Simulation experiment

22 October 2019

Define baseline intensity `Z` for the endemic component and an outbreak area `win`.

```
library(spatstat)
```

```
## Loading required package: spatstat.data
## Loading required package: spatstat.geom
## spatstat.geom 2.3-0
## Loading required package: spatstat.core
## Loading required package: nlme
## Loading required package: rpart
## spatstat.core 2.3-2
## Loading required package: spatstat.linnet
## spatstat.linnet 2.3-0
##
## spatstat 2.2-0       (nickname: 'That's not important right now')
## For an introduction to spatstat, type 'beginner'
```

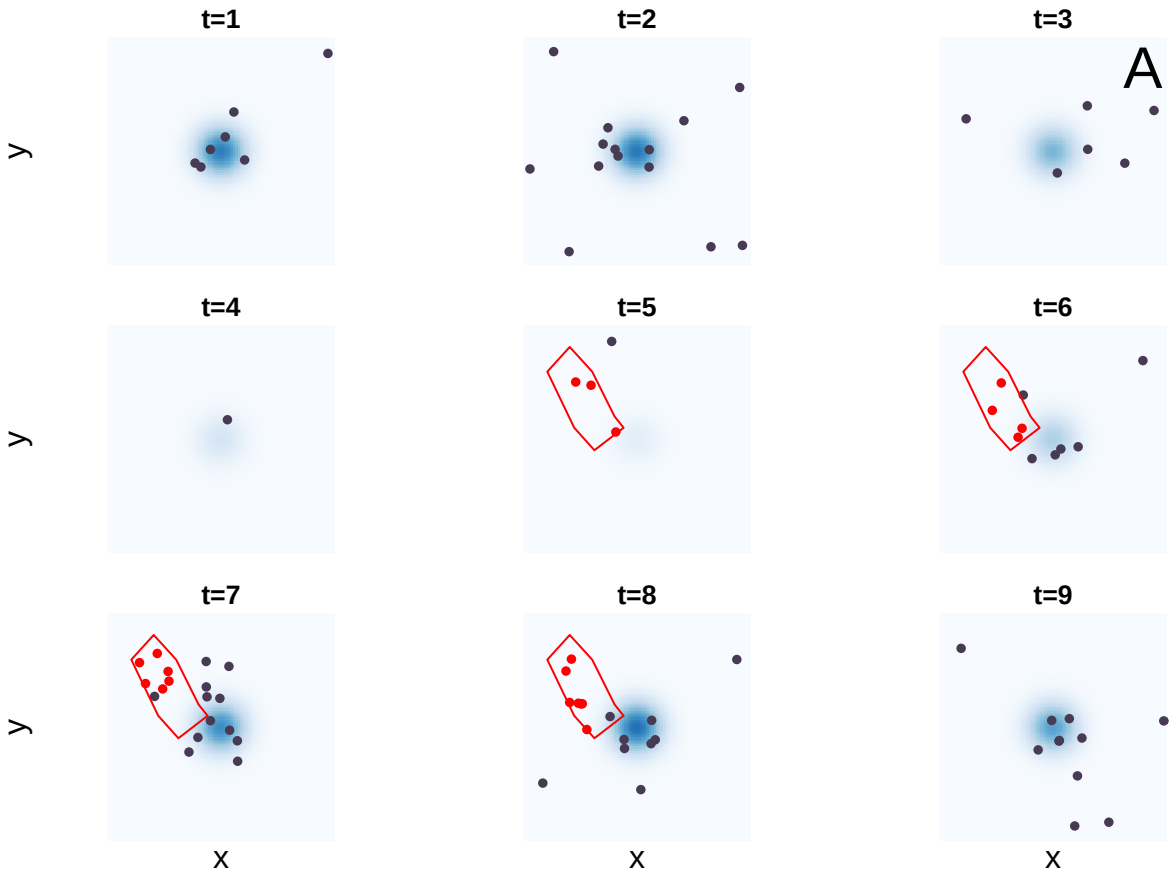
```
temporal.trend.function<-function(t,A=7, B=0.9){
  return (A/2*(sin(t) + 1) + B)
}

spatial.trend.function<-function(x,y, C=80, B=0.5, x0=0.5, y0=0.5){
  return(B + 20*exp(-C * ( (x-x0)^2 + (y - y0)^2)))
}
n.weeks = 20

delta = 0.01
delta2 = delta ^ 2
xx = seq(0,1,delta)
yy= seq(0,1,delta)

Z = lapply(1:n.weeks, function(t)
  outer(xx, yy, function(x,y){temporal.trend.function(t) * spatial.trend.function(x,y)}))

win = owin(poly = data.frame(x=rev(c(0.1,0.2,0.3,0.4,0.44,0.31,0.22)),
                             y=rev(c(0.8,0.91,0.8,0.6,0.55,0.45,0.55))) )

my_blues = c("#F7FBFF", "#DEEBF7", "#C6DBEF", "#9ECAE1", "#6BAED6","#4292C6","#2171B5")
#blues9
ColorRamp=colorRampPalette(c(my_blues))(1000)
```

Generate point events for the baseline and the outbreak with `rpoispp`.

```
set.seed(1)
baseline=lapply(1:n.weeks, function(t){
  rpoispp(function(x,y){ temporal.trend.function(t) * spatial.trend.function(x,y) })})
set.seed(1)
epidemics=lapply(c(0,0,0,0,1,1,1,1,0,0, 0,0,0,0,1,1,1,1,0,0), function(t){
  rpoispp(t * 50, win=win)})

Max = max(unlist(lapply(Z, max)))
```

Plot baseline intensity and point events for each time `t`.

```
# include=FALSE}
# png('../Manuscript/data_points__.png', width = 3.25 *2, height = 3.25 *2, units = 'in', res=600, poin

par(mfrow=c(3,3), mar=c(0.9, 0.9, 0.9, 0.5))
for (i in 1:9){
  ex = max(Z[[i]])/ Max
  ColorRamp_ex = ColorRamp[1:(ex * length(ColorRamp))]

  plot(baseline[[i]], cols='#473B54', pch=16, main=paste0('t=', as.character(i)), show.window = F)
  image(xx, yy, Z[[i]], add=T, col=ColorRamp_ex)
  plot(baseline[[i]], cols='#473B54', pch=16, add=T)
  if ((i == 7) | (i == 8) | (i == 9))
    mtext(c('x' ), c(1), outer = F)
  if ((i == 1) | (i == 4) | (i == 7))
    mtext(c('y' ), c(2), outer = F)
  if (i == 3)
    mtext('A', at = 0.9, cex = 1.8, padj = 1.5)


  if (i %in% 5:8){
    plot(win, add=T, border='red')
  }
  plot.ppp(epidemics[[i]], cols='red', pch=16, add=T)
}
```




```
# dev.off()
#palette = colorRampPalette(c('blue', 'red'))(max(cases2[,2]))
```

Aggregate all observations in a `list` data structure.

```
writeLines(c("The number of observations is:", nrow(observations)))
```

```
## The number of observations is:
## 159
```

Define function to draw covering cylinders and compute the exceedance statistics for each cylinder.

```
# x = seq(0,1,delta)
# y = seq(0,1,delta)
# z = lapply(1:10,
#            function(i){outer(x, y, function(x,y){temporal.trend(i) * spatial.trend.function(x,y)})}


compute_cylinders<-function(cylinder, observations, tabulated.baseline){
  t.low = as.numeric(cylinder['t.low'])
  t.upp = as.numeric(cylinder['t.upp'])
  x0 = as.numeric(cylinder['x'])
  y0 = as.numeric(cylinder['y'])
  rho = as.numeric(cylinder['rho'])
  d = sqrt((tabulated.baseline$x-x0)^2 + (tabulated.baseline$y-y0)^2)
  in_circle = (d < rho) & (!is.na(d))
  in_height = (tabulated.baseline$t >= t.low) & (tabulated.baseline$t <= t.upp)
  mu = sum(tabulated.baseline[in_circle & in_height,]$z) * delta2 #multiply by delta2 as this is the bi
```

```
#   in_of_square = sum(in_circle & in_height) * delta2
# #  out_of_square = pi * rho* rho - in_of_square
#   mu = mu * pi * rho *rho / in_of_square

  d = sqrt((observations$x-x0)^2 + (observations$y-y0)^2)
  in_circle = (d < rho) & (!is.na(d))
  in_height = (observations$t >= t.low) & (observations$t <= t.upp)
  n_cases_in_cylinder = sum(in_circle & in_height)

 # ci = qpois(c(0.25,0.95) , lambda=mu)
  p.val = ppois(n_cases_in_cylinder, lambda=mu, lower.tail=FALSE)
  return (c(n_cases_in_cylinder, mu, p.val))
}
```

Check timeliness

```
tabulated.baseline = expand.grid(xx,yy,1:n.weeks)
names(tabulated.baseline) = c('x', 'y', 't')
tabulated.baseline$z = apply(tabulated.baseline, 1,
                             function(x){
  temporal.trend.function(x['t']) * spatial.trend.function(x['x'],x['y'])})
# mean.baseline = mean( do.call(rbind, z) )

mean.baseline = mean(tabulated.baseline$z)

total.cases =  nrow(observations)
total.expected.cases = sum(tabulated.baseline$z) * delta2
correction.factor = total.cases / total.expected.cases
tabulated.baseline$z = tabulated.baseline$z * correction.factor
```

```
cylinders.list = list()
for  (t.index in 5:n.weeks){
  n.cylinders = 4000 / n.weeks * t.index
  t.max = min(5, t.index)
  radia = sapply(1:t.max, function(h){sqrt(1 / mean.baseline / pi / h )} )
  radia_and_heights = cbind(1:t.max, radia)

  radia_and_heights = radia_and_heights[sample(1:nrow(radia_and_heights), n.cylinders, replace=T),]
  rho = radia_and_heights[,2]
  random_radia = runif(n.cylinders, 0, rho)
  theta = runif(n.cylinders, 0, 2* pi)

  observations.tmp = observations[observations$t <= t.index,]
  tabulated.baseline.tmp = tabulated.baseline[tabulated.baseline$t <= t.index,]

  idx = sample(1:nrow(observations.tmp), n.cylinders, replace=T)
  y0 = observations.tmp$y[idx] + sin(theta) * random_radia
  x0 = observations.tmp$x[idx] + cos(theta) * random_radia
  # t = observations.tmp$t[idx] + round(runif(n.cylinders, -radia_and_heights[,1]/2, radia_and_heights[
  # t.low = t - round(radia_and_heights[,1]/2)
  # t.low = ifelse(t.low > 0, t.low, 1)
  # t.upp = t + round(radia_and_heights[,1]/2)
  # t.max = max(observations.tmp$t)
  # t.upp = ifelse(t.upp <= t.max, t.upp, t.max)
```

```
  v.sample.int<-Vectorize(sample.int, 'n')
  rrr = v.sample.int(as.integer(radia_and_heights[,1]) + 1, 1) - 1
  t.low =  observations$t[idx] - rrr
  t.upp = t.low + as.integer(radia_and_heights[,1])
  t.min = min(observations$t)
  t.max = max(observations$t)
  t.upp = ifelse(t.low > t.min, t.upp, t.upp + (t.min - t.low))
  t.low = ifelse(t.low > t.min, t.low, t.min)
  t.low = ifelse(t.upp < t.max, t.low, t.low - (t.upp - t.max) )
  t.upp = ifelse(t.upp < t.max, t.upp, t.max)
  t.low = as.integer(ifelse( (t.upp==t.max) & (t.low < t.min), t.min, t.low))

  cylinders.tmp = data.frame(x=x0, y=y0, rho=rho, t.low=t.low, t.upp=t.upp, original.x=observations.tmp$

  cylinders.tmp[,c('n_obs', 'mu', 'p.val')] = t(apply(cylinders.tmp, 1, compute_cylinders,
                                                      observations.tmp, tabulated.baseline.tmp))

 # cylinders.tmp$warning = apply(cylinders.tmp, 1, function(x){ifelse((x['p.val'] < 0.05) & (x['n_obs']
  cylinders.tmp$warning = (cylinders.tmp$p.val < 0.05) & (cylinders.tmp$n_obs > 0)

  cylinders.list[[t.index]] = cylinders.tmp
}
rm(cylinders.tmp)
rm(tabulated.baseline.tmp)
rm(observations.tmp)
```

Compute the warning scores.

```
warning.score<-function(observation, cylinders){
  # check if the location
  x = as.numeric(observation['x'])
  y = as.numeric(observation['y'])
  TT = as.numeric(observation['t'])
  in_circle = as.integer(sqrt((as.numeric(cylinders$x) - x)^2 + (as.numeric(cylinders$y) - y)^2) < as.nu
  in_cylinder_height = as.integer( (as.numeric(cylinders$t.low) <= TT) & (as.numeric(cylinders$t.upp) >=

  # number of cylinders that include geo-coordinate of `case`
  in_cylinder = sum(in_circle * in_cylinder_height, na.rm=T)

  # number of cylinder with `warning` flag that include location `i`
  warning = sum(cylinders$warning * in_circle * in_cylinder_height,na.rm=T)
  if (in_cylinder>0){
    re = warning / in_cylinder
  }else{
    re = 0
  }
  return(re)
}
# init = Sys.time()
# observations$warning.score2 = apply(observations, 1, warning.score, cylinders)
# print(Sys.time() - init)
```

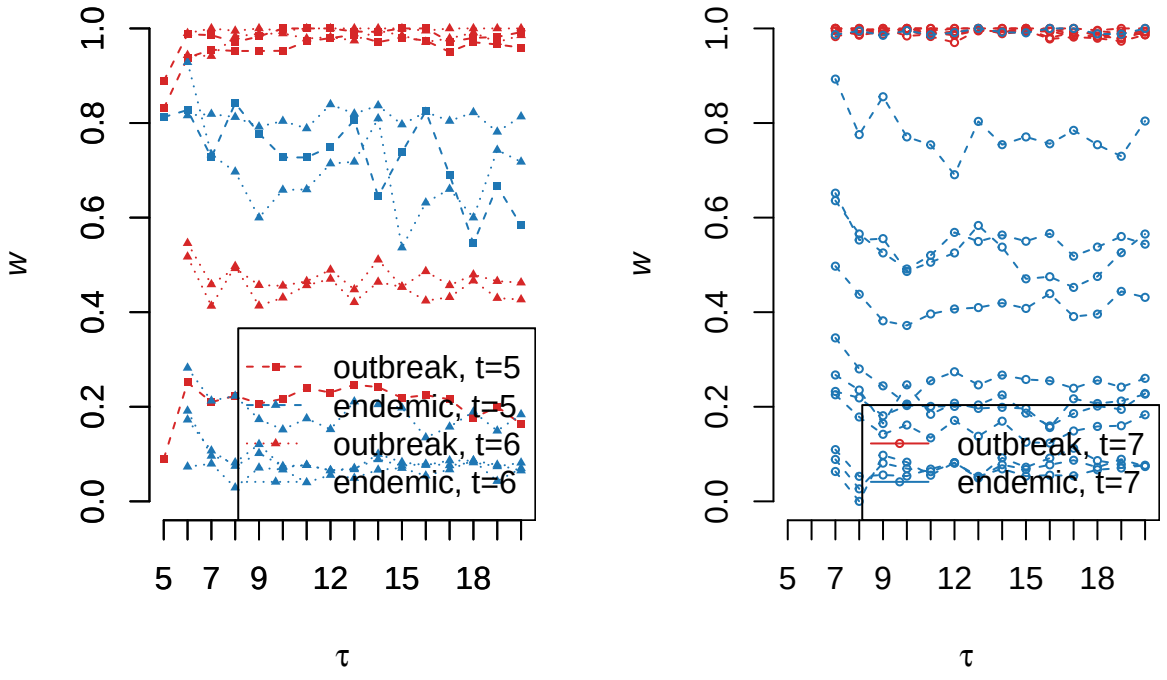
Timeliness warning scores

```
observations.list = list()
for (t.index in 5:n.weeks){
  observations.list[[t.index]] = observations[observations$t <= t.index,]
}
```

```
for (t.index in 5:n.weeks){
  observations.list[[t.index]]$warning.score = apply(observations.list[[t.index]], 1, warning.score, cy
}
```

```
tab.red = '#d62728'
tab.blue = '#1f77b4'
init_size = as.data.frame(table(observations.list[[20]]$t ))
#png(paste0('../Manuscript/timeliness_',.png'), width = 5 * 2, height = 4, units = 'in', res=600, poin
par(mfrow=c(1, 2)) #, mar=c(0.9,0.9,0.9,0.7))
plot(
  c(5,20), c(0,1), type='n', xlab = expression(tau), ylab=expression(italic(w)), axes=FALSE)
for (t in 5:6){
  # idx = init_size$. == t
  idx = init_size$Var1 == t
  if (sum(idx) > 0){
    ws= matrix(ncol = init_size$Freq[idx])
    idx = observations.list[[20]]$t == t
    color = ifelse(observations.list[[20]][idx,]$warning, tab.red, tab.blue)

    for (tau in t:20){
      idx = observations.list[[tau]]$t == t
      w = observations.list[[tau]][idx,]$warning.score
      ws = rbind(ws, w)
    }
    ws = ws[-1,]

    if (t==5){
      pch=15
      lty=2
    }else if(t==6){
      pch=17
      lty=3
    }

    for(i in 1:ncol(ws)){
      lines(t:(t+nrow(ws)-1), ws[,i], col=color[i], lty =  lty)
      points(t:(t+nrow(ws)-1), ws[,i], col=color[i], pch = pch, cex=0.5)
    }
    axis(side = 1, at = 5:20)
    axis(side = 2)
  }
}
legend('bottomright', c('outbreak, t=5','endemic, t=5', 'outbreak, t=6','endemic, t=6'),
       lty=c(2,2,3,3), lwd=c(1,1,1,1),
       col=c(tab.red, tab.blue, tab.red, tab.blue),
       pch=c(15,17,17,17), pt.cex=0.5)
# legend('bottom', c('t=5','t=6'), pch=c(5,6) )
plot(
  c(5,20), c(0,1), type='n', xlab = expression(tau), ylab=expression(italic(w)), axes=FALSE)
```

```
for (t in 7){
  # idx = init_size$. == t
  idx = init_size$Var1 == t
  if (sum(idx) > 0){
    ws= matrix(ncol = init_size$Freq[idx])
    idx = observations.list[[20]]$t == t
    color = ifelse(observations.list[[20]][idx,]$warning,  tab.red, tab.blue)

    for (tau in t:20){
      idx = observations.list[[tau]]$t == t
      w = observations.list[[tau]][idx,]$warning.score
      ws = rbind(ws, w)
    }
    ws = ws[-1,]
    for(i in ncol(ws):1){
      lines(t:(t+nrow(ws)-1), ws[,i], col=color[i], lty =  2)
      points(t:(t+nrow(ws)-1), ws[,i], col=color[i], pch = 1, cex = 0.5)
    }
    axis(side = 1, at = 5:20)
    axis(side = 2)
  }
}
legend('bottomright', c('outbreak, t=7','endemic, t=7'), lwd=c(1,1,NA),
       lty=c(1,1), pch=c(1,1), col=c( tab.red, tab.blue), pt.cex=0.5)
```

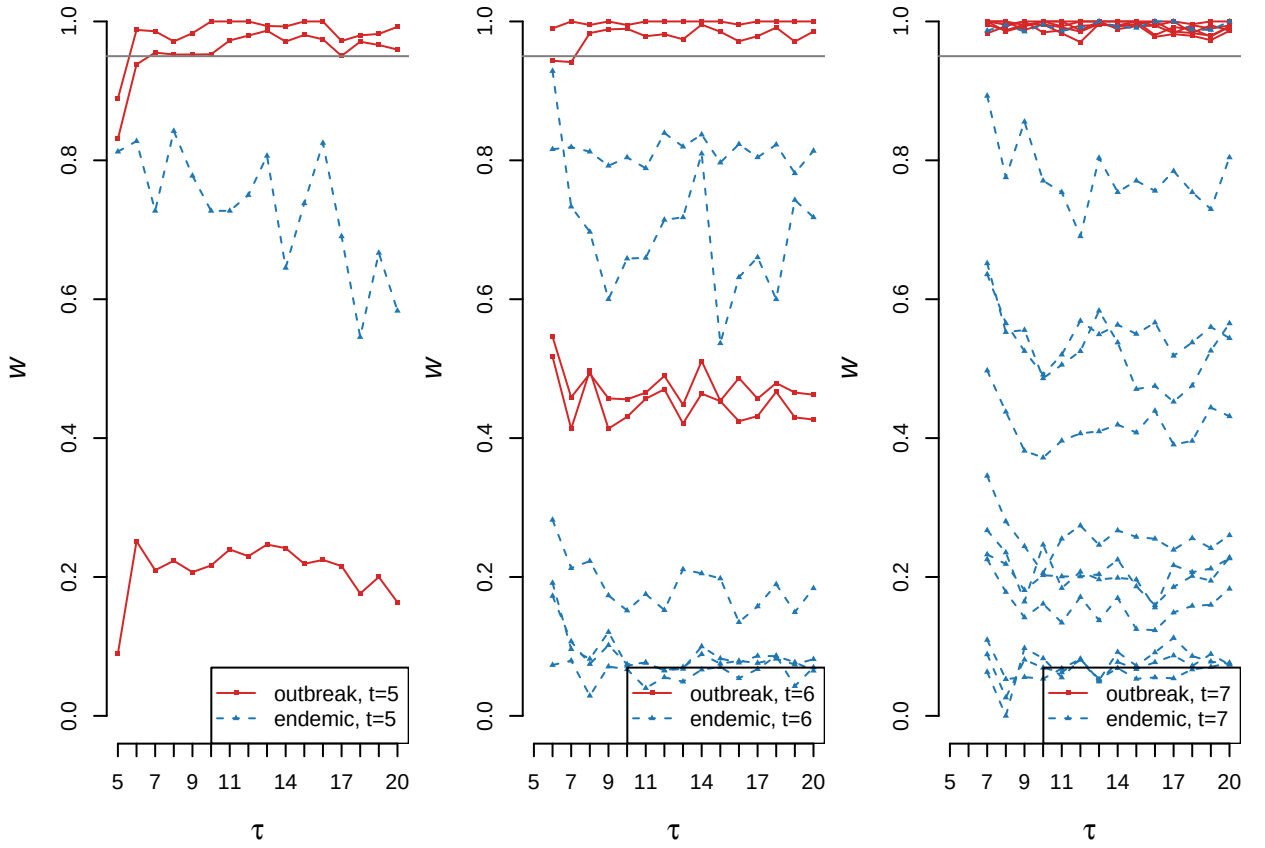



```
# legend('bottom', c('t=7'), pch=c(1) )
# dev.off()
```

```
tab.red = '#d62728'
tab.blue = '#1f77b4'
init_size = as.data.frame(table(observations.list[[20]]$t ))
# png(paste0('../Manuscript/timeliness__','.png'), width = 5 * 2.8, height = 4, units = 'in', res=600,
```

```r
par(mfrow=c(1, 3), mar= c(4, 4, 0.5, 0.5) )
plot(
  c(5,20), c(0,1), type='n', xlab = expression(tau), ylab=expression(italic(w)), axes=FALSE, cex.lab=1.4
t = 5
  # idx = init_size$. == t
  idx = init_size$Var1 == t
  if (sum(idx) > 0){
    ws= matrix(ncol = init_size$Freq[idx])
    idx = observations.list[[20]]$t == t
    color = ifelse(observations.list[[20]][idx,]$warning, tab.red, tab.blue)
    pch = ifelse(observations.list[[20]][idx,]$warning, 15, 17)
    lty = ifelse(observations.list[[20]][idx,]$warning, 1, 2)

    for (tau in t:20){
      idx = observations.list[[tau]]$t == t
      w = observations.list[[tau]][idx,]$warning.score
      ws = rbind(ws, w)
    }
    ws = ws[-1,]

    for(i in 1:ncol(ws)){
      lines(t:(t+nrow(ws)-1), ws[,i], col=color[i], lty =  lty[i])
      points(t:(t+nrow(ws)-1), ws[,i], col=color[i], pch = pch[i], cex=0.5)
    }
    axis(side = 1, at = 5:20)
    axis(side = 2)
  }
abline(h=0.95, col='#7f7f7f')
legend('bottomright', c('outbreak, t=5','endemic, t=5'), lwd=c(1,1),
       lty=c(1,2), col=c(tab.red, tab.blue), pch=c(15,17), pt.cex=0.5)
plot(
  c(5,20), c(0,1), type='n', xlab = expression(tau), ylab=expression(italic(w)), axes=FALSE, cex.lab=1.4
t = 6
  # idx = init_size$. == t
  idx = init_size$Var1 == t
  if (sum(idx) > 0){
    ws= matrix(ncol = init_size$Freq[idx])
    idx = observations.list[[20]]$t == t
    color = ifelse(observations.list[[20]][idx,]$warning, tab.red, tab.blue)
    pch = ifelse(observations.list[[20]][idx,]$warning, 15, 17)
    lty = ifelse(observations.list[[20]][idx,]$warning, 1, 2)

    for (tau in t:20){
      idx = observations.list[[tau]]$t == t
      w = observations.list[[tau]][idx,]$warning.score
      ws = rbind(ws, w)
    }
    ws = ws[-1,]

    for(i in 1:ncol(ws)){
      lines(t:(t+nrow(ws)-1), ws[,i], col=color[i], lty = lty[i])
      points(t:(t+nrow(ws)-1), ws[,i], col=color[i], pch = pch[i], cex=0.5)
    }
```

```
    axis(side = 1, at = 5:20)
    axis(side = 2)
  }
abline(h=0.95, col='#7f7f7f')
legend('bottomright', c('outbreak, t=6','endemic, t=6'), lwd=c(1,1),
       lty=c(1, 2), col=c(tab.red, tab.blue),  pch=c(15,17), pt.cex=0.5)
plot(
  c(5,20), c(0,1), type='n', xlab = expression(tau), ylab=expression(italic(w)), axes=FALSE,  cex.lab=1
  t=7
  # idx = init_size$. == t
  idx = init_size$Var1 == t
  if (sum(idx) > 0){
    ws= matrix(ncol = init_size$Freq[idx])
    idx = observations.list[[20]]$t == t
    color = ifelse(observations.list[[20]][idx,]$warning,  tab.red, tab.blue)
    pch = ifelse(observations.list[[20]][idx,]$warning, 15, 17)
    lty = ifelse(observations.list[[20]][idx,]$warning, 1, 2)

    for (tau in t:20){
      idx = observations.list[[tau]]$t == t
      w = observations.list[[tau]][idx,]$warning.score
      ws = rbind(ws, w)
    }
    ws = ws[-1,]
    for(i in ncol(ws):1){
      lines(t:(t+nrow(ws)-1), ws[,i], col=color[i], lty = lty[i])
      points(t:(t+nrow(ws)-1), ws[,i], col=color[i], pch = pch[i], cex = 0.5)
    }
    axis(side = 1, at = 5:20)
    axis(side = 2)
  }
abline(h=0.95, col='#7f7f7f')
legend('bottomright', c('outbreak, t=7','endemic, t=7'), lwd=c(1, 1),
       lty=c(1,2), pch=c(15,17), col=c(tab.red, tab.blue), pt.cex=0.5)
```


W
τ
outbreak, t=5
endemic, t=5
outbreak, t=6
endemic, t=6
outbreak, t=7
endemic, t=7

```
# dev.off()
```