Simulation experiment England

```
knitr::opts_knit$set(root.dir = "~/Documents/outbreak-detection/")
getwd()
```

```
## [1] "/home/massimo/Documents/outbreak-detection/Simulation_experiments"
```

```
source('R/utils.R')
source("R/Evaluate.r")
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
## Linking to GEOS 3.5.1, GDAL 2.2.2, PROJ 4.9.2
```

```
## 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'
```

Simulation of cases in England

We first simulate *endemic* cases according to a given baseline. The baseline is assumed proportional to postcode population (spatial factor, saved in `population`) and and to a temporal trend (temporal factor, saved in `time.factor`). `no.endemic_cases` is the total number of endemic cases to simulate. We make use of a utility function `Simulate()` that calls `rpois()` to generate the count data and returns the simulated cases in matrix format (rows correspond to postcodes, columns correspond to detected time).

```
n.endemic_cases = 5000
size_factor_epi = 4 # multiply by this factor to scale the number of epidemic cases.
label.str = paste(as.character(size_factor_epi), as.character(n.endemic_cases),  sep = '_')
load("Data/population_of_england.RData_")
load('Data/time.factor.RData_')
head(population)
```

```
## AL1 1AG AL1 1AJ AL1 1AR AL1 1AS AL1 1BH AL1 1BX
##      14     124      32      34      52      54
```

```
# Set the random generator seed
set.seed(1)
# subsample 10000 postcodes for speed
idx=sort(sample(length(population), 10000))
sample.population = population[idx]
end.matrix = Simulate(sample.population, time.factor[-1][1:100], n.endemic_cases)
```

Create a baseline matrix of the same dimension as the observation matrix:

```
b.matrix = sample.population %o% time.factor[-1][1:100]
b.matrix = b.matrix / sum(b.matrix) * n.endemic_cases
```

Map data to geographical coordinates:

```
#geo.location = t(sapply(names(population), postcode.to.location3))
#save(geo.location, file = "Data/geo.location_of_england.RData")
load("Data/geo.location_of_england.RData_")
df.population = cbind(geo.location[idx,], sample.population)
df.cases = cbind(geo.location[idx,], rowSums(end.matrix))
colnames(df.cases) = c('latitude', 'longitude', 'n.cases')
df.cases = as.data.frame(df.cases)
head(df.cases)
```

```
##         latitude longitude n.cases
## AL1 1DX 51.73172 -0.301875       2
## AL1 1JL 51.74460 -0.322917       3
## AL1 1UW 51.73384 -0.312063       2
## AL1 2JS 51.73943 -0.339446       0
## AL1 3BT 51.74802 -0.330787       0
## AL1 3FR 51.75332 -0.328534       0
```

```
df = data.frame(longitude = rep(df.cases[,'longitude'], df.cases[,'n.cases']),
             latitude = rep(df.cases[,'latitude'], df.cases[,'n.cases']))
```

Simulate more cases around St Albans (postcode starting with AL1 and AL2) from time 40 to time 60. These represent an outbreak (the *epidemic* component).

```
idx1 = grepl('AL1', rownames(end.matrix), fixed=T)
idx1 = idx1 | grepl('AL2', rownames(end.matrix), fixed=T)
cat("the number of postcodes starting with AL1 or AL2 is", sum(idx1))
```

```
## the number of postcodes starting with AL1 or AL2 is 19
```

```
# set.seed(1)
save(".Random.seed",file="random_state_seed_1.RData") ## save current RNG state
load("random_state_seed_1.RData")
epi = rpois(n = sum(idx1) * 20, lambda=rep(dnorm(-9:10, sd = 4) * size_factor_epi, rep(sum(idx1), 20)))
save(epi, file = "Data/simulated_larger_epidemic.RData")
cat("the numer of epidemic cases is", sum(epi))
```

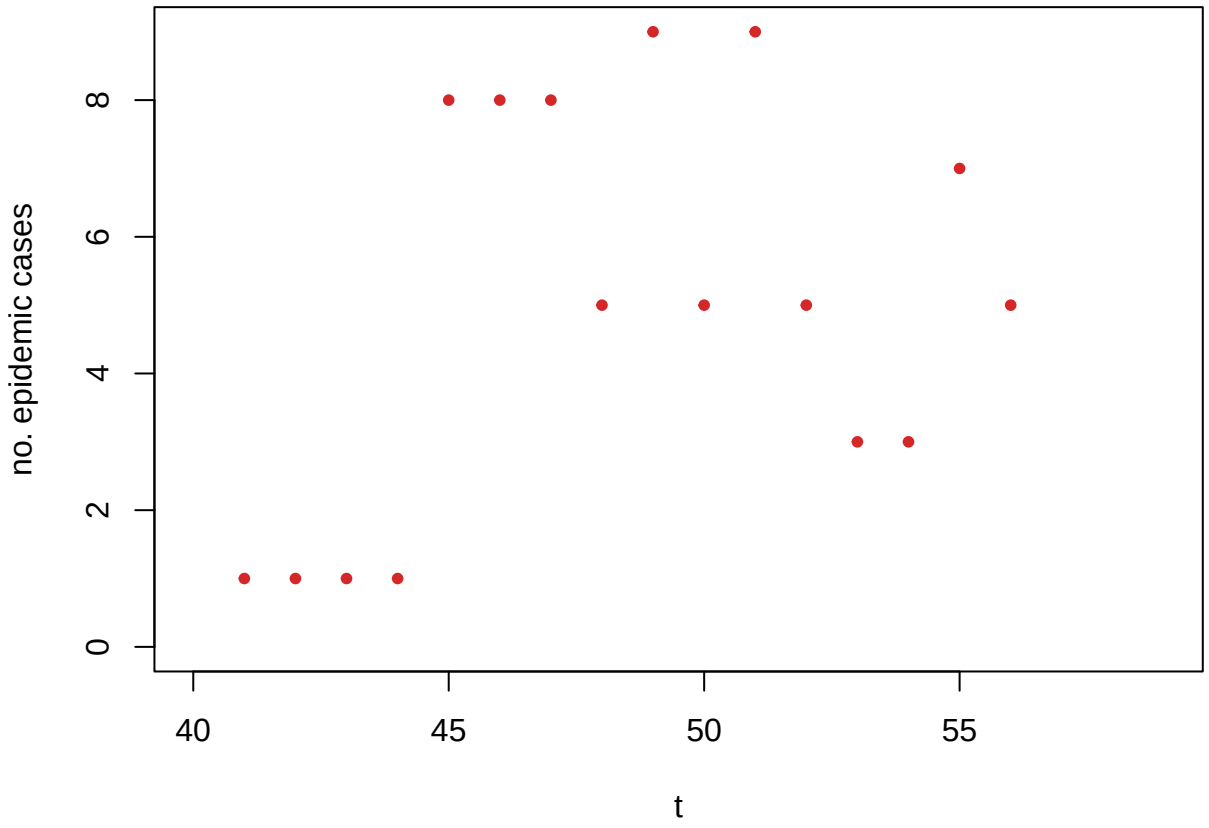
```
## the numer of epidemic cases is 79
```

```
epi.matrix = matrix(epi, ncol = 20)
rownames(epi.matrix) = row.names(end.matrix)[1:sum(idx1)]
colnames(epi.matrix) = 40:59
epi.matrix[1:3,1:3]
```

```
##         40 41 42
```

```
## AL1 1DX  0  0  0
## AL1 1JL  0  0  0
## AL1 1UW  0  0  0
```

```
#png(paste0('Fig/epi_time_', label.str, '.png'), width = 4 * 1.2, height = 3, units = 'in', res=400, po
par(mar=c(4,4,1,1)+0.1)
plot(40:59, colSums(epi.matrix), col=ifelse(colSums(epi.matrix) > 0,'#d62728','white'), xlab='t', pch=2
```

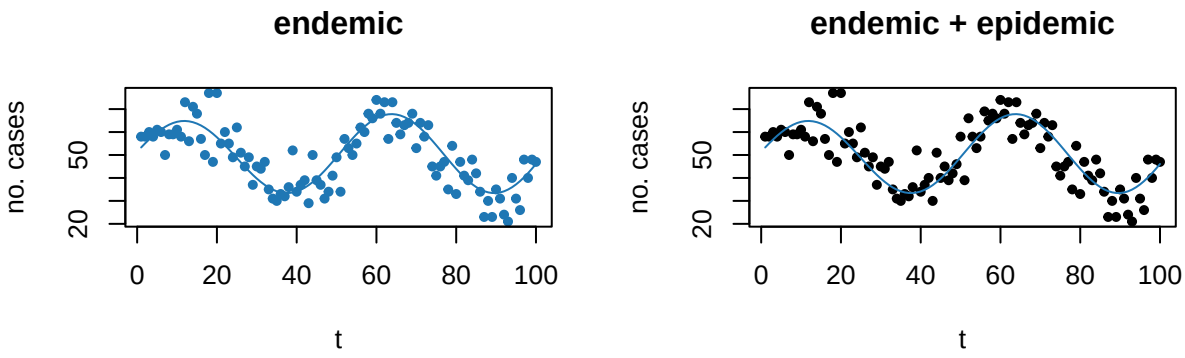



```
#dev.off()
```

Aggregate epidemic and endemic cases and plot versus time and space.

```
all.matrix = end.matrix
all.matrix[1:sum(idx1), as.character(40:59)] = all.matrix[1:sum(idx1), as.character(40:59)] + epi.matri

# code that requires UK boundaries shape files is commented
# source('R/plotBaseMap.r')
# png(paste0('Fig/end_epi_panel_',label.str,'.png'), width = 4 * 1.2 * 2, height = 3 * 1.2 * 2, units =
par(mfrow = c(2, 2)) #, mar=c(4,4,1,1)+0.1)
plot(colSums(end.matrix), xlab='t', col='#1f77b4', pch=20, ylab='no. cases', main='endemic')
lines(time.factor[-1][1:100] * n.endemic_cases / sum(time.factor[-1][1:100]), col='#1f77b4')
# mtext('A', side=3, padj=2, at=95, cex=2)
plot(colSums(all.matrix), pch=20, xlab='t', main='endemic + epidemic',  ylab='no. cases')
lines(time.factor[-1][1:100] * n.endemic_cases / sum(time.factor[-1][1:100]), col='#1f77b4')
# mtext('B', side=3, padj=2, at=95, cex=2)

df.cases2 = cbind(geo.location[idx,], rowSums(all.matrix))
df.cases2 = as.data.frame(df.cases2)
df.cases2 = cbind(rownames(geo.location[idx,]), df.cases2)
```

```
colnames(df.cases2) = c('Postcode','latitude', 'longitude', 'n.cases')
df2 = data.frame(longitude = rep(df.cases2[,'longitude'], df.cases2[,'n.cases']),
                 latitude = rep(df.cases2[,'latitude'], df.cases2[,'n.cases']))

plot(df$longitude, df$latitude,axes=F, pch=20, col='#1f77b4', cex=0.1, xlab=NA, ylab=NA, main='endemic'
# plotBaseMap(add=T, Wales=F)
# mtext('C', side=3, padj=2, at=1, cex=2)

plot(df2$longitude, df2$latitude,axes=F, pch=20, cex=0.1, xlab=NA, ylab=NA, main='endemic + epidemic', y
```




```
# plotBaseMap(add=T, Wales=F)
# mtext('D', side=3, padj=2, at=1, cex=2)
```

Convert the matrix data containing the simulated observations into a data frame `case.df` (as in a realistic records of detected cases).

```
case.df = as.data.frame(which(all.matrix == 1, arr.ind = TRUE))
case.df$postcode = rownames(all.matrix)[case.df$row]
for (i in 2:max(c(all.matrix))){
  case.df.tmp = as.data.frame(which(all.matrix == i, arr.ind = TRUE))
  case.df.tmp$postcode = rownames(all.matrix)[case.df.tmp$row]
  case.df = rbind(case.df, case.df.tmp[rep(seq_len(NROW(case.df.tmp)), i),])
}

case.df = cbind(case.df, df.population[case.df[,'row'],])
case.df$col = case.df$col - 1
names(case.df) = c('row','SAMPLE_DT_numeric', 'postcode', 'latitude', 'longitude', 'population')
case.df[, c('y','x')] = vlatlong2km(case.df[,c('latitude', 'longitude')])
tail(case.df)
```

```
##            row SAMPLE_DT_numeric postcode latitude longitude population
## RH2.7NN.1 7111                83  RH2 7NN 51.22575 -0.194216        111
## E14.9FA.1 2731                95  E14 9FA 51.49664 -0.015921         94
## LE672HE.1 4476                99  LE672HE 52.71245 -1.379711        116
```

```
## OL103NB   6076               55  OL103NB 53.59020 -2.245998        208
## OL103NB.1 6076               55  OL103NB 53.59020 -2.245998        208
## OL103NB.2 6076               55  OL103NB 53.59020 -2.245998        208
##                    y          x
## RH2.7NN.1 5696.303  -7.0569065
## E14.9FA.1 5726.426  -0.5675292
## LE672HE.1 5861.625 -44.8914334
## OL103NB   5959.230 -67.9957500
## OL103NB.1 5959.230 -67.9957500
## OL103NB.2 5959.230 -67.9957500
```

Mark the entries in `case.df` corresponding to the simulated outbreak (epidemic component) as `true_positive`:

```
case.df.epidemic =  which(epi.matrix > 0,  arr.ind = T)
case.df$true_positive = apply(case.df, 1, function(x){ ( as.numeric(x['row']) %in% case.df.epidemic[,'r
# add 39 because epi.matrix starts from time index 40.
```

Apply RaNCover.

We detect the outbreak events with RaNCover in two steps. First, we randomly draw $N = 1000000$ covering cylinders using the function `CreateCylinders()`. Then, we apply the function `warning.score()` to all entries in `case.df` in order to compute the warning scores w (along with their Wilson 95% confidence intervals) and test that w is significantly greater than 0.95.

```
# set.seed(1)
save(".Random.seed",file="random_state_seed_2.RData") ## save current RNG state
load("random_state_seed_2.RData")
cylinders = CreateCylinders(observation.matrix = all.matrix, baseline.matrix = b.matrix, emmtype = 'sim
case.df[,c('warning.score', 'low','upp','p.value')] = t(apply(case.df, 1, FUN=warning.score, cylinders)
```

```
head(case.df)
```

```
##         row SAMPLE_DT_numeric postcode latitude longitude population        y
## B14.6TN  79                 0  B14 6TN 52.42452 -1.906414         80 5829.607
## B15.2BQ  80                 0  B15 2BQ 52.47047 -1.908348         59 5834.716
## B42.2RZ 165                 0  B42 2RZ 52.53757 -1.902311        177 5842.178
## B61.0DB 203                 0  B61 0DB 52.34426 -2.052947         18 5820.682
## B91.3GX 295                 0  B91 3GX 52.40486 -1.775618         17 5827.421
## BH178AN 675                 0  BH178AN 50.75126 -1.961944        128 5643.541
##                 x true_positive warning.score        low       upp p.value
## B14.6TN -63.43771         FALSE     0.1447099 0.13227219 0.1580905       1
## B15.2BQ -63.27718         FALSE     0.1205594 0.11015663 0.1317875       1
## B42.2RZ -62.74949         FALSE     0.1111463 0.10050152 0.1227496       1
## B61.0DB -68.73608         FALSE     0.1444548 0.12425147 0.1672508       1
## B91.3GX -59.17484         FALSE     0.1845004 0.16971118 0.2002541       1
## BH178AN -73.64748         FALSE     0.1137885 0.09235665 0.1392876       1
```

Generate a second set of warning scores using slightly larger covering cylinders (`cylinders2`).

```
#set.seed(1)
load("random_state_seed_2.RData")
cylinders2 = CreateCylinders(observation.matrix = all.matrix, baseline.matrix =  b.matrix, emmtype = 'si
case.df[,c('warning.score2','low2','upp2','p.value2')] = t(apply(case.df, 1, FUN=warning.score, cylinde
```

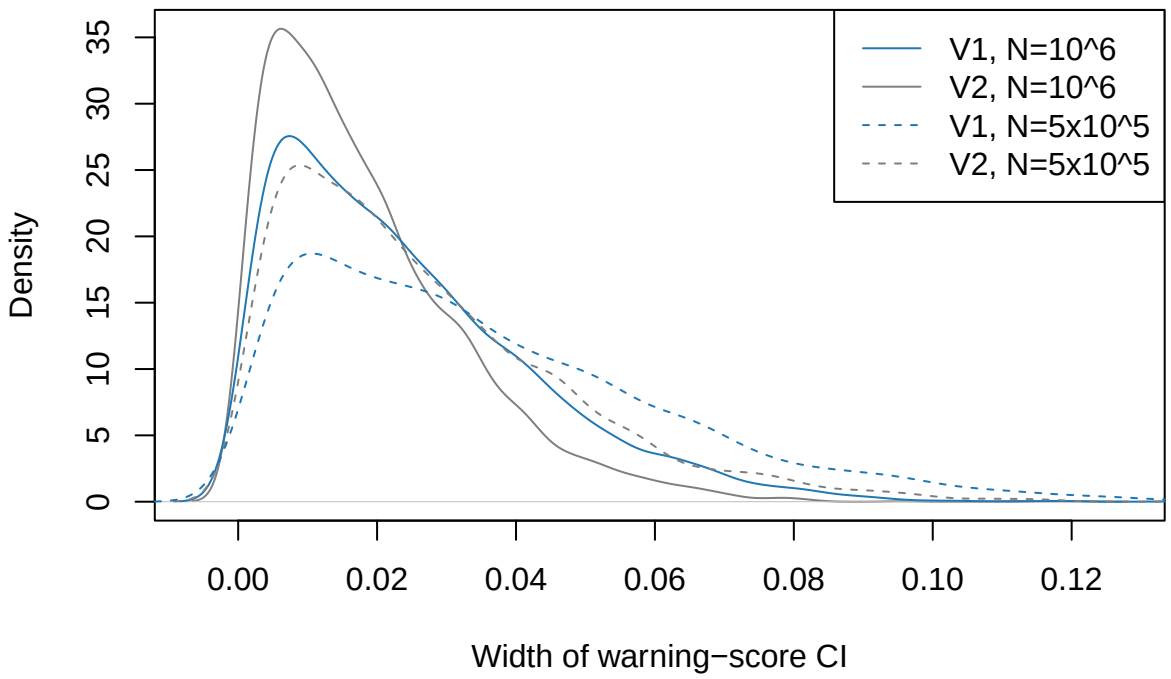
Let us perform other two replicates with fewer cylinders (`n.cylinders = 500000`).

```
cylinders_a = CreateCylinders(observation.matrix = all.matrix, baseline.matrix =  b.matrix, emmtype = '
ws_a = as.data.frame(t(apply(case.df, 1, FUN=warning.score, cylinders_a)))
cylinders_b = CreateCylinders(observation.matrix = all.matrix, baseline.matrix =  b.matrix, emmtype = '
ws_b = as.data.frame(t(apply(case.df, 1, FUN=warning.score, cylinders_b)))
names(ws_b) = c('warning.score', 'low','upp','p.value')
names(ws_a) = c('warning.score', 'low','upp','p.value')
```

Plot results

Plot widths of Wilson CIs.

```
# png(paste0('widths_plot', label.str, '.png'), width = 4.25, height = 3.25, units = 'in', res=600, poi
# par(mfrow=c(1,1), mar=c(4,4,1,1))
plot(density(case.df$upp2 - case.df$low2), main=NA, col='#7f7f7f', xlab='Width of warning-score CI')
lines(density(case.df$upp - case.df$low), col='#1f77b4')
lines(density(ws_b$upp - ws_b$low), col='#7f7f7f', lty=2)
lines(density(ws_a$upp - ws_a$low), col='#1f77b4', lty=2)
#
legend('topright', legend = c("V1, N=10^6", "V2, N=10^6","V1, N=5x10^5", "V2, N=5x10^5"), col=c('#1f77b
```




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

Compare the warning scores generated using different cylinder sizes.

```
#case.df.warning.score.old = case.df$warning.score
ctest = cor.test( case.df$warning.score2, case.df$warning.score)
ctest
```

```
##
##  Pearson's product-moment correlation
##
```

```
## data:  case.df$warning.score2 and case.df$warning.score
## t = 306.23, df = 4987, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9729860 0.9757904
## sample estimates:
##       cor
## 0.9744261
```

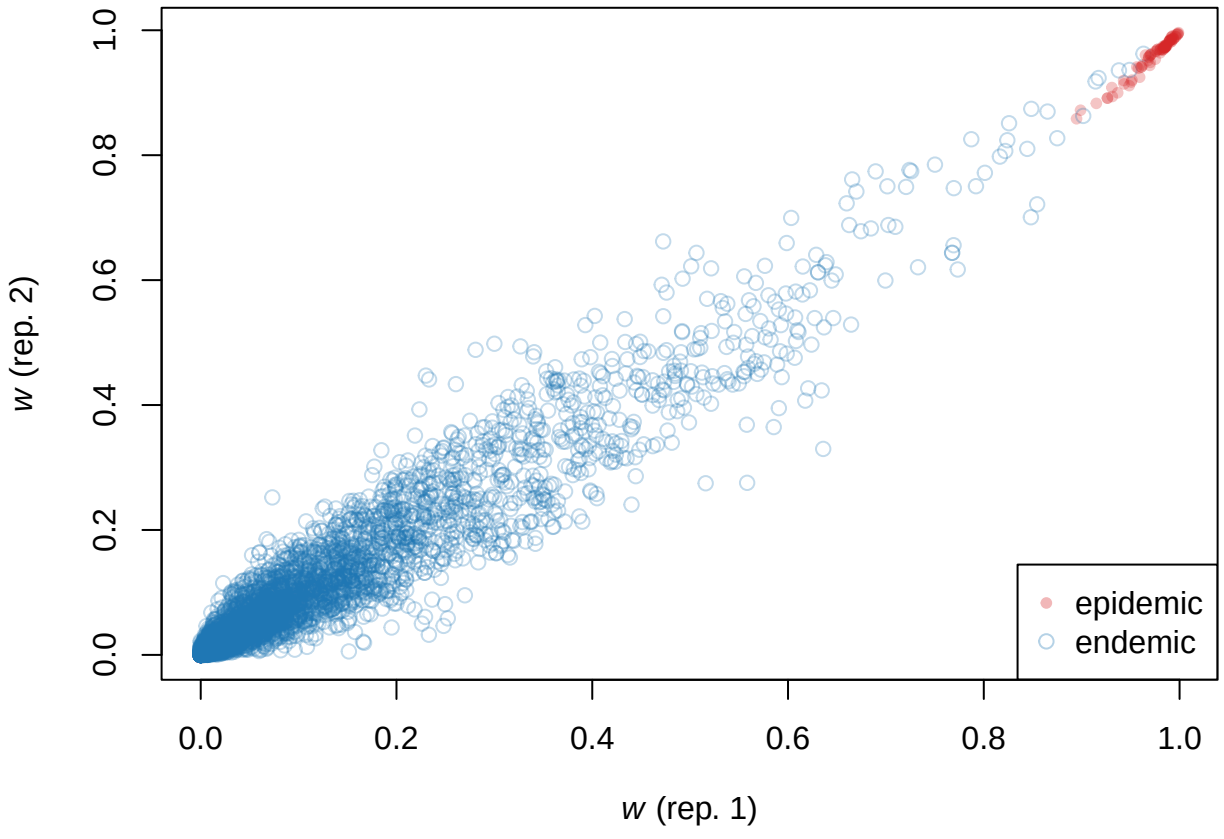
```
print(ctest$estimate)
```

```
##       cor
## 0.9744261
```

```
print(ctest$p.value)
```

```
## [1] 0
```

```
colors = ifelse(case.df$true_positive, "#d6272844", "#1f77b444")
# plot(case.df$warning.score, case.df$warning.score2,
#      xlab = 'replicate 1', ylab = 'replicate 2', col=colors, pch=ifelse(case.df$true_positive, 20,1))
#png(paste0('Fig/corrplot', label.str, '.png'), width = 3.25, height = 3.25, units = 'in', res=400, poi
par(mfrow=c(1,1), mar=c(4,4,1,1))
plot(case.df$warning.score, case.df$warning.score2,
     xlab = expression(paste(italic(w),' (rep. 1)')), ylab = expression(paste(italic(w),' (rep. 2)')),
legend('bottomright', c('epidemic','endemic'), col=c("#d6272855", "#1f77b455"), pch=c(20,1))
```




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

Assess performance of using the area under the Receiver Operating Characteristic curve (ROC-AUC).

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:spatstat.core':
##
##     auc, roc
```

```
## The following object is masked from 'package:spatstat.geom':
##
##     coords
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
ROC = roc(case.df$true_positive, case.df$warning.score)
```

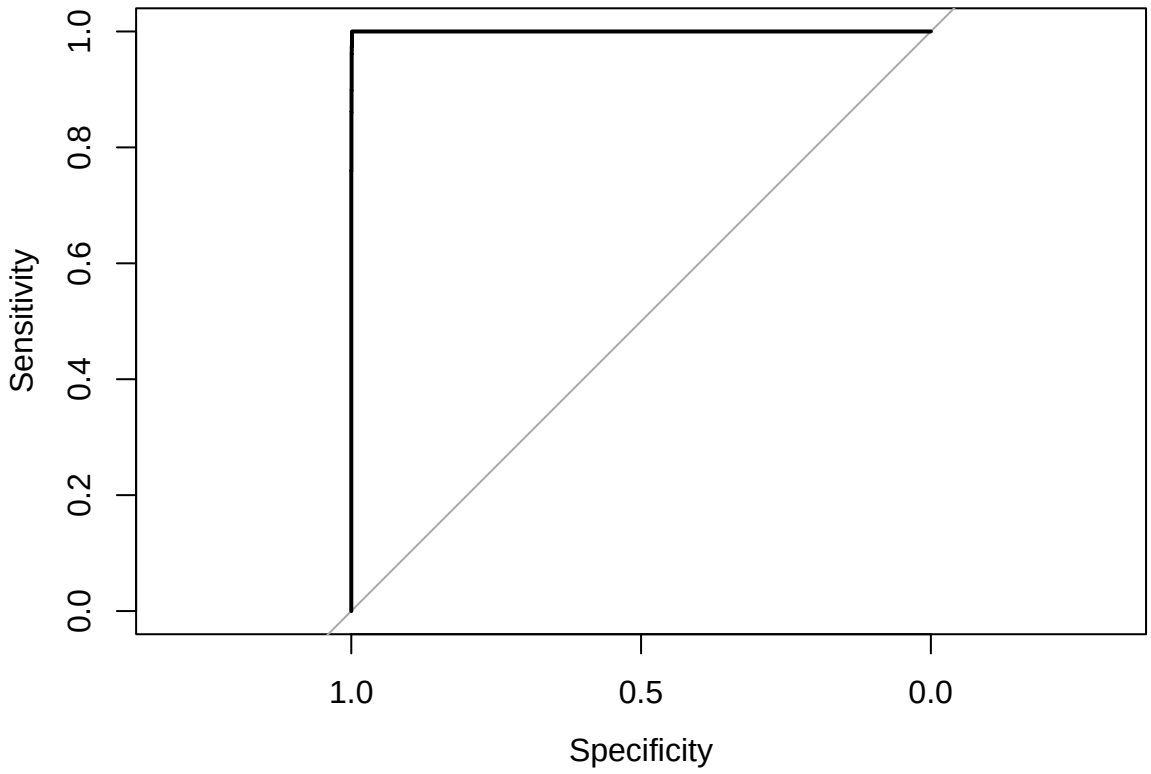
```
## Setting levels: control = FALSE, case = TRUE
```

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

```
print(ROC)
```

```
##
## Call:
## roc.default(response = case.df$true_positive, predictor = case.df$warning.score)
##
## Data: case.df$warning.score in 4910 controls (case.df$true_positive FALSE) < 79 cases (case.df$true_
## Area under the curve: 0.9999
```

```
plot(ROC)
```




```
L = NROW(case.df)
col = '#00000033'
#png(paste0('Fig/ROC_',label.str,'.png'), width = 3.25, height = 3.25, units = 'in', res=400, pointsize
par(mfrow=c(1,1), mar=c(4,4,0,0))
auc = numeric(0)
#plot(ROC, col='black', identity.col='black',  grid.col='red')
for (i in 1:5){
  cc = roc(true_positive ~ warning.score, case.df[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]]))
}
```

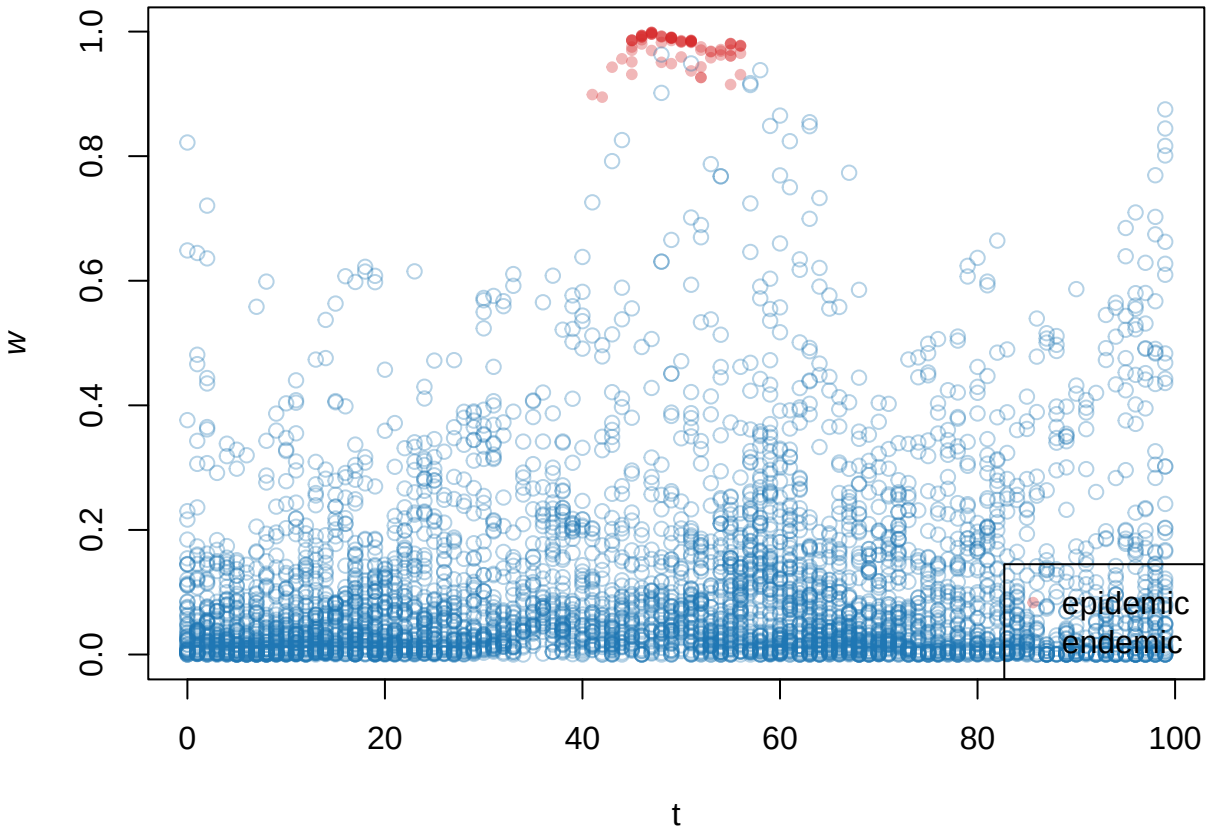
```
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
```

```
#dev.off()
print(quantile(auc, c(0.025,0.5,0.975) ))
```

```
##      2.5%       50%     97.5%
## 0.9998479 0.9998617 0.9999323
```

Plotting the warning scores vs time highlights when the outbreak happened.

```
# png(paste0('Fig/ws_time_',label.str,'.png'), width = 4 * 1.2, height = 3 * 1.2, units = 'in', res=400
par(mfrow=c(1,1), mar=c(4,4,1,1))
plot(case.df$SAMPLE_DT_numeric, case.df$warning.score, xlab='t',
     ylab=expression(italic(w)),
     col=ifelse(case.df$true_positive, "#d6272855", "#1f77b455"),
     pch=ifelse(case.df$true_positive, 20, 1))
legend('bottomright', c('epidemic','endemic'), col=c("#d6272855", "#1f77b455"),
       pch=c(20,1))
```

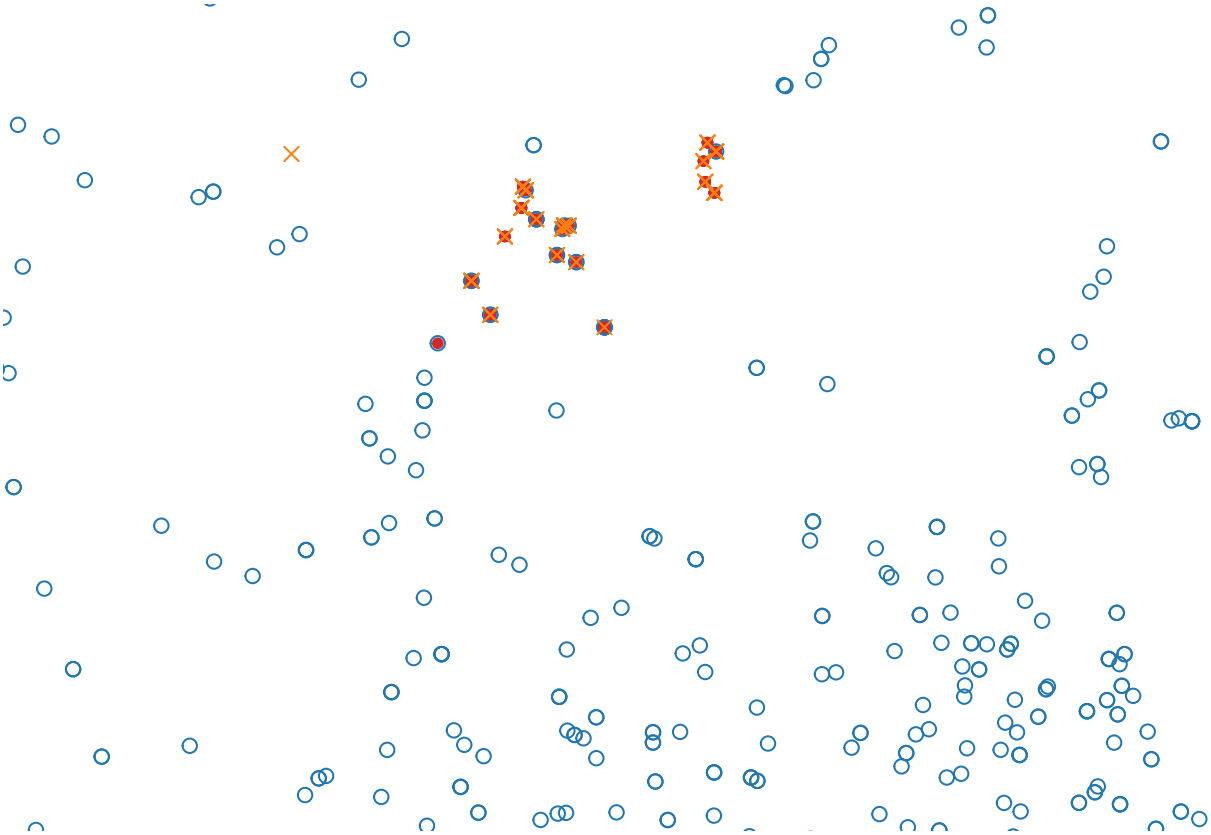



```
# dev.off()
```

Plot the location of detected cases and colour by their warning scores ($w > 0.95$ in orange).

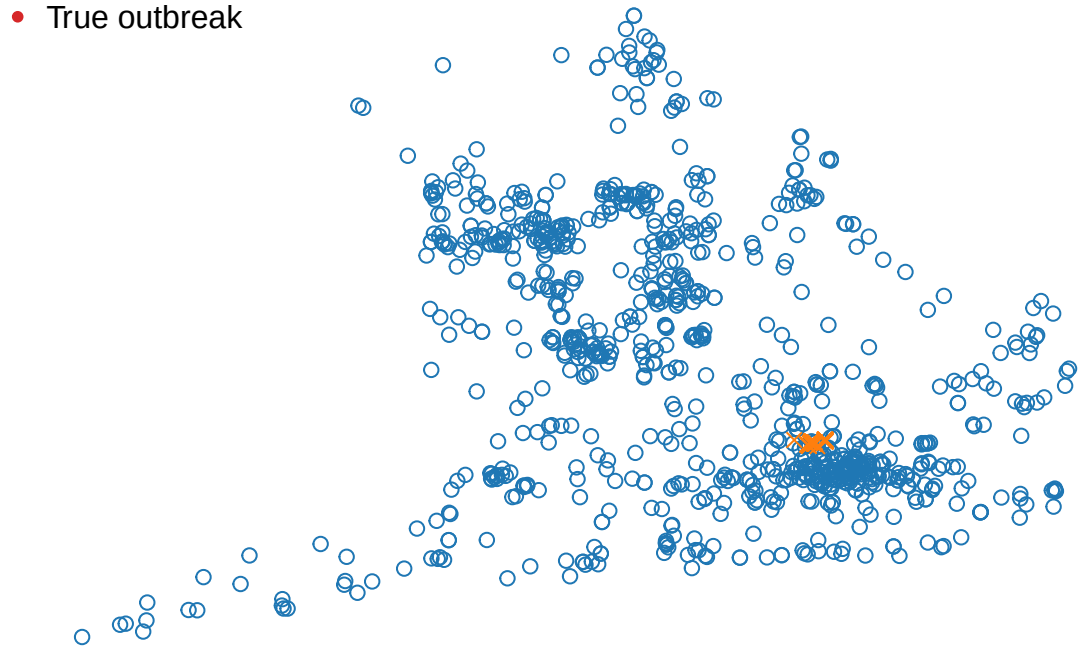
```
# palette = colorRampPalette(c('#1f77b4', '#d62728'))(100)
idx1 = (case.df$SAMPLE_DT_numeric > 40 ) & ( case.df$SAMPLE_DT_numeric < 60) & (case.df$warning.score <
#colors = palette[as.integer(case.df[idx,]$warning.score * 100)+1]
idx2 = (case.df$SAMPLE_DT_numeric > 40 ) & ( case.df$SAMPLE_DT_numeric < 60) & (case.df$warning.score >
# plot(case.df[idx1,]$longitude, case.df[idx1,]$latitude, col='#1f77b4',axes=F, pch=1, cex=1, ylab = NA
# points(case.df[idx2,]$longitude, case.df[idx2,]$latitude, col='#d62728', pch=20, cex=1, ylab = NA, xl
#
# png(paste0('Fig/satscan_',label.str,'.png'), width =  4 *1.2 / 1.5, height = 3 *1.2 /1.5, units = 'in
```

```r
par(mfrow = c(1, 1), mar=c(0,0,0,0) + 0.5)
idx3=case.df$warning.score > 0.95
plot(case.df[!idx3,]$longitude, case.df[!idx3,]$latitude, col='#1f77b4',axes=F, pch=1, cex=1, ylab = NA
points(case.df[case.df$true_positive,]$longitude, case.df[case.df$true_positive,]$latitude, col='#d62728
points(case.df[idx3,]$longitude, case.df[idx3,]$latitude, col='#ff7f0e', pch=4, cex=1, ylab = NA, xlab=
```



```r
# plotBaseMap(add=T, Wales = F, onlyregion = T)
# sim.col=st_read('shape_files/satscan_output.shp')
# plot(sim.col[1], add=T, col=NA, border='#7f7f7f')
# dev.off()
```

```r
# png(paste0('Fig/end_epi_spatial_',label.str,'.png'), width =  4 *1.6, height = 3 *1.6, units = 'in',
par(mfrow = c(1, 1), mar=c(1,1,1,1)+0.1)

plot(case.df[idx1,]$longitude, case.df[idx1,]$latitude, col='#1f77b4',axes=F, pch=1, cex=1, ylab = NA,
points(case.df[idx2,]$longitude, case.df[idx2,]$latitude, col='#ff7f0e', pch=4, cex=1, ylab = NA, xlab=
# plotBaseMap(add=T, Wales = F)
legend('topleft',
       c( expression(italic(w)>=0.95),
          expression(italic(w)<0.95),
          "True outbreak"), col=c("#ff7f0e", "#1f77b4", '#d62728'), pch=c(4,1,20), box.col = "white")
```




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

Test timeliness:

```
save(".Random.seed",file="random_state_seed_3.RData")
load("random_state_seed_3.RData")
for (week in 40:59){
  week.range = as.character(0:week)
  cylinders = CreateCylinders(observation.matrix = all.matrix[,week.range], baseline.matrix = b.matrix[
  case.df[,paste0('warning.score', as.character(week))] = apply(case.df, 1, FUN=warning.score, cylinder
}
```

```
library(viridisLite)
palette=inferno(length(40:59),begin=0.15, end=0.85, direction=-1)

plot(c(40,59), range(case.df[case.df$true_positive,11:30]), col='white', xlab = expression(tau),
     ylab=expression(italic(w)), axes = FALSE)
#abline(h=0.95, col='#7f7f7f')

for(i in 1:NROW(case.df[case.df$true_positive,  ])){
  t1 = case.df[case.df$true_positive,][i,'SAMPLE_DT_numeric']
  id = t1 - 40 + 18
  # 40 is the week of the start of the outbreak
  # 18 is the number of columns before the column `warning.score40`
  color = palette[t1-39]
  if (t1 <= 59){
  #  print(c(dim(case.df[case.df$true_positive,]),  i, id:31))
#    print( unlist(case.df[case.df$true_positive,][i, id:31]))
    lines(t1:59, case.df[case.df$true_positive,][i, id:37], col=color)
    points(t1:59, case.df[case.df$true_positive,][i, id:37], pch=20, cex=0.4, col=color)
  }
}
```

```
axis(2, at=c(0,0.25,0.5,0.75,1))
axis(1, at = seq(40,59))
```