Simulation Experiment

23 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'
```

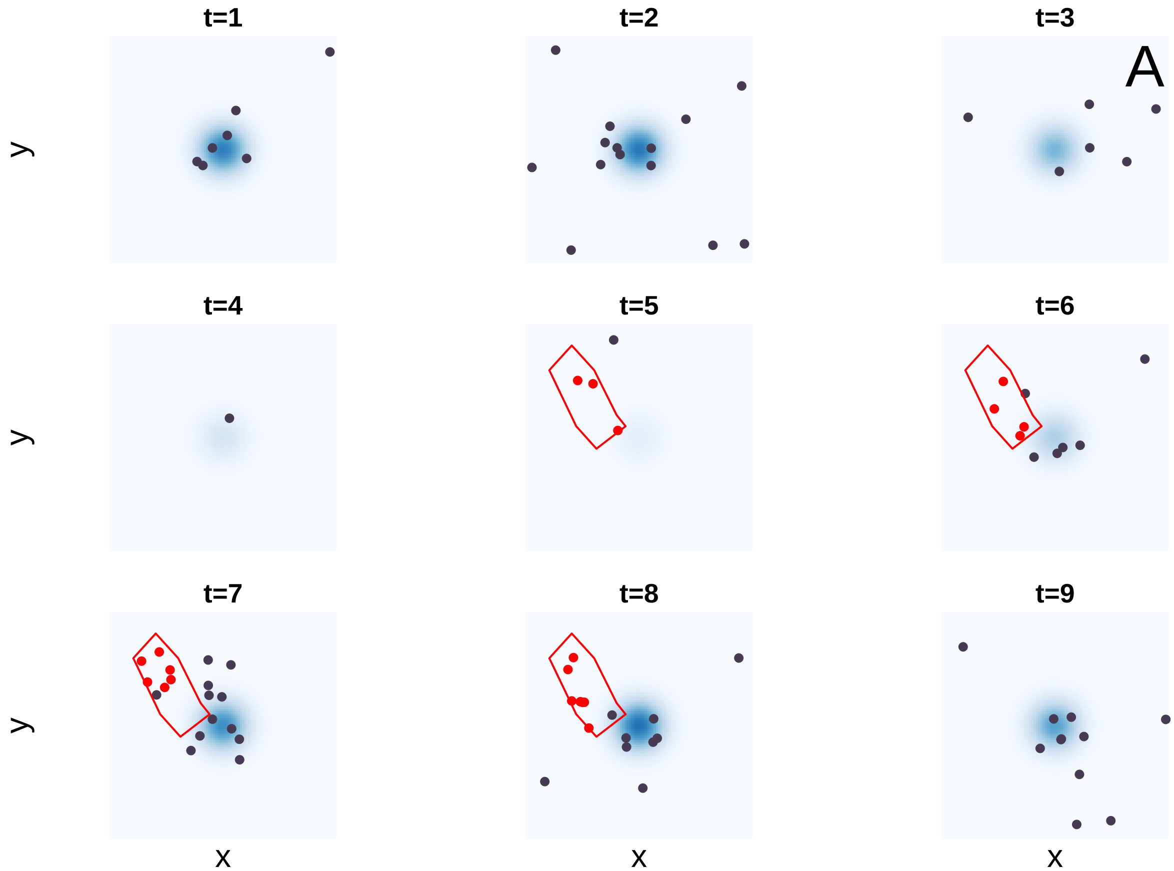
```
temporal.trend.function<-function(t, A=7, B=0.9){
  return (A/2*(sin(t) + 1) + B)
}

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, point

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()
```

Aggregate all observations in a `list` data structure.

```
observations = list()
for (i in 1:n.weeks){
  b = baseline[[i]]
  s = epidemics[[i]]
  observations[[i]] = data.frame(x=c(b$x, s$x),
                                 y=c(b$y, s$y),
                                # color=c(rep('#473B54', length(b$x)), rep('red', length(s$x)))
                                 warning=c(rep(F, length(b$x)), rep(T, length(s$x))),
  t=rep(i, length(c(b$x, s$x))))
}
observations = do.call(rbind,  observations)
```

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

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

Randomly 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))
}

init = Sys.time()
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

radia = sapply(1:5, function(h){sqrt(1 / mean.baseline / pi / h )} )
radia_and_heights = cbind(1:5, radia)
n.cylinders = 10000
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)

idx = sample(1:nrow(observations), n.cylinders, replace=T)
y0 = observations$y[idx] + sin(theta) * random_radia
x0 = observations$x[idx] + cos(theta) * random_radia
```

```
t = observations$t[idx] + round(runif(n.cylinders, -radia_and_heights[,1]/2, radia_and_heights[,1]/2))

t.low = t - round(radia_and_heights[,1]/2)
t.low = ifelse(t.low>0, t.low, 1)
t.upp = t.low + round(radia_and_heights[,1]/2)
t.max = max(observations$t)
t.upp = ifelse(t.upp <= t.max, t.upp, t.max)

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

cylinders[,c('n_obs', 'mu',  'p.val')] = t(apply(cylinders, 1, compute_cylinders,
                                                  observations, tabulated.baseline))
cylinders$warning = (cylinders['p.val'] < 0.05) & (cylinders['n_obs'] > 0)
print(Sys.time() - init)
```

```
## Time difference of 1.870083 mins
```

To demonstrate that the methodology is robust to changes in cylinder size, make another set of cylinders with increased radia.

```
init=Sys.time()
cylinders3 = cylinders
cylinders3$rho = cylinders3$rho * 1.4
cylinders3[,c('n_obs', 'mu', 'p.val')] = t(apply(cylinders3, 1, compute_cylinders, observations, tabula

# cylinders3$warning = apply(cylinders3, 1, function(x){ifelse((x['p.val'] < 0.05) & (x['n_obs'] > 0),
cylinders3$warning = (cylinders3['p.val'] < 0.05) & (cylinders3['n_obs'] > 0)

print(Sys.time() - init)
```

```
## Time difference of 2.168572 mins
```

Compute the warning scores.

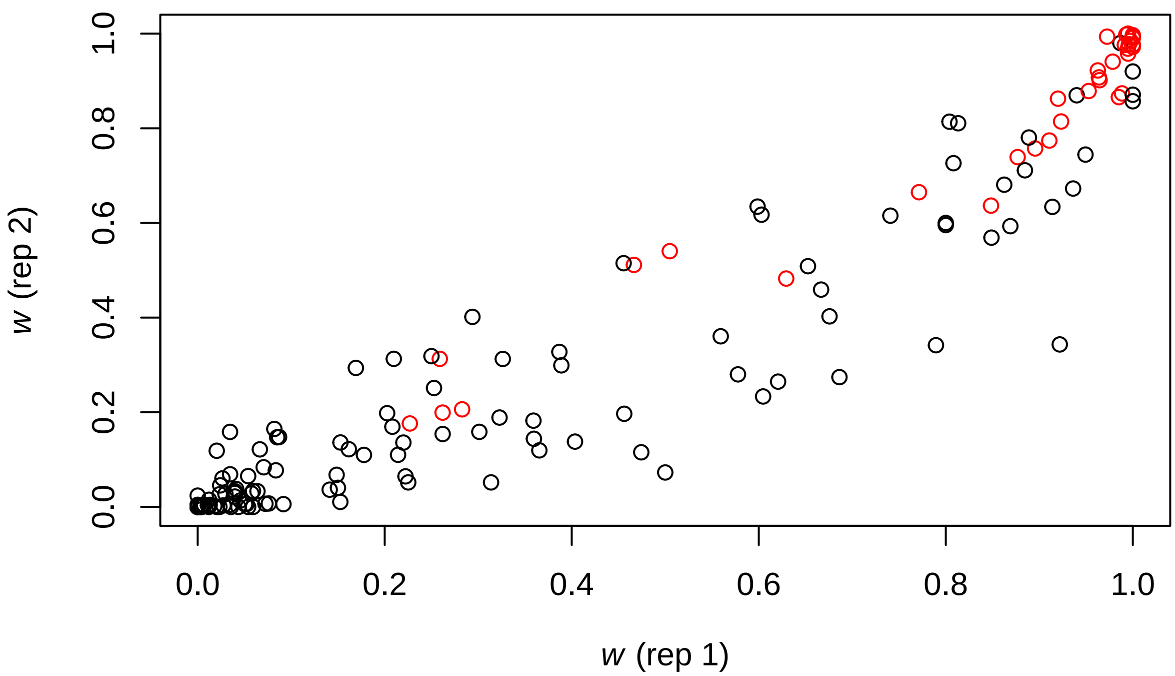
```
warning.score<-function(observation, cylinders){
  # check if the location
  x = as.numeric(observation['x'])
  y = as.numeric(observation['y'])
  TT = as.numeric(observation['t'])

  in_circle = as.integer(sqrt((cylinders$x - x)^2 + (cylinders$y - y)^2) < cylinders$rho)
  in_cylinder_height = as.integer((cylinders$t.low <= TT) & (cylinders$t.upp >= TT))

  # 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$warning.score = apply(observations, 1, warning.score, cylinders)
#observations$warning.score2 = apply(observations, 1, warning.score, cylinders2)
observations$warning.score3 = apply(observations, 1, warning.score, cylinders3)

print(Sys.time() - init)
```

```
## Time difference of 0.326313 secs
```

Show that warning scores obtained from cylinders of different volumes are correlated and both informative.

```
#png('../Manuscript/corr_rho_simulation.png', width = 3.25 * 1.2, height = 3.25* 1.2, units = 'in', res
# par(mfrow=c(1,1))
color = ifelse(observations$warning, 'red', 'black')
plot(warning.score3 ~ warning.score, observations, xlab=expression(paste(italic(w), " (rep 1)")), ylab =
```




```
#dev.off()

c.test=cor.test(observations$warning.score, observations$warning.score3) #pearson correlation
print(c(c.test$estimate, c.test$p.value))
```

```
##          cor
## 9.527856e-01 3.591341e-83
```

Compute the ROC as an appropriate performance metric using the library `pROC`.

```
cc1 = pROC::roc(as.integer(warning) ~ warning.score, observations)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
cc2 = pROC::roc(as.integer(warning) ~ warning.score3, observations)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
df1 = data.frame(cc1$specificities, cc1$sensitivities, cc1$thresholds)
df2 = data.frame(cc2$specificities, cc2$sensitivities, cc2$thresholds)

idxa = which(df1$cc1.thresholds > 0.95)
idxb = which(df1$cc1.thresholds < 0.95)
idx = c(idxa[1], idxb[length(idxb)])
# print(df1)
spec_sens_cc1 = colMeans(df1[idx,])
print(spec_sens_cc1)
```

```
## cc1.specificities cc1.sensitivities    cc1.thresholds
##         0.9640000         0.5882353         0.9478725
```

```
idxa = which(df2$cc2.thresholds > 0.95)
idxb = which(df2$cc2.thresholds < 0.95)
idx = c(idxa[1], idxb[length(idxb)])
# print(df2)
spec_sens_cc2 = colMeans(df2[idx,])

print(spec_sens_cc2)
```

```
## cc2.specificities cc2.sensitivities    cc2.thresholds
##         0.9920000         0.3676471         0.9561957
```

Sensitivity: the ability of a test to correctly identify epidemic events. Specificity: the ability of a test to correctly identify non-epidemic events.

```
options(warn = - 1)
#conf_matrix<-table(ifelse(observations$warning.score2 > 0.95, T, F), observations$warning)
# sens = caret::sensitivity(conf_matrix)
# spec = caret::specificity(conf_matrix)
# cat("sensitivity=", sens, "\n")
# cat("specificity=", spec, "\n")
# png('../Manuscript/ROC_.png', width = 3.25, height = 3.25, units = 'in', res=400, pointsize = 13)
par(mfrow=c(1,1), mar=c(0.5, 0.5, 0.5, 0.5))
col = '#00000033'
cc = pROC::roc(as.integer(warning) ~ warning.score, observations)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(cc, col='black', identity.col='black',  grid.col='red', grid.v=spec_sens_cc1[1], grid.h = spec_sens
auc = as.numeric(cc['auc'][[1]])
L = nrow(observations)
for (i in 1:50){
  cc = pROC::roc(as.integer(warning) ~ warning.score, observations[sample(1:L, L, replace = T),])
  if(i < 11){
    plot(cc, add=T, col=col,  identity.col='black')
  }
  auc = c(auc,as.numeric(cc['auc'][[1]]))
}
```

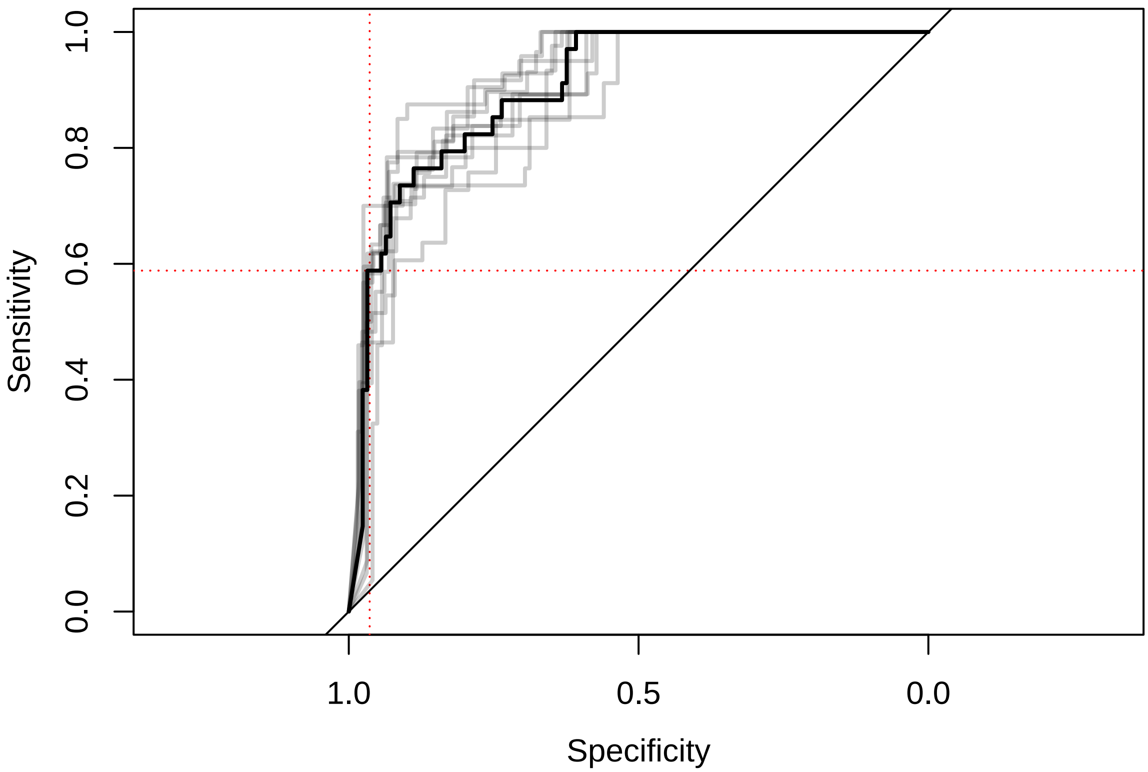
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```




```
print(quantile(auc, c(0.025,0.5,0.975) ))
```

```
##      2.5%       50%     97.5%
## 0.8520541 0.9025210 0.9432191
```

```
par(mfrow=c(1,1), mar=c(0.5, 0.5, 0.5, 0.5))

cc = pROC::roc(as.integer(warning) ~ warning.score3, observations)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

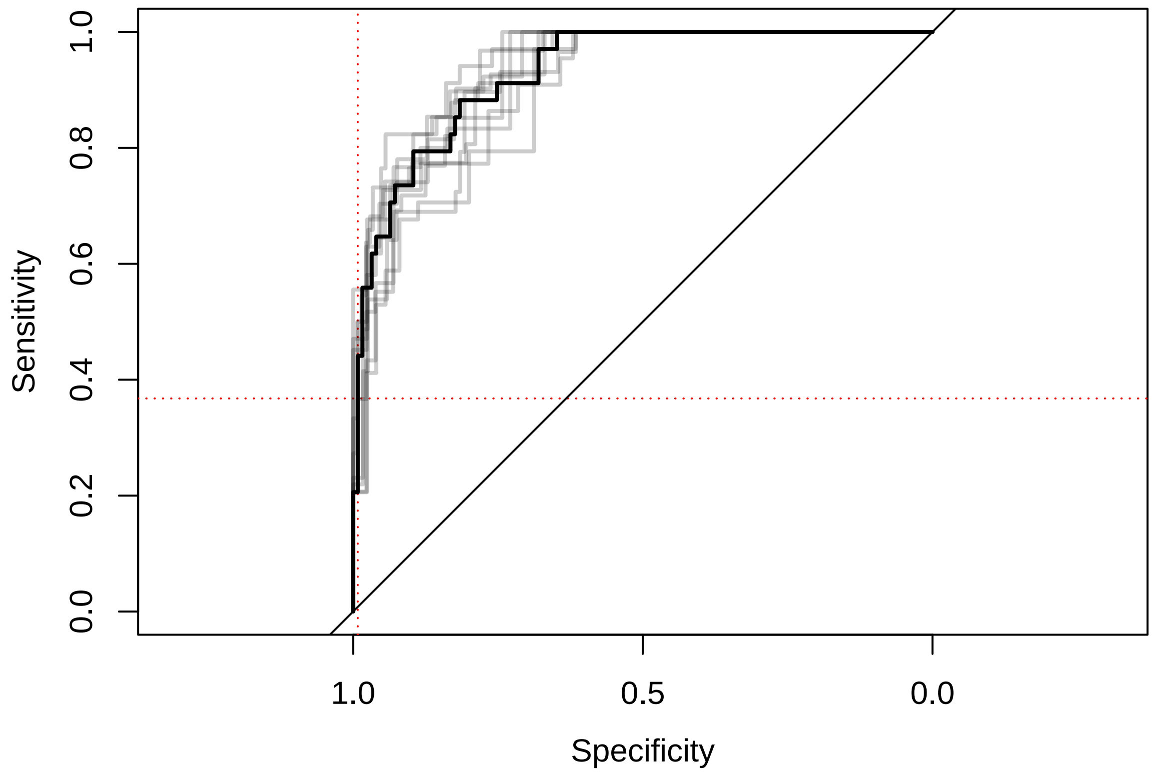
```
plot(cc, col='black', identity.col='black',
     grid.col='red', grid.v=spec_sens_cc2[1], grid.h = spec_sens_cc2[2])
auc = as.numeric(cc['auc'][[1]])
L = nrow(observations)
for (i in 1:50){
  cc = pROC::roc(as.integer(warning) ~ warning.score3, observations[sample(1:L, L, replace = T),])
  if(i < 11){
    plot(cc, add=T, col=col,  identity.col='black')
  }
  auc = c(auc,as.numeric(cc['auc'][[1]]))
}
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```




```
print(quantile(auc, c(0.025,0.5,0.975) ))
```

```
##      2.5%       50%     97.5%
## 0.8937076 0.9304147 0.9638357
```

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

```
# png('../Manuscript/simulation_experiment_ROC_CORR.png',
#  width = 3.25 *3, height = 3.25, units = 'in', res=600, pointsize = 20)
# par(mfrow=c(1,3))

col = '#00000033'
cc = pROC::roc(as.integer(warning) ~ warning.score, observations)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(cc, col='black', identity.col='black',  grid.col='red', grid.v=spec_sens_cc1[1], grid.h = spec_sens
auc = as.numeric(cc['auc'][[1]])
L = nrow(observations)
for (i in 1:50){
  cc = pROC::roc(as.integer(warning) ~ warning.score, observations[sample(1:L, L, replace = T),])
  if(i < 11){
    plot(cc, add=T, col=col,  identity.col='black')
  }
  auc = c(auc,as.numeric(cc['auc'][[1]]))
}
```

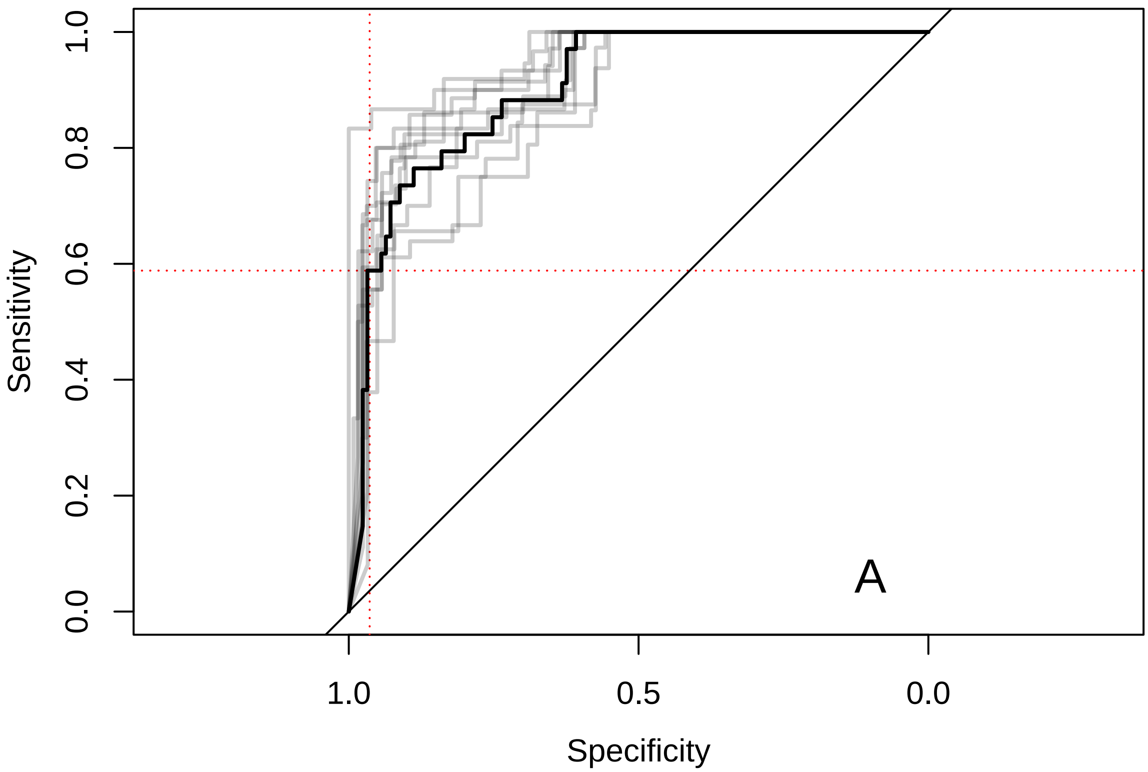
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
mtext('A', 1, at = 0.1, padj = -1, cex=1.5)
```




```
cc = pROC::roc(as.integer(warning) ~ warning.score3, observations)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(cc, col='black', identity.col='black',  grid.col='red', grid.v=spec_sens_cc2[1], grid.h = spec_sens
auc = as.numeric(cc['auc'][[1]])
L = nrow(observations)
for (i in 1:50){
  cc = pROC::roc(as.integer(warning) ~ warning.score3, observations[sample(1:L, L, replace = T),])
  if(i < 11){
    plot(cc, add=T, col=col,  identity.col='black')
  }
  auc = c(auc,as.numeric(cc['auc'][[1]]))
}
```

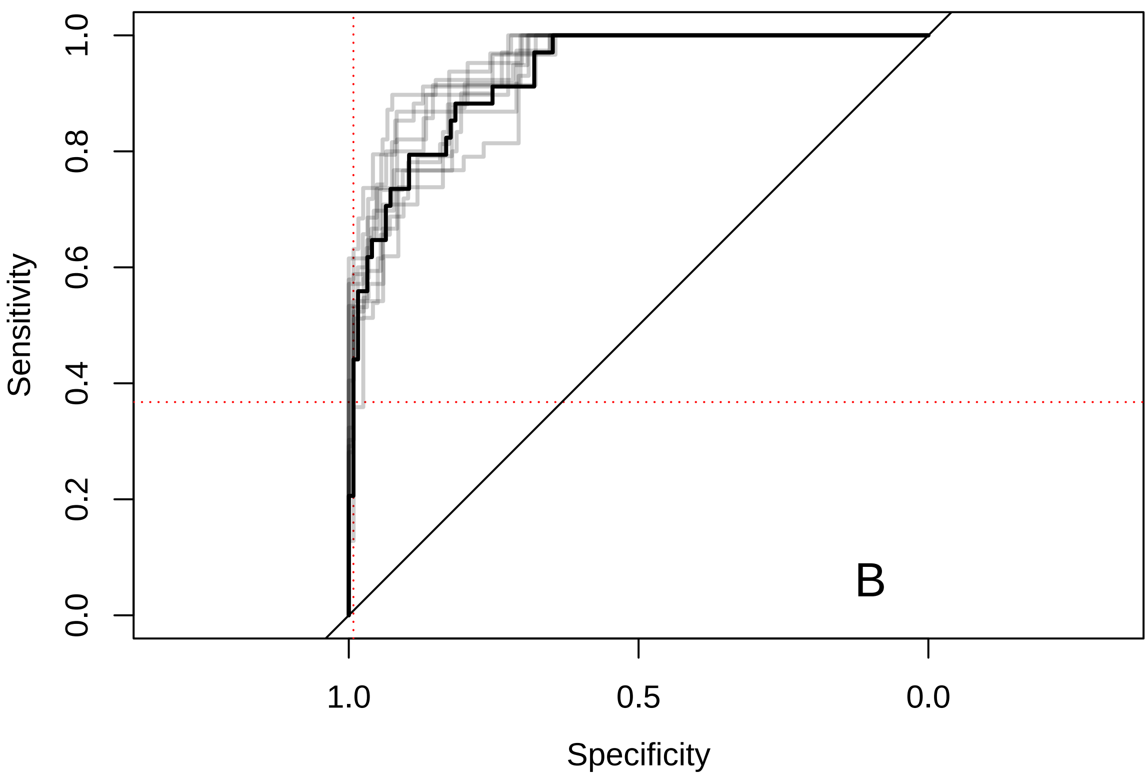
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

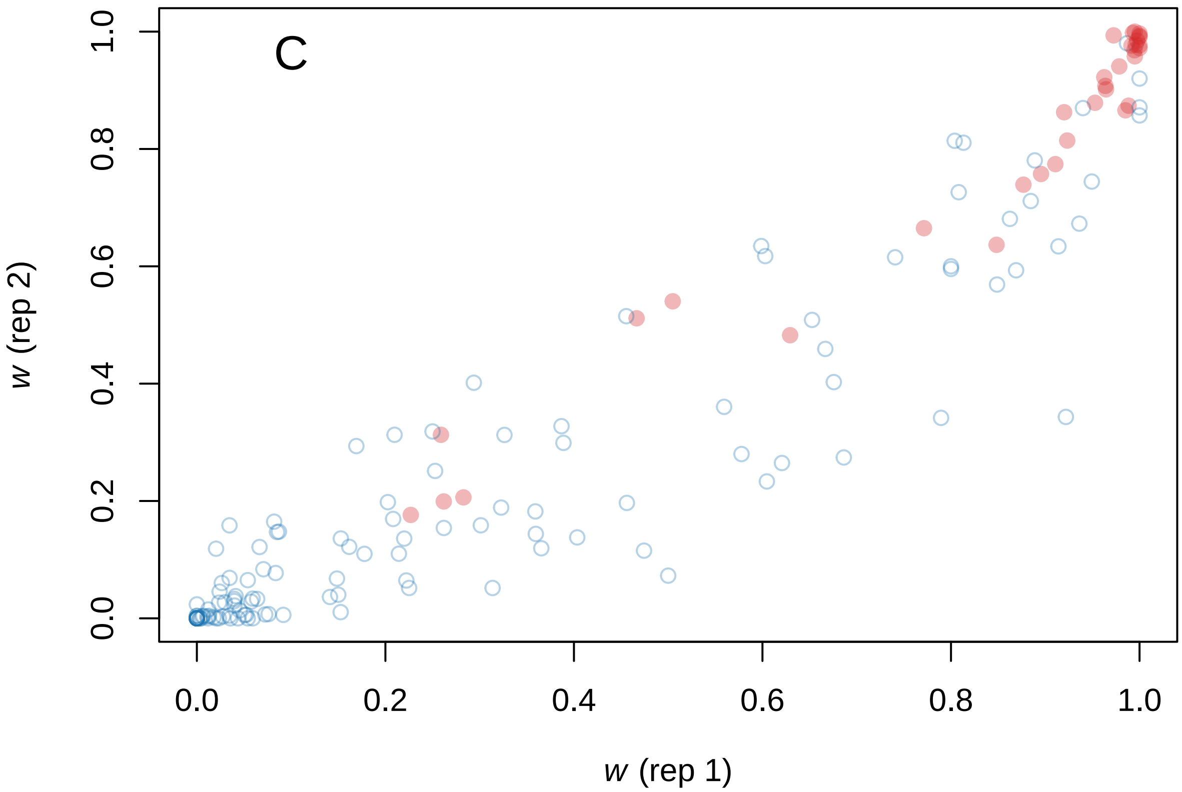
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
mtext('B', 1, at = 0.1, padj = -1, cex=1.5)
```



```
par(mar=c(4,4,2,2))
color = ifelse(observations$warning, '#d6272855', '#1f77b455')
pch = ifelse(observations$warning, 19, 1)
plot(warning.score3 ~ warning.score, observations, xlab=expression(paste(italic(w), " (rep 1)")), ylab =
mtext('C', 3, at = 0.1, padj = 2, cex=1.5)
```




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