TDLNM

Daniel Mork

6/9/2020

Code example of TDLNM

```
library(dlmtree)
set.seed(1)
D <- tdlnm.sim(effect = "A", error.to.signal = 1) # try effects B, C, and D
```

Run TDLNM

- Warning: For speed, this simulation runs only a small number of iterations. We recommend ≥ 5000 burn-in with ≥ 15000 iterations thinned by 10, using 20 trees. Model convergence can be checked by comparing the consistency of results across multiple runs.

```
splits <- seq(min(D$exposure), max(D$exposure), length.out = 52)[-c(1,52)]
smoothing <-
res <- tdlnm(formula = y ~ ., data = D$dat, exposure.data = as.matrix(D$exposure),
             #exposure.se = sd(D$exposure)/2, # uncomment this line to try smoothing
             exposure.splits = list("type" = "values", "split.vals" = splits),
             n.trees = 10, n.burn = 500, n.iter = 1000, n.thin = 1)
```

```
## Preparing data...
## Running model '.' = 100 iterations
## .....
## Burn-in time: 3.64 seconds
## Estimated time to completion: 7.28 seconds
## .........
```

```
res.sum <- summary(res,
                   pred.at = seq(min(D$exposure), max(D$exposure), length.out = 102),
                   cenval = D$cenval, conf.level = 0.95)
```

```
## Centered DLNM at exposure value 1
```

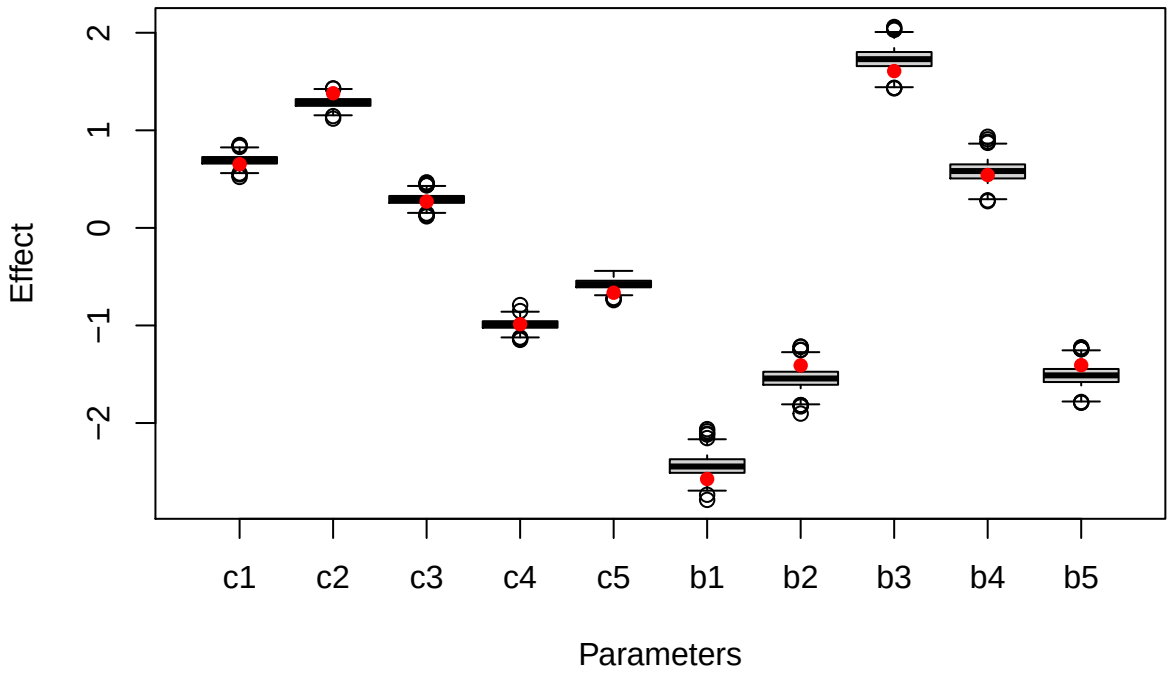
```
res.sum
```

```
## TDLNM summary
##
## Model run info
## - 10 trees
## - 500 burn-in iterations
## - 1000 post-burn iterations
## - 1 thinning factor
## - 0.95 confidence level
##
## Fixed effect coefficients:
```

```
##                Mean Lower.Bound Upper.Bound
## (Intercept) 60.255    -496.335     619.685
## *c1          0.693       0.596       0.788
## *c2          1.286       1.180       1.388
## *c3          0.292       0.191       0.391
## *c4         -0.990      -1.089      -0.898
## *c5         -0.576      -0.672      -0.479
## *b1         -2.441      -2.643      -2.229
## *b2         -1.542      -1.753      -1.338
## *b3          1.729       1.520       1.929
## *b4          0.580       0.379       0.785
## *b5         -1.513      -1.729      -1.307
## ---
## * = CI does not contain zero
##
## DLNM effect:
## range = [-0.924, 0.115]
## signal-to-noise = 1.006
## critical windows: 11 12 13 14 15
```

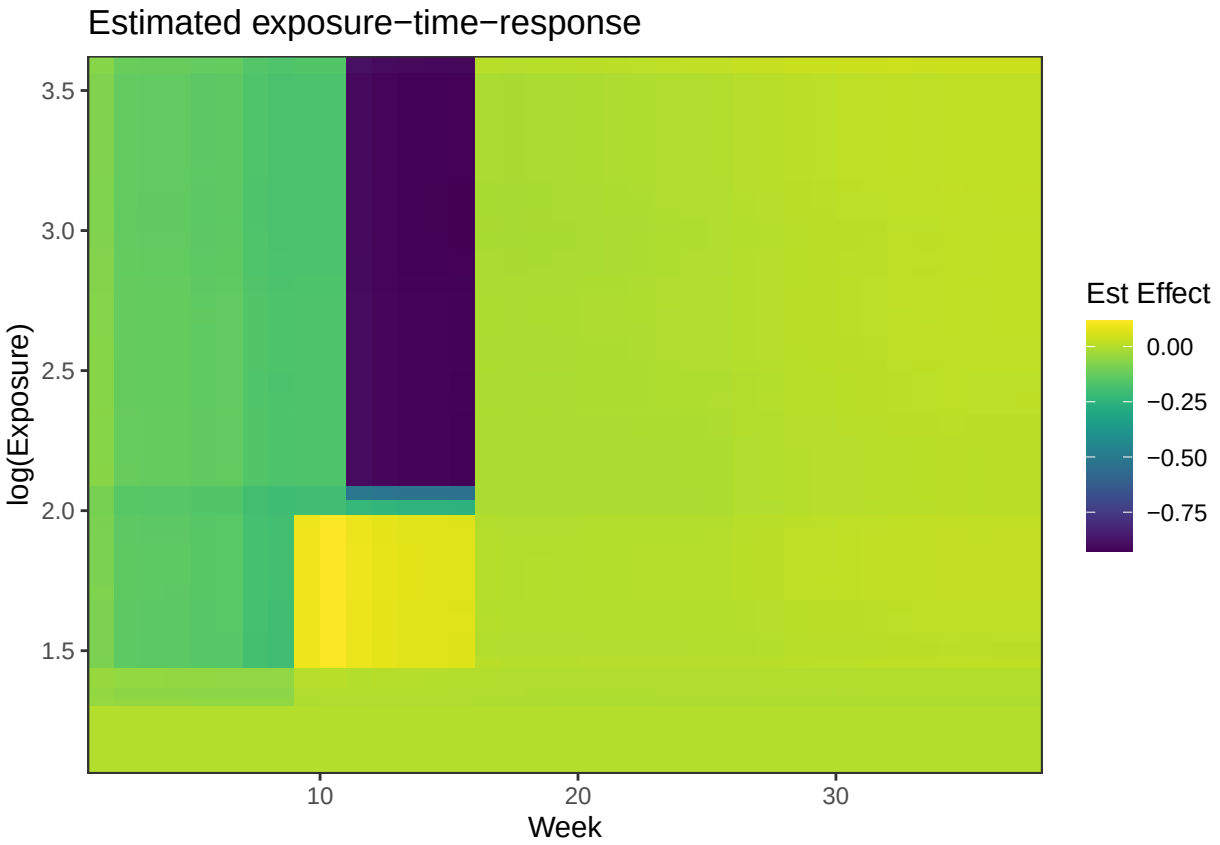
Fixed effect estimates vs. truth

```
boxplot(res$gamma[,-1], xlab = "Parameters", ylab = "Effect") # Boxplot estimated effects
points(1:10, D$params, col = "red", pch = 16) # Red dots = truth
```



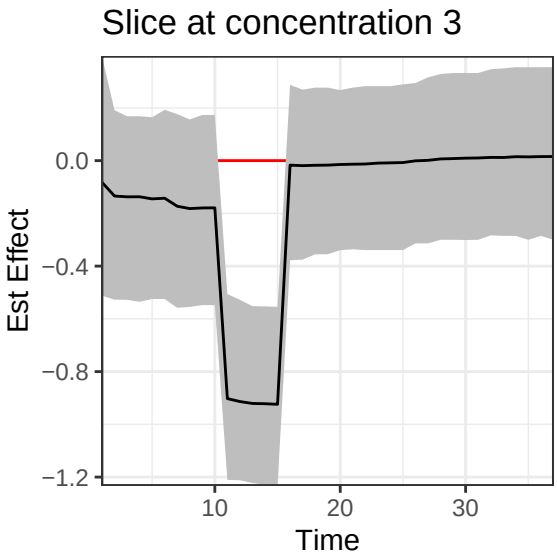
Plot of exposure-time response surface

```
plot(res.sum, main = "Estimated exposure-time-response",
     xlab = "Week", ylab = "log(Exposure)", start.time = 1)
```

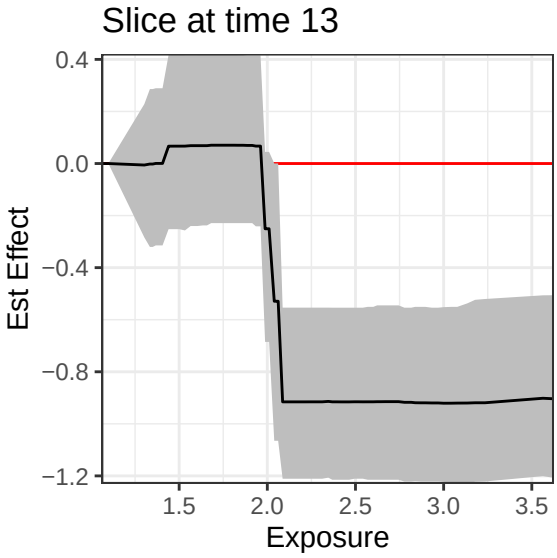


Slices of surface

```
plot(res.sum, plot.type = "slice", val = 3, main = "Slice at concentration 3")
```




```
plot(res.sum, plot.type = "slice", time = 13, main = "Slice at time 13")
```




```
#plot(res.sum, plot.type = "animate") # try animated slice plot in console
```

Compare estimated surface to truth

```
truth <- D$dlnm.fun(sapply(1:37, function(i) res.sum$pred.vals), D$cenval, F)
# RMSE
sqrt(mean((res.sum$matfit - truth)^2))
```

```
## [1] 0.09084211
```

```r
# Coverage
mean(res.sum$cilower < truth & res.sum$ciupper > truth)
```

```
## [1] 0.972271
```

```r
# True positive effect classification
(length(which(res.sum$cilower > 0 & truth > 0)) +
    length(which(res.sum$ciupper < 0 & truth < 0))) /
  length(which(truth != 0))
```

```
## [1] 0.9531915
```

```r
# False positive effect classification
(length(which(res.sum$cilower > 0 & truth == 0)) +
    length(which(res.sum$ciupper < 0 & truth == 0))) /
  length(which(truth == 0))
```

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