        rect(xleft=hits$V4-1,xright = hits$V4,   ybottom=125-hits$V3,ytop=125-hits$V3+1,col="red",borde

      #}
    }
    return(hits)
  }
}
```

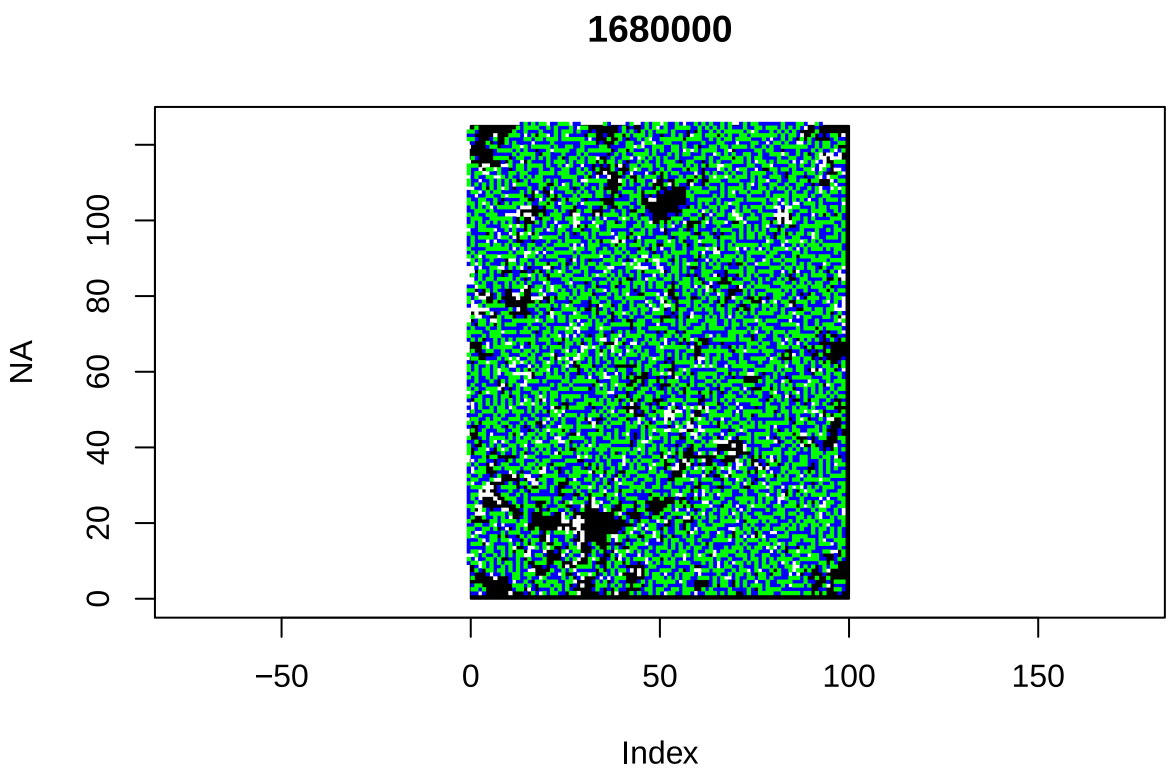
```
for(t in seq(1600000,1700000,20000)){

  cf <- read.table(sprintf("%sout1_%d.conf",froot,t),as.is = T, fill = T, sep=" ",comment.char="",string
get_pos_plot(cf,t)
}
```

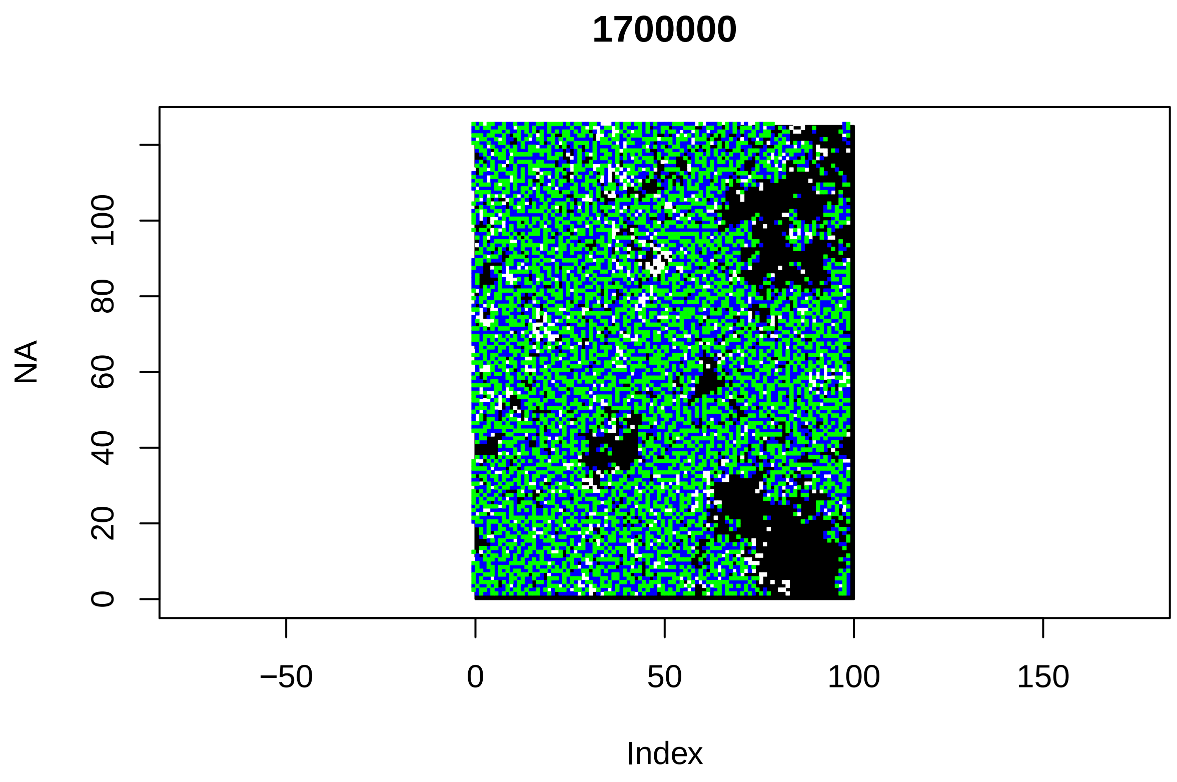



1620000
NA
Index
0
20
40
60
80
100
−50
0
50
100
150


1640000
NA
Index
0
20
40
60
80
100
−50
0
50
100
150


1660000
NA
0
20
40
60
80
100
−50
0
50
100
150
Index

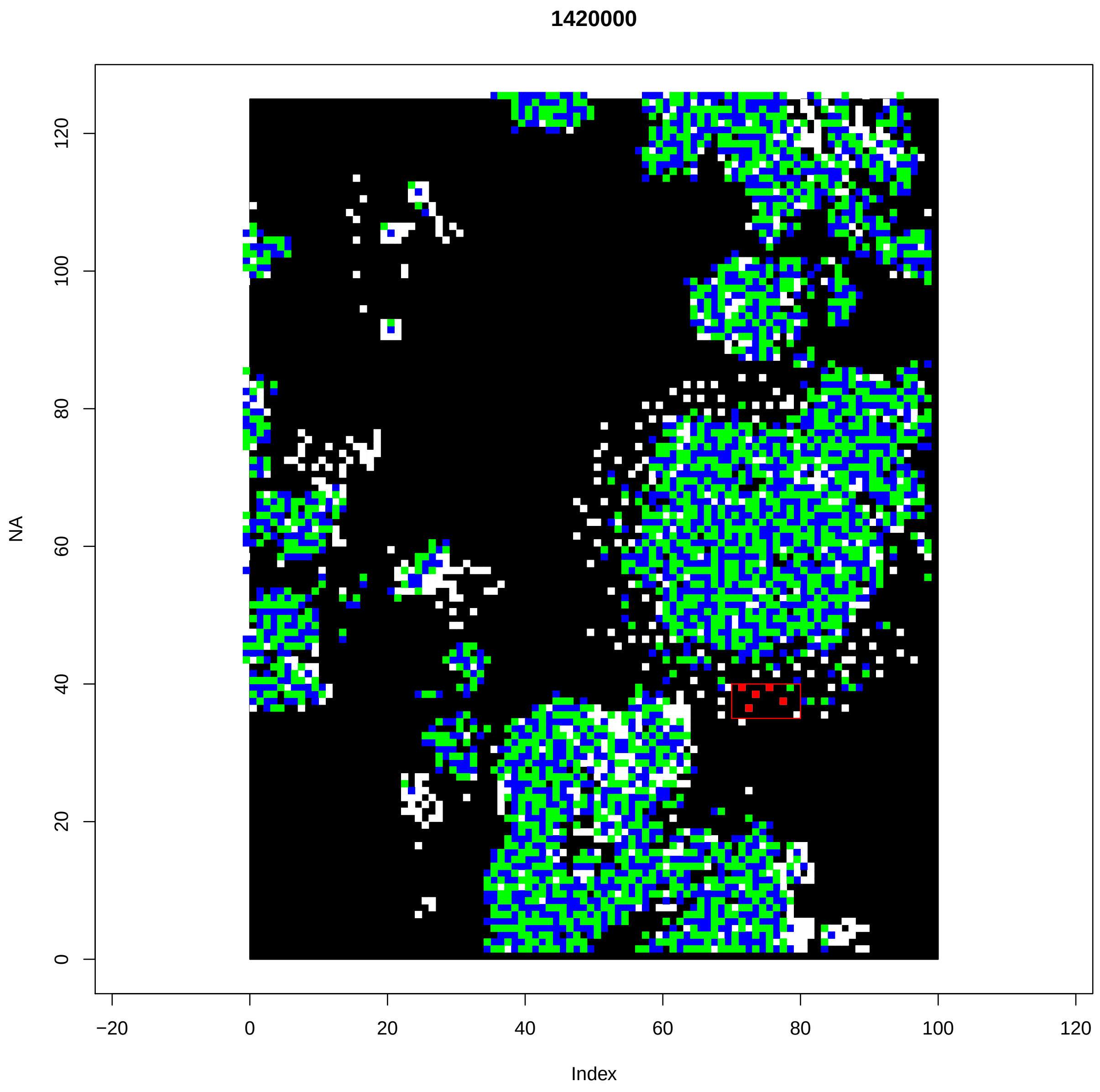

1680000
NA
Index
0
20
40
60
80
100
−50
0
50
100
150



OK, it's looking like t=1540000 has jumpers. We'll extend the gridplot function to isolate a region

```
  cf <- read.table(sprintf("%sout1_%d.conf",froot,1420000),as.is = T, fill = T, sep=" ",comment.char=""
hits <- get_pos_plot(cf,1420000,c(70,80,35,40))
```

```
## Warning in if (!is.na(box)) {: the condition has length > 1 and only the
## first element will be used
```



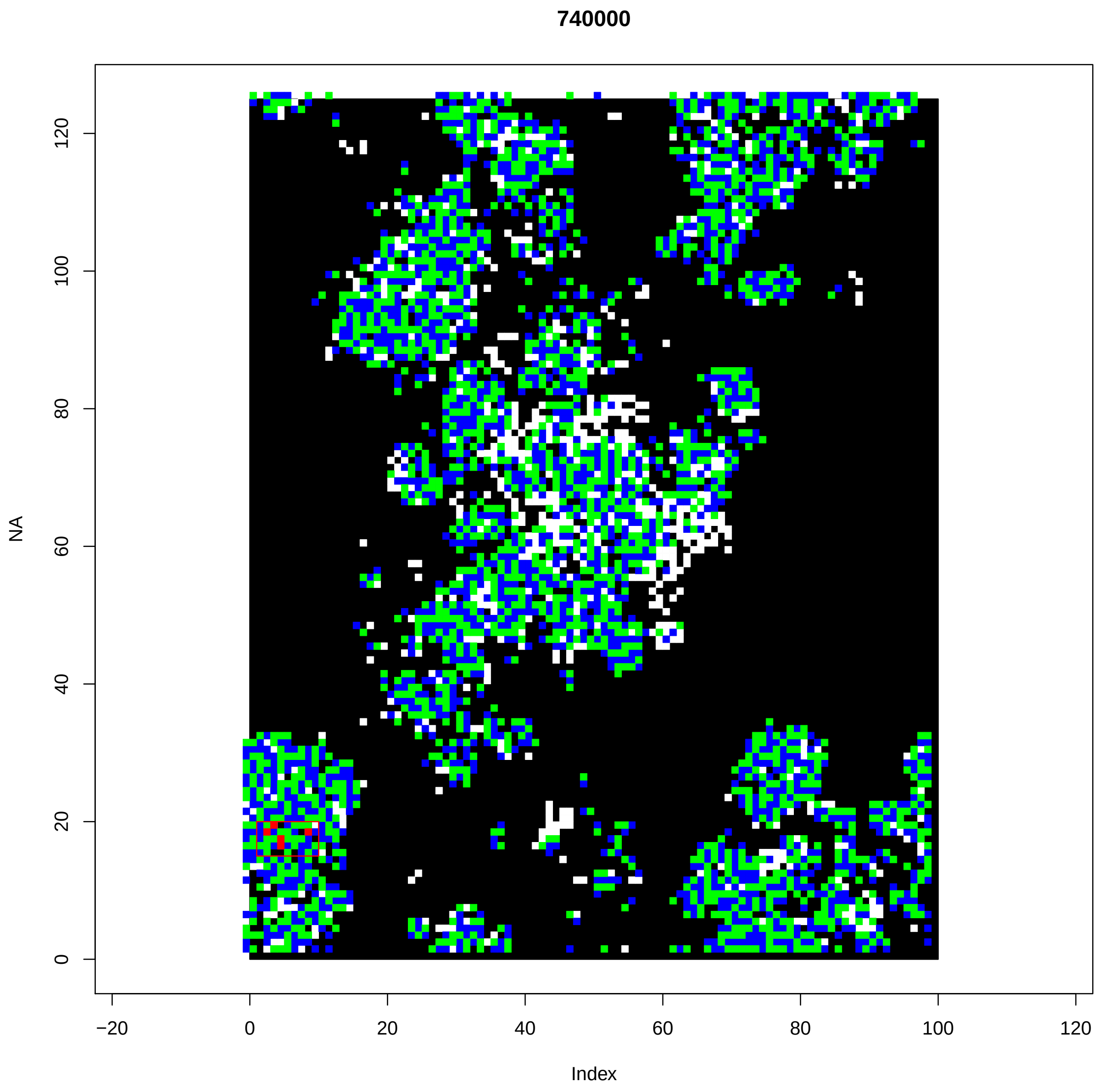
`$=?>CAVBO%}HOB$` is an example of a 'jumper'

'Standard' RP molcules

for comparision, it's worth looking at the R-P moleulces that evolve. Run three has these at t=740,000

```
  cf <- read.table(sprintf("%sout1_%d.conf",froot,740000),as.is = T, fill = T, sep=" ",comment.char="",
hits <- get_pos_plot(cf,740000,c(1,10,15,20))
```

```
## Warning in if (!is.na(box)) {: the condition has length > 1 and only the
## first element will be used
```



R-R reaction takes 171 steps

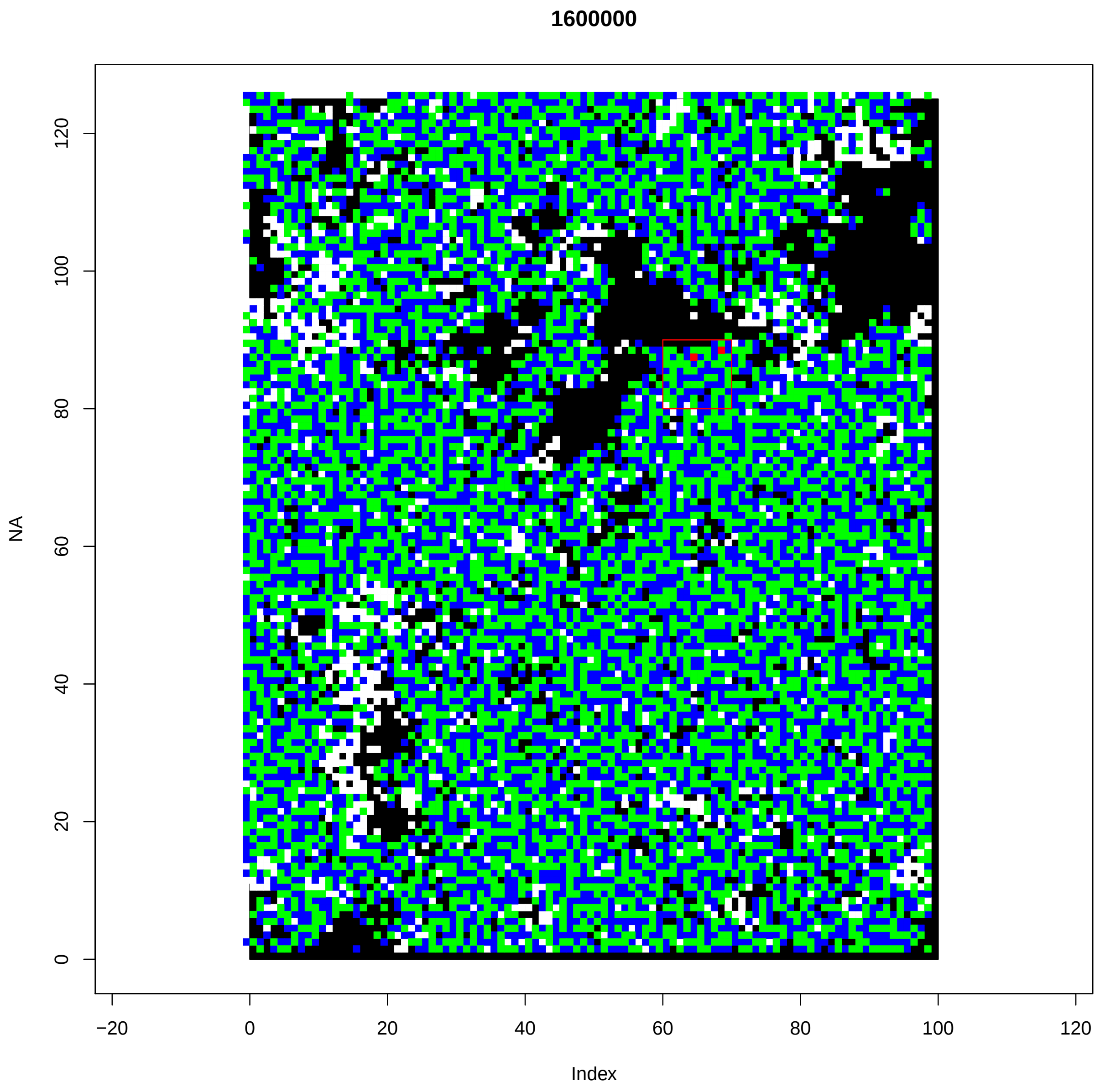
R-P reaction takes 126 steps

‘Flood’ molecules

These occur at t=1,600,000 in run 3.

```
cf <- read.table(sprintf("%sout1_%d.conf",froot,1600000),as.is = T, fill = T, sep=" ",comment.char=""
hits <- get_pos_plot(cf,1600000,c(60,70,80,90))
```

```
## Warning in if (!is.na(box)) {: the condition has length > 1 and only the
## first element will be used
```



`C=?PPC=?NPP^C>B$=K?P>$ALUOGGBC$=P?>F$BLU%}PYH` - 209 t - no product

Diversity wars

*This happens right at the end of the run

Notes from run 1

```
P=?>G^C>B$=?>$A?UO%}PYHHGGP 67,58
P=?>G^C>C$=?>$A?UO%}PYHHGGPC>B$=?V>$AL=OH}PYH 67,63 (splits itself in half after $=?>$)
P=?>G^C>B$=?>$A?UO%}PYHHPPP 67,72
PYHHPPP 68,69
```

```
P=?>G^C>B$=?>$A?UO%}PYHHPPP
C=?PPC=?QP^C>B$=?P>$ALVOHPP
P=?>G^C>B$=?>$A?UO%}PYHHQPPP
P=?>G^C>B$=?>$A?UO%}PYHHPPP
P=?>G^C>B$=?>$A?UO%}PYHHGGP
```

`C=?>G^BC$=P?>$BLU%PYH` and `P=?>G^BC$=P?>$BLU%PYH` form a hypercycle. Identical apart from 1st character, but this difference allows a bind site to form. This one is really efficient because there's nowhere else for a bind to form

`P=?PP^C>B$=?V>$ALUOG}PYHH` binds to `C=?PP^C>B$=?V>$ALUOG}PYHH` and converts it back to `P=?PP^C>B$=?V>$ALUOG}PYHH` by overwriting - there is no cleave