TDLNM

Daniel Mork

6/5/2020

Code example of TDLNM

```
library(dlmtree)
data("pm25Exposures")
pm25Exposures <- log(pm25Exposures[which(pm25Exposures$S == "Colorado"),-c(1:2)])[,1:37]
n <- nrow(pm25Exposures)
```

Create fixed effect

```
set.seed(1)
data <- as.data.frame(cbind(matrix(rnorm(5*n), n, 5), matrix(rbinom(5*n, 1, .5), n, 5)))
colnames(data) <- c(paste0("c", 1:5), paste0("b", 1:5))
params <- rnorm(10)
z.gamma <- c(as.matrix(data)[,1:10] %*% params)
```

Create DLNM effect: piecewise in time and exposure

```
dlnm.fun <- function(exposure.data) {
  dlnm <- t(sapply(1:nrow(exposure.data), function(i) {
      sapply(1:ncol(exposure.data), function(j) {
        ifelse(j %in% 11:15, ifelse(exposure.data[i, j] > 2, -1, 0), 0)
    })}))
  colnames(dlnm) <- paste0("Time", 1:ncol(exposure.data))
  return(dlnm)
}
f <- rowSums(dlnm.fun(pm25Exposures))
```

Combine fixed effect, DLNM effect, and error to create continuous response

```
data$y <- z.gamma + f + rnorm(n, sd = sd(f))
```

Run TDLNM

```
splits <- seq(min(pm25Exposures), max(pm25Exposures), length.out = 52)[-c(1,52)]
res <- tdlnm(y ~ ., data, as.matrix(pm25Exposures),
             exposure.splits = list("type" = "values", "split.vals" = splits),
             n.trees = 10, n.burn = 500, n.iter = 1000, n.thin = 1)
```

```
## Preparing data...
```

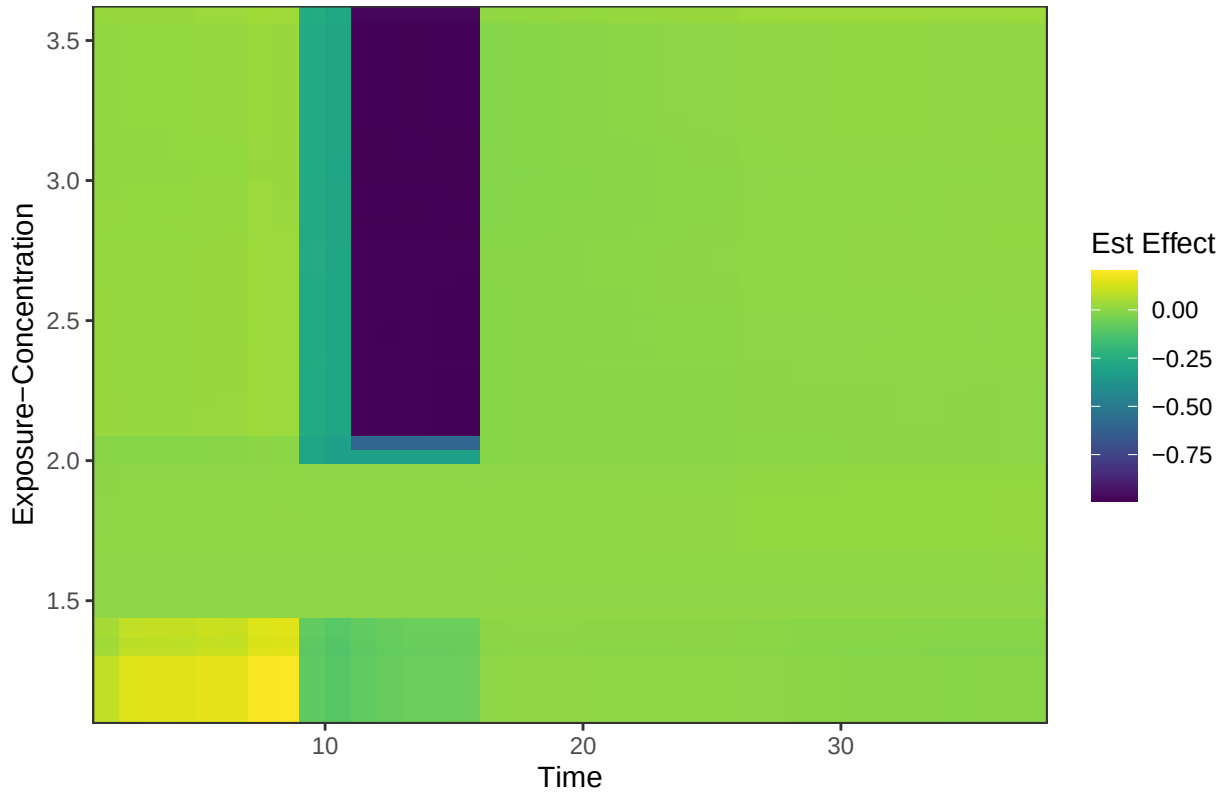
```
## Running model '.' = 100 iterations
## .....
## Burn-in time: 3.05 seconds
## Estimated time to completion: 6.1 seconds
## ..........
```

```
res.sum <- summary(res,
                   pred.at = seq(min(pm25Exposures), max(pm25Exposures), length.out = 102),
                   cenval = 1.5)
```

```
## Centered DLNM at exposure value 1.5
```

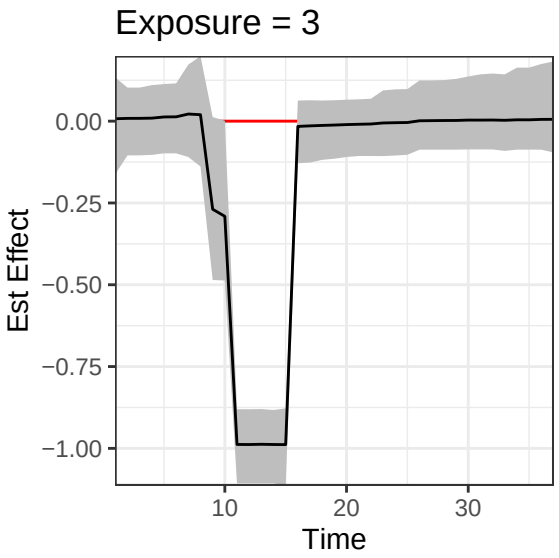
Plot of exposure-time response surface

```
plot(res.sum)
```

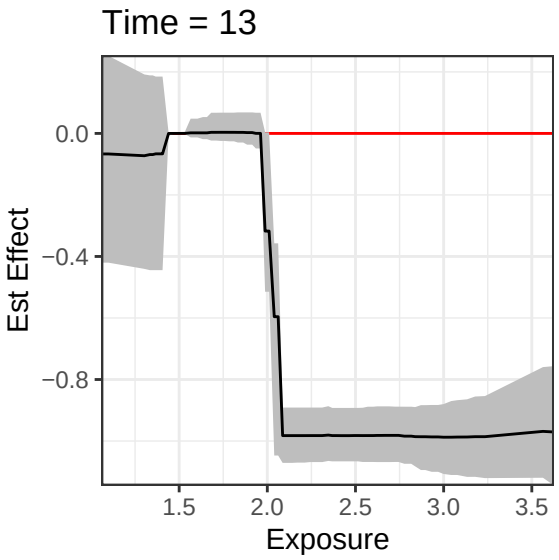


Slices of surface

```
plot(res.sum, "slice", val = 3)
```




```
plot(res.sum, "slice", time = 13)
```



Compare estimated surface to truth

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

```
## [1] 0.06835563
```

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

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

```
# 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] 0.9787234
```

```
# 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
```