In [2]:
```
library(tidyverse)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse
 1.3.0 ──

✔ ggplot2 3.2.1     ✔ purrr   0.3.3
✔ tibble  2.1.3     ✔ dplyr   0.8.3
✔ tidyr   1.0.0     ✔ stringr 1.4.0
✔ readr   1.3.1     ✔ forcats 0.4.0

── Conflicts ────────────────────────────────────────── tidyverse_conflic
ts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

In [3]:
```
n <- 400 # population size
N <- 25 # grid is N x N
initial_size <- 20 # initial no. of infected people

rtime <- 10 # recovery time
```

In [4]:
```
init_state <- function(n, N, initial_size, c_rate) {


    state <- tibble(

        compliant = as.logical(rbinom(n,1,c_rate)),

        status = rep("H",times=n),


        xpos = sample(1:N,n,replace=TRUE),


        ypos = sample(1:N,n,replace=TRUE),

        infected = NA,


        recovered = NA
    )


    first_infections <- sample(1:n,initial_size)


    state$status <-
    ifelse(as.numeric(rownames(state))%in% first_infections,"I","H")


    state$infected <- now

    return(state)
}
```

In [5]:
```
plot_state <- function() {
    ggplot(state) +
        geom_jitter(aes(xpos, ypos, color = status), size = 4) + # jitter t
        scale_colour_manual(values = c("H" = "blue", "I" = "red", "R" = "gr
}
```

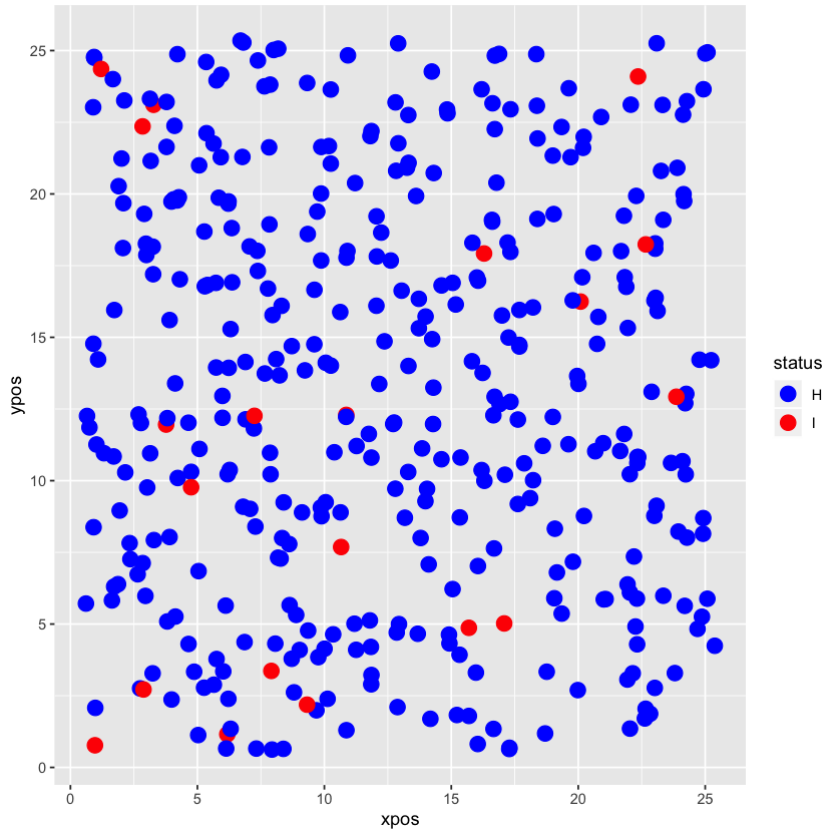
In [6]:
```
plot_history <- function() {
    ggplot(history) +
        geom_line(aes(day, count, color = status), size = 2) +
        scale_colour_manual(values = c("H" = "blue", "I" = "red", "R" = "gr
        facet_wrap(~ c_rate)
}
```

In [7]:
```
clip<-function(x){
    new=case_when(
        x<1~1,
        x>N~N,
        TRUE~x
    )
    return(new)
}
```

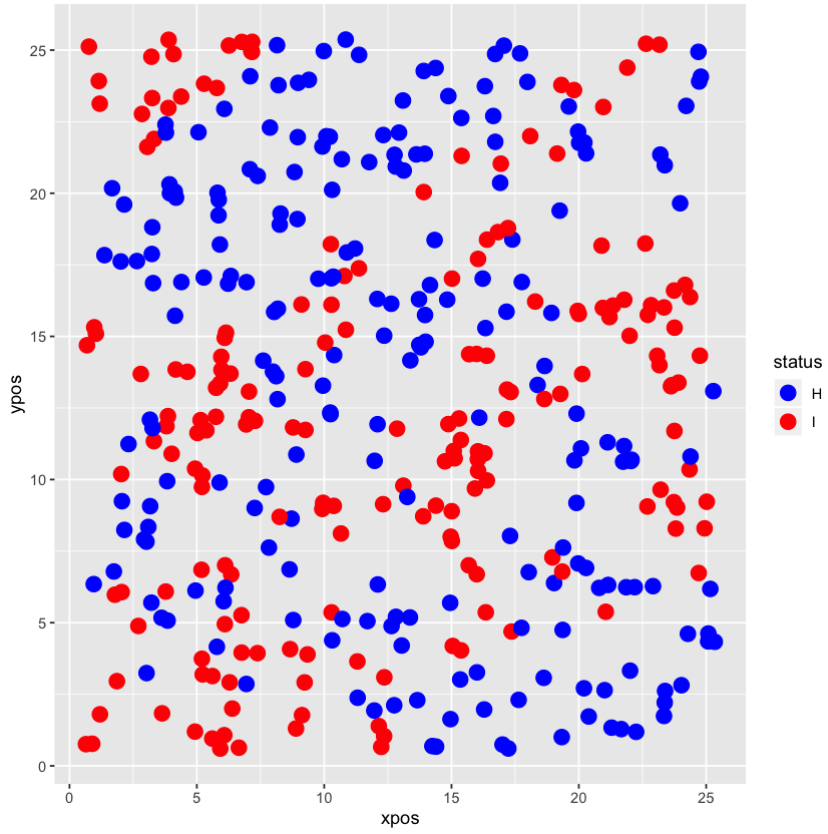
In [8]:
```
move_people <-function(state){
    for(i in 1:nrow(state)){
        if (state$compliant[i] == FALSE){
            move_x=sample(c(-1,1),1)
            move_y=sample(c(-1,1),1)
            state$xpos[i]=clip(state$xpos[i]+move_x)
            state$ypos[i]=clip(state$ypos[i]+move_y)
        }
    }
    return(state)
}
```

In [9]:
```
update_status <- function(state) {

    # see if individuals have recovered
    condition1 <- state$status == "I" & now - state$infected > rtime
    state$status <- ifelse(condition1, "R", state$status)
    state$recovered <- ifelse(condition1, now, state$recovered)


    for (i in 1:n) {

        ix <- state$xpos[i]
        iy <- state$ypos[i]

        # if individual i is infected, they infect every healthy individual
        if (state$status[i] == "I") {
            condition2 <- state$xpos == ix & state$ypos == iy & state$statu
            state$status <- ifelse(condition2, "I", state$status)
            state$infected <- ifelse(condition2, now, state$infected)
        }

    }

    return(state)
}
```

In [10]:
```
now <- 0
state <- init_state(n, N, initial_size, c_rate=0.5)
plot_state()
```



In [11]:
```
# run the simulation for 10 steps
for (k in 1:10) {
        now <- now + 1
        state <- move_people(state)
        state <- update_status(state)
}

# then visualize the state again
plot_state()
```



In [12]:
```
history <- tibble(day = integer(), status = character(), count = integer(),


T <- 50 # length of simulation

for (c_rate in seq(0.3,0.9,by=0.2)) {

    now <- 0 # current time
    state <- init_state(n, N, initial_size, c_rate)

    for (k in 1:T) {
        now <- now + 1
        state <- move_people(state)
        state <- update_status(state)

        history <- add_row(history, day = now, status = "H", count = sum(st
        history <- add_row(history, day = now, status = "I", count = sum(st
        history <- add_row(history, day = now, status = "R", count = sum(st
    }
}
```

In [13]:
```
plot_history()
```