Analysis code for *Developments in statistical inference when assessing spatiotemporal disease clustering with the tau statistic*

written by Timothy M Pollington and reviewed by Lloyd AC Chapman

April 2020

Contents

---

1 Citing this resource

Please kindly cite as:

@ARTICLE{Pollington2020, author={Pollington, T.M. and Tildesley, M.J. and Hollingsworth, T.D. and Chapman, L.A.C.},
year={2020},
title={{Developments in statistical inference when assessing spatiotemporal disease clustering with the tau statistic}},
journal={Spatial Statistics},
doi={10.1016/j.spasta.2020.100438},
}

2 Introduction

Our code implements *graphical hypothesis testing* of the *tau statistic* and examines how different aspects of implementation may affect the *endpoint of spatiotemporal clustering* and its *precision*, by comparing against the standard analysis (Lessler et al. 2016)—a measles dataset(Pfeilsticker 1863; Oesterle 1992; Neal and Roberts 2004) with the time-relatedness interval defined as $[T_1 = 0, T_2 = 14]$ and an overlapping distance band set.

3 How to use this document

This `.Rmd` document can be *knitted* in *RStudio*(RStudio Team 2020) to output a results `.pdf` file for easier printing or device accessibility. It is composed of chunks that all run when the document is knit to print the results, or they can be run separately in *RStudio* (note: they may depend on variables from earlier chunks having been run, in which case run the previous chunks using the ↓ button first, found at the top of the chunk of interest). A References section is provided. Other sections are cross-linked throughout using internal links.

Cloning this GitHub repository makes sense if you want to easily amend code. If you find errors please contact us using an Issue request before doing a Pull request to save you time, in case we can easily answer it first.

For reproducibility we provide the `set.seed` numbers for initialising any use of random number generations in the analysis from jittering to bootstrap sampling. Major outputs are also saved in the `/intrmd8`.

For computer-intensive chunks we indicate typical runtimes[1] and load the pre-run graph instead, and use `eval = FALSE` to skip the chunk; this can be relaxed by changing to `eval = TRUE` for the chunk(s) of

interest.[1] We also do data checks and tests to show the code runs as expected. All code was run in R v3.6.3(R Core Team 2020) on *RStudio*® v1.2.5033 in Microsoft® Windows® 10.0.18363.

4 Prerequisites before your first knit of this document

You will need to install Rtools, the following R packages and be running *RStudio* `as administrator`; this is useful too in modifying the file structure of your cloned repository from within *RStudio*.

```
# install.packages(pkgs = c("IDSpatialStats","surveillance","latex2exp",
# "scales","GET","devtools","coxed","testthat"))
# You'll need to copy and paste them (uncommented) into your R console to run.
```

Correct knitting of this document requires cloning its GitHub repository into an *RStudio* project (*RStudio* cmds: `File > New Project > Version Control > Git > repository URL = https://github.com/t-pollington/developments_tau_statistic`). Furthermore to use the tau functions the `tauodds` (internal) package from within this repository needs to be built and installed before use (*RStudio* cmds: `Build > Configure Build Tools` and setting package directory to `.../measles/tauodds` and then in the top-right Build pane choose `Install and Restart`); this is why we require `devtools` package. Sometimes if an error appears when building in *RStudio*'s Build pane then you will need to check *RStudio* is running the 64-bit R in `Tools > Global Options > General> Basic tab > R Sessions > Rversion box. Also in the Build pane just choosing`More > Clean & Rebuild`can fix the problem. We hope that these amended tau functions will be integrated into`IDSpatialStats' package, via pull requests by us over the course of the next few months.

If you haven't compiled LaTeX before on your Windows OS then downloading the `tinytex` package and running `tinytex::install_tinytex()` can help. Also amend the `setup` chunk according to your filepaths for the repository, note these are often different across OS.

```
knitr::opts_chunk$set(echo = TRUE)
```

5 Descriptive analyses on the Hagelloch measles dataset

5.1 Dataset check

We use the open access Hagelloch dataset from the `surveillance` R package(Höhle et al. 2020). Like Lessler et al's shared code (unpublished) we take the start of the prodromal period (`tPRO`) as the start of onset. There are 188 cases, one data row per case. In a reduced version of the dataset (`X`), all five variables have reasonable ranges and data types as shown below.

```
rm(list = ls()) # clear the R workspace
figspath = "C:/Users/mochu/Documents/GitHub/developments_tau_statistic/figs"
intrmd8path = "C:/Users/mochu/Documents/GitHub/developments_tau_statistic/intrmd8"
data("hagelloch", package = "surveillance")
library(surveillance)
```

```
# help("hagelloch") # uncomment to load dataset info in RStudio's Help pane/browser if
# knitted
rm(hagelloch) # additional time series object that loads is not required
```

[1] based on a Dell™ Precision M2800 laptop with Intel® Core™ i7-4810MQ CPU @ 2 · 80GHz × 8 with 16MB RAM

```r
# X = {house no, patient no, x/y-location, prodromal start}
X = subset(hagelloch.df, select = c("HN","PN","x.loc","y.loc","tPRO"))
dim(X)
```

```
## [1] 188   5
```

```r
str(X) # data types as expected
```

```
## 'data.frame':    188 obs. of  5 variables:
##  $ HN   : int  61 61 61 62 63 63 23 69 69 31 ...
##  $ PN   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ x.loc: num  142 142 142 165 145 ...
##  $ y.loc: num  100 100 100 102 120 ...
##  $ tPRO : num  22.7 24.2 29.6 28.1 23.1 ...
```

```r
length(unique(X$HN)) # 56 case hhlds
```

```
## [1] 56
```

```r
length(unique(X$PN)) == length(X$PN) # all PN unique. PASS
```

```
## [1] TRUE
```

```r
apply(X,2,range) # no gaps in the PNs and reasonable ranges for the others PASS
```

```
##      HN  PN x.loc y.loc       tPRO
## [1,]  2   1   7.5     5  0.7349822
## [2,] 80 188 280.0   240 86.6882978
```

```r
sum(is.na(X)) + sum(is.null(X)) + sum(apply(X, 2, is.nan)) # PASS
```

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

```r
# Remove PN as all found to be unique.
X = subset(hagelloch.df, select = c("x.loc","y.loc","tPRO","HN"))
```

5.2 About the epidemic

```r
SI.mean = 14.9 # mean serial interval (days) from Cori et al 2013, Table 1, p1507
tPRO.range = c(floor(min(X$tPRO)), ceiling(max(X$tPRO)))
round(diff(range(X$tPRO))/SI.mean) # approximate number of generations covered
```

```
## [1] 6
```

The epidemic lasts nearly 3 months which could have covered up to ~6 disease generations based on estimated serial intervals for this specific epidemic(Cori et al. 2013). Five generations can be discerned from the epidemic curve of this propagated epidemic.

```
setwd(figspath)
pdf("Re.pdf")
hist(X$tPRO, xlab="Study time (days)", ylab="Measles cases", cex.lab = 1.5, main=NULL,
breaks = seq.int(tPRO.range[1],tPRO.range[2],by = 1), xaxs="i", yaxs="i", xaxt = "n",
yaxt = "n", col = "ivory1", panel.first = {
  grid(NA,10,lty = 3,lwd = 0.5, col = "grey")
})
axis(2, at=c(0,5,10,15,20), labels=c(0,5,10,15,20), col.axis="black", las=1, lwd = 4)
axis(1, at=c(0,23,47,60,80,87), labels=c(0,23,47,60,80,87), col.axis="black", las=1,
     lwd = 4)
dev.off()
```

5.3 Spatiotemporal plots

```
# generate STplot----
# define palette----
set.seed(seed = 2)
df = data.frame(x = X$x.loc, y = X$y.loc, t = X$tPRO)
rbPal = colorRampPalette(colors = c("red","orange","yellow","green","blue","purple",
                                    "violet")) # plot rainbow colours
df$col = rbPal(10)[as.numeric(cut(df$t, breaks = 10))] # coloured deciles

setwd(figspath)
pdf("STplot.pdf")
# jitter in x & y separately, using Uniform[-2,+2] distribution .
par(las = 1)
plot(jitter(X$x.loc,amount = 5), jitter(X$y.loc,amount = 5), col = df$col, main = NULL,
xlab = "", ylab = "", xaxs="i", yaxs="i", xaxt = "n", yaxt = "n",
pch = 20, cex = 2, cex.lab = 1.5, xlim = c(0,(max(X$x.loc)+5)),
ylim = c(0,(max(X$y.loc)+5)), lwd = 4)
axis(2, las=1, at=c(0,50,100,150,200,240), labels = c("0","50","100","150","200","240"),
     lwd = 4)
axis(1, las=1, at=c(0,50,100,150,200,250,280), labels = c("0","50","100","150","200",
                                                          "250","280"), lwd = 4)
par(las = 0)
mtext(latex2exp::TeX('$y$ (m)'), side=2, line=3, cex = 1.5, padj = 1)
mtext(latex2exp::TeX('$x$ (m)'), side=1, line=3, cex = 1.5)
par(xpd=TRUE)
legend.text = c("1-9","10-19","20-28","29-38","39-47","48-57","58-67","68-76","77-86",
"87")
legend(x = 5,y = 235, title = "Onset time decile (days)", legend = legend.text,
col  = rbPal(10), pch = 20, cex = 1.05, pt.cex = 2, ncol = 2)
dev.off()
```

The study window was a ~280 m x 240 m rectangle. Households have single or multiple cases. The jittered plot shows that household cases often have very similar onsets because they share the same colour. Nearby houses also tend to share similar onsets to their neighbours but there are exceptions; the range in which the onsets are similar is likely of the order of tens of metres.

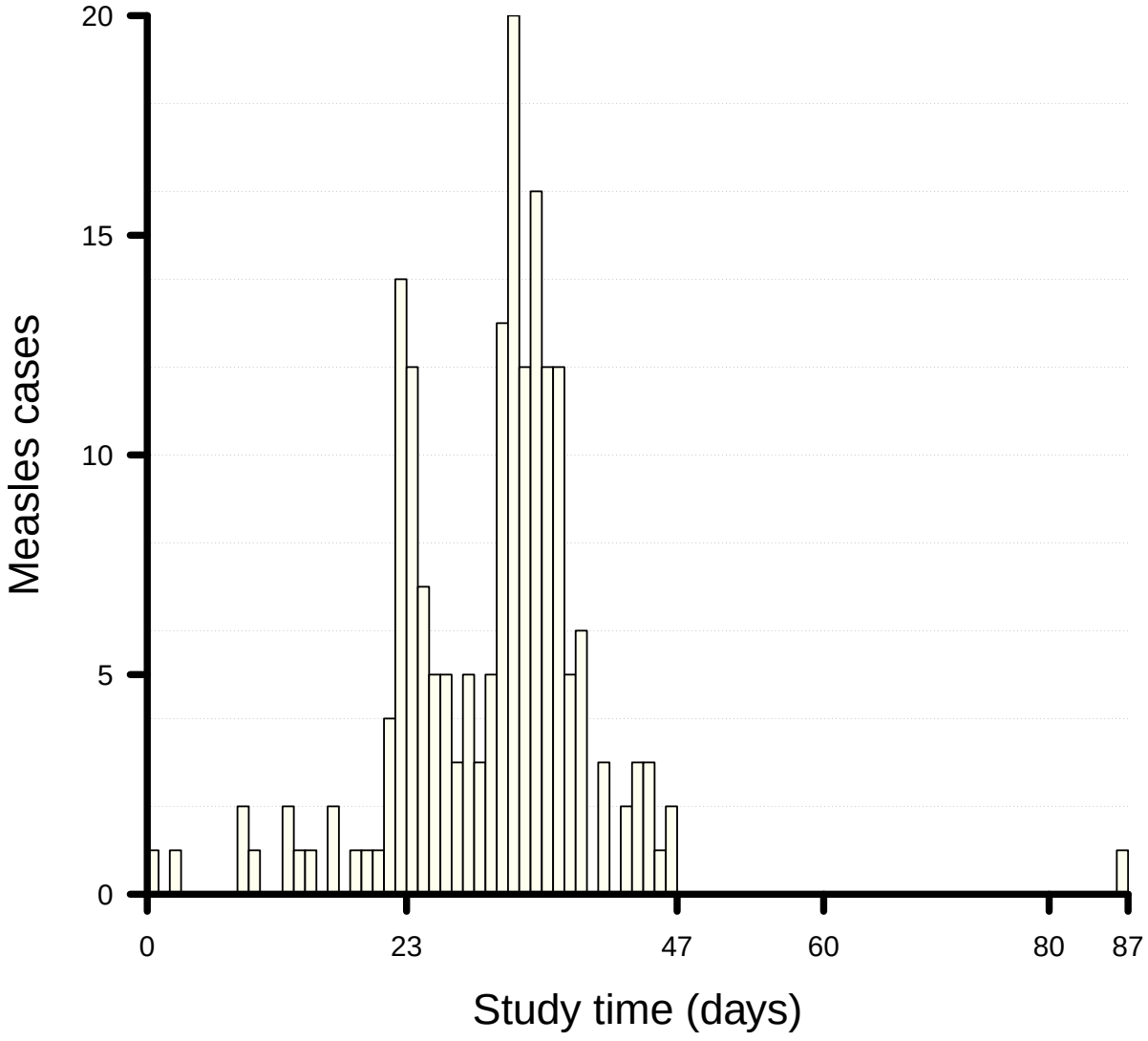


Figure 1: Epidemic curve of the 188 measles cases in Hagelloch in 1861.

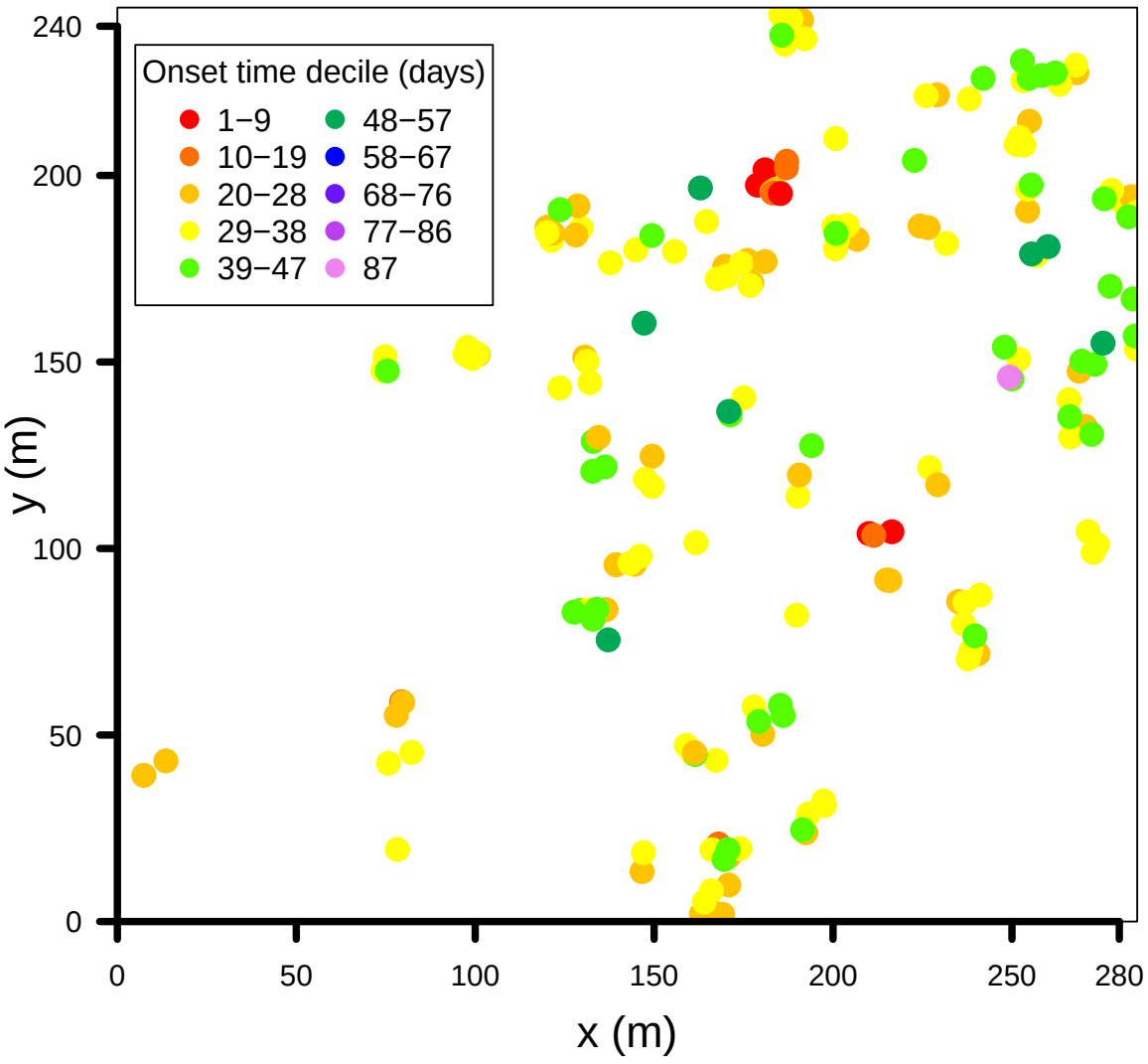


Figure 2: Spatial plot of cases' locations (jittered) with colour marks for onset date

```r
# fig_label() from @January2017 for later figure labelling----
fig_label <- function(text, region="figure", pos="topleft", cex=NULL, ...) {
  region <- match.arg(region, c("figure", "plot", "device"))
  pos <- match.arg(pos, c("topleft", "top", "topright",
                          "left", "center", "right",
                          "bottomleft", "bottom", "bottomright"))
  if(region %in% c("figure", "device")) {
    ds <- dev.size("in")
    # xy coordinates of device corners in user coordinates
    x <- grconvertX(c(0, ds[1]), from="in", to="user")
    y <- grconvertY(c(0, ds[2]), from="in", to="user")
    # fragment of the device we use to plot
    if(region == "figure") {
      # account for the fragment of the device that
      # the figure is using
      fig <- par("fig")
      dx <- (x[2] - x[1])
      dy <- (y[2] - y[1])
      x <- x[1] + dx * fig[1:2]
      y <- y[1] + dy * fig[3:4]
    }
  }
  # much simpler if in plotting region
  if(region == "plot") {
    u <- par("usr")
    x <- u[1:2]
    y <- u[3:4]
  }
  sw <- strwidth(text, cex=cex) * 60/100
  sh <- strheight(text, cex=cex) * 60/100
  x1 <- switch(pos,
    topleft     =x[1] + sw,
    left        =x[1] + sw,
    bottomleft  =x[1] + sw,
    top         =(x[1] + x[2])/2,
    center      =(x[1] + x[2])/2,
    bottom      =(x[1] + x[2])/2,
    topright    =x[2] - sw,
    right       =x[2] - sw,
    bottomright =x[2] - sw)
  y1 <- switch(pos,
    topleft     =y[2] - sh,
    top         =y[2] - sh,
    topright    =y[2] - sh,
    left        =(y[1] + y[2])/2,
    center      =(y[1] + y[2])/2,
    right       =(y[1] + y[2])/2,
    bottomleft  =y[1] + sh,
    bottom      =y[1] + sh,
    bottomright =y[1] + sh)
  old.par <- par(xpd=NA)
  on.exit(par(old.par))
  text(x1, y1, text, cex=cex, ...)
```

```
  return(invisible(c(x,y)))
}
```

Third-party code in the `figlabel` chunk assists us with figure labelling(January 2017).

6 Tau functions

The actual tau functions (coded in the C language) can be found in `./tauodds/src/*.cpp` based on adaptations from an older fork (v0.3.7) of the `IDSpatialStats` package (Giles, Salje, and Lessler 2018).

6.1 Defining tau functions

```
# faster version of IDSpatialStats::get.tau()----
summonTau = function(X.region, r.min, r.max, T1, T2){
  tau = tauodds::getTau2OddsMeasles(X.region[,"x"], X.region[,"y"], X.region[,"tPRO"],
  r.min, r.max, as.integer(1:nrow(X.region)), T1, T2)
  return(tau)
}
# faster version of IDSpatialStats::get.tau.bootstrap() using resampled-index spatial----
# bootstrap
summonTauBstrap = function(X.region, r.min, r.max, bootiters, T1, T2){
  tauboots = matrix(NA, nrow = bootiters, ncol = length(r.max))
  for (i in 1:bootiters) {
    inds = sample(nrow(X.region), replace = TRUE)
    tauboots[i,] = tauodds::getTau2OddsMeasles(X.region[,"x"], X.region[,"y"],
    X.region[,"tPRO"], r.min, r.max, as.integer(inds), T1, T2)
  }
  return(tauboots)
}
# faster version of IDSpatialStats::get.tau.bootstrap() using resampled-index spatial----
# bootstrap which also ignores and repeats bootstrapped estimate if any Infs are computed.
summonTauBstrapnoinfs = function(X.region, r.min, r.max, bootiters, T1, T2){
  tauboots = matrix(NA, nrow = bootiters, ncol = length(r.max))
  i = 1
  while (i <= bootiters) {
    inds = sample(nrow(X.region), replace = TRUE)
    tauboots[i,] = tauodds::getTau2OddsMeasles(X.region[,"x"], X.region[,"y"],
    X.region[,"tPRO"], r.min, r.max, as.integer(inds), T1, T2)
    if(sum(is.infinite(tauboots[i,]))==0){
      i = i + 1
    }
  }
  return(tauboots)
}
# faster version of IDSpatialStats::get.tau.bootstrap() using Loh & Stein's marked----
# point bootstrap
summonTauBstraploh = function(X.region, r.min, r.max, bootiters, T1, T2){
tauboots = matrix(NA, nrow = bootiters, ncol = length(r.max))
for (i in 1:bootiters) {
  inds = sample(nrow(X.region), replace = TRUE)
```

```
  tauboots[i,] = tauodds::getTau2Loh(X.region[,"x"], X.region[,"y"], X.region[,"tPRO"],
  r.min, r.max, as.integer(inds), T1, T2)
}
return(tauboots)
}
# faster version of IDSpatialStats::get.tau.bootstrap() using modified Loh & Stein's
# marked point bootstrap----
summonTauBstraplohv2 = function(X.region, r.min, r.max, bootiters, T1, T2){
tauboots = matrix(NA, nrow = bootiters, ncol = length(r.max))
for (i in 1:bootiters) {
  inds = sample(nrow(X.region),replace = TRUE)
  tauboots[i,] = tauodds::getTau2Lohv2(X.region[,"x"], X.region[,"y"], X.region[,"tPRO"],
  r.min, r.max, as.integer(inds), T1, T2)
}
return(tauboots)
}
# obtains percentile CIs from summonTauBstrap() result----
summonTauCI <- function(tauboots,r.max){
  tau.ci = matrix(nrow=2, ncol=length(r.max))
  for (i in 1:length(r.max)) {
    tau.ci[,i] = quantile(tauboots[,i], probs=c(0.025, 0.975), type = 7)
  }
  return(tau.ci)
}
```

6.2 Comparing IDSpatialStats::get.tau() & tauodds::getTau2OddsMeasles()

```
testthat::test_that("comparing point estimator functions",{
setwd(intrmd8path)
load(file = "tau.hagg.RData") # created later in the "validate tau functions" chunk
dist.gap = 50
r.max = seq(10,120,2)
r.min = r.max-dist.gap
r.min[which(r.min < 0)] = 0
tau.hagg1 = tauodds::getTau2OddsMeasles(X$x.loc, X$y.loc, X$tPRO, r.min, r.max, index = as.integer(1:nr
testthat::expect_equal(tau.hagg, tau.hagg1, tolerance=1e-07) # equal up to 1e-07 but
# results are different at 1e-08.
})
```

6.3 Toy problem to validate the bootstrap methods

We use a toy system of 5 cases to test `IDSpatialStats::get.tau.bootstrap()` against the tau functions. `testthat` package (Wickham 2011) will throw an error if these functions do not produce the expected output, otherwise they will complete silently.

```
# specify the toy system----
toy = matrix(c(0,0,1,0,1,2,1,0,5,1,1,14,1,2,9), nrow = 5, ncol = 3,
dimnames = list(NULL,c("x","y","t")), byrow = TRUE)
toy.r.max = c(1.1,2) # two distance bands
toy.r.min = c(0,1.1)
```

```
hagg.func <- function(a, b, tlimit=4){
  # with a time-relatedness interval of [T_1 = 0, T_2 = 4]
  if(abs(a[3]-b[3]) <= tlimit){rc=1}
  else{rc=2}
  return(rc)
}

# run Lessler's and our function on this----
testthat::test_that("comparing Lessler RISB, RISB, MPSB and MMPSB, against non-computer calcs",{
set.seed(seed = 1)
lessler.RISB = IDSpatialStats::get.tau.bootstrap(toy, hagg.func, r = toy.r.max,
  r.low = toy.r.min, boot.iter = 1, comparison.type = "independent") # the inds that
  # get.tau.bootstrap() used were read from additional print statements added to their
  # code and input into ours.
testthat::expect_equal(object = lessler.RISB[,3], expected = c(3.5,0.0)) #PASS

inds = c(1,4,1,2,5) # this is the inds that is used for get.tau.bootstrap as printed off
  # when it had an extra printf statement in the get.tau.bootstrap program
pollington.RISB = tauodds::getTau2OddsMeasles(toy[,"x"], toy[,"y"], toy[,"t"], toy.r.min,
  toy.r.max, as.integer(inds), 0, 4)
testthat::expect_equal(object = pollington.RISB, expected = c(3.5,0.0), tolerance = 1e-06) #PASS

pollington.MPSB = tauodds::getTau2Loh(toy[,"x"], toy[,"y"], toy[,"t"], toy.r.min,
  toy.r.max, as.integer(inds), 0, 4)
testthat::expect_equal(object = pollington.MPSB, expected = c(0.5,0.25))

pollington.MMPSB = tauodds::getTau2Lohv2(toy[,"x"], toy[,"y"], toy[,"t"], toy.r.min,
  toy.r.max, as.integer(inds), 0, 4)
testthat::expect_equal(object = pollington.MMPSB, expected = c(13/7,13/35),
             tolerance = 1e-03)
})
```

6.4 Loading Lessler et al's measles setup to validate tau functions

```
# the following (unpublished) code is abridged from that kindly provided by Lessler et al.
# of their measles analysis----
hag.dat = cbind(hagelloch.df$x.loc, hagelloch.df$y.loc, hagelloch.df$tPRO)
colnames(hag.dat) = c("x","y","tPRO")

hagg.func <- function(a, b, tlimit=14){ # time-relatedness interval [T_1 = 0, T_2 = 14]
  if(abs(a[3]-b[3]) <= tlimit){rc=1}
  else{rc=2}
  return(rc)
}

dist.gap = 50
r.max = seq(10,120,2)
r.min = r.max-dist.gap
r.min[which(r.min < 0)] = 0
# r.mid = (r.max+r.min)/2 # note that plotting the midpoint is deprecated to reduce
# reader error of reading off the result and stating this as the *endpoint* of the
# distance band which this is not.
```

6.5 The baseline: validation against Lessler et al's published graph

```
# compare Lessler et al.'s measles analysis function output with ours----
tau.hagg = IDSpatialStats::get.tau(hag.dat, hagg.func, r=r.max, r.low=r.min,
comparison.type = "independent")$tau # note in IDSpatialStats functions the r.min & r.max
# order is swapped
set.seed(seed = 2)
ptm = proc.time()
tau.ci = IDSpatialStats::get.tau.ci(hag.dat, hagg.func, r=r.max, r.low=r.min,
boot.iter=100, comparison.type = "independent")[,4:5]
proc.time() - ptm # 39s
set.seed(seed = 2) # same seed for fair comparison
ptm = proc.time()
tauCItmp = summonTauBstrap(as.matrix(hag.dat), r.min, r.max, bootiters = 100, T1 = 0,
T2 = 14)
tauCI = summonTauCI(tauCItmp, r.max)
proc.time() - ptm # 1s
setwd(intrmd8path)
save(tau.hagg, file = "tau.hagg.RData")

# plot our reproduction of Lessler's analysis----
setwd(figspath)
pdf("taureproduction.pdf", width = 7*4, height = 7*4, pointsize = 12*4)
plot(r.max, tau.hagg, ylim=c(0.6,max(tau.ci[2,])+0.5), type="l", log="y", xlim=c(0,120),
yaxt="n", axes=FALSE, xaxs="i", yaxs="i", col = "black",
xlab = "Endpoint of the distance band, from an average case (m)",
ylab = "", lwd = 4, cex.lab = 1.5)
mtext(latex2exp::TeX('$\\tau (d_l,d_m)$'),side=2, line=2, cex = 1.5)
axis(2, las=1, at=c(0.6,1,2,5,12), labels = c("0·6","1","2·0","5·0","12·0"), lwd = 4)
axis(1, las=1, at=c(0,20,29,40,61,seq.int(80,120,20)),
     labels = c("0","20","29","40","61","80","100","120"), lwd = 4)
abline(h=1, lty=2, col="black", lwd = 4)
lines(r.max, tauCI[1,], lty = 2, col = "green", lwd = 4)
lines(r.max, tauCI[2,], lty = 2, col = "green", lwd = 4)
lines(r.max, tau.ci[,1], lty = 3, col = "slategrey", lwd = 4)
lines(r.max, tau.ci[,2], lty = 3, col = "slategrey", lwd = 4)
abline(v = 29, col = "red", lwd = 4)
abline(v = 61, col = "red", lty = 2, lwd = 4)
legend(x = 35, y = 8, bg = "white",
       legend=c(bquote(hat(tau) ~ "point estimate"),
  latex2exp::TeX('95% pointwise percentile CI of $\\hat{\\tau}$ (our function)'),
  latex2exp::TeX(
    '95% pointwise percentile CI of $\\hat{\\tau}$ ($\\textit{IDSpatialStats}$)'),
  "End of clustering (defined by Lessler et al.)",
  latex2exp::TeX('Clustering endpoint point estimate, $\\hat{D}$ (our definition)')),
col=c("black", "green", "slategrey", "red", "red"), lty=c(1,2,3,1,2), cex=1.05,
yjust = 0.5, lwd = 6)
dev.off()

# graphical abstract version----
pdf("taureproduction.ga.pdf", width = 7*4, height = 7*4, pointsize = 12*4)
plot(r.max, tau.hagg, ylim=c(0.6,max(tau.ci[2,])+0.5), type="l", xlim=c(0,100), yaxt="n",
     axes=FALSE, xaxs="i", yaxs="i", col = "black", log = "y",
```

```
     xlab = latex2exp::TeX('Distance from average case, $d$ (m)'), ylab = "",
     lwd = 8, cex.lab = 2)
mtext(latex2exp::TeX('$\\tau$'), side = 2, cex = 2, las = 1, line = 2, lwd = 4)
axis(2, las=1, at=c(0.6,1,12.0), labels = c("0·6","1","12"), cex = 2, lwd = 4)
axis(1, las=1, at = c(0,50,100), labels = c("0","50","100"), cex = 2, lwd = 4)
abline(h=1, lty=2, col=1, lwd = 4)
lines(r.max, tauCI[1,], col = "slategrey", lwd = 8)
lines(r.max, tauCI[2,], col = "slategrey", lwd = 8)
dev.off()
```

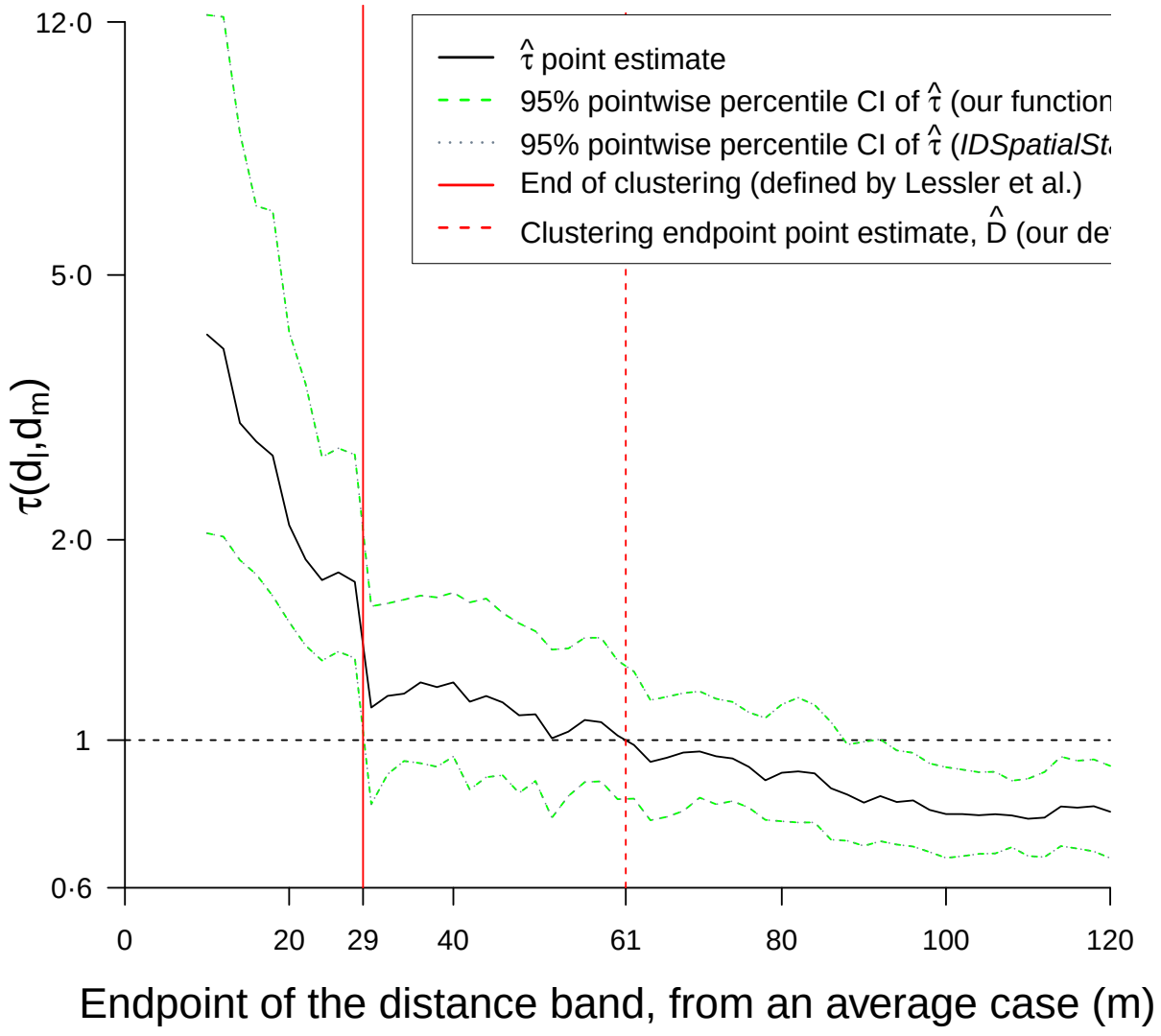


Figure 3: Reproducing Lessler et al.'s Fig4C using our own tau functions

We are confident that `seed = 2` produces a result identical to the original graph in (Lessler et al. 2016 Fig

4C p10/13) as the lower bound touches $14 \cdot 5$m which matches the "up to 15m" result in Lessler et al. and the CIs coincide perfectly. In fact due to the convention of plotting a distance band's midpoint this refers to clustering "up to 30m"; also the exact matchup of the confidence intervals shows that the faster code (`summonTauBstrap()` 1s, `IDSpatialStats` 39s) emulates the original.

7 Hypothesis testing

7.1 Global envelope tests

We construct a *global envelope* around the null hypothesis H_0 = no spatial clustering nor inhibition represented by $\tau = 1$ (simulated by time-mark permutations of the dataset). We assess if a region exists where the tau point estimate is above the upper bound or below the lower bound (as a two-tailed test) of this null envelope, using 2,500 iterations. A two-sided test is necessary as until you plot the graph you don't know whether there is clustering, inhibition or both. Unlike for indicative purposes of plotting a tau-distance graph or parameter estimation that we see later, when performing a global hypothesis test we should use a distance band set that covers the entire pairwise distances; first we check at what distance the majority of null simulations unacceptably diverge from $\tau = 1$. We then apply the global envelope test on the range [0,220 m].

```
# determine range of entire distance band set
dist.cases = spatstat::crossdist(X$x.loc, X$y.loc, X$x.loc, X$y.loc)
range(dist.cases)
dist.gap = 50
r.max.entire = seq(10,320,2)
r.min.entire = r.max.entire-dist.gap
r.min.entire[which(r.min.entire < 0)] = 0

# generate 'time mark' permuted dataset----
# We continued to use IDSpatialStats::get.tau.permute() for this purpose
set.seed(seed = 4)
ptm = proc.time()
hag.permute = IDSpatialStats::get.tau.permute(posmat = as.matrix(hag.dat),
fun = hagg.func, r = r.max.entire, r.low = r.min.entire, permutations = 2510,
comparison.type = "independent") # draw 10 more than the 2500 permutations reqd as will
proc.time() - ptm # 34mins for entire distance band
sum(apply(hag.permute,1,is.infinite)) # 7 infs, need to drop some which contain Infs as
# the GET functions cannot run otherwise
setwd(intrmd8path)
save(hag.permute, file = "hag.permute.RData") #change filename here

# subset hag.permute to drop the infs and just get enough for 2500 permutations
j=3
count = 1
hag.permute.noinfs = matrix(0, nrow = 156, ncol = 2500)
while (j <= 2512 & count < 2501) {
  if(sum(is.infinite(hag.permute[,j]))==0){
    hag.permute.noinfs[,count] = hag.permute[,j]
    count = count + 1
  }
  j = j + 1
}
sum(apply(hag.permute.noinfs,1,is.infinite)) # 0 PASS
```

```r
# check that the null simulations is acceptably close to tau = 1
plot(r.max.entire, apply(hag.permute.noinfs, 1, median), type = "l", col = "red",
     ylim = c(0.5,10), log = "y", xaxs = "i", yaxs = "i", xlab = "d", xaxt = "n",
     ylab = latex2exp::TeX('$\\tau (d)$'), lwd = 4, cex.lab = 1.5)
lines(r.max.entire, apply(hag.permute.noinfs, 1, mean), col = "blue", lwd = 4)
lines(r.max.entire, apply(hag.permute.noinfs, 1, quantile, probs = 0.75),
      col = "green", lwd = 4)
lines(r.max.entire, apply(hag.permute.noinfs, 1, max),
      col = "darkgreen", lwd = 4)
lines(r.max.entire, apply(hag.permute.noinfs, 1, quantile, probs = 0.25),
      col = "darkorange", lwd = 4)
lines(r.max.entire, apply(hag.permute.noinfs, 1, min), col = "darkorange3", lwd = 4)
abline(h = 1, lty = 2, lwd = 4)
abline(v = 220, col = "darkgray", lwd = 4)
abline(h = seq(0.5,10,0.2), v = seq.int(50,300,50), lty = "dotted", lwd = 4
       col = "lightgray")
axis(1, las=1, at=c(0,50,100,150,200,220,250,300),
     labels = c("0","50","100","150","200","220","250","300"), lwd = 4)
legend(x = 30, y = 5, title = "Null simulations", ncol = 2,
       legend=c("maximum",
"upper quartile",
"mean",
"median",
"lower quartile",
"minimum",
latex2exp::TeX('$\\tau = 1$')),
col=c("darkgreen", "green", "blue", "red", "darkorange","darkorange3","black"),
lty=c(1,1,1,1,1,1,2), cex=1.05, yjust = 0.5, lwd = 3)
tau.hagg.entire = IDSpatialStats::get.tau(hag.dat, hagg.func, r=r.max.entire,
                                          r.low=r.min.entire,
                                          comparison.type = "independent")$tau

# redo above analysis with distance band set up to and incl. 220 m
r.max.220 = seq(10,220,2)
r.min.220 = r.max.220-dist.gap
r.min.220[which(r.min.220 < 0)] = 0
tau.hagg.220 = IDSpatialStats::get.tau(hag.dat, hagg.func, r=r.max.220,
                                          r.low=r.min.220,
                                          comparison.type = "independent")$tau

set.seed(seed = 4)
ptm = proc.time()
hag.permute.220 = IDSpatialStats::get.tau.permute(posmat = as.matrix(hag.dat),
fun = hagg.func, r = r.max.220, r.low = r.min.220, permutations = 2510,
comparison.type = "independent") # draw 10 more than the 2500 permutations reqd as will
proc.time() - ptm # 30mins for entire distance band
sum(apply(hag.permute.220,1,is.infinite)) # 0 infs, no need to drop Infs
hag.permute.220 = hag.permute.220[,c(-1,-2)]
hag.permute.220 = hag.permute.220[,1:2500]
setwd(intrmd8path)
save(hag.permute.220, file = "hag.permute220.RData") #change filename here
```

```
# GET analysis with entire distance band
# create curve set for null hypothesis for GET----
curveset = GET::create_curve_set(list(r = r.max.220, obs = tau.hagg.220,
                                      sim_m = as.matrix(hag.permute.220)))
res.cr = GET::global_envelope_test(curve_sets = curveset, type = "rank", alpha = 0.05,
alternative = c("two.sided"), ties = "erl", probs = c(0.025, 0.975), quantile.type = 7,
central = "median")
round(attr(res.cr,"p_interval"), digits = 3) # p-value range [0·000, 0·022]

# GET plot----
setwd(figspath)
pdf("get.pdf", width = 7*4, height = 7*4, pointsize = 12*4)
plot(NULL, xlim = c(0,220), ylim = c(min(res.cr$lo, res.cr$obs),max(res.cr$obs)),
     xaxt = "n", yaxt = "n", xaxs = "i", yaxs = "i", ylab = "", xlab = "", lwd = 4,
     cex.lab = 1.5)
mtext(latex2exp::TeX('$\\tau (d_l,d_m)$'), side=2, line=2, cex = 1.5)
mtext(latex2exp::TeX(
'Distance band endpoint ($d_m$)'), side=1, line=3, cex = 1.5)
mtext(latex2exp::TeX(
'from an average case (m)'), side=1, line=4, cex = 1.5)
for (i in 1:2500) {
  lines(r.max.220, hag.permute.220[,i], col = scales::alpha("grey", alpha = 0.3),
        lwd = 4)
}
axis(2, las=1, at=c(0.7,1,2,3,4), labels = c("0·7","1","2·0","3·0","4·0"), lwd = 4)
lines(res.cr$r, res.cr$lo, col = "slategrey", lwd = 4)
lines(res.cr$r, res.cr$hi, col = "slategrey", lwd = 4)
lines(res.cr$r, res.cr$central, col = "red", lwd = 4)
lines(res.cr$r, res.cr$obs, lwd = 4)
axis(1, las=1, at=c(0,50,100,150,200,220), labels = c("0","50","100","150","200","220"),
     lwd = 4)
abline(h=1, lty = 2, lwd = 4)
legend(x = 120, y = 3.4, legend=c(latex2exp::TeX('$\\hat{\\tau}$ point estimate'),
"95% global envelope",latex2exp::TeX('$H_0$ simulations'), "median simulation",
latex2exp::TeX('$\\tau = 1$')), col=c("black", "slategrey", "grey", "red", "black"),
lty=c(1,1,1,1,2), cex=1.05, yjust = 0.5, lwd = 6)
par(xpd = TRUE)
text(latex2exp::TeX('$\\textit{p}$-value$\\in\\lbrack$0,0·022$\\rbrack$'), x = 85, y = 1.4)
dev.off()

# graphical abstract version----
r.max1 = r.max.220[1:46]
res.cr1 = res.cr[1:46,]
hag.permute1 = hag.permute.220[1:46,]

pdf("get.ga.pdf", width = 7*4, height = 7*4, pointsize = 12*4)
plot(NULL, xlim = c(10,100), ylim = c(min(res.cr1$lo,res.cr1$obs),max(res.cr1$obs)),
     xaxt = "n",
yaxt = "n", xaxs = "i", yaxs = "i", ylab = "",
xlab = latex2exp::TeX(
'$d$ (m)'), lwd = 8, cex.lab = 2)
mtext(latex2exp::TeX('$\\tau$'), side = 2, cex = 2, las = 1, line = 2, lwd = 4)
for (i in 1:2500) {
```

```
  lines(r.max1, hag.permute1[,i], col = scales::alpha("grey", alpha = 0.3), lwd = 4)
}
axis(2, las=1, at=c(0.7,1,4), labels = c("0·7","1","4·0"), cex = 2, lwd = 4)
axis(1, las=1, at=c(10, 50, 100), labels = c("10","50","100"), cex = 2, lwd = 4)
lines(res.cr1$r, res.cr1$lo, col = "slategrey", lwd = 4)
lines(res.cr1$r, res.cr1$hi, col = "slategrey", lwd = 4)
lines(res.cr1$r, res.cr1$obs, lwd = 4)
lines(res.cr1$r[1:10], res.cr1$obs[1:10], lwd = 16)
lines(x = c(res.cr1$r[10],res.cr1$r[10]+0.5*(res.cr1$r[11] - res.cr1$r[10])), y = c(res.cr1$obs[10],res
      col = "black")
lines(res.cr1$r[44:46], res.cr1$obs[44:46], lwd = 4)
lines(x = c(10,100), y = c(1,1), lty = 2, lwd = 4)
par(xpd = TRUE)
mtext(side = 3, text = latex2exp::TeX('p-value$\\in\\lbrack$ 0,0·022 $\\rbrack$'),
outer = 0, cex = 2, lwd = 4)
dev.off()
```

8 Parameter estimation of $\hat{D}$—the endpoint of spatiotemporal clustering

Our `ciIntercept()` function identifies the first value of d where $\hat{\tau}(d) = 1$ for all $\underline{D}$ to generate `d.envelope`.

```
# ciIntercept() function----
ciIntercept <- function(n.sim, end.set, tau.sim, verbose = 1) {
  j.max = length(end.set)
  # now define d.envelope
  alwaysabove1 = 0
  d.envelope = NULL
  for (i in 1:n.sim) {
    j = 1
    if(tau.sim[i,j] > 1){ # else ignore simulation as starting from below tau = 1
      stillabove1 = TRUE
      while (stillabove1 & (j < j.max)) {
        j = j + 1
        if(tau.sim[i,j] <= 1){ # else it stays above tau = 1 until the next j is tested
          stillabove1 = FALSE
          root.tau1 = ((1-tau.sim[i,(j-1)])*(end.set[j]-end.set[j-1])/
                       (tau.sim[i,j]-tau.sim[i,(j-1)]))+end.set[j-1]
          d.envelope = c(d.envelope, root.tau1)
        }
      }
      if(stillabove1 & j==j.max){
        alwaysabove1 = alwaysabove1 + 1
      }
    }
  }
  # print warnings as if the value is much below 100% then a CI can't be constructed as
  # it has not been drawn from a random sample.
  if(verbose == 1){
  print(paste0("sims cross tau = 1 from above = ",length(d.envelope)/n.sim*100,"%"))
  print(paste0("alwaysabove1 = ",alwaysabove1/n.sim*100,"%"))
```

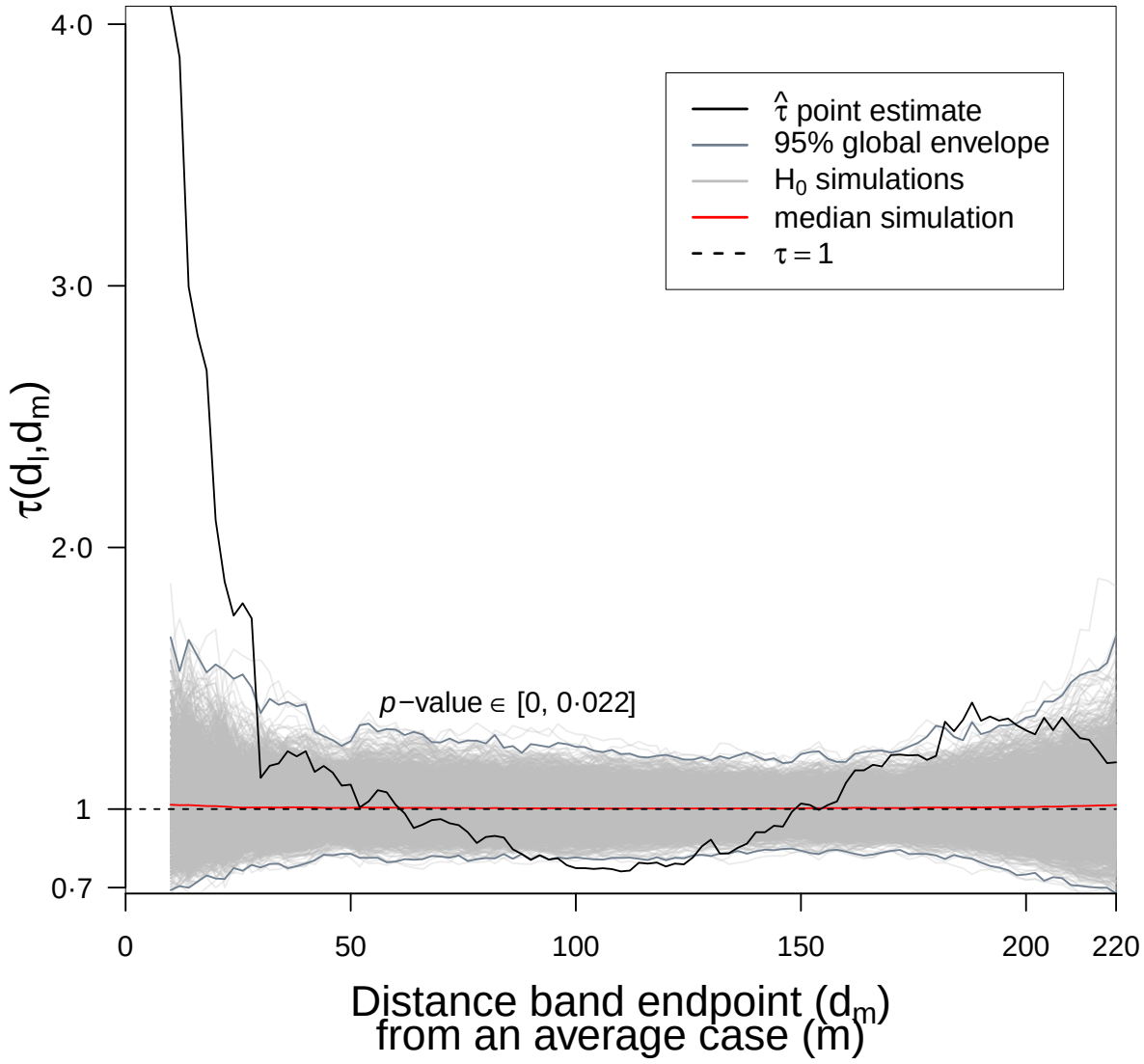


Figure 4: Global Envelope Test

```
  }
  return(d.envelope)
}
```

8.1 Number of bootstrap estimates

```
# generate 100 and 2500 bootstrap estimates----
set.seed(seed = 5)
ptm = proc.time()
tauCItmp2500noinfs = summonTauBstrapnoinfs(as.matrix(hag.dat), r.min, r.max,
bootiters = 2500, T1 = 0, T2 = 14)
proc.time() - ptm # 29s
setwd(intrmd8path)
save(tauCItmp2500noinfs, file = "tauCItmp2500noinfs.RData")
set.seed(seed = 6)
tauCItmp100noinfs = summonTauBstrapnoinfs(as.matrix(hag.dat), r.min, r.max,
bootiters = 100, T1 = 0, T2 = 14)
save(tauCItmp100noinfs, file = "tauCItmp100noinfs.RData")
```

```
# compute d.envelope by number of bootstrap estimates (100 or 2500)----
setwd(intrmd8path)
load(file = "tau.hagg.RData")
load(file = "tauCItmp2500noinfs.RData")
load(file = "tauCItmp100noinfs.RData")
d.envelope2500 = ciIntercept(2500, end.set = r.max, tau.sim = tauCItmp2500noinfs)
d.envelope100 = ciIntercept(100, end.set = r.max, tau.sim = tauCItmp100noinfs)
quantile(d.envelope2500, probs = c(0.025,0.975))
quantile(d.envelope100, probs = c(0.025,0.975))
save(d.envelope2500, file = "d.envelope2500.RData")
save(d.envelope100, file = "d.envelope100.RData")

# compute where on d-axis the point estimate intercepts tau(d) = 1----
firstbelow1 = which(tau.hagg < 1)[1] # when does the point estimate first fall below tau=1
dintercept.pointestimate = ((1-tau.hagg[firstbelow1-1])*
(r.max[firstbelow1]-r.max[firstbelow1-1])/
(tau.hagg[firstbelow1]-tau.hagg[firstbelow1-1]))+r.max[firstbelow1-1]
dintercept.pointestimate #print
save(dintercept.pointestimate, file = "dintercept.pointestimate.RData")

setwd(figspath)
pdf("nbstrap.pdf", width = 4*7, height = 4*7, pointsize = 4*12)
plot(NULL, xlim = c(0,120), log="y", ylim = c(0.5,93), xaxt = "n", yaxt = "n",
     xaxs = "i", yaxs = "i", ylab = "", xlab = "", lwd = 4, cex.lab = 1.5)
mtext(latex2exp::TeX('$\\tau (d_l,d_m)$'), side=2, line=2, cex = 1.5)
mtext(latex2exp::TeX(
'Distance band endpoint (1/2$(d_l + d_m)$)'), side=1, line=3, cex = 1.5)
mtext(latex2exp::TeX(
'from an average case (m)'), side=1, line=4, cex = 1.5)
for (i in 1:2500) {
  lines(r.max, tauCItmp2500noinfs[i,], col = scales::alpha("grey", alpha = 0.2), lwd = 4)
}
```

```
for (i in 1:100) {
  lines(r.max, tauCItmp100noinfs[i,], col = scales::alpha("green", alpha = 0.2), lwd = 4)
}
axis(2, las=1, at=c(0.5,1,2,4,16,32,64,93),
     labels = c("0·5","1","2·0","4·0","16·0","32·0","64·0","93·0"), lwd = 4)
axis(1, lwd = 4)
lines(x = c(0,120), y = c(1,1), lty = 2, lwd = 4) # as abline seems to overlap
par(lend=1);
lines(x = as.numeric(quantile(d.envelope2500, probs = c(0.025,0.975))), y=c(1.03,1.03),
type = "l", lwd = 20, col = "red")
lines(x = as.numeric(quantile(d.envelope100, probs = c(0.025,0.975))), y=c(0.97,0.97),
type = "l", lwd = 20, col = "blue")
lines(x=c(dintercept.pointestimate,dintercept.pointestimate), y = c(0.9,1.1), lwd = 8)
lines(r.max, tau.hagg, lwd = 4)
legend(x = 40, y = 32,
legend=c(latex2exp::TeX('$\\hat{\\tau}$ point estimate & $\\hat{D}$'),
latex2exp::TeX('$\\hat{\\tau}^*$ bootstrap estimates (N=2500)'),
latex2exp::TeX('    $\\bullet$  95% percentile CI of $\\hat{D}$'),
latex2exp::TeX('$\\hat{\\tau}^*$ bootstrap estimates (N=100)'),
latex2exp::TeX('    $\\bullet$  95% percentile CI of $\\hat{D}$'),
latex2exp::TeX('$\\tau = 1$')), col=c("black", "grey", "red", "green", "blue", "black"),
lty=c(1,1,1,1,1,2), lwd = c(6,6,30,6,30,6), pch = c(124,NA,NA,NA,NA,NA), cex=1.05,
yjust = 0.5)
dev.off()
```

Unexpectedly the number of bootstrap estimates of 100 versus 2500 don't appear to impact the precision of `d.envelope`; both CIs used 100% of simulations.

```
setwd(intrmd8path)
load("d.envelope100.RData")
load("d.envelope2500.RData")
load("dintercept.pointestimate.RData")
setwd(figspath)
pdf("bootstraphist.pdf")
par(mfrow = c(1,2))
hist(d.envelope100, breaks = seq.int(0,110,10), xaxs = "i", yaxs = "i", main = "N=100",
xlab = latex2exp::TeX('Samples of $\\hat{D}$ (m)'), las = 1, col = "ivory1")
abline(v = dintercept.pointestimate, lty = 2, lwd = 4, col = "red")
abline(v = mean(d.envelope100), lty = 2, lwd = 4, col = "green")
abline(v = median(d.envelope100), lty = 2, lwd = 4, col = "blue")
hist(d.envelope2500, breaks = seq.int(0,110,10), xaxs = "i", yaxs = "i", ylab = NULL,
main = "N=2500", xlab = latex2exp::TeX('Samples of $\\hat{D}$ (m)'), las = 1,
col = "ivory1")
abline(v = dintercept.pointestimate, lty = 2, lwd = 4, col = "red")
abline(v = mean(d.envelope2500), lty = 2, lwd = 4, col = "green")
abline(v = median(d.envelope2500), lty = 2, lwd = 4, col = "blue")
dev.off()
```

The asymmetric distribution of $\underline{D}$ suggests the usual *percentile confidence interval* would be a bad choice.

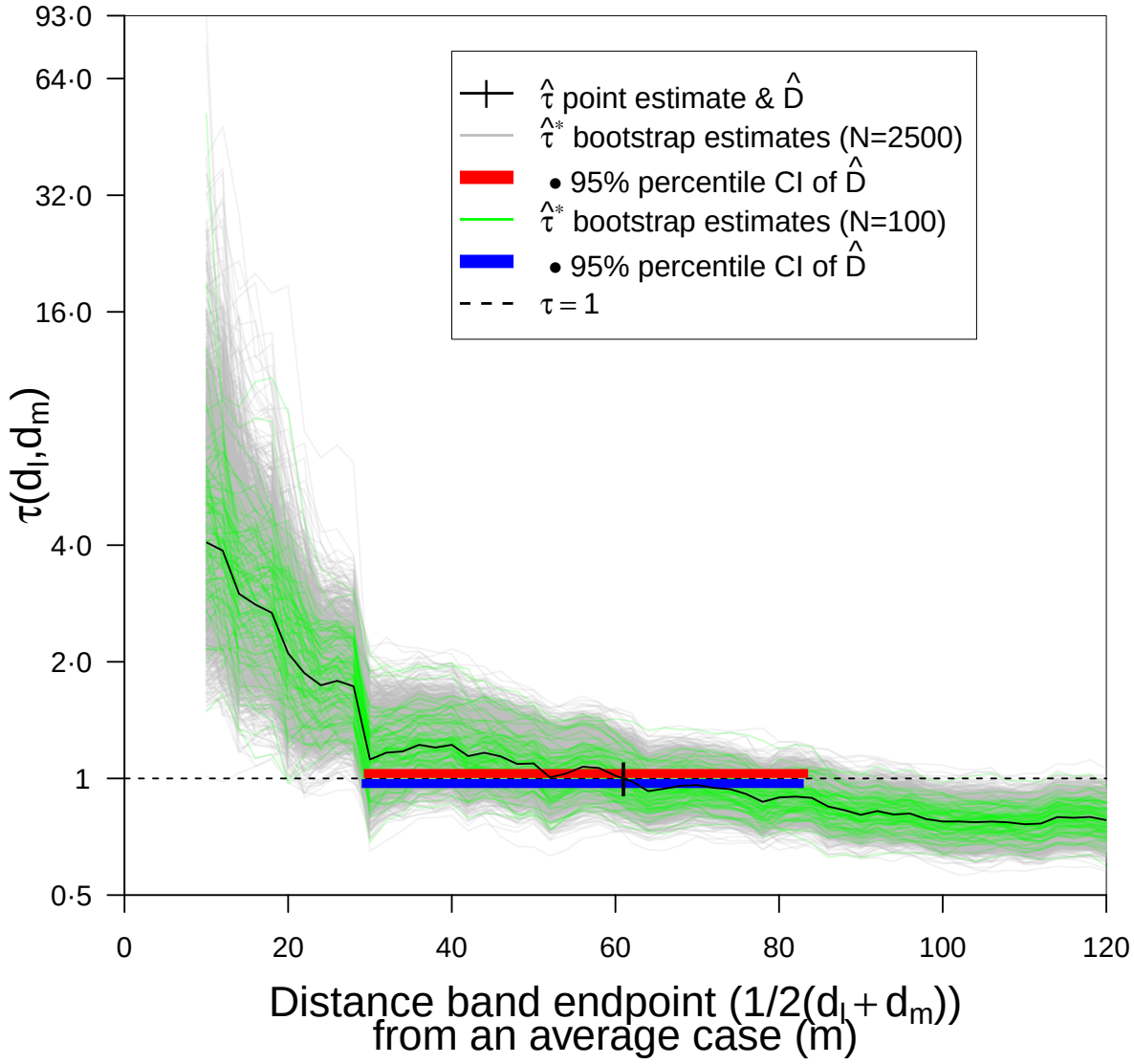


Figure 5: Bootstrap simulations versus the point estimate for 100 or 2,500 bootstrap estimates

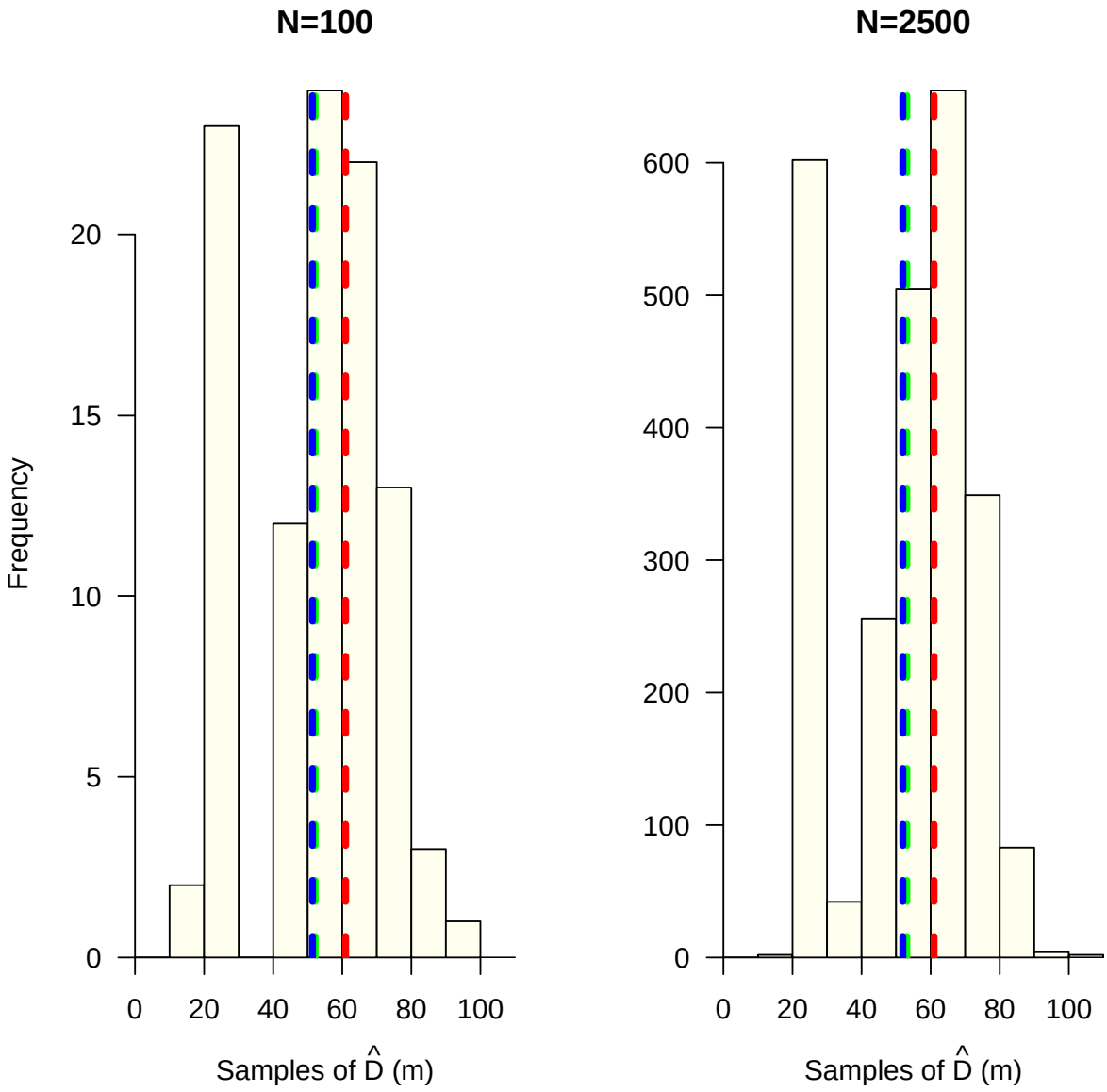


Figure 6: Histogram of the $\hat{D}$ estimates by number of bootstrap estimates

8.2 BCa confidence interval

A better alternative is the *BCa confidence interval* using `coxed::bca()` which takes a few seconds extra to run; we decide to test for N=100 or N=2500 as it may be more sensitive to low numbers of bootstrap samples than the percentile CI.

8.3 Spatial bootstrap

8.3.1 Number of bootstrap estimates & bootstrap sampling method

```
setwd(intrmd8path)
load("d.envelope2500.RData")
load("d.envelope100.RData")
load("dintercept.pointestimate.RData")
load("tauCItmp2500noinfs.RData")
load("tau.hagg.RData")

BCa.ci.2500 = coxed::bca(d.envelope2500, conf.level = 0.95)
percentile.ci.2500 = quantile(d.envelope2500, probs=c(0.025, 0.975), type = 7)

# compute modified marked point bootstrap for 100 and 2500 bootstrap estimates using----
# percentile or BCa CIs
set.seed(seed = 5) # set at seed = 5 to compare with tauCItmp2500noinfs
ptm = proc.time()
tauCI2500lohv2 = summonTauBstraplohv2(X.region = as.matrix(hag.dat), r.min = r.min,
r.max = r.max, bootiters = 2500, T1 = 0, T2 = 14)
proc.time() - ptm # 17s
d.envelope2500lohv2 = ciIntercept(2500, end.set = r.max, tau.sim = tauCI2500lohv2)
setwd(intrmd8path)
save(d.envelope2500lohv2, file = "d.envelope2500lohv2.RData")
set.seed(seed = 6) # set at seed = 6 to compare with tauCItmp100noinfs
tauCI100lohv2 = summonTauBstraplohv2(X.region = as.matrix(hag.dat), r.min = r.min,
r.max = r.max, bootiters = 100, T1 = 0, T2 = 14)
d.envelope100lohv2 = ciIntercept(100, end.set = r.max, tau.sim = tauCI100lohv2)

BCa.ci.lohv2.2500 = coxed::bca(d.envelope2500lohv2, conf.level = 0.95)
percentile.ci.lohv2.2500 = quantile(d.envelope2500lohv2, probs=c(0.025, 0.975), type = 7)
BCa.ci.lohv2.100 = coxed::bca(d.envelope100lohv2, conf.level = 0.95)
percentile.ci.lohv2.100 = quantile(d.envelope100lohv2, probs=c(0.025, 0.975), type = 7)

dintercept.pointestimate
BCa.ci.2500
percentile.ci.2500
BCa.ci.lohv2.2500
percentile.ci.lohv2.2500

setwd(figspath)
pdf("bootstraphistv2.pdf", width = 4*7, height = 4*7, pointsize = 4*12)
par(mfrow = c(2,2))
hist(d.envelope100,breaks = seq.int(0,110,10), xaxs = "i", yaxs = "i", ylab = "Frequency",
main = "N=100, resampled-index", xlab = NULL, lwd = 4, ylim = c(0,41), las = 1
, col = "ivory1")
```

```
fig_label("a", cex = 3, region = "plot", lwd = 4)
lines(x = coxed::bca(d.envelope100, 0.95), y = c(0,0), lty = 1, lwd = 24, col = "purple")
abline(v = dintercept.pointestimate, lty = 2, lwd = 12, col = "red")
abline(v = mean(d.envelope100), lty = 2, lwd = 8, col = "green")
abline(v = median(d.envelope100), lty = 2, lwd = 8, col = "blue")
hist(d.envelope100lohv2,breaks = seq.int(0,110,10), xaxs = "i", yaxs = "i", ylab = NULL,
main = "N=100, modified marked point", xlab = NULL, lwd = 4, ylim = c(0,41),
las = 1, col = "ivory1")
fig_label("b", cex = 3, region = "plot", lwd = 4)
lines(x = coxed::bca(d.envelope100lohv2, 0.95), y = c(0,0), lty = 1, lwd = 24,
      col = "purple")
abline(v = dintercept.pointestimate, lty = 2, lwd = 12, col = "red")
abline(v = mean(d.envelope100lohv2), lty = 2, lwd = 8, col = "green")
abline(v = median(d.envelope100lohv2), lty = 2, lwd = 8, col = "blue")
hist(d.envelope2500,breaks = seq.int(0,110,10), xaxs = "i", yaxs = "i",
     ylab = "Frequency", col = "ivory1",main = "N=2500, resampled-index",
xlab = latex2exp::TeX('$\\underline{D}$ (m)'), lwd = 4, ylim = c(0,1160), las = 1)
fig_label("c", cex = 3, region = "plot", lwd = 4)
lines(x = coxed::bca(d.envelope2500, conf.level = 0.95), y = c(0,0), lty = 1, lwd = 24,
      col = "purple")
abline(v = dintercept.pointestimate, lty = 2, lwd = 12, col = "red")
abline(v = mean(d.envelope2500), lty = 2, lwd = 8, col = "green")
abline(v = median(d.envelope2500), lty = 2, lwd = 8, col = "blue")
hist(d.envelope2500lohv2,breaks = seq.int(0,110,10), xaxs = "i", yaxs = "i", ylab = NULL,
main = "N=2500, modified marked point", xlab = latex2exp::TeX('$\\underline{D}$ (m)'),
lwd = 4, ylim = c(0,1160), las = 1, col = "ivory1")
fig_label("d", cex = 3, region = "plot", lwd = 4)
lines(x = coxed::bca(d.envelope2500lohv2), y = c(0,0), lty = 1, lwd = 24, col = "purple")
abline(v = dintercept.pointestimate, lty = 2, lwd = 12, col = "red")
abline(v = mean(d.envelope2500lohv2), lty = 2, lwd = 8, col = "green")
abline(v = median(d.envelope2500lohv2), lty = 2, lwd = 8, col = "blue")
dev.off()

# graphical abstract version----
pdf("bootstraphistv2.ga.pdf", width = 7*4, height = 7*4, pointsize = 12*4)
hist(d.envelope2500lohv2,breaks = seq.int(0,110,7.5), xaxs = "i", yaxs = "i", ylab = NULL,
     yaxt = "n", main = NULL, xlab = latex2exp::TeX('$\\underline{D}$ (m)'), cex = 2,
     cex.lab = 2, lwd = 4, ylim = c(0,1160), col = "ivory1")
axis(2, las = 1, at = c(0,1000), labels = c("0","1000"), lwd = 4)
abline(v = dintercept.pointestimate, lty = 2, lwd = 16, col = "black")
lines(x = coxed::bca(d.envelope2500lohv2), y = c(0,0), lty = 1, lwd = 24, col = "purple")
dev.off()

coxed::bca(d.envelope2500, conf.level = 0.95)
coxed::bca(d.envelope2500lohv2, conf.level = 0.95)

pdf("nbstrapv2.pdf", width = 4*7, height = 4*7, pointsize = 4*12)
plot(NULL, xlim = c(0,120), log="y", ylim = c(0.5,max(tauCItmp2500noinfs,tauCI2500lohv2)),
xaxt = "n", yaxt = "n", xaxs = "i", yaxs = "i", ylab = "", xlab = "", lwd = 4,
cex.lab = 1.5)
mtext(latex2exp::TeX('$\\tau (d_l,d_m)$'), side=2, line=2, cex = 1.5)
mtext(latex2exp::TeX(
'Distance band endpoint (1/2$(d_l + d_m)$)'), side=1, line=3, cex = 1.5)
```

```
mtext(latex2exp::TeX('from an average case (m)'), side=1, line=4, cex = 1.5)

for (i in 1:2500) {
  lines(r.max, tauCItmp2500noinfs[i,], col = scales::alpha("grey", alpha = 0.2), lwd = 4)
}
for (i in 1:2500) {
  lines(r.max, tauCI2500lohv2[i,], col = scales::alpha("green", alpha = 0.2), lwd = 4)
}
axis(2,las=1,at=c(0.5,1,2,4,16,32,64,92), labels = c("0·5","1","2·0","4·0","16·0","32·0","64·0","92·0")
axis(1, lwd = 4)
lines(x = c(0,120),y = c(1,1),lty = 4)
par(lend=1);
lines(x = coxed::bca(d.envelope2500, conf.level = 0.95), y=c(1.03,1.03), type = "l",
lwd = 20, col = "red")
lines(x = coxed::bca(d.envelope2500lohv2, conf.level = 0.95), y=c(0.97,0.97), type = "l",
lwd = 20, col = "blue")
lines(x=c(dintercept.pointestimate,dintercept.pointestimate), y = c(0.9,1.1), lwd = 8)
lines(r.max,tau.hagg, lwd = 4)
legend(x = 40, y = 32,
legend=c(latex2exp::TeX('$\\hat{\\tau}$ point estimate & $\\hat{D}$'), latex2exp::TeX('$\\hat{\\tau}^*$
latex2exp::TeX('    $\\bullet$  95% BCa CI of $\\hat{D}$'),
latex2exp::TeX('$\\hat{\\tau}^*$: modified marked point (N=2500)'),
latex2exp::TeX('    $\\bullet$  95% BCa CI of $\\hat{D}$'),
latex2exp::TeX('$\\tau = 1$')),
col=c("black", "grey", "red", "green", "blue", "black"), lty=c(1,1,1,1,1,2),
lwd = c(6,6,30,6,30,6), pch = c(124,NA,NA,NA,NA,NA), cex=1.05, yjust = 0.5)
dev.off()

# graphical abstract version----
pdf("nbstrapv2.ga.pdf", width = 4*7, height = 4*7, pointsize = 4*12)
plot(NULL, xlim = c(10,100), log="y", ylim = c(0.6,max(tauCI2500lohv2)),
xaxt = "n", yaxt = "n", xaxs = "i", yaxs = "i",
ylab = "",
xlab = latex2exp::TeX('$d$ (m)'), cex.lab = 2, lwd = 8)
for (i in 1:2500) {
  lines(r.max, tauCI2500lohv2[i,], col = scales::alpha("green", alpha = 0.1), lwd = 8)
}
axis(2, las=1, at=c(0.6,1,5,10), labels = c("0·6","1","5·0","10·0"), cex = 2, lwd = 4)
axis(1, las=1, at=c(10,50,100), labels = c("10","50","100"), cex = 2, lwd = 4)
lines(x = c(0,100),y = c(1,1),lty = 2, lwd = 4) # as abline seems to overlap
mtext(latex2exp::TeX('$\\tau$'), side = 2, cex = 2, las = 1, line = 2, lwd = 4)
par(lend=1);
lines(x = coxed::bca(d.envelope2500lohv2, conf.level = 0.95), y=c(1,1), type = "l",
col = "blue", lwd = 20)
lines(x=c(dintercept.pointestimate,dintercept.pointestimate), y = c(0.95,1.05), lwd = 8)
lines(r.max,tau.hagg, lwd = 4)
dev.off()
```

It turns out that the number of bootstrap estimates does not make a large difference to the precision of BCa confidence intervals. They appear to be slightly narrower for N=2500. However as we know the $\underline{D}$ distribution is non-symmetric, the BCa CI should be relied upon rather than the percentile CI.

However the effect of the sampling method is far more obvious and goes to show that the *modified marked point bootstrap* (MMPB) method far outperforms the resampled index method, especially when used together

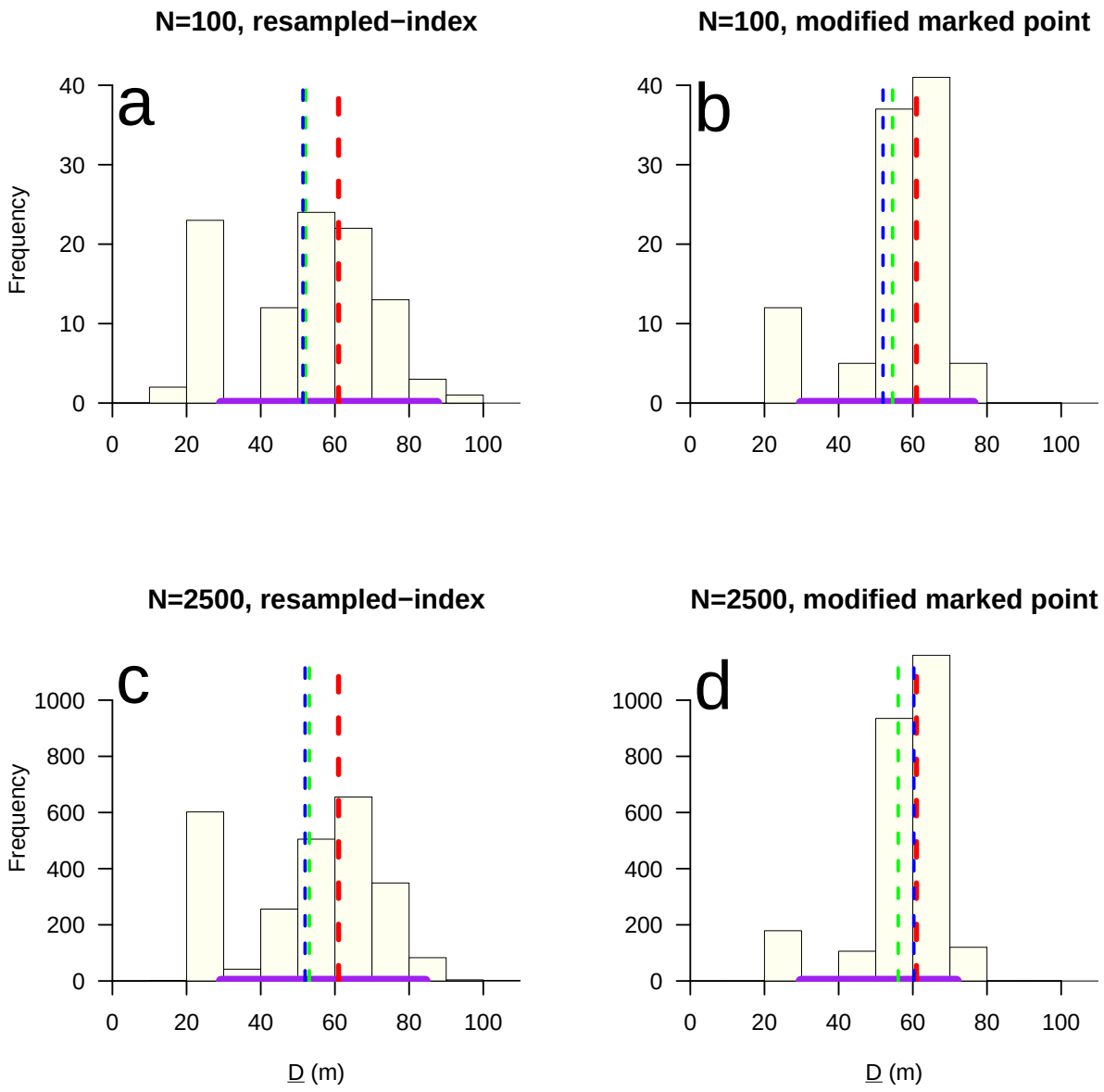


Figure 7: Histogram of the $\hat{D}$ estimates by number of bootstrap estimates and bootstrap sampling method. All four CIs used 100% of simulations

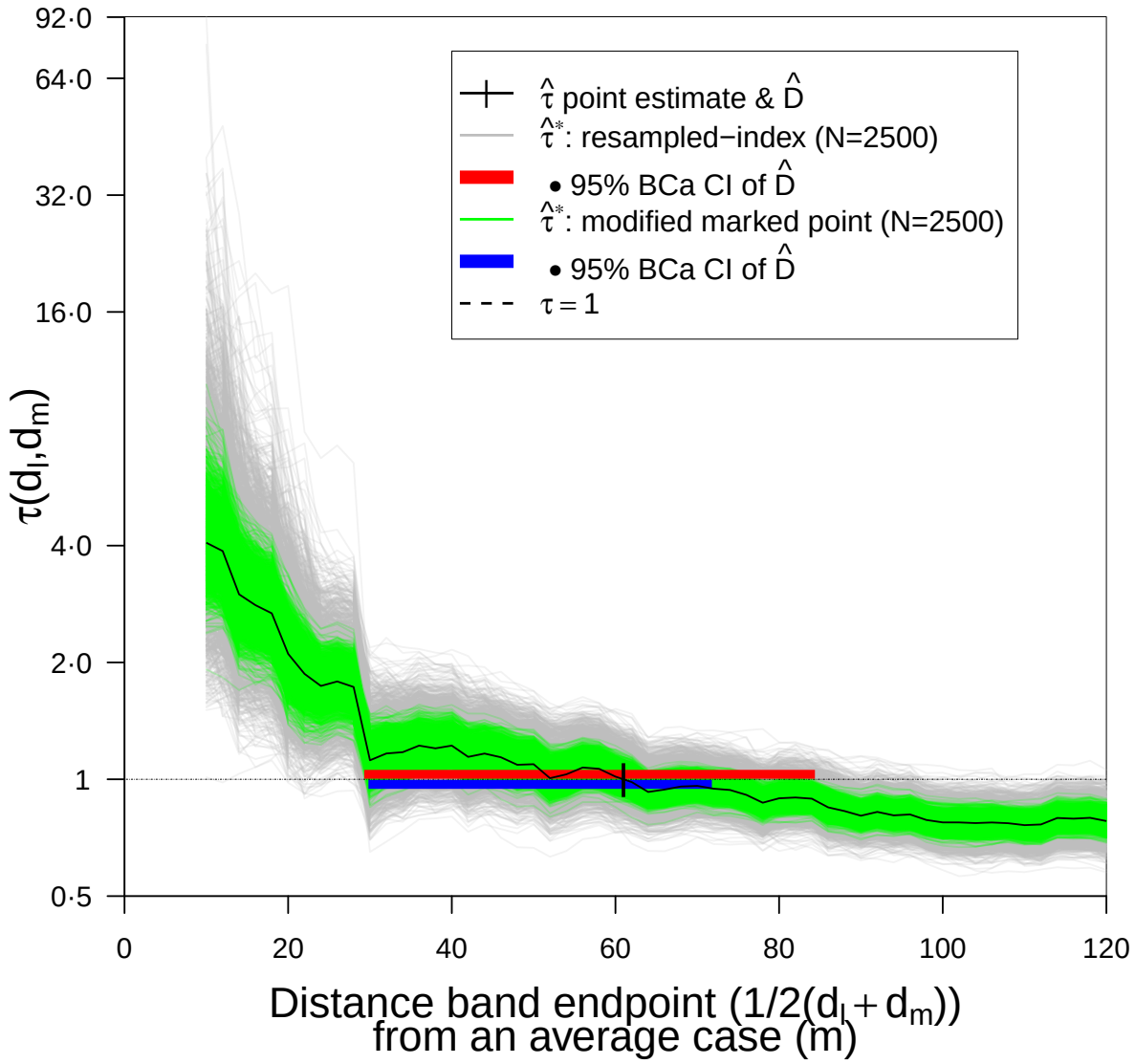


Figure 8: Bootstrap simulations versus the point estimate for different bootstrap sampling methods using 2,500 bootstrap estimates; both CIs used 100% of simulations.

with a higher number of sampling estimates; here the skew also reverses unlike the other three plots. As a result, MMPB CIs are narrower (more precise).

8.3.2 Typical information loss when bootstrapping

We examine the typical information lost for the MMPB approach versus the resampled-index.

```
total = 0
nsims = 1000
cases = 188 # using the measles dataset as an example
set.seed(seed = 9)
for (i in 1:nsims) {
  x = sample(cases, replace = TRUE)
  count = length(unique(x))
  total = total + count
}
uniqueinds = round(total/nsims)
original = cases*(cases - 1) # number of pairs involved in a point estimate calc
ri = uniqueinds*(uniqueinds - 1)
ri/original*100 # % of pairs involved in a resampled-index approach
```

```
## [1] 39.94197
```

```
mmpb = uniqueinds*(cases - 1)
mmpb/original*100 # % of pairs involved in a modified marked point bootstrap approach
```

```
## [1] 63.29787
```

Lessler et al. use the resampled-index as the standard bootstrap sampling method i.e. just sampling from the row indices with replacement to get their bootstrapped version of the data to apply the τ estimator to. This is then repeated N=`bootiters` times. Instead we use a modified form of Loh & Stein, to calculate local mark functions and take a bootstrap sample from these. More information in the main paper and appendix.

8.3.3 Modified marked point bootstrap vs Loh & Stein

This chunk explains why MMPB is better than the standard Loh & Stein's implementation for the tau statistic.

```
setwd(intrmd8path)
load("tau.hagg.RData")
load("d.envelope2500lohv2.RData")
load("dintercept.pointestimate.RData")
set.seed(seed = 10)
ptm = proc.time()
tausims.loh1 = summonTauBstraploh(as.matrix(hag.dat), r.min, r.max, 2500, T1= 0, T2 = 14)
proc.time() - ptm # 17s
tauCI.loh1 = summonTauCI(tausims.loh1, r.max)
set.seed(seed = 10) # set as same seed for fair comparison
ptm = proc.time()
tausims.loh2 = summonTauBstraplohv2(as.matrix(hag.dat), r.min, r.max, 2500, T1= 0,
T2 = 14)
```

```
proc.time() - ptm # 17s
setwd(intrmd8path)
save(tausims.loh2, file = "tausims.loh2.RData")
tauCI.loh2 = summonTauCI(tausims.loh2, r.max)
d.envelope2500lohv1 = ciIntercept(2500, end.set = r.max, tau.sim = tausims.loh1)
# warning only 72·6% of sims cross tau = 1

setwd(figspath)
pdf("loh.pdf", width = 4*7, height = 4*7, pointsize = 4*12)
plot(NULL, xlim = c(0,120), log="y", ylim = c(0.5,max(tausims.loh1,tausims.loh2)),
xaxt = "n", yaxt = "n", xaxs = "i", yaxs = "i",
ylab = "",
xlab = "", lwd = 4, cex.lab = 1.5)
mtext(latex2exp::TeX('$\\tau (d_l,d_m)$'), side=2, line=2, cex = 1.5)
mtext(latex2exp::TeX(
'Distance band endpoint (1/2$(d_l + d_m)$)'), side=1, line=3,
cex = 1.5)
mtext(latex2exp::TeX(
'from an average case (m). '), side=1, line=4, cex = 1.5)
for (i in 1:2500) {
  lines(r.max, tausims.loh1[i,], col = scales::alpha("grey", alpha = 0.2), lwd = 4)
}
for (i in 1:2500) {
  lines(r.max, tausims.loh2[i,], col = scales::alpha("green", alpha = 0.2), lwd = 4)
}
axis(2,las=1,at=c(0.5,1,4,8,10), labels = c("0·5","1","4·0","8·0","10·0"), lwd = 4)
axis(1, lwd = 4)
lines(x = c(0,120),y = c(1,1),lty = 2, lwd = 4) # as abline seems to overlap
par(lend=1);
lines(x = coxed::bca(d.envelope2500lohv1, conf.level = 0.95), y=c(1.03,1.03), type = "l",
lwd = 20, col = "red")
lines(x = coxed::bca(d.envelope2500lohv2, conf.level = 0.95), y=c(0.97,0.97), type = "l",
lwd = 20, col = "blue")
lines(x=c(dintercept.pointestimate,dintercept.pointestimate), y = c(0.9,1.1), lwd = 8)
lines(r.max,tau.hagg, lwd = 4)
legend(x = 35, y = 4.5,
legend=c(latex2exp::TeX(
'$\\hat{\\tau}$ point estimate & $\\hat{D}$'),
latex2exp::TeX(
'$\\hat{\\tau}^*$: Loh & Stein\'s marked point (N=2500)'),
latex2exp::TeX('    $\\bullet$  95% BCa CI of $\\hat{D}$'),
latex2exp::TeX(
'$\\hat{\\tau}^*$: modified marked point (N=2500)'),
latex2exp::TeX('    $\\bullet$  95% BCa CI of $\\hat{D}$'),
latex2exp::TeX('$\\tau = 1$')),
col=c("black", "grey", "red", "green", "blue", "black"), lty=c(1,1,1,1,1,2),
lwd = c(6,6,30,6,30,6), pch = c(124,NA,NA,NA,NA,NA), cex=1.05, yjust = 0.5)
dev.off()
```

The CIs for Loh & Stein are unreliable as they are only formed from only $77 \cdot 4\%$ of simulations that are crossing $\tau = 1$, as a result it does not contain the point estimate line $\hat{\tau}(d)$. Loh & Stein's simulation lines poorly underestimate $\hat{\tau}$ for short distances and overestimate it for most medium to large distances.

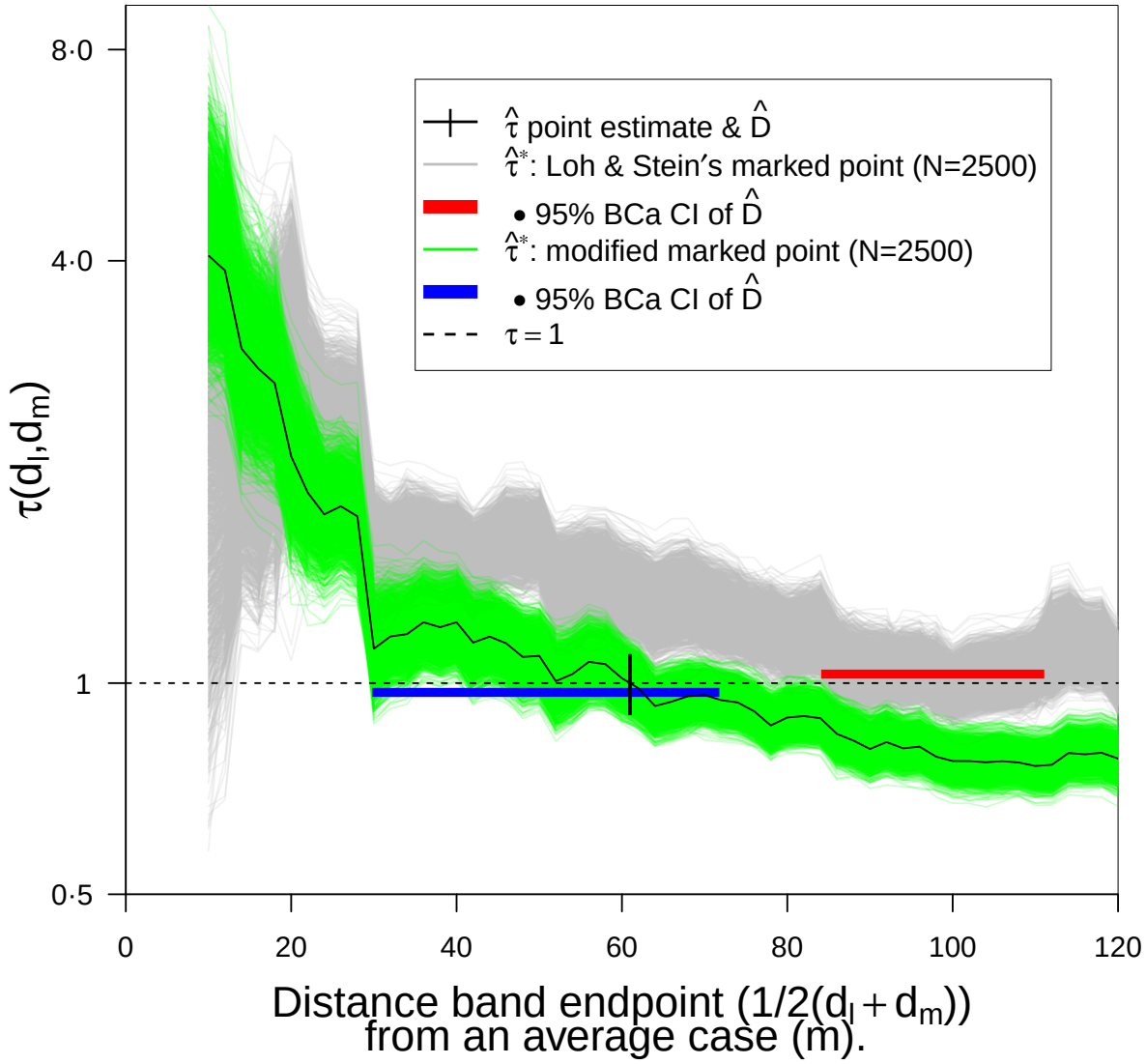


Figure 9: Bootstrap simulations versus the point estimate for the MMPB sampling methods versus Loh & Stein, using 2,500 bootstrap estimates.

8.4 Distance bands

We do a simple analysis of the case distance distribution to propose a reasonable distance band set. We then compare this to the one used by Lessler et al. in their measles analysis and see how the estimates differ.

```
setwd(intrmd8path)
load(file = "d.envelope2500lohv2.RData")
load("dintercept.pointestimate.RData")

dist.cases = spatstat::crossdist(X$x.loc, X$y.loc, X$x.loc, X$y.loc)
range(dist.cases)
round(median(dist.cases))
dist.cases = sort(dist.cases, decreasing = FALSE)
first.nonzero.dist = dist.cases[max(which(dist.cases==0))+1]
dist.bands = c(0, floor(first.nonzero.dist),
seq.int(from = ceiling(first.nonzero.dist/5)*5,
to = (ceiling(max(dist.cases)/5)*5),by = 5))
hist(dist.cases, xlab = "Pairwise distance between cases (m)", breaks = dist.bands,
main="Distance histogram (too few in 2nd dist band)", freq = FALSE, xaxs = "i",
yaxs = "i")
dist.bands = c(0, floor(first.nonzero.dist),
seq.int(from = 15, to = (ceiling(max(dist.cases)/5)*5),by = 5))
hist(dist.cases, xlab = "Pairwise distance between cases (m)", breaks = dist.bands,
main="Distance histogram (ignoring pairs separated by >200m)", freq = FALSE, xaxs = "i",
yaxs = "i")
abline(v = 200, col = "red", lwd = 4)

dist.bands = c(0, floor(first.nonzero.dist), seq.int(from = 15,to = 200,by = 5))
r.min.newdb = sort(rev(dist.bands)[-1])
r.max.newdb = dist.bands[-1]
tau.newdb = summonTau(as.matrix(hag.dat), r.min = r.min.newdb,
r.max = r.max.newdb, T1 = 0, T2 = 14)

dist.gap = 50
r.max = seq(10,200,2)
r.min = r.max-dist.gap
r.min[which(r.min < 0)] = 0
tausims.loh2 = summonTauBstraplohv2(as.matrix(hag.dat), r.min, r.max, 2500, T1= 0,
T2 = 14)

tau.hagg.newdb = IDSpatialStats::get.tau(hag.dat, hagg.func, r=r.max, r.low=r.min,
comparison.type = "independent")$tau

set.seed(seed = 12)
ptm = proc.time()
tau.sims.newdb = summonTauBstraplohv2(X.region = as.matrix(hag.dat),
r.min = r.min.newdb, r.max = r.max.newdb, bootiters = 2500,
T1= 0, T2 = 14)
proc.time() - ptm # 12s
d.envelope2500lohv2.newdb = ciIntercept(2500, end.set = r.max.newdb,
tau.sim = tau.sims.newdb)
firstbelow1.newdb = which(tau.newdb < 1)[1]
y1 = tau.newdb[firstbelow1.newdb-1]
y2 = tau.newdb[firstbelow1.newdb]
```

```
x1 = r.max.newdb[firstbelow1.newdb-1]
x2 = r.max.newdb[firstbelow1.newdb]
m = (y2-y1)/(x2-x1)
dintercept.pointestimate.newdb = (1+m*x1-y1)/m

coxed::bca(d.envelope2500lohv2.newdb, conf.level = 0.95)
coxed::bca(d.envelope2500lohv2, conf.level = 0.95)

hist(d.envelope2500lohv2.newdb) # any CI struggles to contain the point estimate due to
# the strongly bimodal distribution of D for the non-overlapping distance bands.
hist(d.envelope2500lohv2)

setwd(figspath)
pdf("distband.pdf", width = 4*7, height = 4*7, pointsize = 4*12)
plot(NULL, xlim = c(0,120), log="y", ylim = c(0.3, (max(tausims.loh2[,1],
tau.sims.newdb[,1])+4)), xaxt = "n", yaxt = "n", xaxs = "i", yaxs = "i",
ylab = "",
xlab = "", lwd = 4, cex.lab = 1.5)
mtext(latex2exp::TeX('$\\tau (d_l,d_m)$'), side=2, line=2, cex = 1.5)
mtext(latex2exp::TeX(
'Distance band endpoint (1/2$(d_l + d_m)$),'), side=1, line=3, cex = 1.5)
mtext(latex2exp::TeX(
'from an average case (m). See caption for details.'), side=1, line=4, cex = 1.5)
for (i in 1:2500) {
  lines(r.max.newdb, tau.sims.newdb[i,], col = scales::alpha("grey",
  alpha = 0.2), lwd = 4)
}
for (i in 1:2500) {
  lines(r.max, tausims.loh2[i,], col = scales::alpha("green", alpha = 0.2), lwd = 4)
}
axis(2, las=1, at=c(0.3,0.5,1,4,11), labels = c("0·3","0·5","1","4·0","11·0"), lwd = 4)
axis(1, lwd = 4)
lines(x = c(0,120), y = c(1,1), lty = 2, lwd = 4) # as abline seems to overlap
par(lend=1);
lines(x = coxed::bca(d.envelope2500lohv2.newdb, conf.level = 0.95), y=c(1.03,1.03),
type = "l",lwd = 20, col = "red")
lines(x = coxed::bca(d.envelope2500lohv2, conf.level = 0.95), y=c(0.97,0.97),type = "l",
lwd = 20, col = "blue")
lines(x=c(dintercept.pointestimate.newdb,dintercept.pointestimate.newdb), y = c(0.9,1.1),
      lwd = 8)
lines(r.max,tau.hagg.newdb, col = "black", lty = 1, lwd = 4)
lines(x=c(dintercept.pointestimate,dintercept.pointestimate), y = c(0.9,1.1), lwd = 8,
      col = "gray30")
lines(r.max.newdb, tau.newdb, col = "gray38", lty = 1, lwd = 4)
legend(x = 20, y = 6.5,
legend=c(latex2exp::TeX(
'$\\hat{\\tau}$ point estimate & $\\hat{D}$ (overlapping dist. band)'), latex2exp::TeX('$\\hat{\\tau}^*$
latex2exp::TeX('    $\\bullet$  95% BCa CI of $\\hat{D}$'),
latex2exp::TeX(
'$\\hat{\\tau}$ point estimate & $\\hat{D}$ (non-overlapping dist. band)'),
latex2exp::TeX('$\\hat{\\tau}^*$ simulations'),
latex2exp::TeX('    $\\bullet$  95% BCa CI of $\\hat{D}$'),
latex2exp::TeX('$\\tau = 1$')),
```

```
col=c("black", "green", "blue", "gray38", "grey", "red", "black"), lty=c(1,1,1,1,1,1,2),
lwd = c(6,6,30,6,6,30,6), pch = c(124,NA,NA,124,NA,NA,NA), cex=1.05, yjust = 0.5)
dev.off()
```

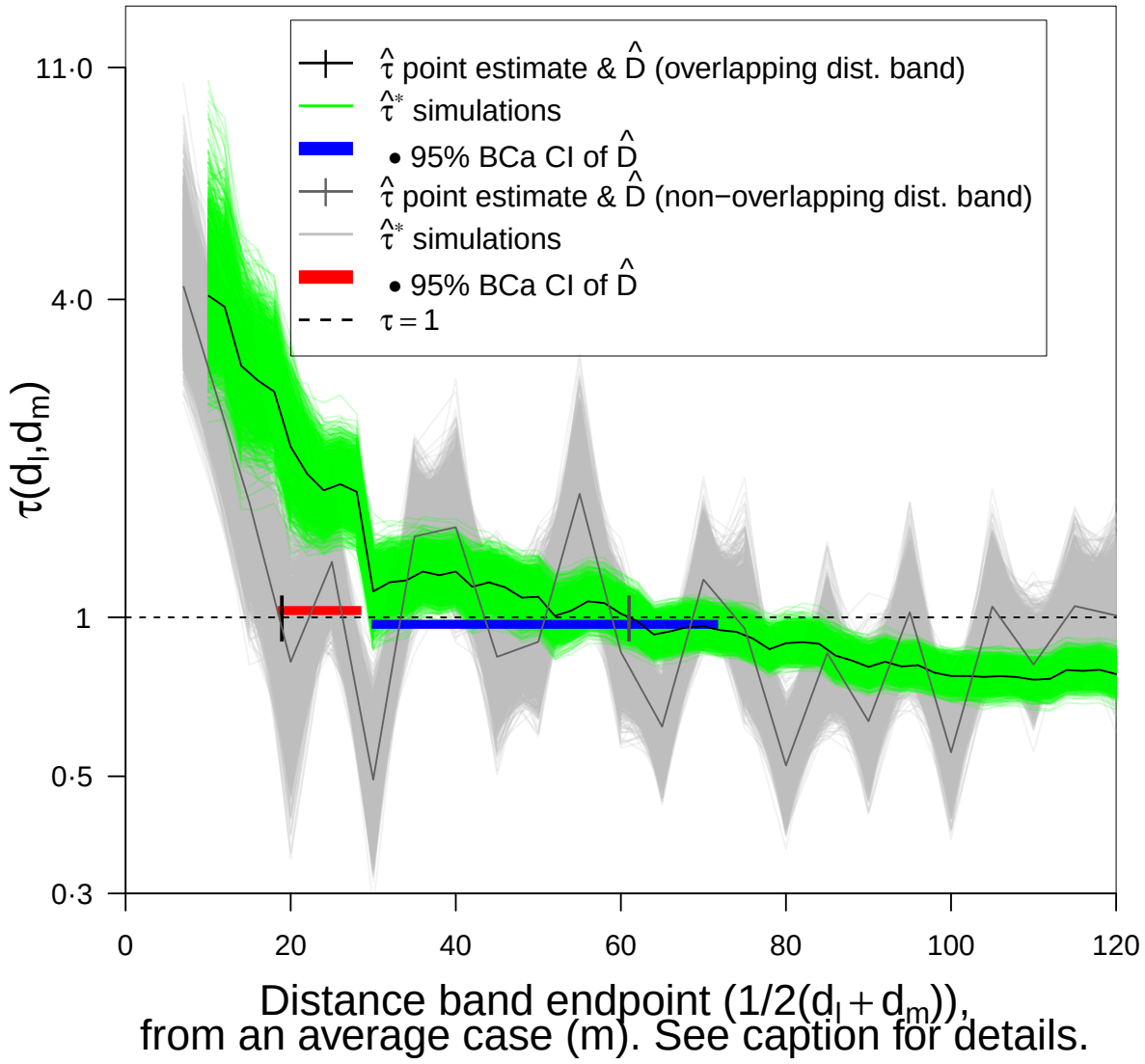


Figure 10: Bootstrap simulations versus the point estimate for a reasonably-proposed example non-overlapping distance band set versus the overlapping one used by Lessler et al. in their measles analysis.

Cases are separated from 0m to just over 300m with a median of 108m. Using distance bands starting at 0m (within household) and then including the first non-zero distance separations from $7 \cdot 9$m to 10m and in 5m increments up to 200m, does not properly cover the $[7 \cdot 9,10]$ band properly which may affect the estimation. We therefore combine the second and third distance bands to get $\{0,7,15,20,25,\ldots,200\}$. We end the distance bands at 200m as there the number of pairs fall off after this point. This now forms the distance band set for our analysis—we now compute `tau.sims.newdb` and compare with Lessler's analysis.

The effect of the distance band sets is enormous in terms of the precision of the `d.envelope` CI, and also where the point estimate intersects $\tau = 1$; both CIs used 100% of simulations. The non-overlapping distance band set (as expected) produces a more erratic tau estimate. From this graph it would seem that for understanding trends the overlapping statistic is better but it is unclear how to construct this as there are infinite combinations for its construction too!

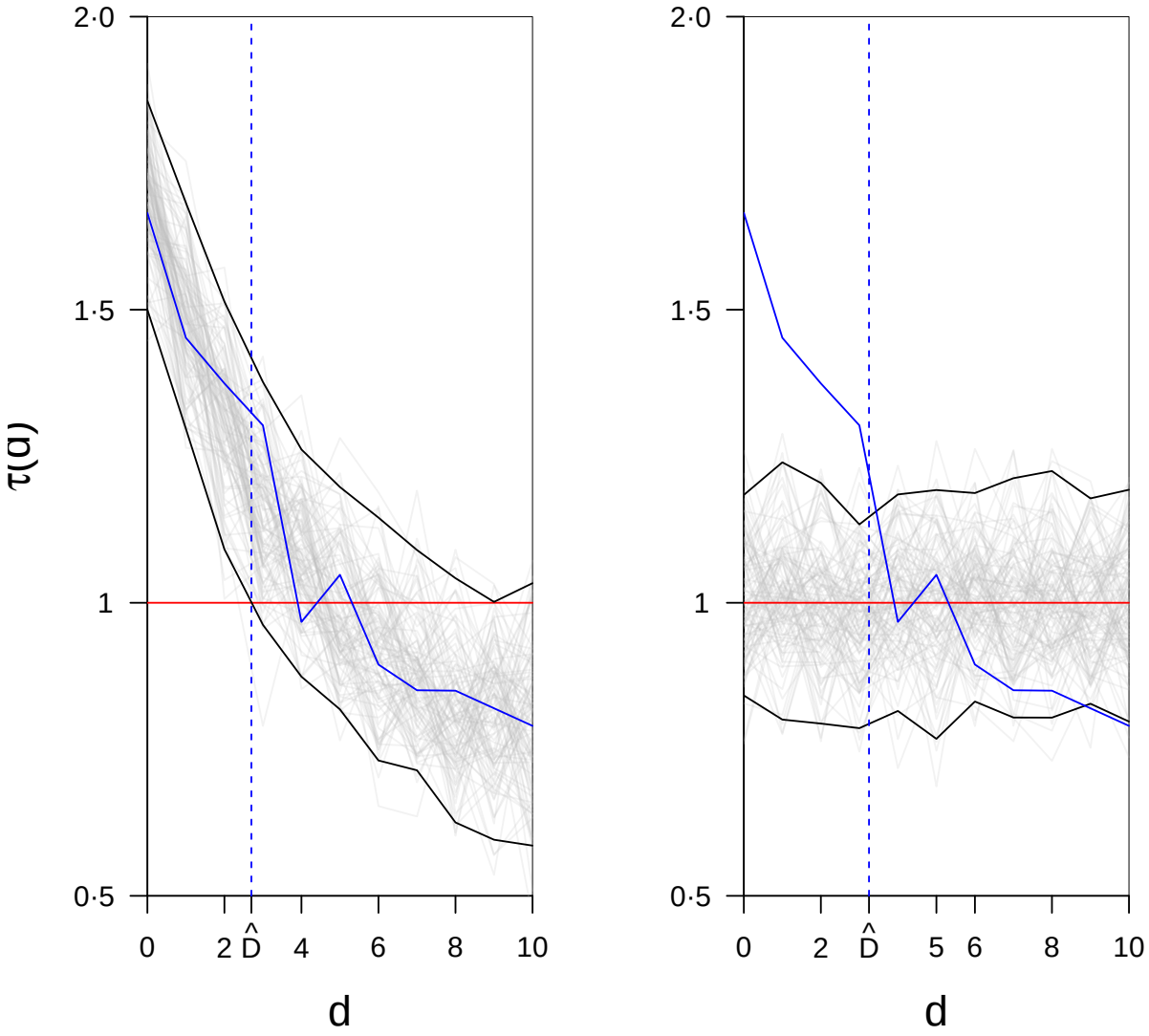
```
# test set----
d2.set = seq.int(0,10)
j.max = length(d2.set)
n.sim = 100
tau.sim = matrix(NA,n.sim,j.max)
set.seed(seed = 30)
for (i in 1:n.sim) {
  alpha = rnorm(1,1,0.1)
  noise = rnorm(j.max,0,0.1)
  tau.sim[i,] = exp(-0.25*d2.set*alpha) + rep.int(0.7,j.max) + noise
}
null.sim = matrix(NA,n.sim,j.max)
set.seed(seed = 31)
for (i in 1:n.sim) {
  noise = rnorm(j.max,0,0.1)
  null.sim[i,] = 1 + noise
}

# get envelope----
central.env = apply(tau.sim,2, quantile, probs = c(0.025,0.975))
null.env = apply(null.sim,2, quantile, probs = c(0.025,0.975))
tau.sim[1,9:11] = c(0.85,0.82, 0.79)  # tweak dummy 'point estimate'
tau.sim[1,3:4] = tau.sim[1,3:4] - 0.1 # tweak dummy 'point estimate'

# plot test set and envelope----
setwd("figs/")
pdf("centralnull.pdf", width = 7*4, height = 7*4, pointsize = 12*4)
par(mfrow = c(1,2))
plot(NULL, NULL, xlim = c(0,10), ylim = c(0.5,2), type = "l", xlab = "d", ylab = latex2exp::TeX('$\\tau
axis(1, labels = c(0,2,latex2exp::TeX('$\\hat{D}$'),4,6,8,10), at = c(0,2,2.7,4,6,8,10),
     lwd = 4, las = 1)
axis(2, labels = c("0·5","1","1·5","2·0"), at = c(0.5,1,1.5,2.0), lwd = 4, las = 1)
for (i in 1:100) {
  lines(seq.int(0,10), tau.sim[i,], col = scales::alpha("grey",0.2), lwd = 4)
}
abline(h=1,col = "red", lwd = 4)
lines(seq.int(0,10), central.env[1,], type = "l", lwd = 4)
lines(seq.int(0,10), central.env[2,], type = "l", lwd = 4)
lines(seq.int(0,10), tau.sim[1,], type = "l", lwd = 4, col = "blue")
abline(v = 2.7, col = "blue", lty = 2, lwd = 4)

plot(NULL, NULL, xlim = c(0,10), ylim = c(0.5,2), type = "l", xlab = "d", ylab = "",
     col = scales::alpha("black",0.2), xaxs = "i", yaxs = "i", xaxt = "n", yaxt = "n",
     lwd = 4, cex.lab = 1.5)
axis(1, labels = c(0,2,latex2exp::TeX('$\\hat{D}$'),5,6,8,10), at = c(0,2,3.25,5,6,8,10),
     lwd = 4, las = 1)
axis(2, labels = c("0·5","1","1·5","2·0"), at = c(0.5,1,1.5,2.0), lwd = 4, las = 1)
for (i in 1:100) {
```

```
  lines(seq.int(0,10), null.sim[i,], col = scales::alpha("grey",0.2), lwd = 4)
}
abline(h=1, col = "red", lwd = 4)
lines(seq.int(0,10), null.env[1,], type = "l", lwd = 4)
lines(seq.int(0,10), null.env[2,], type = "l", lwd = 4)
lines(seq.int(0,10), tau.sim[1,], type = "l", lwd = 4, col = "blue")
abline(v = 3.25, col = "blue", lty = 2, lwd = 4)
dev.off()
```



***

9 References

We feature a list of references for citations made within this code document. However for a comprehensive list of works that contributed to the study please consult the main paper.

Cori, Anne, Neil M. Ferguson, Christophe Fraser, and Simon Cauchemez. 2013. "A new framework and software to estimate time-varying reproduction numbers during epidemics." *Am. J. Epidemiol.* 178 (9): 1505–12. https://doi.org/10.1093/aje/kwt133.

Giles, John, Henrik Salje, and Justin Lessler. 2018. "IDSpatialStats R package development version v0.3.7." https://github.com/HopkinsIDD/IDSpatialStats.

Höhle, M., S. Meyer, M. Paul, L. Held, H. Burkom, T. Correa, M. Hofmann, et al. 2020. "`surveillance` R Package V1.18.0." https://cran.r-project.org/package=surveillance.

January, @ztrewq a.k.a. 2017. "Adding figure labels (A, B, C, ...) in the top left corner of the plotting region." https://logfc.wordpress.com/2017/03/15/adding-figure-labels-a-b-c-in-the-top-left-corner-of-the-plotting-region/.

Lessler, Justin, Henrik Salje, M. Kate Grabowski, and Derek A T Cummings. 2016. "Measuring Spatial Dependence for Infectious Disease Epidemiology." *PLoS ONE* 11 (5): 1–13. https://doi.org/10.1371/journal.pone.0155249.

Neal, Peter J., and Gareth O. Roberts. 2004. "Statistical inference and model selection for the 1861 Hagelloch measles epidemic." *Biostatistics* 5 (2): 249–61. https://doi.org/10.1093/biostatistics/5.2.249.

Oesterle, Heike. 1992. "Statistische Reanalyse einer Masernepidemie 1861 in Hagelloch." PhD thesis, Eberhard-Karls-Universitäat Tübingen.

Pfeilsticker, Albert. 1863. "Beiträge zur Pathologie der Masern mit besonderer Berücksichtigung der statistischen Verhältnisse." PhD thesis, Eberhard-Karls-Universität Tübingen. http://www.archive.org/details/beitrgezurpatho00pfeigoog.

R Core Team. 2020. *R V3.6.3: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

RStudio Team. 2020. *RStudio: Integrated Development Environment for R V1.2.5033*. Boston, MA: RStudio, Inc. http://www.rstudio.com/.

Wickham, Hadley. 2011. "Testthat: Get Started with Testing." *The R Journal* 3: 5–10. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.