Voluntary assignment

January 9, 2022

1 Enunciate of the exercise

In Lesson #5 we have seen a cellular automata that simulates the process of synchronization of a population of fireflies. When executing the code, most of your classmates found the system quickly reached an equilibrium comprised of two large clusters of individuals, either in the ON or the OFF state. However, some of you spotted as well cases in which a equilibrium is reached that involves just a single, population-wide cluster of synchronized individuals.

Your task in this exercise is to characterize the behaviour of the system as a function of the different parameters (e.g., lattice size, initial frequency of each type of individuals, boundary conditions, influenceability by the neighbours' consensus . . .).

What determines how quickly the system reaches equilibrium?

What does the average cluster size at equilibrium depends mostly on?

Further instructions:

Please label your document following the format: lastname_MSB_opt.pdf. A high-quality answer would include:

1) a clear description of the methods used (scripts as annex).
2) a minumum of 2-3 well-thought figures.
3) a clear answer to the questions.
4) a brief critical discussion of limitations, future prospects or whatever aspects you deem appropriate

2 Code Lesson 5

I documented better the code to understand every part of the code.

[25]:
```
time <- 50 # maximun number of time steps
th <- 0.25 # threshold
sp <- matrix(0,25,25) #spatial population matrix  # lattice size is 25 x 25
spm <- matrix(0,25,25) # spatial population matrix (mirror) to save last time␣
 ↪step

lat <- round(runif(305,1,25)) # latitude
```

```
lon <- round(runif(305,1,25)) # longitude
coords <- cbind(c(lat), c(lon))

sp[coords]<-1 # to introduce the active fireflies
spm<- sp # copy the spatial population matrix

for (t in 1:time){   # t is the time step
  #image(t(sp[25:1,]), col=c('white', 'black'), breaks=c(0, 0.5, 1))  # sp is the matrix of fireflies
  #grid(25, 25, lwd=0.5, col= 'grey', lty=1)
#  print(sum(sp)) // number of active fireflies of the matrix
#  readline(prompt="Press [enter] to continue") I commented out to don't have the cell running until I press enter 50 times
  for (x in 1:25){
    for (y in 1:25){
      sp[x,y]<- 0 + (spm[x,y]==0) # the pacemaker, if the last to time step the firefly is deactivated get activate.
      xup = x+1 #coordinates of the neighbours
      xdw = x-1
      yup = y+1
      ydw = y-1

      if (x==1) {xdw=x} #dealing with edges, represent the boundary conditions
      if (x==25) {xup=x}
      if (y==1) {ydw=y}
      if (y==25) {yup=y}

      #let's calculate neighbours' consensus
      nei <- median(c(spm[xdw,y], spm[xup,y], spm[x,yup], spm[x, ydw], spm[xdw,ydw], spm[xup,yup], spm[xdw,yup], spm[xup,ydw]))
      if (sp[x,y]==1 & runif(1)>th & nei==1){ # 75% probability of get deactivate if the last time step the majority of neighbours were active and the firefly was inactive before.
        sp[x,y]<-0 # the reset rule
      }
    }
  }
  spm<-sp
}
```

3 First criticism to the code

The code of the coordinates is totally wrong. The code work with the coordinates system because is naming the variables latitude and longitude. Longitud is X and latitude is Y.

To define this system in a matrix:

```
y_border = 4
x_border = 4
s <- matrix(0,y_border, x_border)
x = 2
y = 1
#x_left (subtract 1), x_right (add 1), y_up (subtract 1), y_down (add 1)
s[y, x] <- 1
s
```

Ouput:

```
     [,1] [,2] [,3] [,4]
[1,]    0    1    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
```

And does not exist x_up or x_down, because the X go in direction rigth or left.

Maybe right now this code does not have affect the functionality but if you want to play with these parameters later on, this will be a problem.

Moreover, this whole mistake cause me personally a lot of confussion and make me loose a lot of time because I'm dyslexic, so was very difficult for me deal with this mistake.

[24]:
```
time <- 50 # maximun number of time steps
th <- 0.25 # threshold
sp <- matrix(0,25,25) #spatial population matrix  # lattice size is 25 x 25
spm <- matrix(0,25,25) # spatial population matrix (mirror) to save last time␣
 ↪step

lat <- round(runif(305,1,25)) # latitude
lon <- round(runif(305,1,25)) # longitude
coords <- cbind(c(lat), c(lon))

sp[coords]<-1 # to introduce the active fireflies
spm<- sp # copy the spatial population matrix

for (t in 1:time){   # t is the time step
 # image(t(sp[25:1,]), col=c('white', 'black'), breaks=c(0, 0.5, 1))  # sp is␣
 ↪the matrix of fireflies
 # grid(25, 25, lwd=0.5, col= 'grey', lty=1)
#  print(sum(sp)) // number of active fireflies of the matrix
#  readline(prompt="Press [enter] to continue") I commented out to don't have␣
 ↪the cell running until I press enter 50 times
  for (x in 1:25){
    for (y in 1:25){
      sp[x,y]<- 0 + (spm[x,y]==0) # the pacemaker, if the last to time step the␣
 ↪firefly is deactivated get activate.
      x_right = x+1 #coordinates of the neighbours
```

```
      x_left = x-1
      y_down = y+1
      y_up = y-1

      if (x==1) {x_left=x} #dealing with edges, represent the boundary conditions
      if (x==25) {x_right=x}
      if (y==1) {y_up=y}
      if (y==25) {y_down=y}

      #let's calculate neighbours' consensus
      nei <- median(c(spm[x_right,y], spm[x_left,y], spm[x,y_up], spm[x,
 ↪y_down], spm[x_left,y_down], spm[x_right,y_up], spm[x_left,y_up],
 ↪spm[x_right,y_down]))
      if (sp[x,y]==1 & runif(1)>th & nei==1){ # 75% probability of get
 ↪deactivate if the last time step the majority of neighbours were active and
 ↪the firefly was inactive before.
        sp[x,y]<-0 # the reset rule
      }
    }
  }
  spm<-sp
}
```

4 Second criticism to the code

Right now the way of populate the matrix does not allow to control the initial frequencies of active and inactive fireflies. Moreover, how the replacement is allow when getting the number of coordinates randomly, if you have bad luck, the population of the matrix would be really small or too big. Also put the numbers like variable will allow us to play more with the code. I need to fix this problem:

First I checked my code with a little example:

[3]:
```
y_border = 4
x_border = 4
s <- matrix(0,y_border, x_border)
x = 2
y = 1
#x_left (subtract 1), x_right(add 1), y_up (subtract 1), if y == 1, y_up =
 ↪y_border, y_down (add 1) (if y == y_border, y_down = 1)
s[y, x] <- 1
s
```

0	1	0	0
0	0	0	0
0	0	0	0
0	0	0	0

[4]:
```
non_coords <- c(1)
non_coords

lat <- non_coords%/%5 + 1 # y
lat
```

1

1

[5]:
```
lon <- non_coords%%5 + 1 # x
lon
```

2

[6]:
```
matriz <- matrix(0,4,4)
coords <- cbind(c(lat), c(lon))
matriz[coords] <- 1
matriz
```

0	1	0	0
0	0	0	0
0	0	0	0
0	0	0	0

[7]:
```
print(matriz[0,0])
```

```
<0 x 0 matrix>
```

[8]:
```
median(c(matriz[0,0], 1, 1))
```

1

Then I fix the code :

[26]:
```
time <- 2 # maximun number of time steps
th <- 0 # threshold
n <- 4
total <- n * n
sp <- matrix(0,n,n) #spatial population matrix  # lattice size is n*m
spm <- matrix(0,n,n) # spatial population matrix (mirror) to save last time step

initial_density <- 0.1
k <- total - 1
x <- initial_density*total
n_coords <- sample(0:k, x, replace=FALSE)
lat <- n_coords%/%n + 1
lon <- n_coords%%n + 1
coords <- cbind(c(lat), c(lon))
sp[coords]<-1 # to introduce the active fireflies
spm<- sp # copy the spatial population matrix
```

```
for (t in 1:time){   # t is the time step
#  image(t(sp[n:1,]), col=c('white', 'black'), breaks=c(0, 0.5, 1))  # sp is the␣
 ↪matrix of fireflies
#  grid(n, n, lwd=0.5, col= 'grey', lty=1)
#  print(sum(sp)) // number of active fireflies of the matrix
#  readline(prompt="Press [enter] to continue") I commented out to don't have␣
 ↪the cell running until I press enter 50 times
  for (x in 1:n){
    for (y in 1:n){
      sp[x,y]<- 0 + (spm[x,y]==0) # the pacemaker, if the last to time step the␣
 ↪firefly is deactivated get activate.
      x_right = x+1 #coordinates of the neighbours
      x_left = x-1
      y_down = y+1
      y_up = y-1

      if (x==1) {x_left=x} #dealing with edges, represent the boundary conditions
      if (x==n) {x_right=x}
      if (y==1) {y_up=y}
      if (y==n) {y_down=y}

      #let's calculate neighbours' consensus
      nei <- median(c(spm[x_right,y], spm[x_left,y], spm[x,y_up], spm[x,␣
 ↪y_down], spm[x_left,y_down], spm[x_right,y_up], spm[x_left,y_up],␣
 ↪spm[x_right,y_down]))
      if (sp[x,y]==1 & runif(1)>th & nei==1){ # 75% probability of get␣
 ↪deactivate if the last time step the majority of neighbours were active and␣
 ↪the firefly was inactive before.
        sp[x,y]<-0 # the reset rule
      }
    }
  }
  spm<-sp
}
```

5 Definition of a function

If we want to play with so many parameters, we should have a function, in this way will be easier to study the different situations.

[30]:
```
fireflies <- function(time, th, n, m, initial_density){

total <- n * m
sp <- matrix(0,n,m) #spatial population matrix  # lattice size is n*m
spm <- matrix(0,n,m) # spatial population matrix (mirror) to save last time step
```

```
k <- total - 1
x <- initial_density*total
n_coords <- sample(0:k, x, replace=FALSE)
lat <- non_coords%/%5 + 1
lon <- non_coords%%5 + 1

sp[coords]<-1 # to introduce the active fireflies
spm<- sp # copy the spatial population matrix

for (t in 1:time){   # t is the time step
 # image(t(sp[n:1,]), col=c('white', 'black'), breaks=c(0, 0.5, 1))  # sp is the␣
 ↪matrix of fireflies
 # grid(n, m, lwd=0.5, col= 'grey', lty=1)
#  print(sum(sp)) // number of active fireflies of the matrix
#  readline(prompt="Press [enter] to continue") I commented out to don't have␣
 ↪the cell running until I press enter 50 times
  for (x in 1:n){
    for (y in 1:m){
      sp[x,y]<- 0 + (spm[x,y]==0) # the pacemaker, if the last to time step the␣
 ↪firefly is deactivated get activate.
      x_right = x+1 #coordinates of the neighbours
      x_left = x-1
      y_down = y+1
      y_up = y-1

      if (x==1) {x_left=x} #dealing with edges, represent the boundary conditions
      if (x==25) {x_right=x}
      if (y==1) {y_up=y}
      if (y==25) {y_down=y}

      #let's calculate neighbours' consensus
      nei <- median(c(spm[x_right,y], spm[x_left,y], spm[x,y_up], spm[x,␣
 ↪y_down], spm[x_left,y_down], spm[x_right,y_up], spm[x_left,y_up],␣
 ↪spm[x_right,y_down]))
      if (sp[x,y]==1 & runif(1)>th & nei==1){ # 75% probability of get␣
 ↪deactivate if the last time step the majority of neighbours were active and␣
 ↪the firefly was inactive before.
        sp[x,y]<-0 # the reset rule
      }
    }
  }
  spm<-sp
}}
```

Perfect! Now we can play with the number of time steps, the dimensions of the lattice, the initial density of activate fireflies and the threshold.

Still we need to improve the function to be able to play with the boundaries conditions and the neighbour consensus.

Let's try to simulate different situations for the boundary conditions, right now the fireflies of edges interact with themselves, but there are another possibilities like a toroidal world and a flat world.

[11]:
```
# create a function to define boundary conditions
boundaries <- function(x,y,type_of_world,n,bias='None'){
      x_right = x+1 #coordinates of the neighbours
      x_left = x-1
      y_down = y+1
      y_up = y-1


      if (type_of_world=='Flat' ){
        vec_x <- c(x_left, x_left, x_left, x, x, x_right, x_right, x_right)
        vec_y <- c(y_down, y, y_up, y_down, y_up, y_down, y, y_up)
        if (bias=='direct_neighbours'){           # this idea come from the␣
 ↪function consensus, where I realized that apply this type of bias to this type␣
 ↪of world is easier do it here
          vec_x <- c(vec_x, x_left, x, x, x_right)
          vec_y <- c(vec_y, y, y_up, y_down, y)
        }
        boolean_vec <- which(!(vec_x==0|vec_x==(n+1)|vec_y==0|vec_y==(n+1)))
        vec_x <- vec_x[boolean_vec]
        vec_y <- vec_y[boolean_vec]
      }
      if (type_of_world == 'Normal'){
      if (x==1) {x_left=x}
      if (x==n) {x_right=x}
      if (y==1) {y_up=y}
      if (y==n) {y_down=y}
      vec_x <- c(x_left, x_left, x_left, x, x, x_right, x_right, x_right)
      vec_y <- c(y_down, y, y_up, y_down, y_up, y_down, y, y_up)
      }
      if (type_of_world == 'Toroidal'){
      if (x==1) {x_left= n}
      if (x==n) {x_right= 1}
      if (y==1) {y_up= n}
      if (y==n) {y_down= 1}
      vec_x <- c(x_left, x_left, x_left, x, x, x_right, x_right, x_right)
      vec_y <- c(y_down, y, y_up, y_down, y_up, y_down, y, y_up)
      }
  vec <- cbind(vec_y, vec_x)
}
```

I also want play with the neighbours's consensus, and for that we can define another function.

[10]:
```
consensus <- function(spm, vec, type_of_world, bias='None'){
  vec_without_bias <- spm[vec]
  vec_biased <- vec_without_bias
  if (bias=='inactive'){vec_biased <- append(vec_biased, 1)} # bias against␣
 ↪deactive cells
  if (bias=='active'){vec_biased <- append(vec_biased, 0)} # bias against active␣
 ↪cells
  if (bias=='direct_neighbours'){ # repeat the direct neighbours in the vector␣
 ↪so then they have more influence than the neighbours of the diagonal
  if (type_of_world != 'Flat'){
    direct_nei <- c(vec_without_bias[2], vec_without_bias[4],␣
 ↪vec_without_bias[5], vec_without_bias[7]) # we should repeat the direct␣
 ↪neighbours which are the 2nd, 4th, 5th, 8th elements of the original vector
    vec_biased <- c(vec_biased, direct_nei)}}
  if (bias=='bad_neighbours'){
    random_bad <- sample(1:len(vec), 3, replace=FALSE) # I decided and it is␣
 ↪totally arbitrary define only 3 bad neighbours (neighbours that do not␣
 ↪participate in the consensus) from the usually vector of 8 neighbours
    vec_biased <- vec_biased[-random_bad]
  }
  else {vec_biased}
  vec_biased
}
```

Now, I will put it into the function and I want also define a way to return the time step when the system reach the equilibrium. Also it is a smart idea return the matrix so I can calculate different variables that will be interesting to study.

[18]:
```
fireflies <- function(time, th, n, initial_density, type_of_world, bias){

total <- n * n
sp <- matrix(0,n,n) #spatial population matrix  # lattice size is n*n
spm <- matrix(0,n,n) # spatial population matrix (mirror) to save last time step
comparison = FALSE
equilibrium_timestep = FALSE
k <- total - 1
x <- initial_density*total
n_coords <- sample(0:k, x, replace=FALSE)
lat <- n_coords%/%n + 1
lon <- n_coords%%n + 1
coords <- cbind(c(lat),c(lon))
sp[coords]<-1 # to introduce the active fireflies
spm<- sp # copy the spatial population matrix, to retain the last time step
sp_timestep_2 <- spm # have a third matrix that retains the last second time␣
 ↪step,
# so we can compared with the present one and see if the is a periodic state␣
 ↪which means we reached the equilibrium
```

```
matrix_list <- list()
for (t in 1:time){   # t is the time step
# image(t(sp[n:1,]), col=c('white', 'black'), breaks=c(0, 0.5, 1))  # sp is the␣
 ↪matrix of fireflies
#  grid(n, n, lwd=0.5, col= 'grey', lty=1)
# Was really annoying plot all the time so I comment it out
#  print(sum(sp)) // number of active fireflies of the matrix
#  readline(prompt="Press [enter] to continue") I commented out to don't have␣
 ↪the cell running until I press enter 50 times
  for (x in 1:n){
    for (y in 1:n){
      sp[y,x]<- 0 + (spm[y,x]==0) # the pacemaker, if the last to time step the␣
 ↪firefly is deactivated get activate.
      vec <- boundaries(x,y, type_of_world, n, bias='None') # using this␣
 ↪function we can change the boundaries conditions

      #let's calculate neighbours' consensus
      v <- consensus(spm, vec, type_of_world, bias='None') # using this function␣
 ↪we can change the neighbour's consensus
      nei <- median(v)
      if (sp[y,x]==1 & runif(1)>th & nei==1){ # 75% probability of get␣
 ↪deactivate if the last time step the majority of neighbours were active and␣
 ↪the firefly was inactive before.
        sp[y,x]<-0 # the reset rule
      }
    }
  }
  matrix_list[[t]] <- sp
  if (comparison != TRUE){
     comparison <- all.equal(sp_timestep_2 , sp)
     if (comparison == TRUE){
      equilibrium_timestep <- t
      print(equilibrium_timestep)
     }
  }
 # comparison <- all.equal(sp_timestep_2 , sp)
  sp_timestep_2<-spm
  spm<-sp
 # comparison <- all.equal(sp_timestep_2 , sp)
  print(comparison)
  #if (comparison== TRUE & return_flag==FALSE){
  #  return_flag = TRUE
  #  flag <- list(matrix_list, t)
  #  return(flag) } # to return the when reach the equilibrium and the state of␣
 ↪the matrix in the equilibrium
    # this will make possible to answer the questions of the assignment
```

```
}
return (list(matrix_list, equilibrium_timestep))
}
```

Things I can study in plots:

1. Density in each time step (I can **calculate the density as the number of cell actives divided by the total number of cells**) for different initial conditions (initial frequency of individuals and maeby lattice size and time)
2. Activated zone / initial activated zone (**number of fireflies blinking or cells active**) in each time step also different initial conditions
3. I have the intuition that if there are a big amount of fireflies and enough time, the initial number of fireflies blinking or cells actives will not matter at all because the density and the active zone versus time steps will remain equal, so plot activated zone x density over time steps for different initial conditions
4. Velocity (**How quickly spread the blinking signal of the firefly but how to calculate this? I think a lot about this and I think one option to do this is consider the activated zone as a circle like pi*r^2, so r is the distance from the center and knowing that velocity is x/t, we can define velocity as square root of the activated zone divided by square root of pi, divided by time (2, difference between time steps)**) depending in the initial conditions.
5. Activity depending in the consensus rules and/or the th (whatever is that) maintaining the initial conditions even.
6. Frequency d**efined as number of blinks of the same firefly** over time chaging the consensus rules and/or th.
7. Velocity depending in the consensus rules and/or the th maintaining the initial conditions even.
8. As an idea more original add fireflies without interaction to see how this affect the density, activity and velocity of spreading. **This will be an option of my main function, to define shy fireflies that independently of being active or inactive, will be hidden from the others.**

And of course I should also answer with some plots the following questions :

What determines how quickly the system reaches equilibrium?

What does the average cluster size at equilibrium depends mostly on?

To do all this I should, update and improve still my functions.

[9]:
```
# Improve boundaries condition
boundaries <- function(x,y,type_of_world,n, shy_fireflies, bias){ # shy␣
 ↪fireflies is the matrix of 1 (normal) and 0 (hiding)
      x_right = x+1 #coordinates of the neighbours
      x_left = x-1
      y_down = y+1
      y_up = y-1
```

```
    if (type_of_world=='Flat' ){
      vec_x <- c(x_left, x_left, x_left, x, x, x_right, x_right, x_right)
      vec_y <- c(y_down, y, y_up, y_down, y_up, y_down, y, y_up)
      if (bias=='direct_neighbours'){
        vec_x <- c(vec_x, x_left, x, x, x_right)
        vec_y <- c(vec_y, y, y_up, y_down, y)
      }
      boolean_vec <- which(!(vec_x==0|vec_x==(n+1)|vec_y==0|vec_y==(n+1)))
      vec_x <- vec_x[boolean_vec]
      vec_y <- vec_y[boolean_vec]
    }
    if (type_of_world == 'Normal'){
    if (x==1) {x_left=x}
    if (x==n) {x_right=x}
    if (y==1) {y_up=y}
    if (y==n) {y_down=y}
    vec_x <- c(x_left, x_left, x_left, x, x, x_right, x_right, x_right)
    vec_y <- c(y_down, y, y_up, y_down, y_up, y_down, y, y_up)
    }
    if (type_of_world == 'Toroidal'){
    if (x==1) {x_left= n}
    if (x==n) {x_right= 1}
    if (y==1) {y_up= n}
    if (y==n) {y_down= 1}
    vec_x <- c(x_left, x_left, x_left, x, x, x_right, x_right, x_right)
    vec_y <- c(y_down, y, y_up, y_down, y_up, y_down, y, y_up)
    }


  vec <- cbind(vec_y, vec_x)
  shy <- shy_fireflies[vec]

  boolean_shy <- which(!(shy==0))
  vec_x <- vec_x[boolean_shy]
  vec_y <- vec_y[boolean_shy]
  vec <- cbind(vec_y, vec_x)
  return(vec)
}
```

[16]:
```
# create shy fireflies matrix
# should be a matrix of 1 and 0, 1 meaning normal fireflie, 0 meaning hidden␣
 ↪fireflie
# if we dont want to apply this modification, the matrix will be completely of 1.
# this cell is an example to see how to do it, later on, I will add this to the␣
 ↪function
```

```
shy_fireflies = matrix(1,4,4)
total_shy <- 4 * 4
k_shy <- total_shy - 1
initial_density_shy <- 0.99
x_shy <- initial_density_shy*total_shy
n_coords_shy <- sample(0:k_shy, x_shy, replace=FALSE)

lat_shy <- n_coords_shy%/%4 + 1
lon_shy  <- n_coords_shy%%4 + 1
coords_shy <- cbind(c(lat_shy),c(lon_shy))

shy_fireflies[coords_shy] <- 0
c <- boundaries(2,2, 'Normal', 4, shy_fireflies, bias = 'None')
c # big problem, sometimes there is not neighbours at all, so I need to change␣
 ↪consensus function too
vector.is.empty <- function(x) return(length(x) ==0 ) #helper function
vector.is.empty(c)
```

vec_y	vec_x

TRUE

[8]:
```
consensus <- function(spm, vec, type_of_world, bias='None'){
  vector.is.empty <- function(x) return(length(x) ==0) #helper function

  vec_without_bias <- spm[vec]
  vec_biased <- vec_without_bias
  if (vector.is.empty(vec)==TRUE){vec_biased=c(0.5)} # solution if there is not␣
 ↪neighbours because they are shy
  if (bias=='inactive'){vec_biased <- append(vec_biased, 1)} # bias against␣
 ↪deactive cells
  if (bias=='active'){vec_biased <- append(vec_biased, 0)} # bias against active␣
 ↪cells
  if (bias=='direct_neighbours'){ # repeat the direct neighbours in the vector␣
 ↪so then they have more influence than the neighbours of the diagonal
  if (type_of_world != 'Flat'){
    direct_nei <- c(vec_without_bias[2], vec_without_bias[4],␣
 ↪vec_without_bias[5], vec_without_bias[7]) # we should repeat the direct␣
 ↪neighbours which are the 2nd, 4th, 5th, 8th elements of the original vector
    vec_biased <- c(vec_biased, direct_nei)}}
  if (bias=='bad_neighbours'){
    random_bad <- sample(1:length(vec), 3, replace=FALSE) # I decided and it is␣
 ↪totally arbitrary define only 3 bad neighbours (neighbours that do not␣
 ↪participate in the consensus) from the usually vector of 8 neighbours
    vec_biased <- vec_biased[-random_bad]
  }
  else {vec_biased}
```

```
  vec_biased
}
```

[19]:
```
fireflies <- function(time, th, n, initial_density, shy_density, type_of_world,␣
 ↪bias){

total <- n * n
sp <- matrix(0,n,n) #spatial population matrix  # lattice size is n*n
spm <- matrix(0,n,n) # spatial population matrix (mirror) to save last time step
shy_fireflies <- matrix(1,n,n) # this is the matrix of shy fireflies
comparison = FALSE
equilibrium_timestep = FALSE
k <- total - 1
x <- initial_density*total
n_coords <- sample(0:k, x, replace=FALSE)
lat <- n_coords%/%n + 1
lon <- n_coords%%n + 1
coords <- cbind(c(lat),c(lon))
sp[coords]<-1 # to introduce the active fireflies

s <- shy_density*total
n_coords_shy <-  sample(0:k, s, replace=FALSE)
lat_shy <- n_coords_shy%/%n + 1
lon_shy <- n_coords_shy%%n + 1
coords_shy <- cbind(c(lat_shy),c(lon_shy))
shy_fireflies[coords_shy] <- 0 # to introduce the shy fireflies

spm<- sp # copy the spatial population matrix, to retain the last time step
sp_timestep_2 <- spm # have a third matrix that retains the last second time␣
 ↪step,
# so we can compared with the present one and see if the is a periodic state␣
 ↪which means we reached the equilibrium
matrix_list <- list(sp)
for (t in 1:time){   # t is the time step
# image(t(sp[n:1,]), col=c('white', 'black'), breaks=c(0, 0.5, 1))  # sp is the␣
 ↪matrix of fireflies
#  grid(n, n, lwd=0.5, col= 'grey', lty=1)
# Was really annoying plot all the time so I comment it out
#  print(sum(sp)) // number of active fireflies of the matrix
#  readline(prompt="Press [enter] to continue") I commented out to don't have␣
 ↪the cell running until I press enter 50 times
  for (x in 1:n){
    for (y in 1:n){
      sp[y,x]<- 0 + (spm[y,x]==0) # the pacemaker, if the last to time step the␣
 ↪firefly is deactivated get activate.
      vec <- boundaries(x,y, type_of_world, n, shy_fireflies, bias) # using this␣
 ↪function we can change the boundaries conditions
```

```
      #let's calculate neighbours' consensus
      v <- consensus(spm, vec, type_of_world, bias) # using this function we can␣
↪change the neighbour's consensus
      nei <- median(v)
      if (sp[y,x]==1 & runif(1)>th & nei==1){ # 75% probability of get␣
↪deactivate if the last time step the majority of neighbours were active and␣
↪the firefly was inactive before.
        sp[y,x]<-0 # the reset rule
      }
    }
  }
  matrix_list[[t+1]] <- sp
  if (comparison != TRUE){
     comparison <- all.equal(sp_timestep_2 , sp)
     if (comparison == TRUE){
      equilibrium_timestep <- t
      #print(equilibrium_timestep)
     }
  }
 # comparison <- all.equal(sp_timestep_2 , sp)
  sp_timestep_2<-spm
  spm<-sp
  #print(comparison)
  #if (comparison== TRUE ){ # only use this code if I want the function stop␣
↪when reaches the equilibrium, this save time
  #  flag <- list(matrix_list, t)
  #  return(flag) } # to return the when reach the equilibrium and the state of␣
↪the matrix in the equilibrium
    # but for doing better plots and more beautiful is better use the one is not␣
↪commented

}
return (list(matrix_list, equilibrium_timestep))
}
```

[19]:
```
total <- 20
initial_density <- 0.3
k <- total - 1
x <- initial_density*total
n_coords <- sample(0:k, x, replace=FALSE)
n_coords
```

1. 16 2. 12 3. 13 4. 7 5. 14 6. 0

[20]:
```
#prove that works
time <- 50 # maximun number of time steps
```

```
th <- 0.25 # threshold
n <- 25
initial_density <- 0.5
type_of_world <- 'Normal'
shy_density <- 0.1

b <- fireflies(time, th, n, initial_density, shy_density , type_of_world)
length(b[[1]])
```

51

Very cool! I'm done with this and I like my function, now I going to define some helper functions, to work with the list of matrix returned and calculate:

- Density
- Active zone
- Velocity
- Frequency

[21]:
```
install.packages("wvtool")
```

```
package 'wvtool' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\yaiza\AppData\Local\Temp\RtmpKGDhAs\downloaded_packages
```

[22]:
```
c(0:25)*2 + 1
```

1. 1 2. 3 3. 5 4. 7 5. 9 6. 11 7. 13 8. 15 9. 17 10. 19 11. 21 12. 23 13. 25 14. 27 15. 29 16. 31 17. 33 18. 35 19. 37 20. 39 21. 41 22. 43 23. 45 24. 47 25. 49 26. 51

[23]:
```
a <- list(c(0:50))
m <- (length(a[[1]])-1)%/%2
t <- c(0:m)*2 + 1
a[[1]][t]
```

1. 0 2. 2 3. 4 4. 6 5. 8 6. 10 7. 12 8. 14 9. 16 10. 18 11. 20 12. 22 13. 24 14. 26 15. 28 16. 30 17. 32 18. 34 19. 36 20. 38 21. 40 22. 42 23. 44 24. 46 25. 48 26. 50

[4]:
```
library('wvtool')
# get every second element of a list

second <- function(random_list){
  m <- (length(random_list)-1)%/%2
  t <- c(0:m)*2 + 1
  b <- random_list[t]
  return (b)
}
```

```
# To calculate active zone I need to calculate the sum of every matrix in every␣
 ↪time step

active_zone <- function(x){
  # x is the output of the main function, a list of matrix and the time in which␣
 ↪reach the equilibrium
  active_by_step <- c()
  list_matrix <- second(x[[1]])
  for (matriz in list_matrix){
    active_by_step <- append(active_by_step, sum(matriz))}
  return(active_by_step)
}

# To calculate density I need to calculate the sum of every matrix and
# divide by the amount total of cells in every time step

density <- function(y){
  # y is the output of the main function, a list of matrix and the time in which␣
 ↪reach the equilibrium
  density_by_step <- c()
  list_matrix <- second(y[[1]])
  for (matriz in list_matrix){
    density_by_step <- append(density_by_step, (sum(matriz)/
 ↪(dim(matriz)[[1]])^2))
  }
  return(density_by_step)
}

# To calculate the frequency we need to sum up every matrix and
# divide by the number of time steps

frequency <- function(z){
   # z is the output of the main function, a list of matrix and the time in␣
 ↪which reach the equilibrium
  list_matrix <- z[[1]]
  f <- Reduce("+", list_matrix)/length(list_matrix)
  freq <- as.vector(f)
  return (freq)
}

# Calculate velocity it is maeby the most difficult.
# active_zone = circle; active_zone = pi*r^2; velocity = x/t, r = x
# to obtain r: r = square root of active_zone/pi; so velocity = square root of␣
 ↪active_zone/pi / time

spread_velocity <- function(w){
```

```
  # w is the output of the main function, a list of matrix and the time in which␣
 ↪reach the equilibrium
  circle<- active_zone(w)
  spread<- sqrt(circle/pi)

  return (spread)
}

# The last helper function will be useful to study the equilibrium

equilibrium_time <- function(k){
  # k is the output of the main function, a list of matrix and the time in which␣
 ↪reach the equilibrium
  time_equilibrium_reached <- k[[2]]
  return (time_equilibrium_reached)
}

equilibrium_matrix <- function(m){
  # m is the output of the main function, a list of matrix and the time in which␣
 ↪reach the equilibrium
  t <- equilibrium_time(m)
  list_matriz <- m[[1]]
  equilibrium_m <- list_matriz[[t]]
  return (equilibrium_m)
}

# A helper function to calculate the average cluster sizes

average_cluster_equilibrium <- function(m){
   # m is the output of the main function, a list of matrix and the time in␣
 ↪which reach the equilibrium
  list_output <- cc.label(equilibrium_matrix(m), connect=8, inv=FALSE, img.
 ↪show=FALSE,text.size=0.3)
  # need the library ("wvtool")
  summary <- list_output[[2]]
  area_cluster <- summary$area
  average_cluster_size <- mean(area_cluster)
  return (average_cluster_size)
}
```

```
Warning message:
"package 'wvtool' was built under R version 3.6.3"
```

[25]:
```
length(density(b))
```

26

6 Plots to study the model

The standard parameters of the experiment will be the set up of the Lesson 5:

- Lattice size: 25x25
- Type of world: Normal
- Threshold: 25%
- Shy density: 0%
- Time 50
- Initial density 50%
- Bias = NONE

I will study the parameters, looking to different set ups by changing one parameter at the time.

To obtain a range of possibles values, each experiment will run 5 times.

[15]:
```
# Let's define a function to run each experiment as many times as needed

run_experiment <- function(number, time, th, n, initial_density, shy_density,␣
 ↪type_of_world, bias){
    result_list <- list()
    for (i in 1:number){
       result <- fireflies(time, th, n, initial_density, shy_density ,␣
 ↪type_of_world, bias)
        result_list[[i]] <- result
        }

    return(result_list)
}
```

[12]:
```
# Let's define another function to get the dataframe of results we need to plot

data_experiment <- function(result_list, parameter){
    if (parameter=='density'){
        run_1 <- density(result_list[[1]])
        run_2 <- density(result_list[[2]])
        run_3 <- density(result_list[[3]])
        run_4 <- density(result_list[[4]])
        run_5 <- density(result_list[[5]])
        }
    if (parameter=='active_zone'){
        run_1 <- active_zone(result_list[[1]])
        run_2 <- active_zone(result_list[[2]])
        run_3 <- active_zone(result_list[[3]])
        run_4 <- active_zone(result_list[[4]])
        run_5 <- active_zone(result_list[[5]])
        }
    if (parameter=='frequency'){
        run_1 <- frequency(result_list[[1]])
```

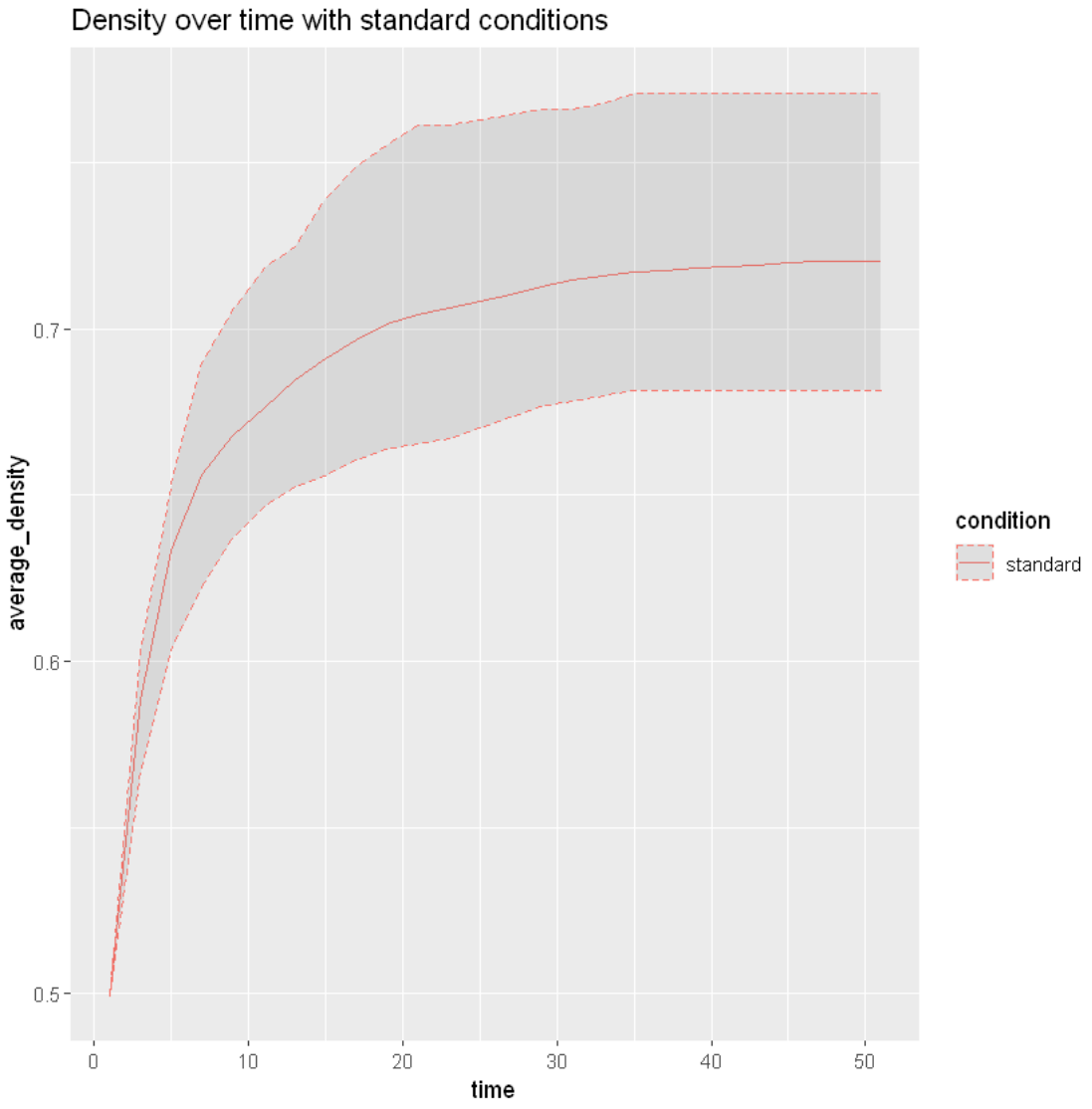
```
        run_2 <- frequency(result_list[[2]])
        run_3 <- frequency(result_list[[3]])
        run_4 <- frequency(result_list[[4]])
        run_5 <- frequency(result_list[[5]])
        }
    if (parameter=='equilibrium_time'){
        run_1 <- equilibrium_time(result_list[[1]])
        run_2 <- equilibrium_time(result_list[[2]])
        run_3 <- equilibrium_time(result_list[[3]])
        run_4 <- equilibrium_time(result_list[[4]])
        run_5 <- equilibrium_time(result_list[[5]])
        }
    if (parameter=='average_cluster_equilibrium'){
        run_1 <- average_cluster_equilibrium(result_list[[1]])
        run_2 <- average_cluster_equilibrium(result_list[[2]])
        run_3 <- average_cluster_equilibrium(result_list[[3]])
        run_4 <- average_cluster_equilibrium(result_list[[4]])
        run_5 <- average_cluster_equilibrium(result_list[[5]])
        }
 data <- data.frame(run_1, run_2, run_3, run_4, run_5)
return (data)
}
```

I needed to discard the velocity of spread because that not works in this type of simulation.

[28]:
```
# standard conditions

number = 5
time = 50
th = 0.25
n = 25
shy_density = 0
type_of_world = 'Normal'
initial_density = 0.5

d <- run_experiment(number, time, th, n, initial_density, shy_density ,␣
 ↪type_of_world)
active_zones <- data_experiment(d, 'active_zone')
frequency <- data_experiment(d, 'frequency')
equilibrium_times <- data_experiment(d, 'equilibrium_time')
densities <- data_experiment(d, 'density')
eq_cluster <- data_experiment(d, 'average_cluster_equilibrium')
```

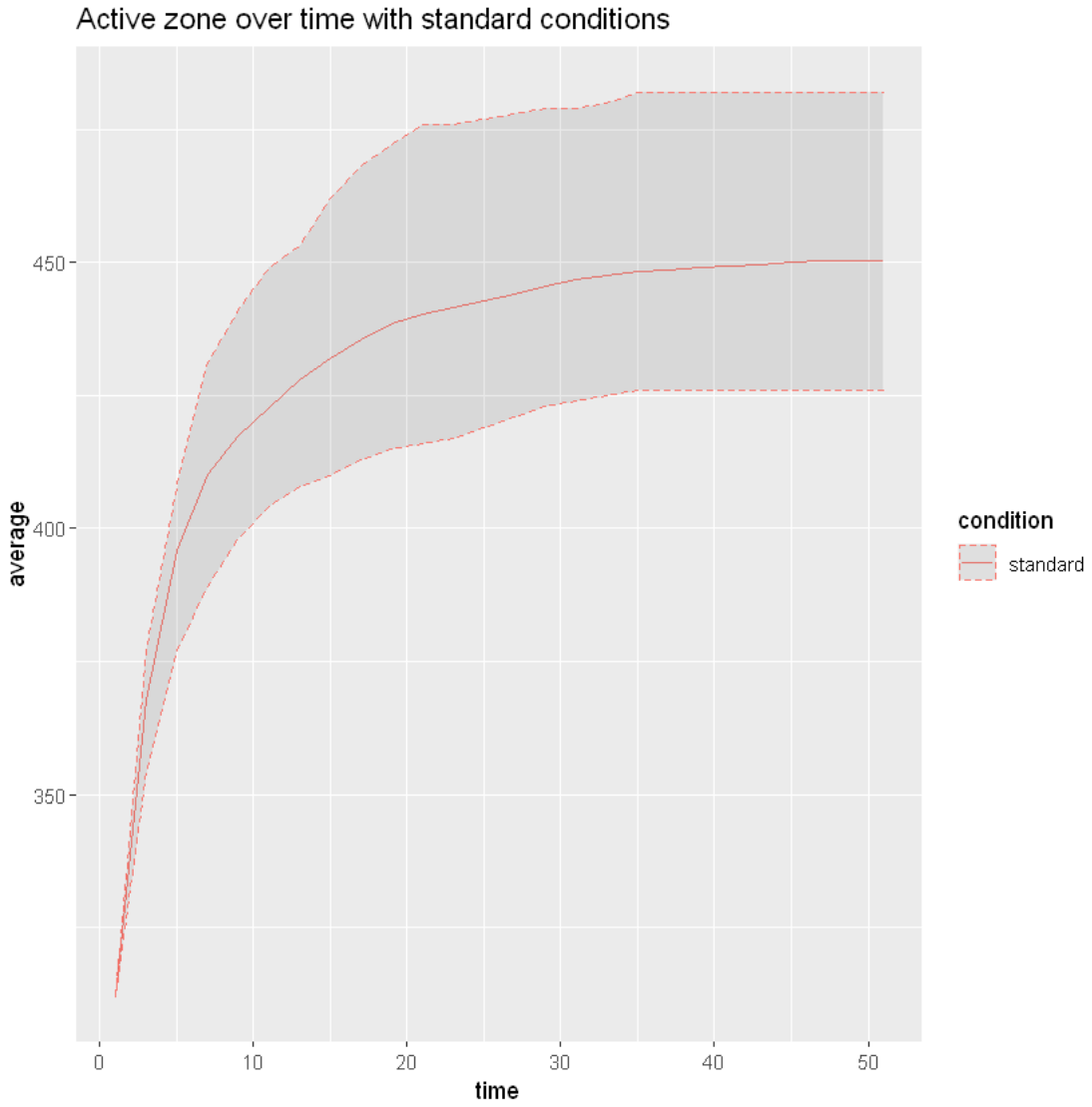
[29]:
```
# study the density
time <- 50
densities$average_density <- rowMeans(densities)
densities$min <- apply(densities, 1, FUN=min, na.rm=TRUE)
```

```
densities$max <- apply(densities, 1, FUN=max, na.rm=TRUE)
densities$time <- c(0:((time)%/%2))*2 + 1
densities$condition <- 'standard' # put labels
library('ggplot2')
ggplot(data=densities, aes(x=time, y=average_density, colour=condition))+␣
↪geom_line() + geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+␣
↪ggtitle("Density over time with standard conditions")
```



[30]:
```
# study the active_zone
time <- 50
active_zones$average<- rowMeans(active_zones)
active_zones$min <- apply(active_zones, 1, FUN=min, na.rm=TRUE)
active_zones$max <- apply(active_zones, 1, FUN=max, na.rm=TRUE)
```

```
active_zones$time <- c(0:((time)%/%2))*2 + 1
active_zones$condition <- 'standard' # put labels
library('ggplot2')
ggplot(data=active_zones, aes(x=time, y=average, colour=condition))+ geom_line()␣
↪+ geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+ ggtitle("Active␣
↪zone over time with standard conditions")
```



[31]:
```
# study the frequency with standard conditions
frequency
```

run_1	run_2	run_3	run_4	run_5
0.5098039	0.5098039	0.5098039	0.5098039	0.4901961
0.5098039	0.5098039	0.4901961	0.5098039	0.4901961
0.5098039	0.4901961	0.5098039	0.5098039	0.4901961
0.4901961	0.5098039	0.5098039	0.4901961	0.4901961
0.4901961	0.4901961	0.5098039	0.4901961	0.4901961
0.5098039	0.4901961	0.5098039	0.4901961	0.4901961
0.5098039	0.4901961	0.4901961	0.4901961	0.5098039
0.5098039	0.4901961	0.5098039	0.4901961	0.4901961
0.5098039	0.4901961	0.4901961	0.4901961	0.5098039
0.4901961	0.5098039	0.4901961	0.4901961	0.5098039
0.5098039	0.4901961	0.4901961	0.4901961	0.5098039
0.4901961	0.5098039	0.5098039	0.5098039	0.5098039
0.4901961	0.5098039	0.5098039	0.4901961	0.5098039
0.5098039	0.4901961	0.5098039	0.5098039	0.4901961
0.4901961	0.4901961	0.5098039	0.5098039	0.4901961
0.5098039	0.5098039	0.4901961	0.5098039	0.4901961
0.5098039	0.4901961	0.4901961	0.5098039	0.4901961
0.4901961	0.5098039	0.4901961	0.4901961	0.5098039
0.4901961	0.5098039	0.5098039	0.4901961	0.4901961
0.4901961	0.4901961	0.5098039	0.5098039	0.5098039
0.5098039	0.5098039	0.4901961	0.5098039	0.4901961
0.5098039	0.5098039	0.4901961	0.4901961	0.4901961
0.5098039	0.4901961	0.5098039	0.5098039	0.4901961
0.5098039	0.4901961	0.4901961	0.4901961	0.5098039
0.4901961	0.5098039	0.4901961	0.4901961	0.5098039
0.4901961	0.5098039	0.4901961	0.5098039	0.4901961
0.4901961	0.4901961	0.5098039	0.5098039	0.5098039
0.5098039	0.5098039	0.5098039	0.4901961	0.4901961
0.4901961	0.5098039	0.4901961	0.4901961	0.4901961
0.5098039	0.5098039	0.4901961	0.4901961	0.4901961
...	...	...	...	...
0.4901961	0.4901961	0.5098039	0.4901961	0.4901961
0.5098039	0.4901961	0.5098039	0.4901961	0.5098039
0.5098039	0.4901961	0.4901961	0.5098039	0.5098039
0.5098039	0.5098039	0.5098039	0.5098039	0.4901961
0.5098039	0.5098039	0.5098039	0.4901961	0.4901961
0.5098039	0.4901961	0.4901961	0.5098039	0.4901961
0.5098039	0.4901961	0.4901961	0.4901961	0.4901961
0.4901961	0.4901961	0.4901961	0.5098039	0.4901961
0.4901961	0.4901961	0.4901961	0.5098039	0.4901961
0.4901961	0.4901961	0.4901961	0.5098039	0.5098039
0.4901961	0.4901961	0.4901961	0.5098039	0.5098039
0.4901961	0.5098039	0.5098039	0.5098039	0.4901961
0.4901961	0.4901961	0.5098039	0.4901961	0.4901961
0.5098039	0.4901961	0.4901961	0.5098039	0.5098039
0.4901961	0.5098039	0.5098039	0.4901961	0.4901961
0.4901961	0.4901961	0.4901961	0.4901961	0.5098039
0.4901961	0.4901961	0.5098039	0.4901961	0.4901961
0.4901961	0.4901961	0.4901961	0.4901961	0.5098039
0.4901961	0.4901961	0.4901961	0.5098039	0.5098039
0.4901961	0.5098039	0.4901961	0.4901961	0.5098039
0.4901961	0.4901961	0.4901961	0.5098039	0.5098039

[32]:
```
# study the equilibrium times with standard conditions
equilibrium_times
```

run_1	run_2	run_3	run_4	run_5
21	23	35	25	47

[33]:
```
rowMeans(equilibrium_times)
```

30.2

[34]:
```
# study the size of the equilibrium cluster
eq_cluster
```

run_1	run_2	run_3	run_4	run_5
238	213	241	422	145

[35]:
```
rowMeans(eq_cluster)
```

251.8

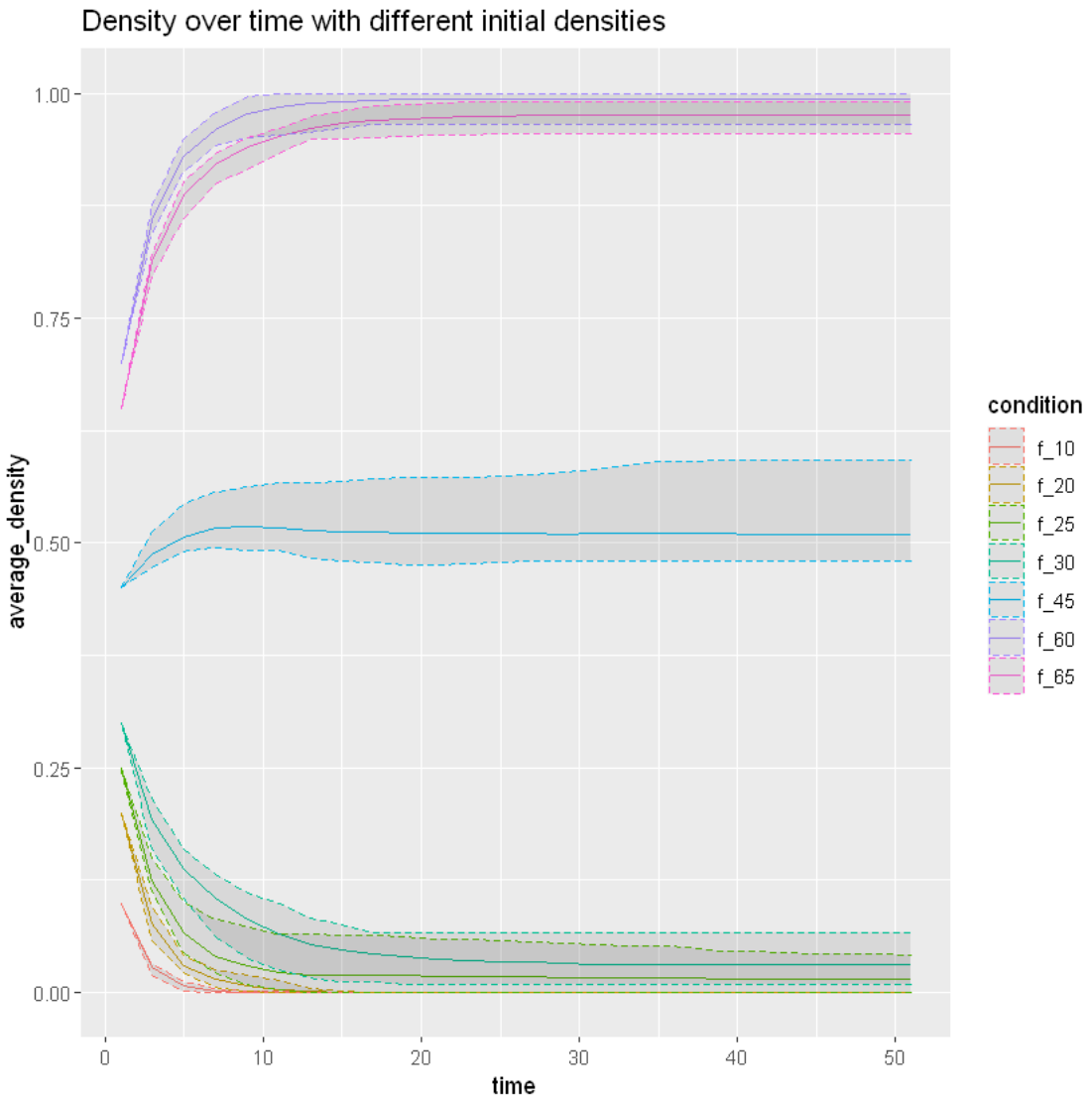
Now, I show how are the results taking these standard conditions, and as we can see the density and actived zones increase over the time. Also, we can observe that the frequency of activation of our fireflies is even for all of then. The equilibrim times go from 19 to 35 (a barplot I found it innecessary since there are only 5 data points) and the size of the cluster in the equilibrium vary a lot, since 96.5 to 486. The average of the size in the equilibrium over 5 runs of the experiment is 297,1 and the average of the time to reach the equilibrium is 26.2 time steps.

[36]:
```
# the parameters of different initial density will be 10%, 15%, 20%, 25%, 30%,␣
 ↪45%, 65%, 70%
number = 5
time = 50
th = 0.25
n = 25
shy_density = 0
type_of_world = 'Normal'
f_10 <- run_experiment(number, time, th, n, initial_density=0.1, shy_density ,␣
 ↪type_of_world)
f_15 <- run_experiment(number, time, th, n, initial_density=0.15, shy_density ,␣
 ↪type_of_world)
f_20 <- run_experiment(number, time, th, n, initial_density=0.2, shy_density ,␣
 ↪type_of_world)
f_25 <- run_experiment(number, time, th, n, initial_density=0.25, shy_density ,␣
 ↪type_of_world)
f_30 <- run_experiment(number, time, th, n, initial_density=0.30, shy_density ,␣
 ↪type_of_world)
f_45 <- run_experiment(number, time, th, n, initial_density=0.45, shy_density ,␣
 ↪type_of_world)
f_65 <- run_experiment(number, time, th, n, initial_density=0.65, shy_density ,␣
 ↪type_of_world)
```

```
f_70 <- run_experiment(number, time, th, n, initial_density=0.70, shy_density ,␣
↪type_of_world)
```

[37]:
```
# study the density
time <- 50
f_10_d <- data_experiment(f_10, 'density')
f_10_d$average_density <- rowMeans(f_10_d)
f_10_d$min <- apply(f_10_d, 1, FUN=min, na.rm=TRUE)
f_10_d$max <- apply(f_10_d, 1, FUN=max, na.rm=TRUE)
f_10_d$time <- c(0:((time)%/%2))*2 + 1
f_10_d$condition <- 'f_10' # put labels
f_15_d <- data_experiment(f_15, 'density')
f_15_d$average_density <- rowMeans(f_15_d)
f_15_d$min <- apply(f_15_d, 1, FUN=min, na.rm=TRUE)
f_15_d$max <- apply(f_15_d, 1, FUN=max, na.rm=TRUE)
f_15_d$time <- c(0:((time)%/%2))*2 + 1
f_15_d$condition <- 'f_15' # put labels
f_20_d <- data_experiment(f_20, 'density')
f_20_d$average_density <- rowMeans(f_20_d)
f_20_d$min <- apply(f_20_d, 1, FUN=min, na.rm=TRUE)
f_20_d$max <- apply(f_20_d, 1, FUN=max, na.rm=TRUE)
f_20_d$time <- c(0:((time)%/%2))*2 + 1
f_20_d$condition <- 'f_20' # put labels
f_25_d <- data_experiment(f_25, 'density')
f_25_d$average_density <- rowMeans(f_25_d)
f_25_d$min <- apply(f_25_d, 1, FUN=min, na.rm=TRUE)
f_25_d$max <- apply(f_25_d, 1, FUN=max, na.rm=TRUE)
f_25_d$time <- c(0:((time)%/%2))*2 + 1
f_25_d$condition <- 'f_25' # put labels
f_30_d <- data_experiment(f_30, 'density')
f_30_d$average_density <- rowMeans(f_30_d)
f_30_d$min <- apply(f_30_d, 1, FUN=min, na.rm=TRUE)
f_30_d$max <- apply(f_30_d, 1, FUN=max, na.rm=TRUE)
f_30_d$time <- c(0:((time)%/%2))*2 + 1
f_30_d$condition <- 'f_30' # put labels
f_45_d <- data_experiment(f_45, 'density')
f_45_d$average_density <- rowMeans(f_45_d)
f_45_d$min <- apply(f_45_d, 1, FUN=min, na.rm=TRUE)
f_45_d$max <- apply(f_45_d, 1, FUN=max, na.rm=TRUE)
f_45_d$time <- c(0:((time)%/%2))*2 + 1
f_45_d$condition <- 'f_45' # put labels
f_65_d <- data_experiment(f_65, 'density')
f_65_d$average_density <- rowMeans(f_65_d)
f_65_d$min <- apply(f_65_d, 1, FUN=min, na.rm=TRUE)
f_65_d$max <- apply(f_65_d, 1, FUN=max, na.rm=TRUE)
f_65_d$time <- c(0:((time)%/%2))*2 + 1
f_65_d$condition <- 'f_65' # put labels
```

```
f_70_d <- data_experiment(f_70, 'density')
f_70_d$average_density <- rowMeans(f_70_d)
f_70_d$min <- apply(f_70_d, 1, FUN=min, na.rm=TRUE)
f_70_d$max <- apply(f_70_d, 1, FUN=max, na.rm=TRUE)
f_70_d$time <- c(0:((time)%/%2))*2 + 1
f_70_d$condition <- 'f_60' # put labels
fdensities <- rbind(f_10_d, f_20_d, f_25_d, f_30_d, f_45_d, f_65_d, f_70_d)
ggplot(data=fdensities, aes(x=time, y=average_density, colour=condition))+␣
↪geom_line() + geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+␣
↪ggtitle("Density over time with different initial densities")
```

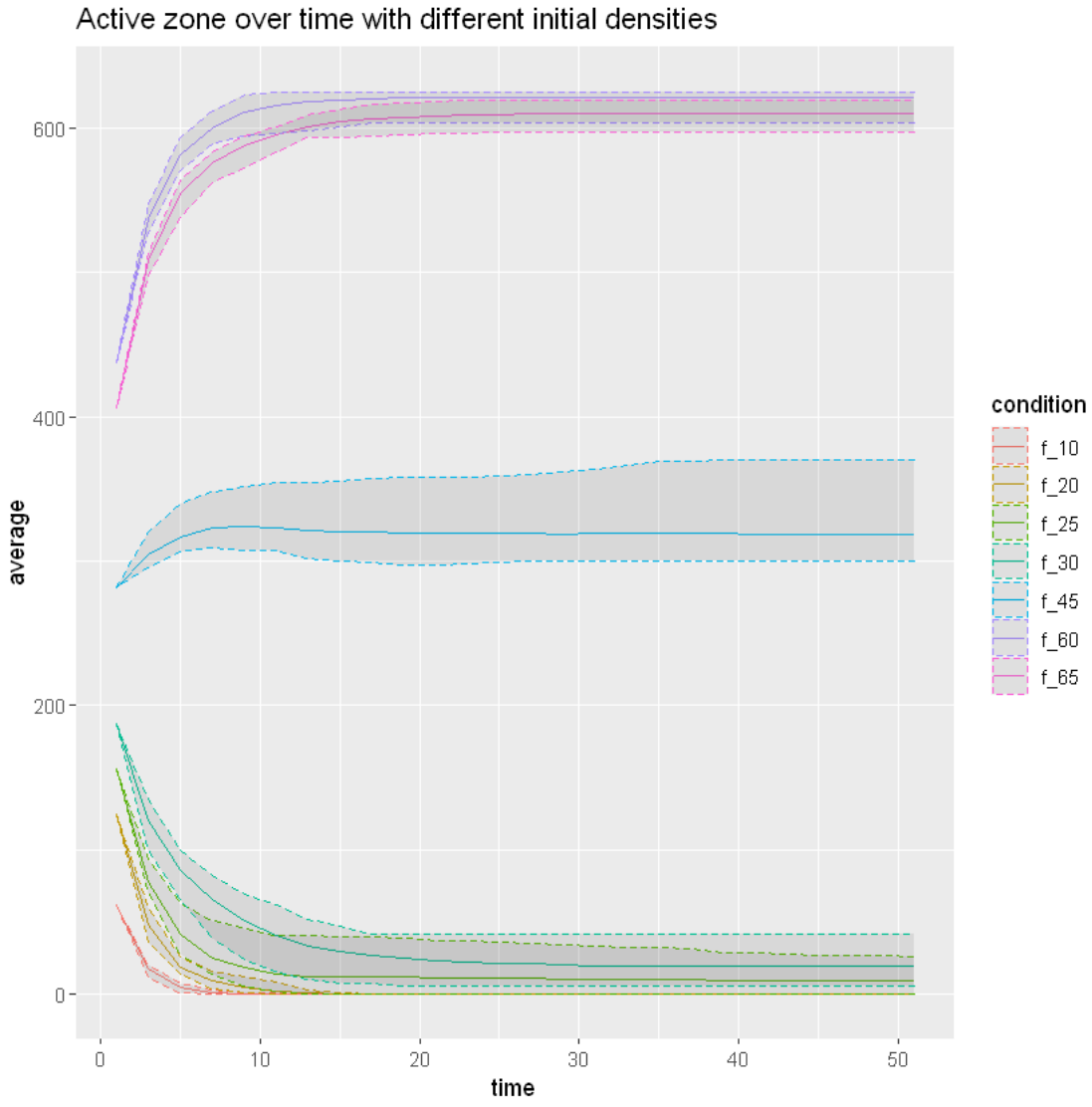


As we can see the density over the time depens a lot in the initial density. The density of the experiments with initial density smaller than 45%, don't increase over the time, the tendencty is arrive

to 0. While in the densities over 45% the increment is really big until all the matriz is populated by blinking fireflies. With a 45% the density remain more or less stable, and I hypothesized seeing the plot that the equilibrium is reach earlier than in the rest of experiments with different initial density, but I will study this later.

[38]:
```
# study the active_zone
time <- 50
f_10_d <- data_experiment(f_10, 'active_zone')
f_10_d$average <- rowMeans(f_10_d)
f_10_d$min <- apply(f_10_d, 1, FUN=min, na.rm=TRUE)
f_10_d$max <- apply(f_10_d, 1, FUN=max, na.rm=TRUE)
f_10_d$time <- c(0:((time)%/%2))*2 + 1
f_10_d$condition <- 'f_10' # put labels
f_15_d <- data_experiment(f_15, 'active_zone')
f_15_d$average <- rowMeans(f_15_d)
f_15_d$min <- apply(f_15_d, 1, FUN=min, na.rm=TRUE)
f_15_d$max <- apply(f_15_d, 1, FUN=max, na.rm=TRUE)
f_15_d$time <- c(0:((time)%/%2))*2 + 1
f_15_d$condition <- 'f_15' # put labels
f_20_d <- data_experiment(f_20, 'active_zone')
f_20_d$average <- rowMeans(f_20_d)
f_20_d$min <- apply(f_20_d, 1, FUN=min, na.rm=TRUE)
f_20_d$max <- apply(f_20_d, 1, FUN=max, na.rm=TRUE)
f_20_d$time <- c(0:((time)%/%2))*2 + 1
f_20_d$condition <- 'f_20' # put labels
f_25_d <- data_experiment(f_25, 'active_zone')
f_25_d$average <- rowMeans(f_25_d)
f_25_d$min <- apply(f_25_d, 1, FUN=min, na.rm=TRUE)
f_25_d$max <- apply(f_25_d, 1, FUN=max, na.rm=TRUE)
f_25_d$time <- c(0:((time)%/%2))*2 + 1
f_25_d$condition <- 'f_25' # put labels
f_30_d <- data_experiment(f_30, 'active_zone')
f_30_d$average <- rowMeans(f_30_d)
f_30_d$min <- apply(f_30_d, 1, FUN=min, na.rm=TRUE)
f_30_d$max <- apply(f_30_d, 1, FUN=max, na.rm=TRUE)
f_30_d$time <- c(0:((time)%/%2))*2 + 1
f_30_d$condition <- 'f_30' # put labels
f_45_d <- data_experiment(f_45, 'active_zone')
f_45_d$average <- rowMeans(f_45_d)
f_45_d$min <- apply(f_45_d, 1, FUN=min, na.rm=TRUE)
f_45_d$max <- apply(f_45_d, 1, FUN=max, na.rm=TRUE)
f_45_d$time <- c(0:((time)%/%2))*2 + 1
f_45_d$condition <- 'f_45' # put labels
f_65_d <- data_experiment(f_65, 'active_zone')
f_65_d$average <- rowMeans(f_65_d)
f_65_d$min <- apply(f_65_d, 1, FUN=min, na.rm=TRUE)
f_65_d$max <- apply(f_65_d, 1, FUN=max, na.rm=TRUE)
```

```
f_65_d$time <- c(0:((time)%/%2))*2 + 1
f_65_d$condition <- 'f_65' # put labels
f_70_d <- data_experiment(f_70, 'active_zone')
f_70_d$average <- rowMeans(f_70_d)
f_70_d$min <- apply(f_70_d, 1, FUN=min, na.rm=TRUE)
f_70_d$max <- apply(f_70_d, 1, FUN=max, na.rm=TRUE)
f_70_d$time <- c(0:((time)%/%2))*2 + 1
f_70_d$condition <- 'f_60' # put labels
factive_zone<- rbind(f_10_d, f_20_d, f_25_d, f_30_d, f_45_d, f_65_d, f_70_d)
ggplot(data=factive_zone, aes(x=time, y=average, colour=condition))+ geom_line()␣
↪+ geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+ ggtitle("Active␣
↪zone over time with different initial densities")
```



The density is proportional to the active zone therefore is logic to think that the result will be even, as I demonstrate here with this plot.

[39]:
```
# study the frequency

f_10_d <- data_experiment(f_10, 'frequency')
f_10_d$condition <- 'f_10' # put labels
f_15_d <- data_experiment(f_15, 'frequency')
f_15_d$condition <- 'f_15' # put labels
f_20_d <- data_experiment(f_20, 'frequency')
f_20_d$condition <- 'f_20' # put labels
f_25_d <- data_experiment(f_25, 'frequency')
f_25_d$condition <- 'f_25' # put labels
f_30_d <- data_experiment(f_30, 'frequency')
f_30_d$condition <- 'f_30' # put labels
f_45_d <- data_experiment(f_45, 'frequency')
f_45_d$condition <- 'f_45' # put labels
f_65_d <- data_experiment(f_65, 'frequency')
f_65_d$condition <- 'f_65' # put labels
f_70_d <- data_experiment(f_70, 'frequency')
f_70_d$condition <- 'f_70' # put labels

ffrequencies<- rbind(f_10_d, f_20_d, f_25_d, f_30_d, f_45_d, f_65_d, f_70_d)
ffrequencies
```

run_1	run_2	run_3	run_4	run_5	condition
1	1	1	1	1	f_10
1	1	1	1	1	f_20
1	1	1	1	1	f_25
1	1	1	1	1	f_30
1	1	1	1	1	f_45
1	1	1	1	1	f_65
1	1	1	1	1	f_70

The frequencies are still even with the change of the differential initial densities.

[40]:
```
# study the equilibrium time

f_10_d <- data_experiment(f_10, 'equilibrium_time')
f_10_d$condition <- 'f_10' # put labels
f_15_d <- data_experiment(f_15, 'equilibrium_time')
f_15_d$condition <- 'f_15' # put labels
f_20_d <- data_experiment(f_20, 'equilibrium_time')
f_20_d$condition <- 'f_20' # put labels
f_25_d <- data_experiment(f_25, 'equilibrium_time')
f_25_d$condition <- 'f_25' # put labels
f_30_d <- data_experiment(f_30, 'equilibrium_time')
f_30_d$condition <- 'f_30' # put labels
```

```
f_45_d <- data_experiment(f_45, 'equilibrium_time')
f_45_d$condition <- 'f_45' # put labels
f_65_d <- data_experiment(f_65, 'equilibrium_time')
f_65_d$condition <- 'f_65' # put labels
f_70_d <- data_experiment(f_70, 'equilibrium_time')
f_70_d$condition <- 'f_70' # put labels

ftime<- rbind(f_10_d, f_20_d, f_25_d, f_30_d, f_45_d, f_65_d, f_70_d)
rownames(ftime) <- ftime$condition
ftime <- subset(ftime, select = -c(condition))
ftime$average_equilibrium_time <- rowMeans(ftime)
ftime
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_time
f_10	8	8	10	10	10	9.2
f_20	14	12	18	12	14	14.0
f_25	12	14	14	14	18	14.4
f_30	18	24	22	34	16	22.8
f_45	42	30	35	27	15	29.8
f_65	19	13	23	11	17	16.6
f_70	19	17	11	19	11	15.4

The initial density influence a lot in the average time to reach the equilibrium. When the density is very low or very high the equilibrium is reach very quickly but when the density is 45% or 50% the equilibrium is reached later. This can be in the case of the lower densities because the fireflies does not interact and in the case of the bigger densities the interaction is very high. In these both scenarios is easier reach the equilibrium.

[41]:
```
# study the equilibrium time

f_10_d <- data_experiment(f_10, 'average_cluster_equilibrium')
f_10_d$condition <- 'f_10' # put labels
f_15_d <- data_experiment(f_15, 'average_cluster_equilibrium')
f_15_d$condition <- 'f_15' # put labels
f_20_d <- data_experiment(f_20, 'average_cluster_equilibrium')
f_20_d$condition <- 'f_20' # put labels
f_25_d <- data_experiment(f_25, 'average_cluster_equilibrium')
f_25_d$condition <- 'f_25' # put labels
f_30_d <- data_experiment(f_30, 'average_cluster_equilibrium')
f_30_d$condition <- 'f_30' # put labels
f_45_d <- data_experiment(f_45, 'average_cluster_equilibrium')
f_45_d$condition <- 'f_45' # put labels
f_65_d <- data_experiment(f_65, 'average_cluster_equilibrium')
f_65_d$condition <- 'f_65' # put labels
f_70_d <- data_experiment(f_70, 'average_cluster_equilibrium')
f_70_d$condition <- 'f_70' # put labels

fcluster<- rbind(f_10_d, f_20_d, f_25_d, f_30_d, f_45_d, f_65_d, f_70_d)
```

```
rownames(fcluster) <- fcluster$condition
fcluster <- subset(fcluster, select = -c(condition))
fcluster$average_equilibrium_cluster_size <- rowMeans(fcluster)
fcluster
```

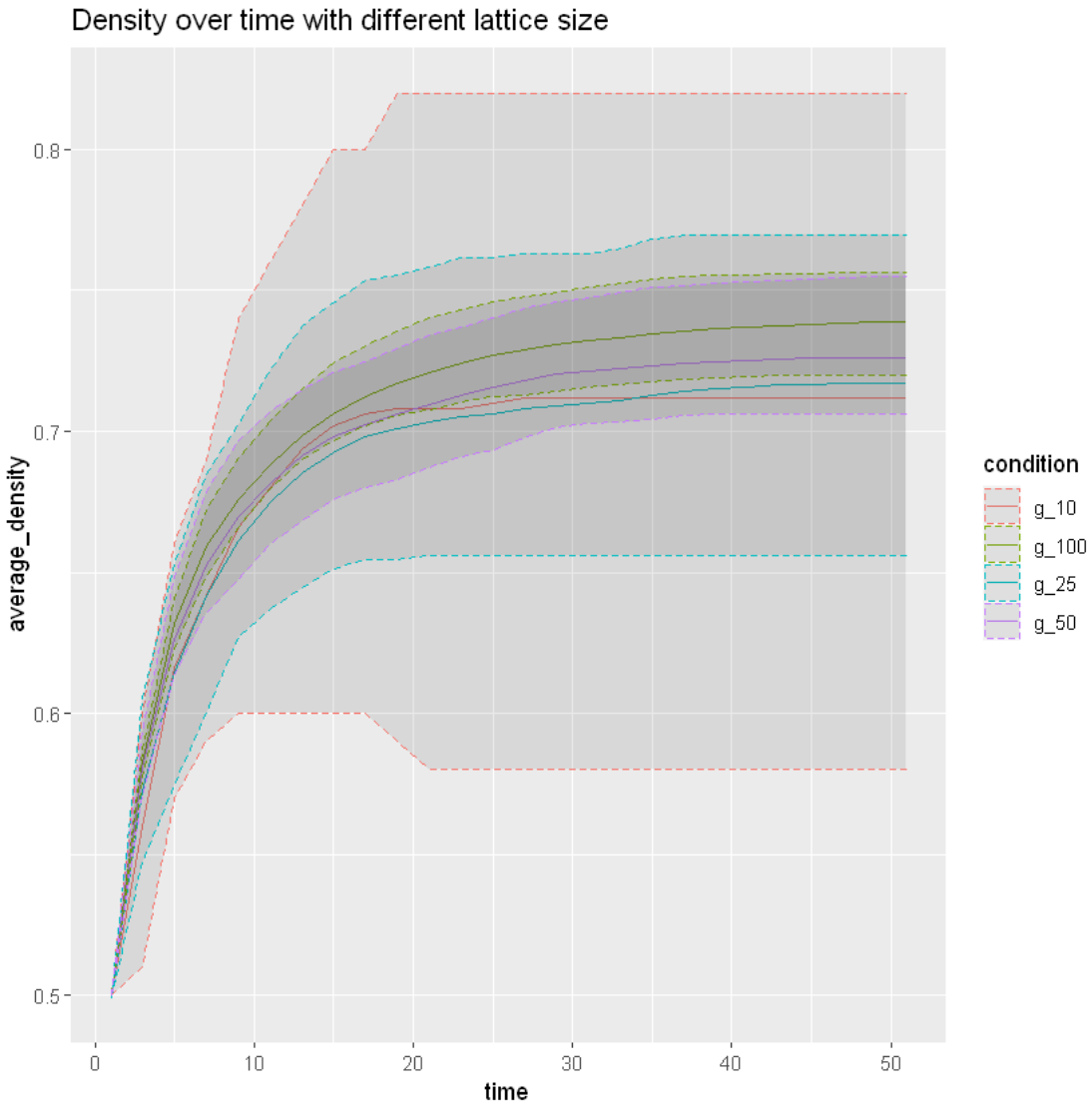
	run_1	run_2	run_3	run_4	run_5	average_equilibrium_cluster_size
f_10	625.0	625.0000	624.0	625	625	624.8000
f_20	625.0	625.0000	625.0	625	625	625.0000
f_25	584.0	625.0000	625.0	606	625	613.0000
f_30	583.0	613.0000	605.0	609	618	605.6000
f_45	79.5	106.6667	184.5	60	103	106.7333
f_65	612.0	593.0000	619.0	594	615	606.6000
f_70	625.0	603.0000	625.0	625	625	620.6000

The initial density influence a lot as well in the average cluster size when reached the equilibrium. When the density is very low or very high the size is bigger but when the density is 45% or 50% the size is smaller. This can be in the case of the lower densities because the fireflies does not interact and in the case of the bigger densities the interaction is very high. In these both scenarios when the equilibrium is reached more or less all the fireflies are active and then inactive (all synchronize) while in the scenario of the 45% or 50% we can see that the equilibrium is reached when there are a stable state with smaller clusters. In contrast to the other experiments the cluster do not begin to merge.

[42]:
```
# the lattice size: 10, 25, 50, 100
number = 5
time = 50
th = 0.25
shy_density = 0
type_of_world = 'Normal'
initial_density = 0.5

g_10 <- run_experiment(number, time, th, n=10, initial_density, shy_density ,␣
 ↪type_of_world)
g_25 <- run_experiment(number, time, th, n=25, initial_density, shy_density ,␣
 ↪type_of_world)
g_50 <- run_experiment(number, time, th, n=50, initial_density, shy_density ,␣
 ↪type_of_world)
g_100 <- run_experiment(number, time, th, n=100, initial_density, shy_density ,␣
 ↪type_of_world)
```

[43]:
```
# study the density
time <- 50
g_10_d <- data_experiment(g_10, 'density')
g_10_d$average_density <- rowMeans(g_10_d)
g_10_d$min <- apply(g_10_d, 1, FUN=min, na.rm=TRUE)
g_10_d$max <- apply(g_10_d, 1, FUN=max, na.rm=TRUE)
g_10_d$time <- c(0:((time)%/%2))*2 + 1
g_10_d$condition <- 'g_10' # put labels
```

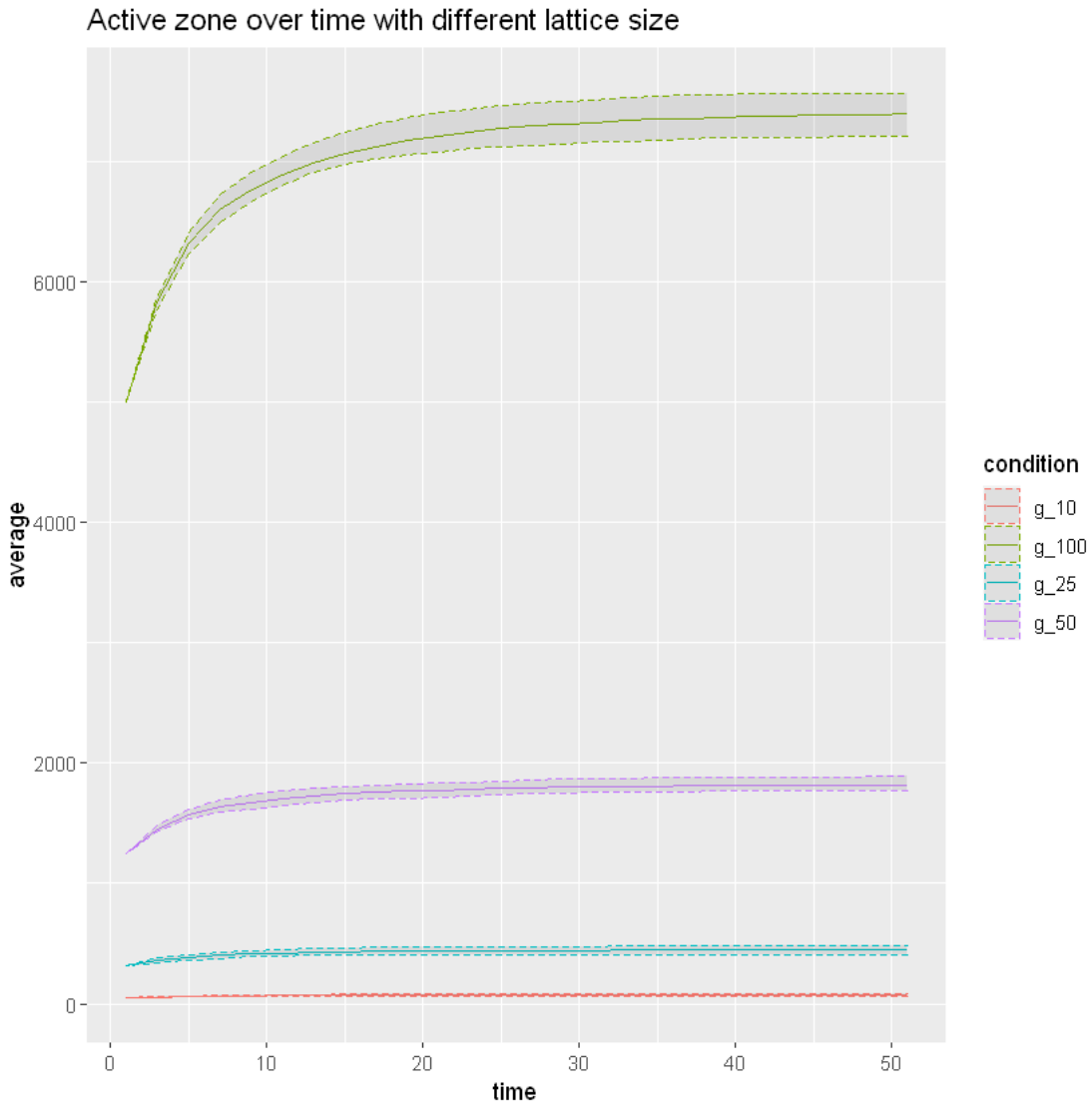
```r
g_25_d <- data_experiment(g_25, 'density')
g_25_d$average_density <- rowMeans(g_25_d)
g_25_d$min <- apply(g_25_d, 1, FUN=min, na.rm=TRUE)
g_25_d$max <- apply(g_25_d, 1, FUN=max, na.rm=TRUE)
g_25_d$time <- c(0:((time)%/%2))*2 + 1
g_25_d$condition <- 'g_25' # put labels
g_50_d <- data_experiment(g_50 , 'density')
g_50_d$average_density <- rowMeans(g_50_d)
g_50_d$min <- apply(g_50_d, 1, FUN=min, na.rm=TRUE)
g_50_d$max <- apply(g_50_d, 1, FUN=max, na.rm=TRUE)
g_50_d$time <- c(0:((time)%/%2))*2 + 1
g_50_d$condition <- 'g_50' # put labels
g_100_d <- data_experiment(g_100, 'density')
g_100_d$average_density <- rowMeans(g_100_d)
g_100_d$min <- apply(g_100_d, 1, FUN=min, na.rm=TRUE)
g_100_d$max <- apply(g_100_d, 1, FUN=max, na.rm=TRUE)
g_100_d$time <- c(0:((time)%/%2))*2 + 1
g_100_d$condition <- 'g_100' # put labels


gdensities <- rbind(g_10_d, g_25_d, g_50_d, g_100_d)
ggplot(data=gdensities, aes(x=time, y=average_density, colour=condition))+␣
 ↪geom_line() + geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+␣
 ↪ggtitle("Density over time with different lattice size")
```



Apparently with bigger lattice size bigger it is the range of values for density over the different runs.

[44]:
```
# study the active zone
time <- 50
g_10_d <- data_experiment(g_10, 'active_zone')
g_10_d$average <- rowMeans(g_10_d)
g_10_d$min <- apply(g_10_d, 1, FUN=min, na.rm=TRUE)
g_10_d$max <- apply(g_10_d, 1, FUN=max, na.rm=TRUE)
g_10_d$time <- c(0:((time)%/%2))*2 + 1
g_10_d$condition <- 'g_10' # put labels
g_25_d <- data_experiment(g_25, 'active_zone')
g_25_d$average <- rowMeans(g_25_d)
```

```
g_25_d$min <- apply(g_25_d, 1, FUN=min, na.rm=TRUE)
g_25_d$max <- apply(g_25_d, 1, FUN=max, na.rm=TRUE)
g_25_d$time <- c(0:((time)%/%2))*2 + 1
g_25_d$condition <- 'g_25' # put labels
g_50_d <- data_experiment(g_50 , 'active_zone')
g_50_d$average <- rowMeans(g_50_d)
g_50_d$min <- apply(g_50_d, 1, FUN=min, na.rm=TRUE)
g_50_d$max <- apply(g_50_d, 1, FUN=max, na.rm=TRUE)
g_50_d$time <- c(0:((time)%/%2))*2 + 1
g_50_d$condition <- 'g_50' # put labels
g_100_d <- data_experiment(g_100, 'active_zone')
g_100_d$average <- rowMeans(g_100_d)
g_100_d$min <- apply(g_100_d, 1, FUN=min, na.rm=TRUE)
g_100_d$max <- apply(g_100_d, 1, FUN=max, na.rm=TRUE)
g_100_d$time <- c(0:((time)%/%2))*2 + 1
g_100_d$condition <- 'g_100' # put labels


gactive_zone<- rbind(g_10_d, g_25_d, g_50_d, g_100_d)
ggplot(data=gactive_zone, aes(x=time, y=average, colour=condition))+ geom_line()␣
 ↪+ geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+ ggtitle("Active␣
 ↪zone over time with different lattice size")
```



Logically bigger it is the lattice size, bigger is the amount of fireflies that can be activated.

[45]:
```
# study the frequency

g_10_d <- data_experiment(g_10, 'frequency')
g_10_d$condition <- 'g_10' # put labels
g_25_d <- data_experiment(g_25, 'frequency')
g_25_d$condition <- 'g_25' # put labels
g_50_d <- data_experiment(g_50, 'frequency')
g_50_d$condition <- 'g_50' # put labels
g_100_d <- data_experiment(g_100, 'frequency')
g_100_d$condition <- 'g_100' # put labels
```

```
ffrequencies<- rbind(g_10_d, g_25_d, g_50_d, g_100_d)
ffrequencies
```

run_1	run_2	run_3	run_4	run_5	condition
1	1	1	1	1	g_10
1	1	1	1	1	g_25
1	1	1	1	1	g_50
1	1	1	1	1	g_100

Again the frequencies are even with these conditions.

[46]:
```
# study the average equilibrium time

g_10_d <- data_experiment(g_10, 'equilibrium_time')
g_10_d$condition <- 'g_10' # put labels
g_25_d <- data_experiment(g_25, 'equilibrium_time')
g_25_d$condition <- 'g_25' # put labels
g_50_d <- data_experiment(g_50, 'equilibrium_time')
g_50_d$condition <- 'g_50' # put labels
g_100_d <- data_experiment(g_100, 'equilibrium_time')
g_100_d$condition <- 'g_100' # put labels

gequilibrium_time<- rbind(g_10_d, g_25_d, g_50_d, g_100_d)
rownames(gequilibrium_time) <- gequilibrium_time$condition
gequilibrium_time <- subset(gequilibrium_time, select = -c(condition))
gequilibrium_time$average_equilibrium_time <- rowMeans(gequilibrium_time)
gequilibrium_time
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_time
g_10	22	15	19	11	17	16.8
g_25	47	23	19	27	21	27.4
g_50	39	47	45	49	33	42.6
g_100	47	0	0	0	0	9.4

This results are really interesting, because with really small lattice size the average equilibrium time is really short. When you have a bigger lattice size the interactions are more difficult and when the lattice size is very small almost all interact. Intituively, you could think that the bigger lattice size take longer (as happen in its first run) but in the rest of runnings get the equilibrium in just one time step. The lattice size for 50 increase the equilibrium time but a really big lattice size like 100 can decreased as well a lot.

[71]:
```
# study the average equilibrium size

g_10_d <- data_experiment(g_10, 'average_cluster_equilibrium')
g_10_d$condition <- 'g_10' # put labels
g_25_d <- data_experiment(g_25, 'average_cluster_equilibrium')
g_25_d$condition <- 'g_25' # put labels
g_50_d <- data_experiment(g_50, 'average_cluster_equilibrium')
```

```
g_50_d$condition <- 'g_50' # put labels



gequilibrium_size<- rbind(g_10_d, g_25_d, g_50_d)
rownames(gequilibrium_size) <- gequilibrium_size$condition
gequilibrium_size <- subset(gequilibrium_size, select = -c(condition))
gequilibrium_size$average_equilibrium_size <- rowMeans(gequilibrium_size)
gequilibrium_size
```

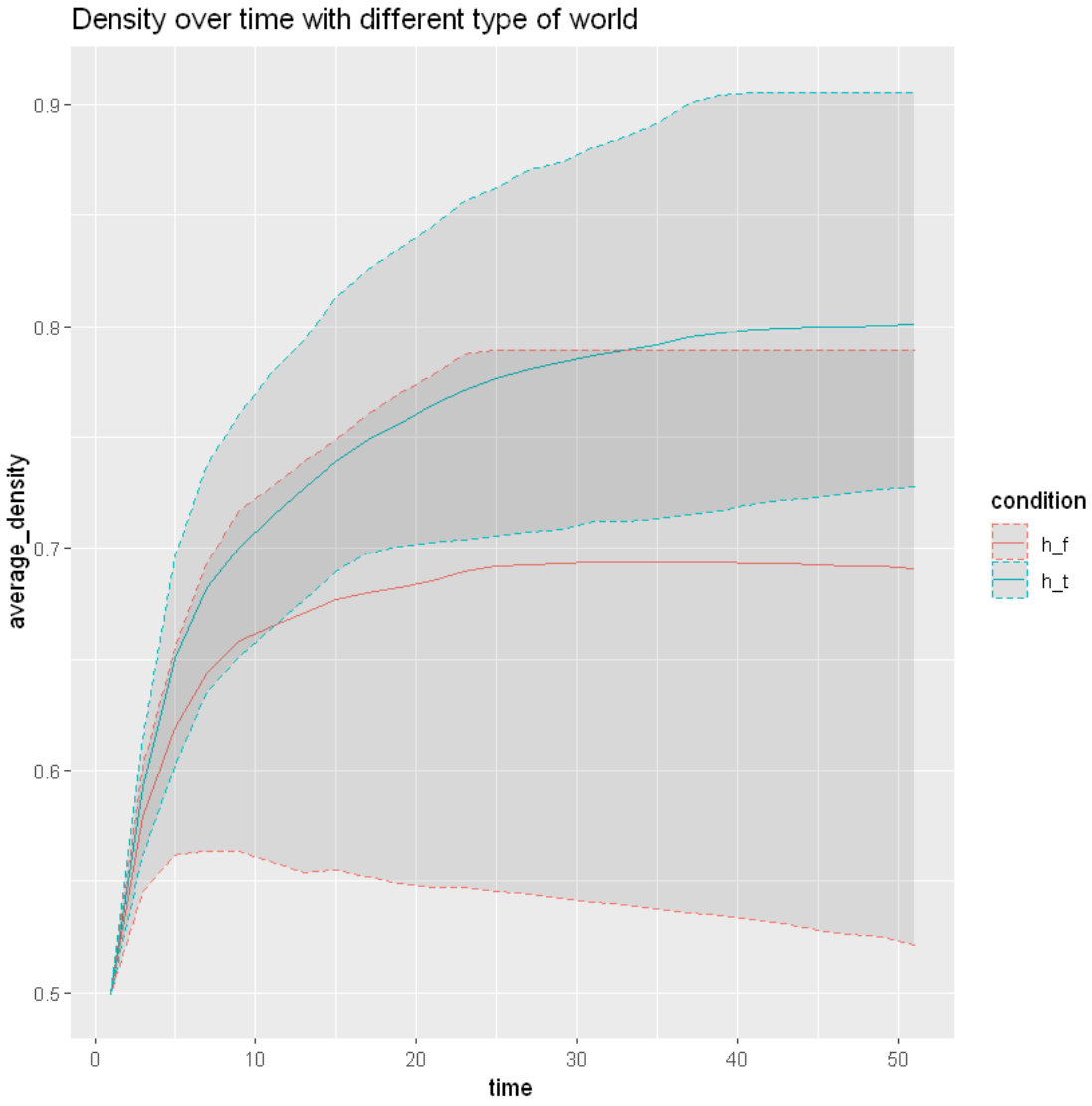
	run_1	run_2	run_3	run_4	run_5	average_equilibrium_size
g_10	42.0	80	82	66.0	67.0	67.40
g_25	444.0	238	236	213.5	205.0	267.30
g_50	441.5	1793	932	943.5	352.8	892.56

Here I needed to remove the average size for 100 lattice size, because my function did not found any cluster, meaning that in the 100 lattice size the equilibrium is reached without any consistent structure. For the rest of lattice size, we can see as it is expected bigger clusters in bigger lattices.

[21]:
```
# Different type of world : Normal, Flat, Toroidal

number = 5
time = 50
th = 0.25
shy_density = 0
n = 25
initial_density = 0.5

h_f <- run_experiment(number, time, th, n, initial_density, shy_density ,␣
 ↪type_of_world= 'Flat', bias='None')
h_t <- run_experiment(number, time, th, n, initial_density, shy_density ,␣
 ↪type_of_world= 'Toroidal', bias='None')
```

[49]:
```
# study the density
time <- 50
h_f_d <- data_experiment(h_f, 'density')
h_f_d$average_density <- rowMeans(h_f_d)
h_f_d$min <- apply(h_f_d, 1, FUN=min, na.rm=TRUE)
h_f_d$max <- apply(h_f_d, 1, FUN=max, na.rm=TRUE)
h_f_d$time <- c(0:((time)%/%2))*2 + 1
h_f_d$condition <- 'h_f' # put labels
h_t_d <- data_experiment(h_t, 'density')
h_t_d$average_density <- rowMeans(h_t_d)
h_t_d$min <- apply(h_t_d , 1, FUN=min, na.rm=TRUE)
h_t_d$max <- apply(h_t_d , 1, FUN=max, na.rm=TRUE)
h_t_d$time <- c(0:((time)%/%2))*2 + 1
h_t_d$condition <- 'h_t' # put labels
```

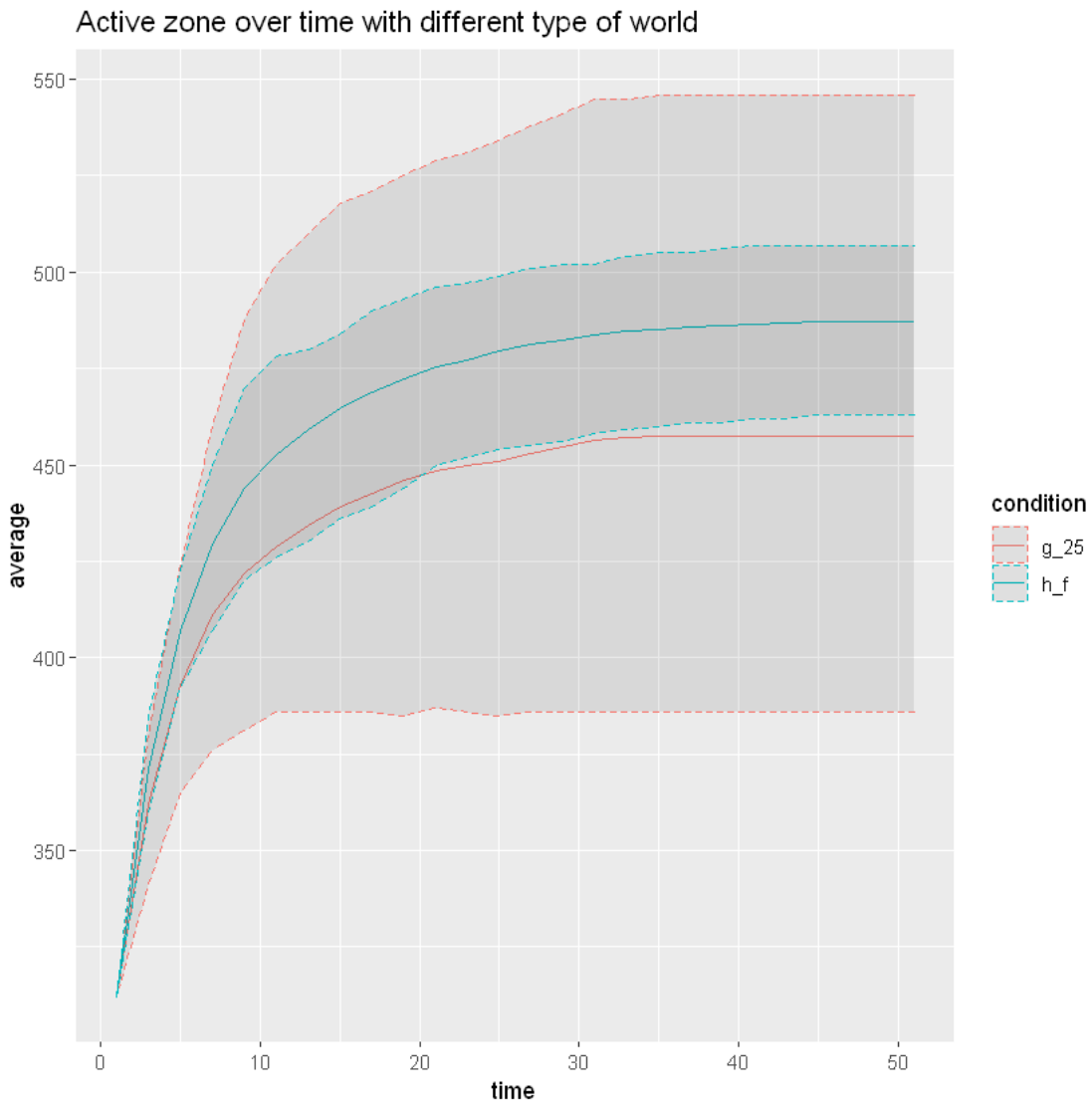
```
hdensities <- rbind(h_t_d , h_f_d)
ggplot(data=hdensities, aes(x=time, y=average_density, colour=condition))+␣
↪geom_line() + geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+␣
↪ggtitle("Density over time with different type of world")
```



As expected, the density is lower in the flat world because the interactions in the borders are limited.

[23]:
```
library('ggplot2')
# study the active zone
time <- 50
h_f_d <- data_experiment(h_f, 'active_zone')
```

```
h_f_d$average <- rowMeans(h_f_d)
h_f_d$min <- apply(h_f_d, 1, FUN=min, na.rm=TRUE)
h_f_d$max <- apply(h_f_d, 1, FUN=max, na.rm=TRUE)
h_f_d$time <- c(0:((time)%/%2))*2 + 1
h_f_d$condition <- 'h_f' # put labels
h_t_d <- data_experiment(h_t, 'active_zone')
h_t_d$average <- rowMeans(h_t_d)
h_t_d$min <- apply(h_t_d, 1, FUN=min, na.rm=TRUE)
h_t_d$max <- apply(h_t_d, 1, FUN=max, na.rm=TRUE)
h_t_d$time <- c(0:((time)%/%2))*2 + 1
h_t_d$condition <- 'g_25' # put labels


hactive_zone<- rbind(h_t_d , h_f_d)
ggplot(data=hactive_zone, aes(x=time, y=average, colour=condition))+ geom_line()␣
 ↪+ geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+ ggtitle("Active␣
 ↪zone over time with different type of world")
```



This results are also accordingly to the results found in density.

[51]:
```
# study the frequency

h_t_d <- data_experiment(h_t, 'frequency')
h_t_d$condition <- 'h_t' # put labels
h_f_d <- data_experiment(h_f, 'frequency')
h_f_d$condition <- 'h_f' # put labels


hfrequencies<-  rbind(h_t_d , h_f_d)
hfrequencies
```

run_1	run_2	run_3	run_4	run_5	condition
1	1	1	1	1	h_t
1	1	1	1	1	h_f

Again the frequencies are even.

[52]:
```
# study the average equilibrium time

h_t_d <- data_experiment(h_t, 'equilibrium_time')
h_t_d$condition <- 'h_t' # put labels
h_f_d <- data_experiment(h_f, 'equilibrium_time')
h_f_d$condition <- 'h_f' # put labels

hequilibrium_time<-  rbind(h_t_d , h_f_d)
rownames(hequilibrium_time) <- hequilibrium_time$condition
hequilibrium_time <- subset(hequilibrium_time, select = -c(condition))
hequilibrium_time$average_equilibrium_time <- rowMeans(hequilibrium_time)
hequilibrium_time
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_time
h_t	45	41	25	39	31	36.2
h_f	29	25	22	27	27	26.0

The average time to reach the equilibrium is bigger in the toroidal world taking in account the number of interactions is bigger and the average equilibrium time for the flat world is really similar to the normal world.

[53]:
```
# study the average equilibrium size
h_t_d <- data_experiment(h_t, 'average_cluster_equilibrium')
h_t_d$condition <- 'h_t' # put labels
h_f_d <- data_experiment(h_f, 'average_cluster_equilibrium')
h_f_d$condition <- 'h_f' # put labels

hequilibrium_size<- rbind(h_t_d , h_f_d)
rownames(hequilibrium_size) <- hequilibrium_size$condition
hequilibrium_size <- subset(hequilibrium_size, select = -c(condition))
hequilibrium_size$average_equilibrium_size <- rowMeans(hequilibrium_size)
hequilibrium_size
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_size
h_t	495	566	526.00000	460	222.5	453.9000
h_f	471	493	94.33333	145	214.0	283.4667

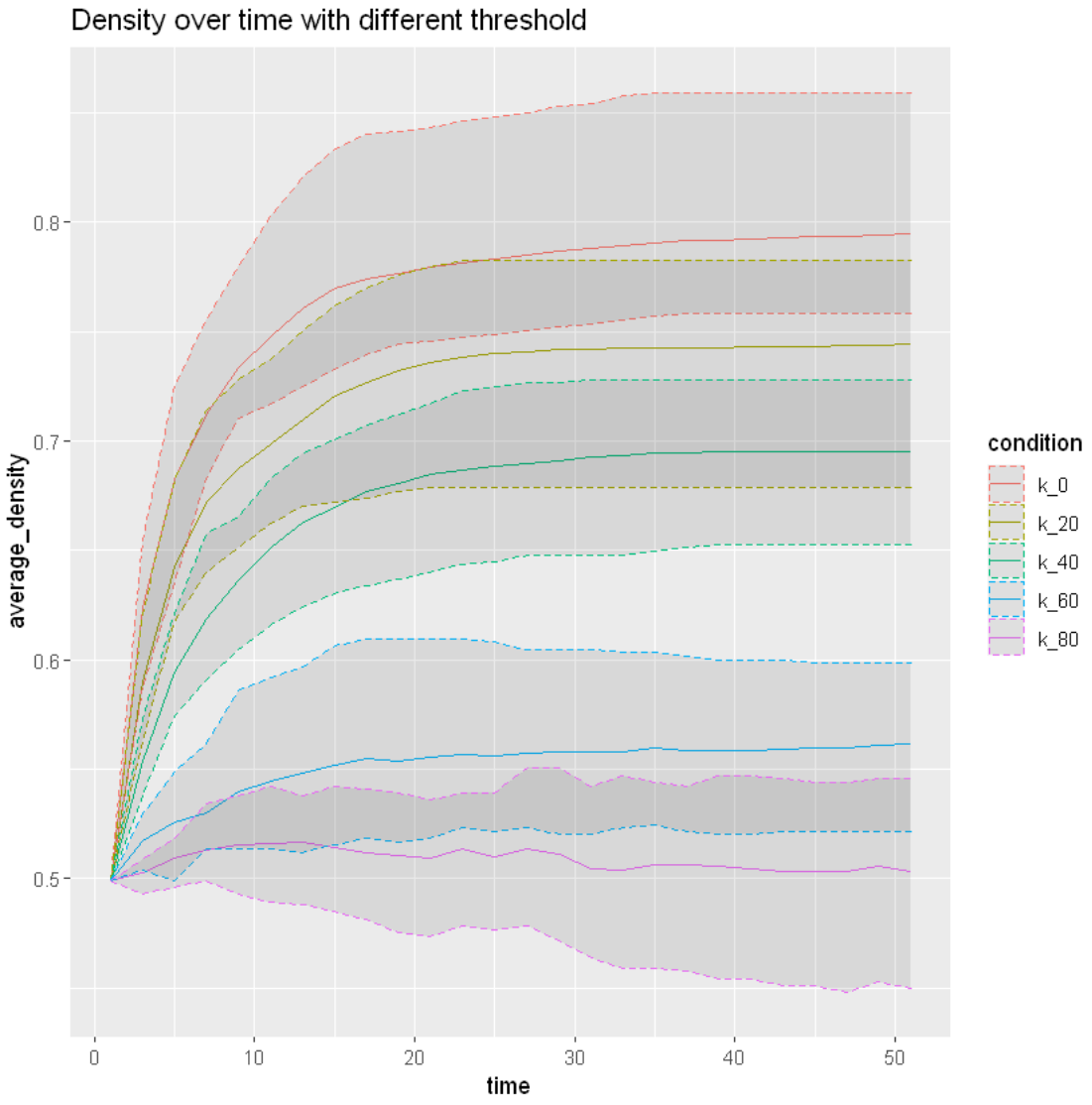
The average equilibrium size is bigger also for the toroidal world because the bigger number of interactions helps to get a more even synchronization, instead have separated clusters.

[54]:
```
#Threshold 0%, 20%, 40%, 60%, 80%

number = 5
time = 50
```

```
shy_density = 0
n = 25
initial_density = 0.5
type_of_world = 'Normal'

k_0 <- run_experiment(number, time, th=0, n, initial_density, shy_density ,␣
 ↪type_of_world)
k_20 <- run_experiment(number, time, th=0.2, n, initial_density, shy_density ,␣
 ↪type_of_world)
k_40 <- run_experiment(number, time, th=0.4, n, initial_density, shy_density ,␣
 ↪type_of_world)
k_60 <- run_experiment(number, time, th=0.6, n, initial_density, shy_density ,␣
 ↪type_of_world)
k_80 <- run_experiment(number, time, th=0.8, n, initial_density, shy_density ,␣
 ↪type_of_world)
```

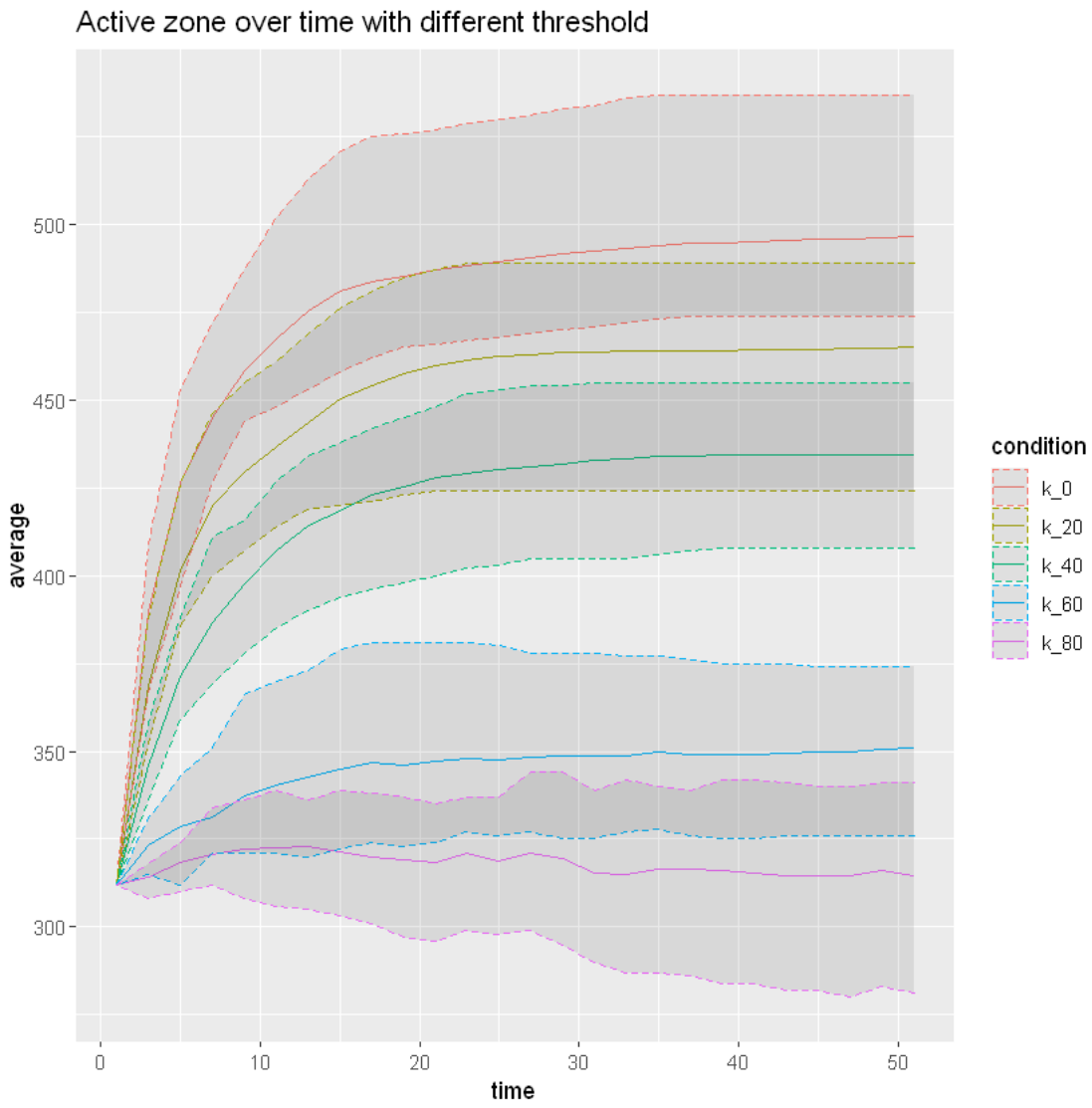
[55]:
```
# study the density
time <- 50
k_0_d <- data_experiment(k_0, 'density')
k_0_d$average_density <- rowMeans(k_0_d)
k_0_d$min <- apply(k_0_d, 1, FUN=min, na.rm=TRUE)
k_0_d$max <- apply(k_0_d, 1, FUN=max, na.rm=TRUE)
k_0_d$time <- c(0:((time)%/%2))*2 + 1
k_0_d$condition <- 'k_0' # put labels
k_20_d <- data_experiment(k_20, 'density')
k_20_d$average_density <- rowMeans(k_20_d)
k_20_d$min <- apply(k_20_d, 1, FUN=min, na.rm=TRUE)
k_20_d$max <- apply(k_20_d, 1, FUN=max, na.rm=TRUE)
k_20_d$time <- c(0:((time)%/%2))*2 + 1
k_20_d$condition <- 'k_20' # put labels
k_40_d <- data_experiment(k_40, 'density')
k_40_d$average_density <- rowMeans(k_40_d)
k_40_d$min <- apply(k_40_d, 1, FUN=min, na.rm=TRUE)
k_40_d$max <- apply(k_40_d, 1, FUN=max, na.rm=TRUE)
k_40_d$time <- c(0:((time)%/%2))*2 + 1
k_40_d$condition <- 'k_40' # put labels
k_60_d <- data_experiment(k_60, 'density')
k_60_d$average_density <- rowMeans(k_60_d)
k_60_d$min <- apply(k_60_d, 1, FUN=min, na.rm=TRUE)
k_60_d$max <- apply(k_60_d, 1, FUN=max, na.rm=TRUE)
k_60_d$time <- c(0:((time)%/%2))*2 + 1
k_60_d$condition <- 'k_60' # put labels
k_80_d <- data_experiment(k_80, 'density')
k_80_d$average_density <- rowMeans(k_80_d)
k_80_d$min <- apply(k_80_d, 1, FUN=min, na.rm=TRUE)
k_80_d$max <- apply(k_80_d, 1, FUN=max, na.rm=TRUE)
k_80_d$time <- c(0:((time)%/%2))*2 + 1
```

```r
k_80_d$condition <- 'k_80' # put labels



kdensities <- rbind(k_0_d, k_20_d, k_40_d, k_60_d, k_80_d)
ggplot(data=kdensities, aes(x=time, y=average_density, colour=condition))+␣
↪geom_line() + geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+␣
↪ggtitle("Density over time with different threshold")
```



A bigger threshold less density and a smaller threshold more density. This is because whith smaller threshold is more likely to the firefly change the state that with bigger thresholds.

[70]:
```
# study the active zone
time <- 50
k_0_d <- data_experiment(k_0, 'active_zone')
k_0_d$average <- rowMeans(k_0_d)
k_0_d$min <- apply(k_0_d, 1, FUN=min, na.rm=TRUE)
k_0_d$max <- apply(k_0_d, 1, FUN=max, na.rm=TRUE)
k_0_d$time <- c(0:((time)%/%2))*2 + 1
k_0_d$condition <- 'k_0' # put labels
k_20_d <- data_experiment(k_20, 'active_zone')
k_20_d$average <- rowMeans(k_20_d)
k_20_d$min <- apply(k_20_d, 1, FUN=min, na.rm=TRUE)
k_20_d$max <- apply(k_20_d, 1, FUN=max, na.rm=TRUE)
k_20_d$time <- c(0:((time)%/%2))*2 + 1
k_20_d$condition <- 'k_20' # put labels
k_40_d <- data_experiment(k_40, 'active_zone')
k_40_d$average<- rowMeans(k_40_d)
k_40_d$min <- apply(k_40_d, 1, FUN=min, na.rm=TRUE)
k_40_d$max <- apply(k_40_d, 1, FUN=max, na.rm=TRUE)
k_40_d$time <- c(0:((time)%/%2))*2 + 1
k_40_d$condition <- 'k_40' # put labels
k_60_d <- data_experiment(k_60, 'active_zone')
k_60_d$average <- rowMeans(k_60_d)
k_60_d$min <- apply(k_60_d, 1, FUN=min, na.rm=TRUE)
k_60_d$max <- apply(k_60_d, 1, FUN=max, na.rm=TRUE)
k_60_d$time <- c(0:((time)%/%2))*2 + 1
k_60_d$condition <- 'k_60' # put labels
k_80_d <- data_experiment(k_80, 'active_zone')
k_80_d$average <- rowMeans(k_80_d)
k_80_d$min <- apply(k_80_d, 1, FUN=min, na.rm=TRUE)
k_80_d$max <- apply(k_80_d, 1, FUN=max, na.rm=TRUE)
k_80_d$time <- c(0:((time)%/%2))*2 + 1
k_80_d$condition <- 'k_80' # put labels

kactivezone <- rbind(k_0_d, k_20_d, k_40_d, k_60_d, k_80_d)
ggplot(data=kactivezone, aes(x=time, y=average, colour=condition))+ geom_line()␣
↪+ geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+ ggtitle("Active␣
↪zone over time with different threshold")
```



The differents between bigger thresholds and smaller thresholds in the active zone is more visible, but again show the same behavior.

[58]:
```
# study the frequency

k_0_d <- data_experiment(k_0, 'frequency')
k_0_d$condition <- 'k_0' # put labels
k_20_d <- data_experiment(k_20, 'frequency')
k_20_d$condition <- 'k_20' # put labels
k_40_d <- data_experiment(k_40, 'frequency')
k_40_d$condition <- 'k_40' # put labels
k_60_d <- data_experiment(k_60, 'frequency')
k_60_d$condition <- 'k_60' # put labels
```

```
k_80_d <- data_experiment(k_80, 'frequency')
k_80_d$condition <- 'k_80' # put labels

kfrequencies<-  rbind(k_0_d, k_20_d, k_40_d, k_60_d, k_80_d)
kfrequencies
```

run_1	run_2	run_3	run_4	run_5	condition
1	1	1	1	1	k_0
1	1	1	1	1	k_20
1	1	1	1	1	k_40
1	1	1	1	1	k_60
1	1	1	1	1	k_80

Again the frequencies of activation are even.

[59]:
```
# study the average equilibrium time
k_0_d <- data_experiment(k_0, 'equilibrium_time')
k_0_d$condition <- 'k_0' # put labels
k_20_d <- data_experiment(k_20, 'equilibrium_time')
k_20_d$condition <- 'k_20' # put labels
k_40_d <- data_experiment(k_40, 'equilibrium_time')
k_40_d$condition <- 'k_40' # put labels
k_60_d <- data_experiment(k_60, 'equilibrium_time')
k_60_d$condition <- 'k_60' # put labels
k_80_d <- data_experiment(k_80, 'equilibrium_time')
k_80_d$condition <- 'k_80' # put labels


kequilibrium_time<- rbind(k_0_d, k_20_d, k_40_d, k_60_d, k_80_d)
rownames(kequilibrium_time) <- kequilibrium_time$condition
kequilibrium_time <- subset(kequilibrium_time, select = -c(condition))
kequilibrium_time$average_equilibrium_time <- rowMeans(kequilibrium_time)
kequilibrium_time
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_time
k_0	35	21	0	31	37	24.8
k_20	21	29	23	25	27	25.0
k_40	27	32	31	35	21	29.2
k_60	18	30	30	0	35	22.6
k_80	49	40	50	47	49	47.0

Here we can see very well that it is necesary at least 80% of threshold, or 0.8, to really affect the average time to reach the equilibrium.

[74]:
```
# study the average equilibrium size
k_20_d <- data_experiment(k_20, 'average_cluster_equilibrium')
k_20_d$condition <- 'k_20' # put labels
k_40_d <- data_experiment(k_40, 'average_cluster_equilibrium')
k_40_d$condition <- 'k_40' # put labels
```

```
k_80_d <- data_experiment(k_80, 'average_cluster_equilibrium')
k_80_d$condition <- 'k_80' # put labels


kequilibrium_size<- rbind( k_20_d, k_40_d, k_80_d)
rownames(kequilibrium_size) <- kequilibrium_size$condition
kequilibrium_size <- subset(kequilibrium_size, select = -c(condition))
kequilibrium_size$average_equilibrium_size <- rowMeans(kequilibrium_size)
kequilibrium_size
```

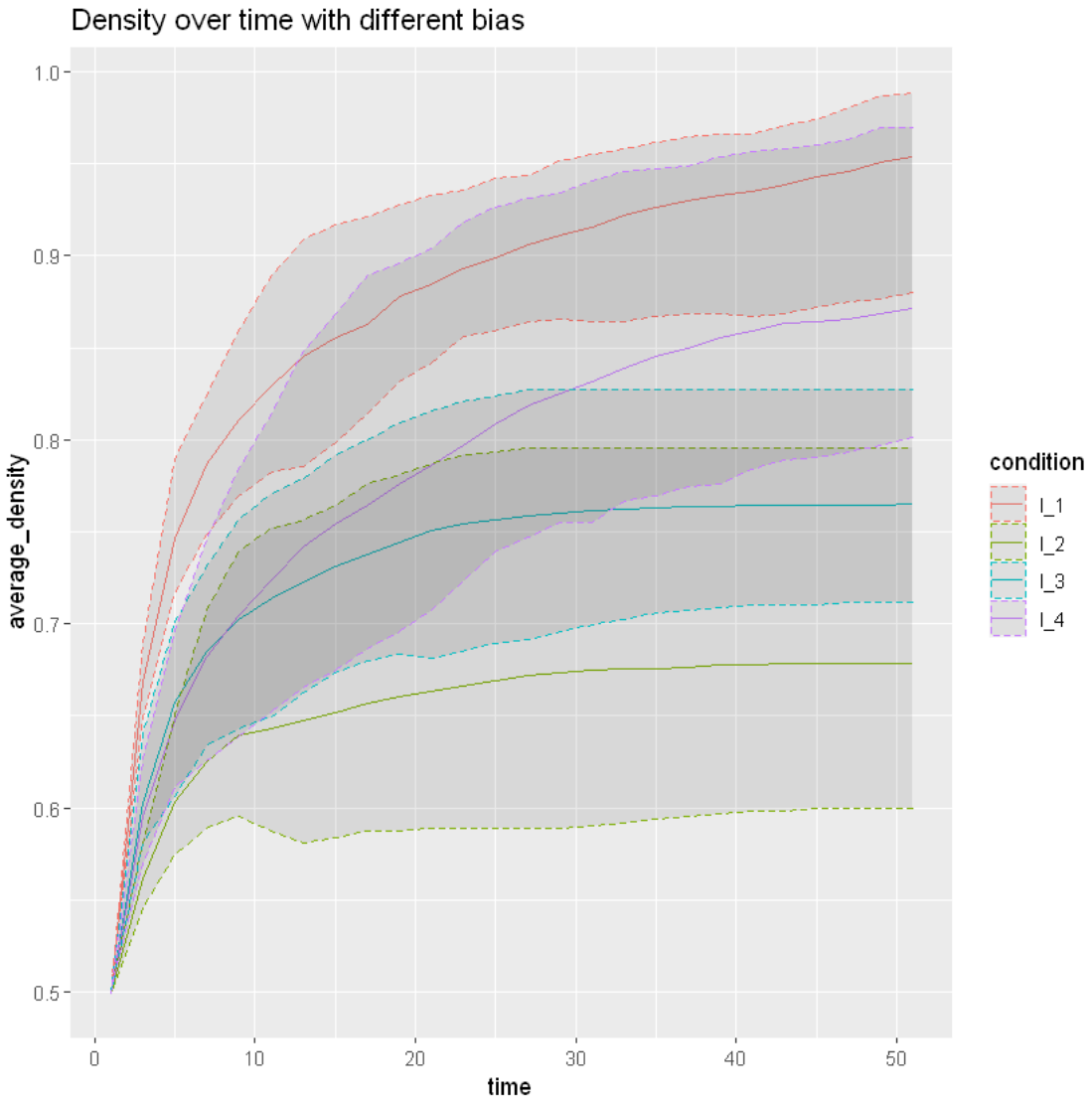
	run_1	run_2	run_3	run_4	run_5	average_equilibrium_size
k_20	212.0000	489.00000	244.5	449.00000	232.5000	325.40000
k_40	454.0000	55.00000	218.5	424.00000	223.5000	275.00000
k_80	113.6667	56.83333	101.0	42.14286	113.3333	85.39524

Here I need it to delete the date of threhold of 10% because the function did not found any cluster. It is interesting to see how with bigger thresholds, the clusters are very little, meaning that the equilibrium is reached with a less even synchronization between all the fireflies.

[89]:
```
# Different bias: inactive, active, direct_neighbours and bad_neighbours

number = 5
time = 50
shy_density = 0
n = 25
initial_density = 0.5
type_of_world = 'Normal'
th = 0.25

l_1 <- run_experiment(number, time, th, n, initial_density, shy_density ,␣
 ↪type_of_world, bias='inactive')
l_2 <- run_experiment(number, time, th, n, initial_density, shy_density ,␣
 ↪type_of_world, bias='active')
l_3 <- run_experiment(number, time, th, n, initial_density, shy_density ,␣
 ↪type_of_world, bias='direct_neighbours')
l_4 <- run_experiment(number, time, th, n, initial_density, shy_density ,␣
 ↪type_of_world, bias='bad_neighbours')
```

[90]:
```
# study the density
time <- 50
l_1_d <- data_experiment(l_1, 'density')
l_1_d$average_density <- rowMeans(l_1_d)
l_1_d$min <- apply(l_1_d, 1, FUN=min, na.rm=TRUE)
l_1_d$max <- apply(l_1_d, 1, FUN=max, na.rm=TRUE)
l_1_d$time <- c(0:((time)%/%2))*2 + 1
l_1_d$condition <- 'l_1' # put labels
l_2_d <- data_experiment(l_2, 'density')
```

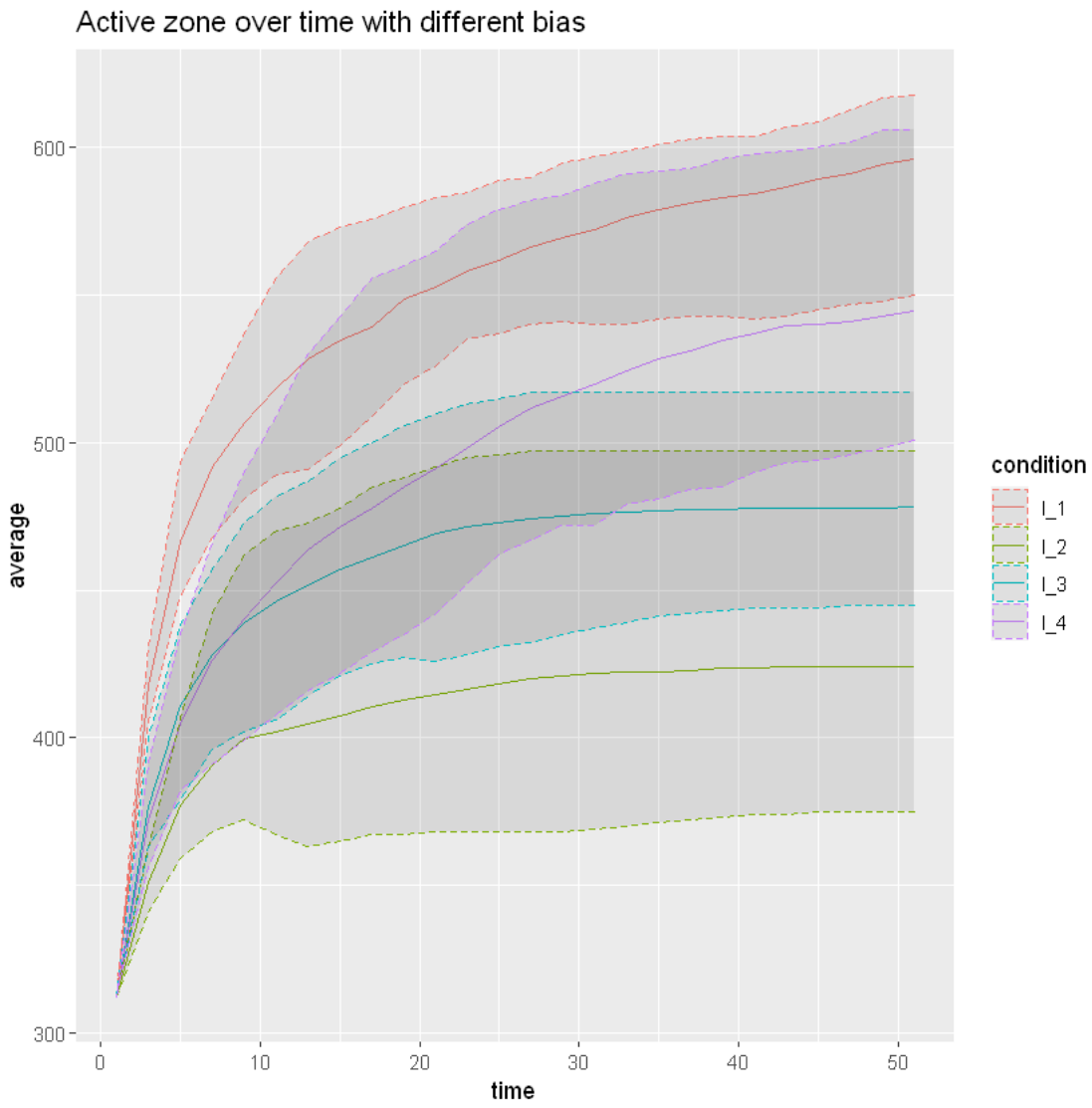
```
l_2_d$average_density <- rowMeans(l_2_d)
l_2_d$min <- apply(l_2_d, 1, FUN=min, na.rm=TRUE)
l_2_d$max <- apply(l_2_d, 1, FUN=max, na.rm=TRUE)
l_2_d$time <- c(0:((time)%/%2))*2 + 1
l_2_d$condition <- 'l_2' # put labels
l_3_d <- data_experiment(l_3 , 'density')
l_3_d$average_density <- rowMeans(l_3_d)
l_3_d$min <- apply(l_3_d, 1, FUN=min, na.rm=TRUE)
l_3_d$max <- apply(l_3_d, 1, FUN=max, na.rm=TRUE)
l_3_d$time <- c(0:((time)%/%2))*2 + 1
l_3_d$condition <- 'l_3' # put labels
l_4_d <- data_experiment(l_4, 'density')
l_4_d$average_density <- rowMeans(l_4_d)
l_4_d$min <- apply(l_4_d, 1, FUN=min, na.rm=TRUE)
l_4_d$max <- apply(l_4_d, 1, FUN=max, na.rm=TRUE)
l_4_d$time <- c(0:((time)%/%2))*2 + 1
l_4_d$condition <- 'l_4' # put labels

gdensities <- rbind(l_1_d, l_2_d, l_3_d, l_4_d)
ggplot(data=gdensities, aes(x=time, y=average_density, colour=condition))+␣
 ↪geom_line() + geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+␣
 ↪ggtitle("Density over time with different bias ")
```



Here we can see how affect the different bias I configured to the density over time. I found interesting that the bias of bad neighbours and the bias of inactive, give you less stability over the time in the density, that make me think that perhaps the equilibrium time is not reached here. We will see it. Obviously, the bias affect as expected, and with less neighbours in the consensus the density is bigger because the rule depends in less fireflies. This can be related to the small world explained in the lesson 5 and I found really interesting being able to reproduce here.

[91]:
```
# study the active zone
time <- 50
l_1_d <- data_experiment(l_1, 'active_zone')
l_1_d$average <- rowMeans(l_1_d)
l_1_d$min <- apply(l_1_d, 1, FUN=min, na.rm=TRUE)
l_1_d$max <- apply(l_1_d, 1, FUN=max, na.rm=TRUE)
```

```
l_1_d$time <- c(0:((time)%/%2))*2 + 1
l_1_d$condition <- 'l_1' # put labels
l_2_d <- data_experiment(l_2, 'active_zone')
l_2_d$average <- rowMeans(l_2_d)
l_2_d$min <- apply(l_2_d, 1, FUN=min, na.rm=TRUE)
l_2_d$max <- apply(l_2_d, 1, FUN=max, na.rm=TRUE)
l_2_d$time <- c(0:((time)%/%2))*2 + 1
l_2_d$condition <- 'l_2' # put labels
l_3_d <- data_experiment(l_3 , 'active_zone')
l_3_d$average <- rowMeans(l_3_d)
l_3_d$min <- apply(l_3_d, 1, FUN=min, na.rm=TRUE)
l_3_d$max <- apply(l_3_d, 1, FUN=max, na.rm=TRUE)
l_3_d$time <- c(0:((time)%/%2))*2 + 1
l_3_d$condition <- 'l_3' # put labels
l_4_d <- data_experiment(l_4, 'active_zone')
l_4_d$average <- rowMeans(l_4_d)
l_4_d$min <- apply(l_4_d, 1, FUN=min, na.rm=TRUE)
l_4_d$max <- apply(l_4_d, 1, FUN=max, na.rm=TRUE)
l_4_d$time <- c(0:((time)%/%2))*2 + 1
l_4_d$condition <- 'l_4' # put labels

gactive_zone<- rbind(l_1_d, l_2_d, l_3_d, l_4_d)
ggplot(data=gactive_zone, aes(x=time, y=average, colour=condition))+ geom_line()␣
 ↪+ geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+ ggtitle("Active␣
 ↪zone over time with different bias")
```



The results for active zone are consintent taking in account the results of densities.

[92]:
```
# study the frequency

l_1_d <- data_experiment(l_1, 'frequency')
l_1_d$condition <- 'l_1' # put labels
l_2_d <- data_experiment(l_2, 'frequency')
l_2_d$condition <- 'l_2' # put labels
l_3_d <- data_experiment(l_3, 'frequency')
l_3_d$condition <- 'l_3' # put labels
l_4_d <- data_experiment(l_4, 'frequency')
l_4_d$condition <- 'l_4' # put labels
```

```
ffrequencies<- rbind(l_1_d, l_2_d, l_3_d, l_4_d)
ffrequencies
```

run_1	run_2	run_3	run_4	run_5	condition
1	1	1	1	1	l_1
1	1	1	1	1	l_2
1	1	1	1	1	l_3
1	1	1	1	1	l_4

Again the frequency of activation is even.

[95]:
```
# study the average equilibrium time

#l_1_d <- data_experiment(l_1_d, 'equilibrium_time')
#l_1_d$condition <- 'l_1' # put labels
# never reached the equilibrium
l_2_d <- data_experiment(l_2, 'equilibrium_time')
l_2_d$condition <- 'l_2' # put labels
l_3_d <- data_experiment(l_3, 'equilibrium_time')
l_3_d$condition <- 'l_3' # put labels
l_4_d <- data_experiment(l_4, 'equilibrium_time')
l_4_d$condition <- 'l_4' # put labels

lequilibrium_time<- rbind(l_2_d, l_3_d, l_4_d)
rownames(lequilibrium_time) <- lequilibrium_time$condition
lequilibrium_time <- subset(lequilibrium_time, select = -c(condition))
lequilibrium_time$average_equilibrium_time <- rowMeans(lequilibrium_time)
lequilibrium_time
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_time
l_2	18	39	29	27	41	30.8
l_3	41	27	25	33	27	30.6
l_4	35	0	0	0	36	14.2

Firs interesting remark is that the equilibrium was never reached in any of the runs for the bias of inactive, and the equilibrium was reached much quicker for the bias of bad neighbours, because again, removing neighbours from the consensus helps to get synchronize.

[100]:
```
# study the average equilibrium size
#l_1_d <- data_experiment(l_1, 'average_cluster_equilibrium')
#l_1_d$condition <- 'l_1' # put labels
#Never reached the equilibrium
l_2_d <- data_experiment(l_2, 'average_cluster_equilibrium')
l_2_d$condition <- 'l_2' # put labels
l_3_d <- data_experiment(l_3, 'average_cluster_equilibrium')
l_3_d$condition <- 'l_3' # put labels
```

```
lequilibrium_size<- rbind( l_2_d, l_3_d)
rownames(lequilibrium_size) <- lequilibrium_size$condition
lequilibrium_size <- subset(lequilibrium_size, select = -c(condition))
lequilibrium_size$average_equilibrium_size <- rowMeans(lequilibrium_size)
lequilibrium_size
```

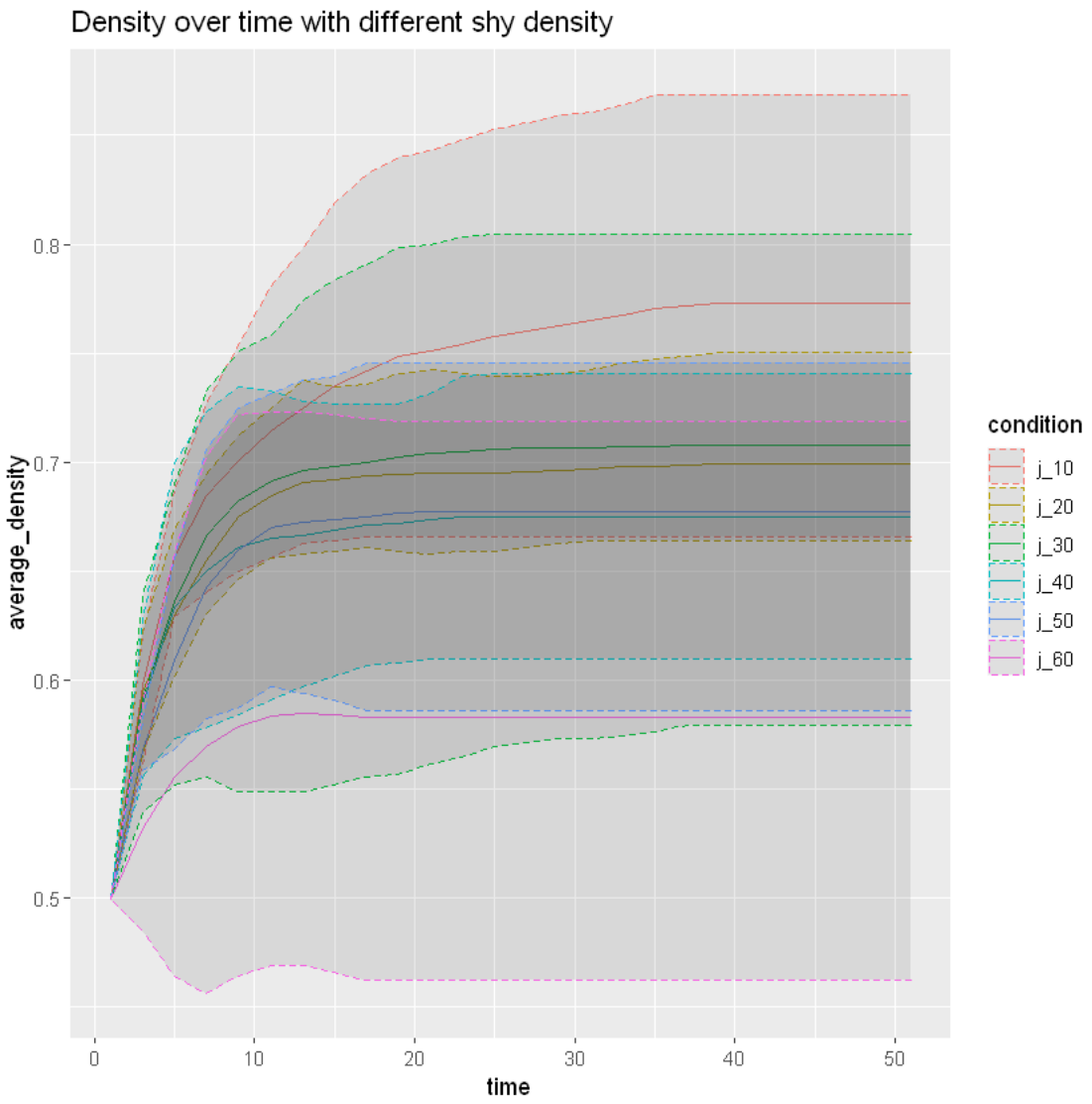
	run_1	run_2	run_3	run_4	run_5	average_equilibrium_size
l_2	64.5	438	140.3333	165.6667	194.5	200.6
l_3	444.0	222	512.0000	470.0000	517.0	433.0

There weren't clusters found for the bias of bad neighbours and in the case of bias inactive the equilibrium was not reached. For the bias of bad neughbours, the equilibrium matrix is more caotic without any real cluster, but still in reaching a stability. In the case of the direct neighbours (were we consider the direct neighbours more important for take the decision that the neighbours of the diagonal), the cluster size in equilibrium is bigger in average. I found this really interesting, maybe could be related with sociology, were there are people that have more influence than anothers, and apparently this help to get bigger amount of fireflies synchronize together.

[64]:
```
# Shy density 10%, 20%, 30%, 40%, 50%, 60%

number = 5
time = 50
th = 0.25
n = 25
initial_density = 0.5
type_of_world = 'Normal'

j_10 <- run_experiment(number, time, th, n, initial_density, shy_density=0.1 ,␣
 ↪type_of_world)
j_20 <- run_experiment(number, time, th, n, initial_density, shy_density=0.2 ,␣
 ↪type_of_world)
j_30 <- run_experiment(number, time, th, n, initial_density, shy_density=0.3 ,␣
 ↪type_of_world)
j_40 <- run_experiment(number, time, th, n, initial_density, shy_density=0.4 ,␣
 ↪type_of_world)
j_50 <- run_experiment(number, time, th, n, initial_density, shy_density=0.5 ,␣
 ↪type_of_world)
j_60 <- run_experiment(number, time, th, n, initial_density, shy_density=0.6 ,␣
 ↪type_of_world)
```

[65]:
```
# study the density
time <- 50
j_10_d <- data_experiment(j_10, 'density')
j_10_d$average_density <- rowMeans(j_10_d)
j_10_d$min <- apply(j_10_d, 1, FUN=min, na.rm=TRUE)
j_10_d$max <- apply(j_10_d, 1, FUN=max, na.rm=TRUE)
j_10_d$time <- c(0:((time)%/%2))*2 + 1
```

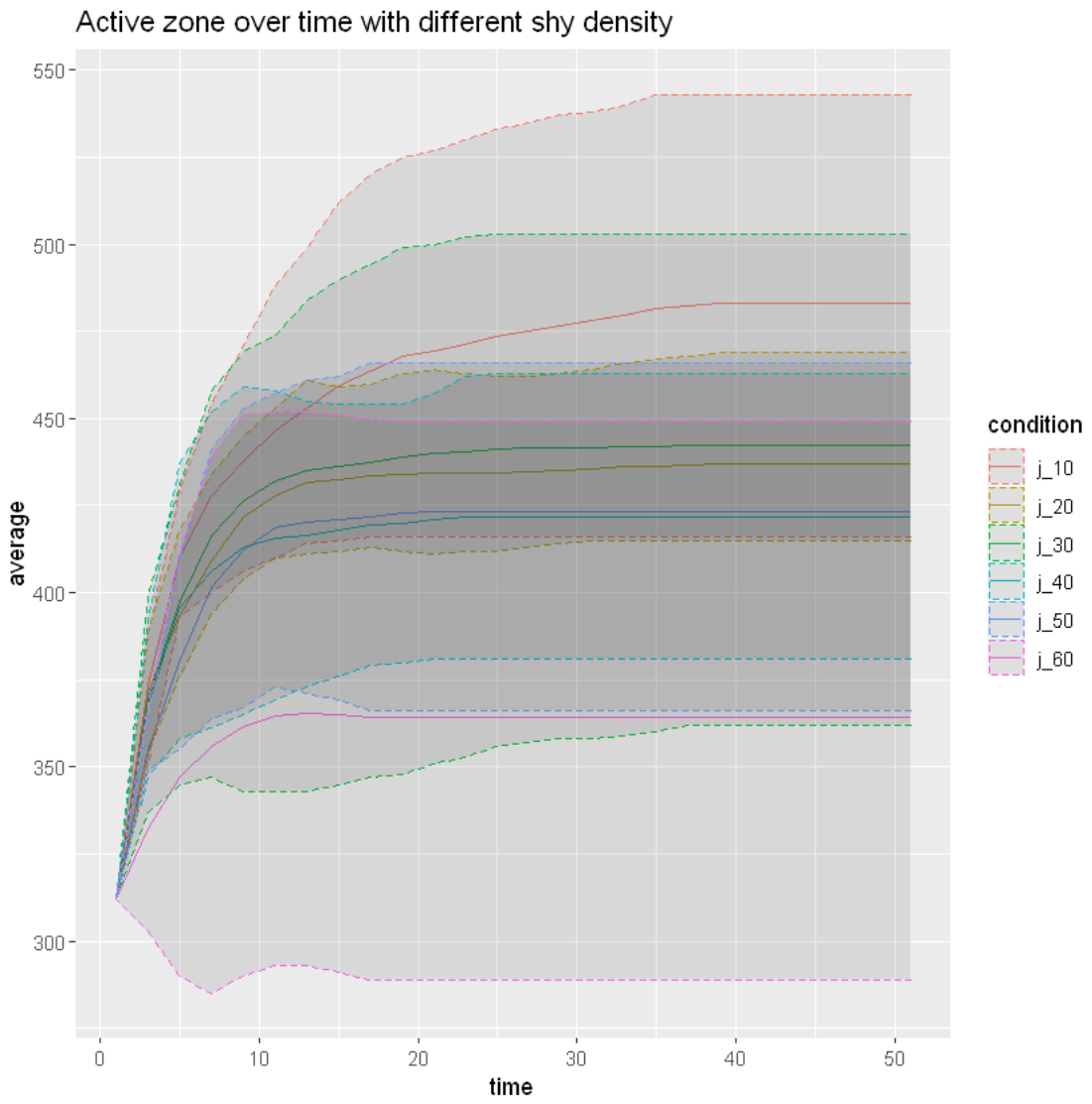
```
j_10_d$condition <- 'j_10' # put labels
j_20_d <- data_experiment(j_20, 'density')
j_20_d$average_density <- rowMeans(j_20_d)
j_20_d$min <- apply(j_20_d, 1, FUN=min, na.rm=TRUE)
j_20_d$max <- apply(j_20_d, 1, FUN=max, na.rm=TRUE)
j_20_d$time <- c(0:((time)%/%2))*2 + 1
j_20_d$condition <- 'j_20' # put labels
j_30_d <- data_experiment(j_30 , 'density')
j_30_d$average_density <- rowMeans(j_30_d)
j_30_d$min <- apply(j_30_d, 1, FUN=min, na.rm=TRUE)
j_30_d$max <- apply(j_30_d, 1, FUN=max, na.rm=TRUE)
j_30_d$time <- c(0:((time)%/%2))*2 + 1
j_30_d$condition <- 'j_30' # put labels
j_40_d <- data_experiment(j_40, 'density')
j_40_d$average_density <- rowMeans(j_40_d)
j_40_d$min <- apply(j_40_d, 1, FUN=min, na.rm=TRUE)
j_40_d$max <- apply(j_40_d, 1, FUN=max, na.rm=TRUE)
j_40_d$time <- c(0:((time)%/%2))*2 + 1
j_40_d$condition <- 'j_40' # put labels
j_50_d <- data_experiment(j_50, 'density')
j_50_d$average_density <- rowMeans(j_50_d)
j_50_d$min <- apply(j_50_d, 1, FUN=min, na.rm=TRUE)
j_50_d$max <- apply(j_50_d, 1, FUN=max, na.rm=TRUE)
j_50_d$time <- c(0:((time)%/%2))*2 + 1
j_50_d$condition <- 'j_50' # put labels
j_60_d <- data_experiment(j_60, 'density')
j_60_d$average_density <- rowMeans(j_60_d)
j_60_d$min <- apply(j_60_d, 1, FUN=min, na.rm=TRUE)
j_60_d$max <- apply(j_60_d, 1, FUN=max, na.rm=TRUE)
j_60_d$time <- c(0:((time)%/%2))*2 + 1
j_60_d$condition <- 'j_60' # put labels

jdensities <- rbind(j_10_d, j_20_d, j_30_d, j_40_d, j_50_d, j_60_d)
ggplot(data=jdensities, aes(x=time, y=average_density, colour=condition))+␣
 ↪geom_line() + geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+␣
 ↪ggtitle("Density over time with different shy density")
```



The final experiment is to see how affect the shy density or in other words, fireflies that independent of being active or inactive, are not seeing by the others and are not taking in account for the consensus. With more amount of shy fireflies, the densities are lower.

[69]:
```
# study the active zone
time <- 50
j_10_d <- data_experiment(j_10, 'active_zone')
j_10_d$average <- rowMeans(j_10_d)
j_10_d$min <- apply(j_10_d, 1, FUN=min, na.rm=TRUE)
j_10_d$max <- apply(j_10_d, 1, FUN=max, na.rm=TRUE)
j_10_d$time <- c(0:((time)%/%2))*2 + 1
j_10_d$condition <- 'j_10' # put labels
j_20_d <- data_experiment(j_20, 'active_zone')
```

```
j_20_d$average <- rowMeans(j_20_d)
j_20_d$min <- apply(j_20_d, 1, FUN=min, na.rm=TRUE)
j_20_d$max <- apply(j_20_d, 1, FUN=max, na.rm=TRUE)
j_20_d$time <- c(0:((time)%/%2))*2 + 1
j_20_d$condition <- 'j_20' # put labels
j_30_d <- data_experiment(j_30 , 'active_zone')
j_30_d$average <- rowMeans(j_30_d)
j_30_d$min <- apply(j_30_d, 1, FUN=min, na.rm=TRUE)
j_30_d$max <- apply(j_30_d, 1, FUN=max, na.rm=TRUE)
j_30_d$time <- c(0:((time)%/%2))*2 + 1
j_30_d$condition <- 'j_30' # put labels
j_40_d <- data_experiment(j_40, 'active_zone')
j_40_d$average <- rowMeans(j_40_d)
j_40_d$min <- apply(j_40_d, 1, FUN=min, na.rm=TRUE)
j_40_d$max <- apply(j_40_d, 1, FUN=max, na.rm=TRUE)
j_40_d$time <- c(0:((time)%/%2))*2 + 1
j_40_d$condition <- 'j_40' # put labels
j_50_d <- data_experiment(j_50, 'active_zone')
j_50_d$average <- rowMeans(j_50_d)
j_50_d$min <- apply(j_50_d, 1, FUN=min, na.rm=TRUE)
j_50_d$max <- apply(j_50_d, 1, FUN=max, na.rm=TRUE)
j_50_d$time <- c(0:((time)%/%2))*2 + 1
j_50_d$condition <- 'j_50' # put labels
j_60_d <- data_experiment(j_60, 'active_zone')
j_60_d$average <- rowMeans(j_60_d)
j_60_d$min <- apply(j_60_d, 1, FUN=min, na.rm=TRUE)
j_60_d$max <- apply(j_60_d, 1, FUN=max, na.rm=TRUE)
j_60_d$time <- c(0:((time)%/%2))*2 + 1
j_60_d$condition <- 'j_60' # put labels

jactive_zone<- rbind(j_10_d, j_20_d, j_30_d, j_40_d, j_50_d, j_60_d)
ggplot(data=jactive_zone, aes(x=time, y=average, colour=condition))+ geom_line()␣
 ↪+ geom_ribbon(aes(ymin=min, ymax=max), linetype=2, alpha=0.1)+ ggtitle("Active␣
 ↪zone over time with different shy density")
```



Here we can see how the results are consitent with the densities.

[66]:
```
# study the frequency

j_10_d <- data_experiment(j_10, 'frequency')
j_10_d$condition <- 'j_10' # put labels
j_20_d <- data_experiment(j_20, 'frequency')
j_20_d$condition <- 'j_20' # put labels
j_30_d <- data_experiment(j_30, 'frequency')
j_30_d$condition <- 'j_30' # put labels
j_40_d <- data_experiment(j_40, 'frequency')
j_40_d$condition <- 'j_40' # put labels
j_50_d <- data_experiment(j_50, 'frequency')
```

```
j_50_d$condition <- 'j_50' # put labels
j_60_d <- data_experiment(j_60, 'frequency')
j_60_d$condition <- 'j_60' # put labels

jfrequencies<- rbind(j_10_d, j_20_d, j_30_d, j_40_d, j_50_d, j_60_d)
jfrequencies
```

run_1	run_2	run_3	run_4	run_5	condition
1	1	1	1	1	j_10
1	1	1	1	1	j_20
1	1	1	1	1	j_30
1	1	1	1	1	j_40
1	1	1	1	1	j_50
1	1	1	1	1	j_60

The frequencies of activation are still even. This make me think that this model it is not appropiate to study the frequency of activation, at wasn't the velocity.

[67]:
```
# study the average equilibrium time

j_10_d <- data_experiment(j_10, 'equilibrium_time')
j_10_d$condition <- 'j_10_d' # put labels
j_20_d <- data_experiment(j_20, 'equilibrium_time')
j_20_d$condition <- 'j_20' # put labels
j_30_d <- data_experiment(j_30, 'equilibrium_time')
j_30_d$condition <- 'j_30' # put labels
j_40_d <- data_experiment(j_40, 'equilibrium_time')
j_40_d$condition <- 'j_40' # put labels
j_50_d <- data_experiment(j_50, 'equilibrium_time')
j_50_d$condition <- 'j_50' # put labels
j_60_d <- data_experiment(j_60, 'equilibrium_time')
j_60_d$condition <- 'j_60' # put labels

jequilibrium_time<- rbind(j_10_d, j_20_d, j_30_d, j_40_d, j_50_d, j_60_d)
rownames(jequilibrium_time) <- jequilibrium_time$condition
jequilibrium_time <- subset(jequilibrium_time, select = -c(condition))
jequilibrium_time$average_equilibrium_time <- rowMeans(jequilibrium_time)
jequilibrium_time
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_time
j_10_d	35	39	31	17	35	31.4
j_20	17	39	20	27	23	25.2
j_30	21	29	25	18	13	21.2
j_40	15	26	16	21	25	20.6
j_50	21	17	14	18	18	17.6
j_60	15	15	18	13	12	14.6

The average equilibrium time is more or less in the same range of values for all the different set ups, but it is true that with very big amount of fireflies the time decreased. This is again related to

the number of fireflies that are important for the consensus.

[68]:
```
# study the average equilibrium size

j_10_d <- data_experiment(j_10, 'average_cluster_equilibrium')
j_10_d$condition <- 'j_10' # put labels
j_20_d <- data_experiment(j_20, 'average_cluster_equilibrium')
j_20_d$condition <- 'j_20' # put labels
j_30_d <- data_experiment(j_30, 'average_cluster_equilibrium')
j_30_d$condition <- 'j_30' # put labels
j_40_d <- data_experiment(j_40, 'average_cluster_equilibrium')
j_40_d$condition <- 'j_40' # put labels
j_50_d <- data_experiment(j_50, 'average_cluster_equilibrium')
j_50_d$condition <- 'j_50' # put labels
j_60_d <- data_experiment(j_60, 'average_cluster_equilibrium')
j_60_d$condition <- 'j_60' # put labels

jequilibrium_size<- rbind(j_10_d, j_20_d, j_30_d, j_40_d, j_50_d, j_60_d)
rownames(jequilibrium_size) <- jequilibrium_size$condition
jequilibrium_size <- subset(jequilibrium_size, select = -c(condition))
jequilibrium_size$average_equilibrium_size <- rowMeans(jequilibrium_size)
jequilibrium_size
```

	run_1	run_2	run_3	run_4	run_5	average_equilibrium_size
j_10	502.00	230.5	493.00000	208.00000	543.00000	395.3000
j_20	428.00	469.0	30.00000	455.00000	412.00000	358.8000
j_30	224.00	179.0	503.00000	28.42857	476.00000	282.0857
j_40	97.25	25.5	21.37500	190.50000	463.00000	159.5250
j_50	447.00	466.0	29.85714	32.37500	28.71429	200.7893
j_60	130.00	379.0	84.00000	52.50000	24.71429	134.0429

Finally we can see how the average equilibrium size it is very diverse. But generally we can see how the lower densities get bigger clusters in the equilibrium.

As conclusion, I discussed the different results to see how the equilibrium time and the size of clusters taking in account all the different parameters I think were interesting. But the more important ones, or at least the parameters that make biggest difference in the state of equilibrium, was the different bias applied and threshold.