EGU short course: 1. Spatiotemporal change detection

Meng Lu

April 18, 2017

Method:

Integrate SAR (simutaneous autocorrelation regression) to the empirical fluctutaion process (efp) structural change test.

What to do in the next 15 minutes:

- Seasonality analysis of a time series
- efp (empirical fluctuation process, from the R package "strucchange") and BFAST (breaks for additive seasonality and trend, from the R package "BFAST") methods for time series structural change detection.
- Spatial correlation of the area
- SAR integrated efp

Start!

Load data, "fevi8" is a 3d array with longitude, latitude, and time as dimensions.

```
load("fevi8.Rdata")
dim(fevi8)
```

```
## [1] 150 150 636
```

Time series structural change analysis

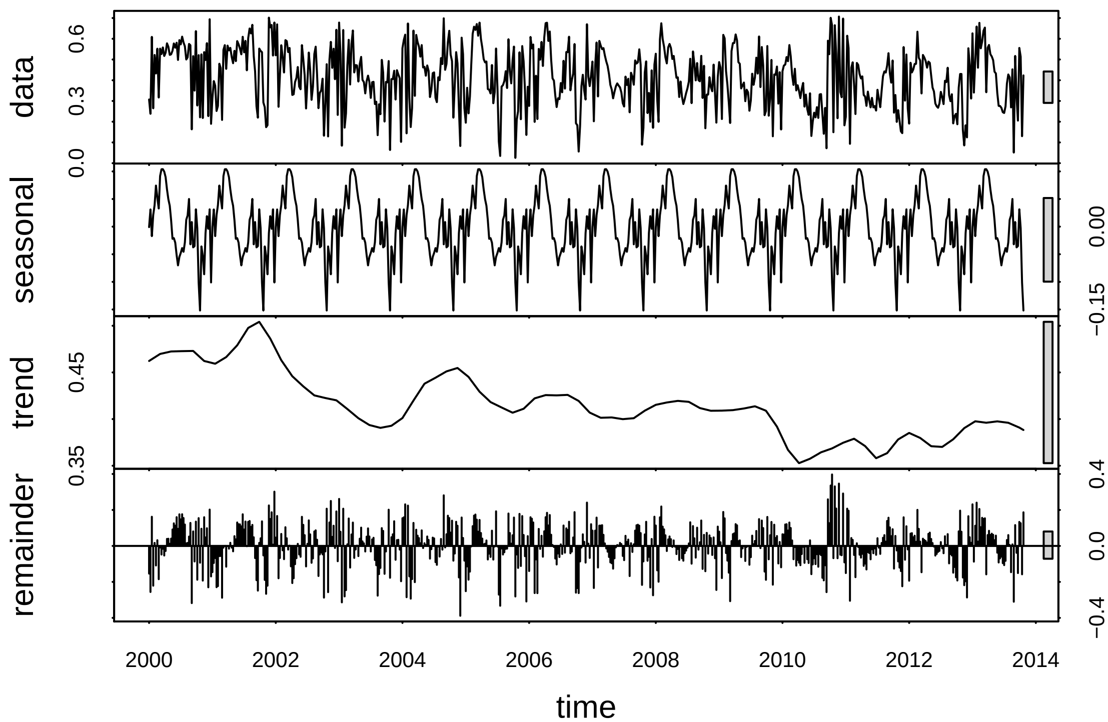
1. BFAST (breakpoints for additive seasonality and trend): detecting change in seasonality and trend interatively

Choose a location and form a time series from the data matrix

```
lon = 60
lat = 40
originalts <- ts(fevi8[lon, lat, ], start = c(2000, 1), frequency = 46)
```

Assuming additive seasonality and trends, use stl (loess) to separate trend, seasonality and residuals.

```
seasonality <- stl(originalts, s.window = "per")$time.series[,
    "seasonal"]
plot(stl(originalts, s.window = "per"))
```




```r
# spec.ar(seasonality)
```

Check seasonality and the ability of using a harmonic model to reduce seasonality.

Form harmonic terms.

$x_t = Acos(2\pi wt + \phi)$

$x_t = U_1cos(2\pi wt) + U_2sin(2\pi wt)$

$(cos(\alpha + \beta) = cos(\alpha)cos(\beta) - sin(\alpha)sin(\beta))$

```r
le = 636  # the length of time series
tl = 1:le
w = 1/46
# harmonic
co <- cos(2 * pi * tl * w)
si <- sin(2 * pi * tl * w)
co2 <- cos(2 * pi * tl * w * 2)
si2 <- sin(2 * pi * tl * w * 2)
co3 <- cos(2 * pi * tl * w * 3)
si3 <- sin(2 * pi * tl * w * 3)
```

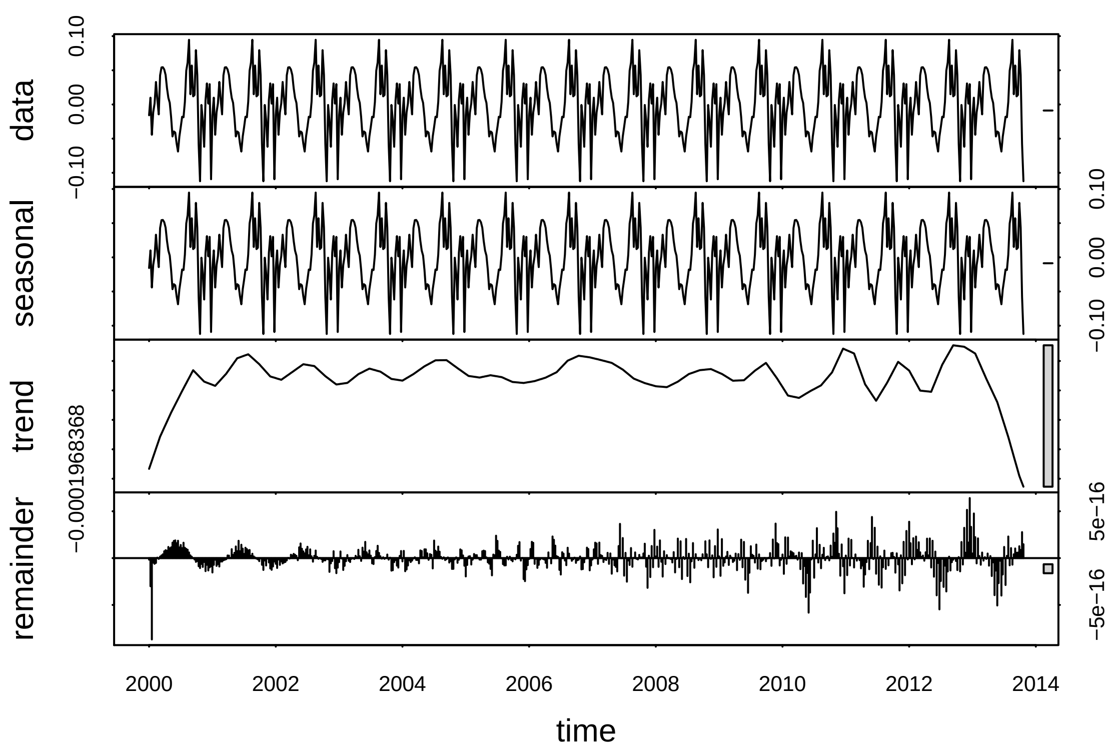
Fit a first order harmonic model to the seasonality and model the seasonality in residuals.

```r
res_har1 <- residuals(lm(seasonality ~ co + si))
summary(lm(seasonality ~ co + si))
```

```
##
## Call:
```

```
## lm(formula = seasonality ~ co + si)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.112342 -0.033067  0.003701  0.032908  0.094615
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0001968  0.0018132   0.109  0.91359
## co          0.0081433  0.0025719   3.166  0.00162 **
## si          0.0489322  0.0025566  19.140  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04572 on 633 degrees of freedom
## Multiple R-squared:  0.3735, Adjusted R-squared:  0.3715
## F-statistic: 188.7 on 2 and 633 DF,  p-value: < 2.2e-16
```

```r
res_har1 <- ts(res_har1, start = c(2000, 1), frequency = 46)
sea_har1 <- stl(res_har1, s.window = "per")$time.series[, "seasonal"]
plot(stl(res_har1, s.window = "per"))
```



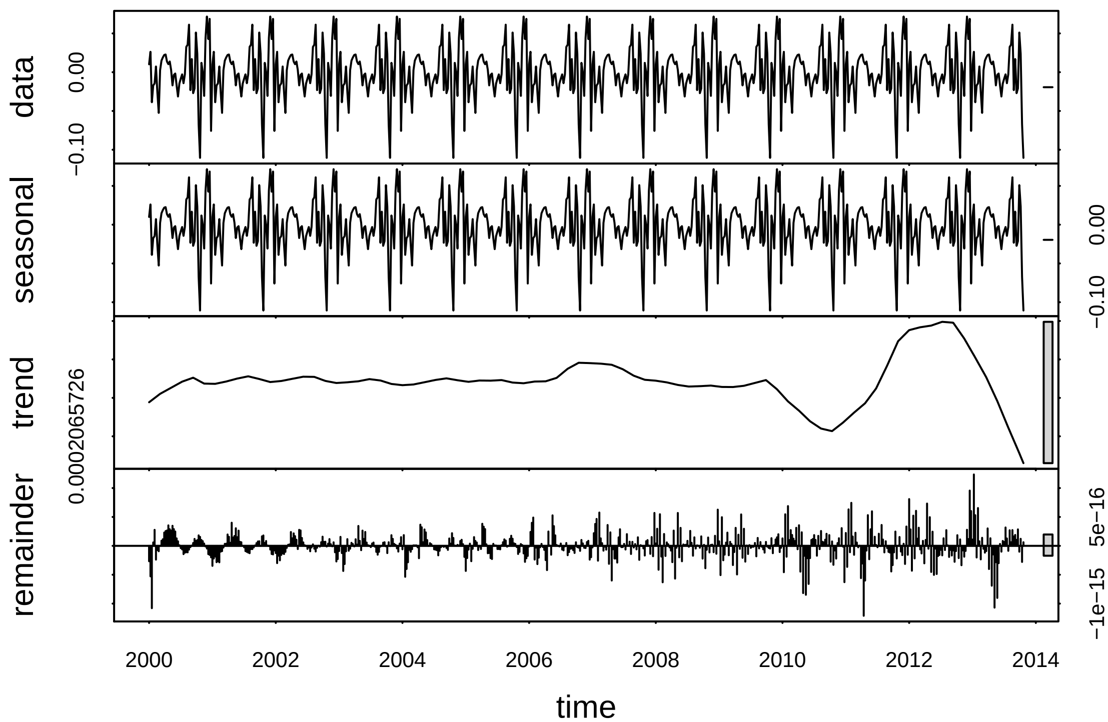

```r
# spec.ar(sea_har1)
```

Fit a second order harmonic model to the seasonality and model the seasonality in residuals. As the second order of harmonics does not explain more variance, we will stick to the first order harmonics.

```r
res_har2 <- residuals(lm(seasonality ~ co + si + co2 + si2))
summary(lm(seasonality ~ co + si + co2 + si2))
```

```
##
## Call:
## lm(formula = seasonality ~ co + si + co2 + si2)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.110493 -0.017217 -0.000125  0.022234  0.071803
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0002066  0.0014155  -0.146 0.884022
## co           0.0074311  0.0020079   3.701 0.000234 ***
## si           0.0492406  0.0019957  24.673  < 2e-16 ***
## co2         -0.0325992  0.0020000 -16.299  < 2e-16 ***
## si2          0.0240451  0.0020034  12.002  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03569 on 631 degrees of freedom
## Multiple R-squared:  0.6195, Adjusted R-squared:  0.6171
## F-statistic: 256.8 on 4 and 631 DF,  p-value: < 2.2e-16
```

```r
res_har2 <- ts(res_har2, start = c(2000, 1), frequency = 46)
sea_har2 <- stl(res_har2, s.window = "per")$time.series[, "seasonal"]
plot(stl(res_har2, s.window = "per"))
```

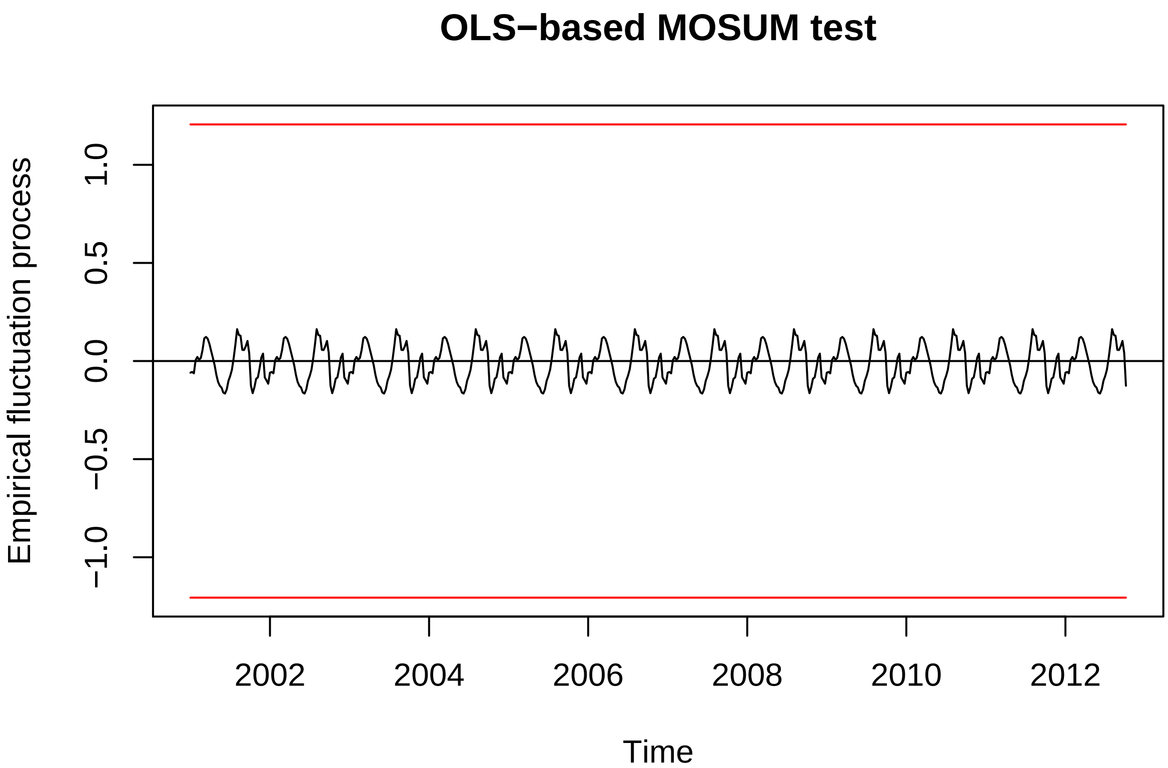



```
# spec.ar(res_har2)
```

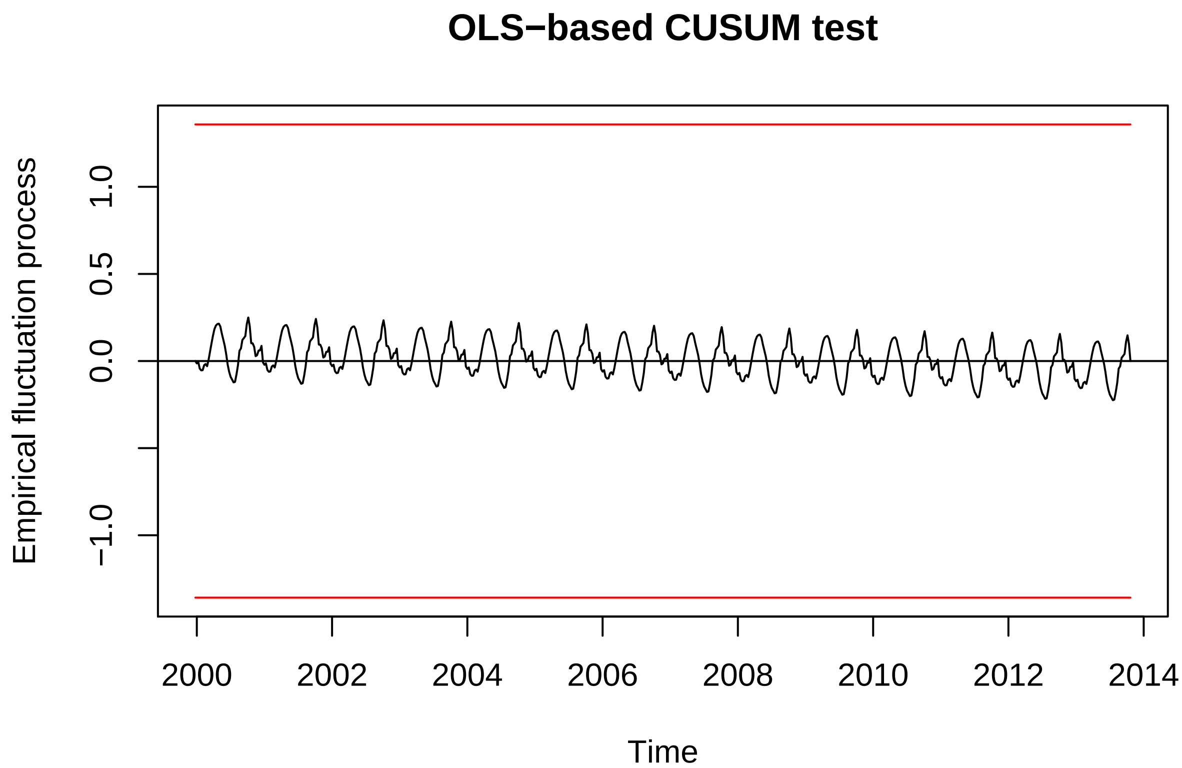
```
# res_har3 <- residuals(lm(seasonality ~ co + si +co2 + si2 +
# co3 + si3)) res_har3 <- ts(res_har3, start=c(2000, 1),
# frequency=46) sea_har3<- stl(res_har3, s.window =
# 'per')$time.series[,'seasonal'] plot(stl(res_har3, s.window
# = 'per')) spec.ar(sea_har3)
```

Use an empirical fluctuation test to test structural change in seasonality. The red lines indicate threshold of a change.

```
efp_har1 <- efp(seasonality ~ co + si, type = "OLS-MOSUM")
plot(efp_har1)
```




```r
efp_har1 <- efp(seasonality ~ co + si, type = "OLS-CUSUM")
plot(efp_har1)
```




```r
sctest(efp_har1)
```

```
##
##  OLS-based CUSUM test
##
## data:  efp_har1
## S0 = 0.24978, p-value = 1
```

Remove the seasonality.

```r
trend_rmstlsea <- originalts - seasonality
```

Check if any harmonic seasonality left after the seasonality is removed by stl:

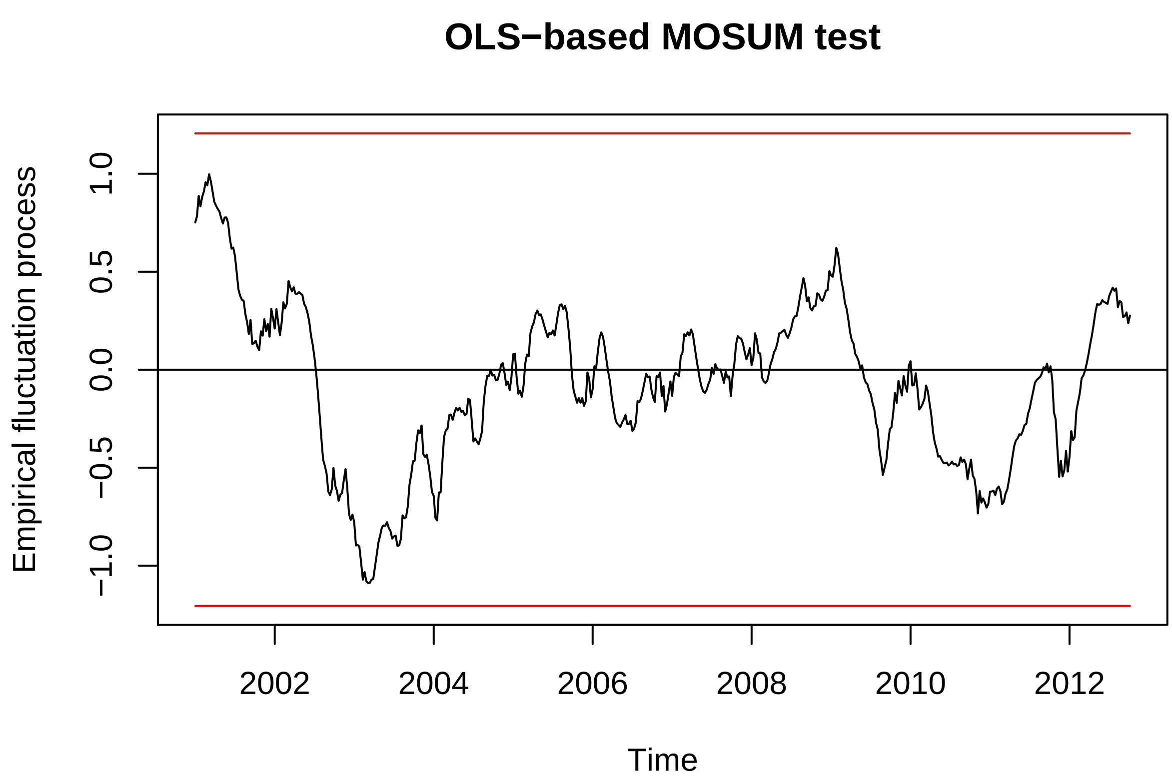
```r
summary(lm(trend_rmstlsea ~ tl + co + si))
```

```
##
## Call:
## lm(formula = trend_rmstlsea ~ tl + co + si)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36592 -0.07655  0.00724  0.08918  0.37666
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.635e-01  1.022e-02  45.349  < 2e-16 ***
## tl          -1.524e-04  2.781e-05  -5.480 6.14e-08 ***
```

```
## co            3.289e-04  7.237e-03   0.045    0.964
## si           -1.526e-03  7.198e-03  -0.212    0.832
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1286 on 632 degrees of freedom
## Multiple R-squared:  0.04541,    Adjusted R-squared:  0.04087
## F-statistic: 10.02 on 3 and 632 DF,  p-value: 1.853e-06
```

Use the empirical fluctuation test to test structural change in trend. The red lines indicate threshold of a change.

```
efp_trend <- efp(trend_rmstlsea ~ tl, type = "OLS-MOSUM")
plot(efp_trend)
```



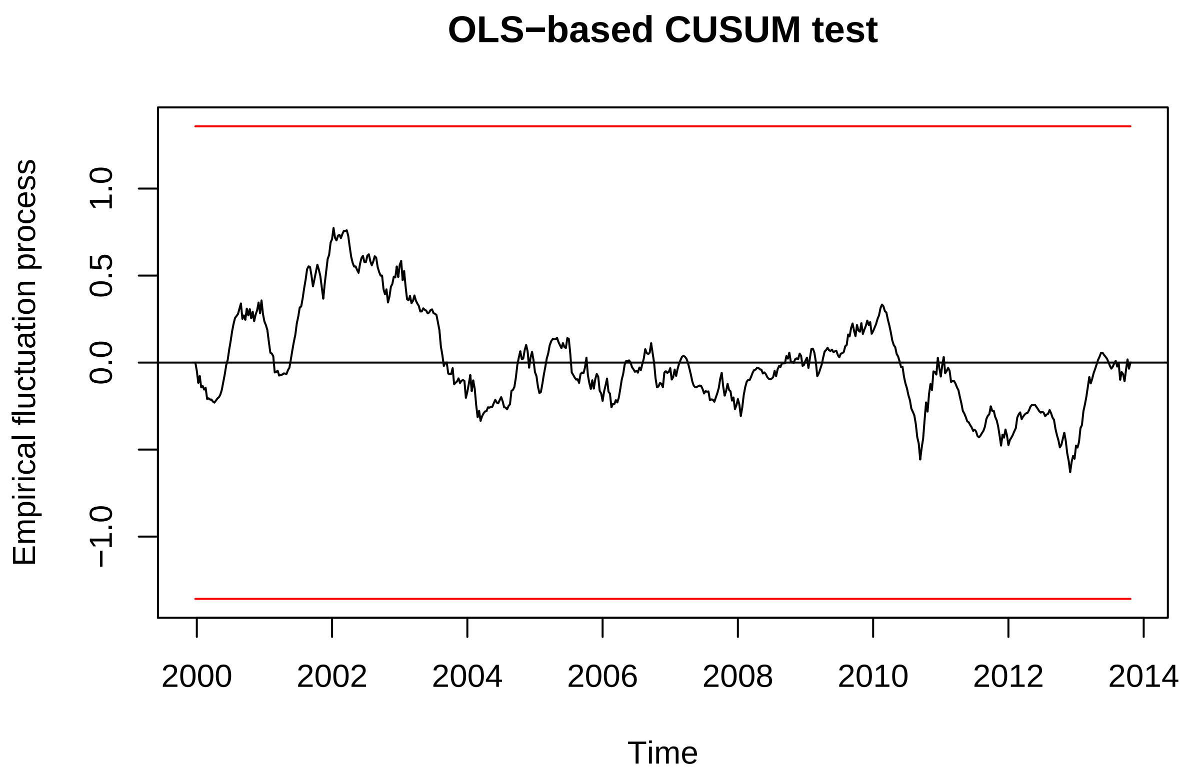

```
sctest(efp_trend)
```

```
##
##  OLS-based MOSUM test
##
## data:  efp_trend
## MO = 1.0889, p-value = 0.1259
```

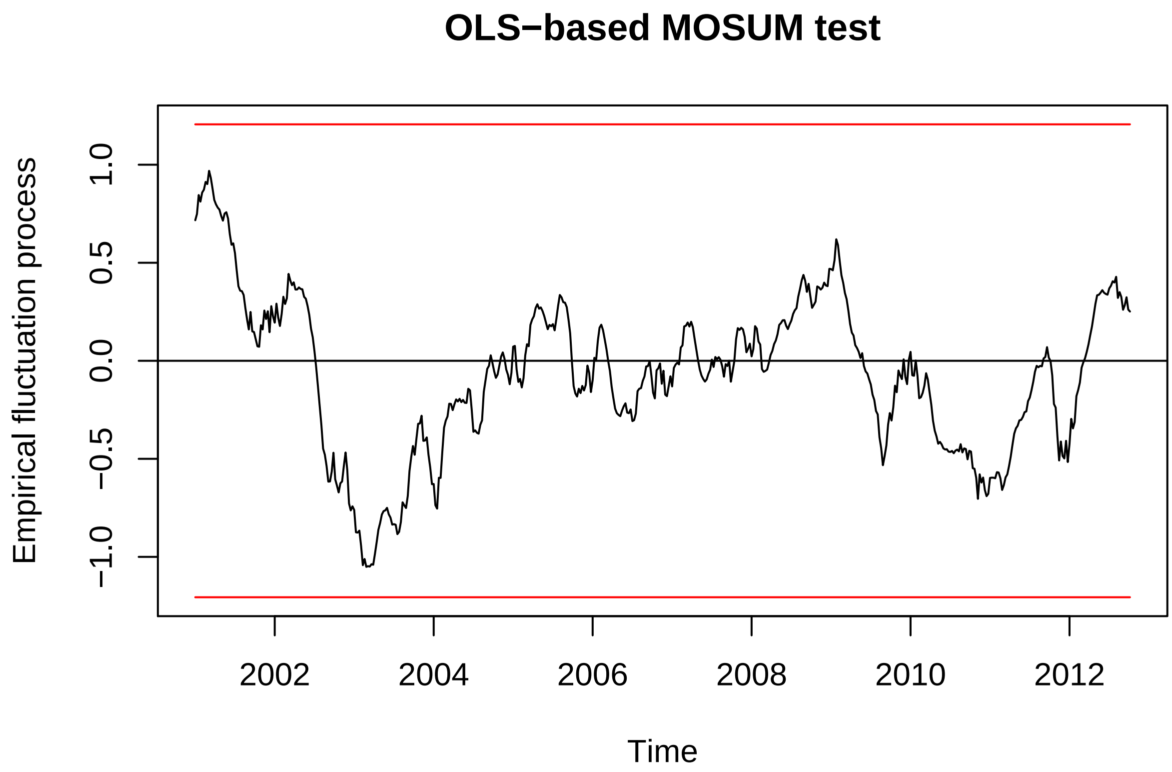
2. Regression on trend and harmonic terms at once: the BFAST Monitor method:

```
p.Vt1 <- efp(originalts ~ tl + co + co2 + co3 + si + si2 + si3,
    h = 0.15, type = "OLS-CUSUM")
```

```r
plot(p.Vt1)
```




```r
p.Vt1 <- efp(originalts ~ tl + co + co2 + co3 + si + si2 + si3,
    h = 0.15, type = "OLS-MOSUM")
plot(p.Vt1)
```



Spatial correlation

Here we checked a parameter (e.g. cosine coefficient) of seasonality. We could also check the spatial correlation in seasonality of model (e.g. BFAST) residuals, trend coefficients, as well as other seasonality coefficients.

```
coef_sea <- function(ts1, whichcoef) {
    originalts <- ts(ts1, start = c(2000, 1), frequency = 46)
    seasonality <- stl(originalts, s.window = "per")$time.series[,
        "seasonal"]
    trend = originalts - seasonality
    coefficients(lm(seasonality ~ co + si))[whichcoef]
}
seacoefco <- apply(fevi8, c(1, 2), coef_sea, 2)  # coefficients of the consine term
seacoefsi <- apply(fevi8, c(1, 2), coef_sea, 3)  # coefficients of the sine term
save(seacoefsi, file = "seacoefsi.Rdata")
save(seacoefco, file = "seacoefco.Rdata")
```

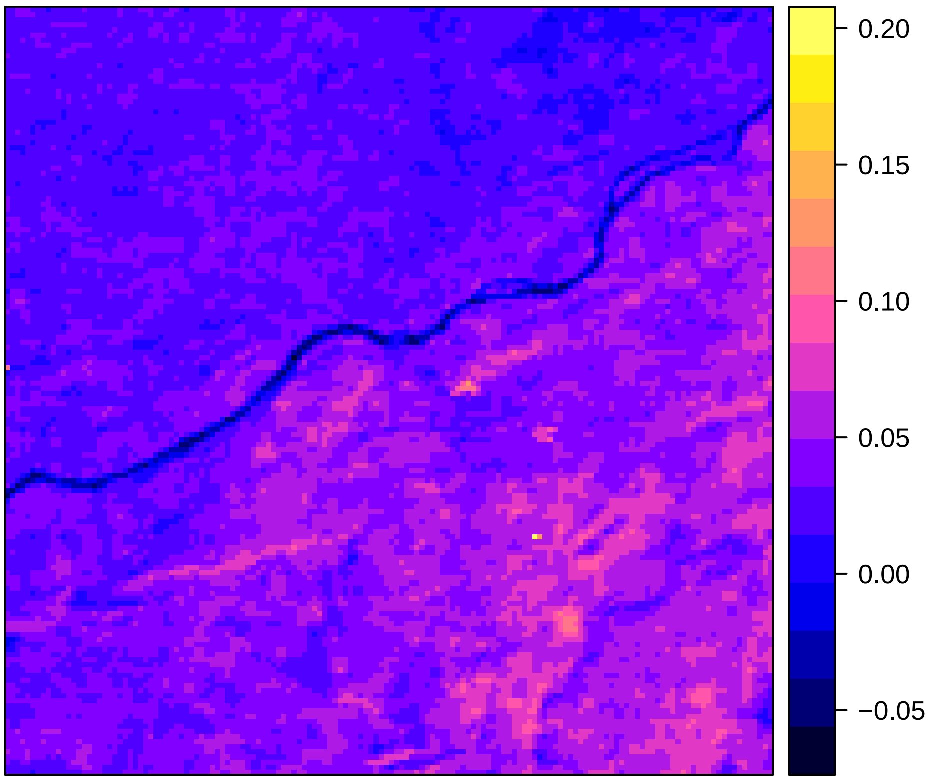
The seasonality coefficient takes around 1 min to run. In case of saving some time, we can load the seasonality coefficeints.

```
load("seacoefsi.Rdata")
load("seacoefco.Rdata")
```

We could see a spatial pattern in the seasonality coefficient: the sine term

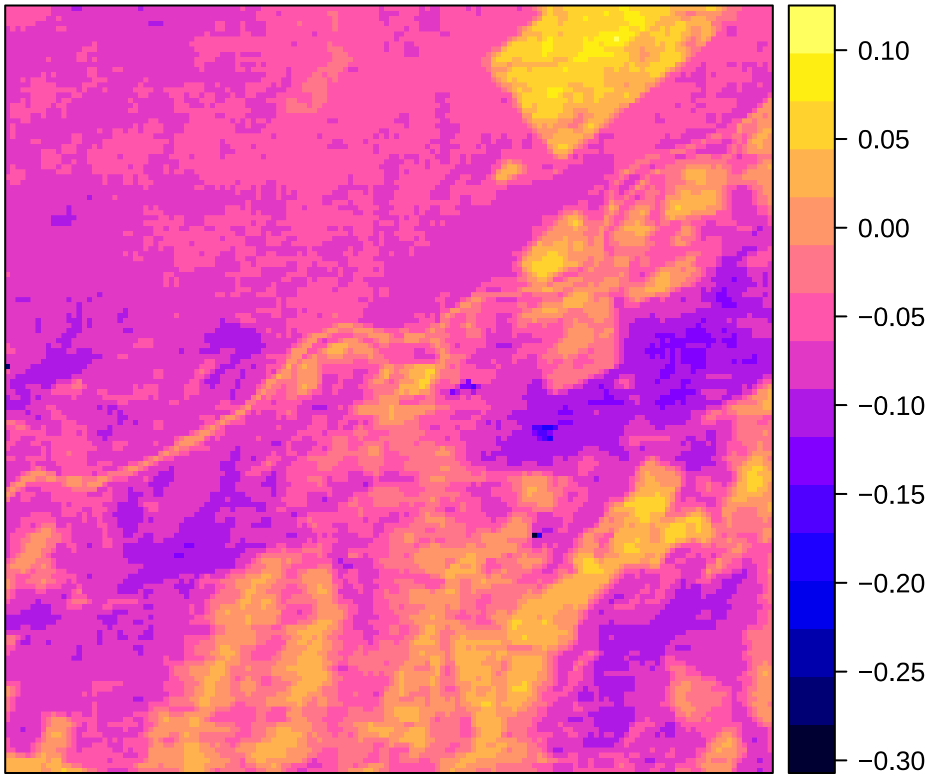
```
coor <- expand.grid(x = 1:dim(fevi8)[1], y = 1:dim(fevi8)[2])
sdfsi <- data.frame(seacoef = as.vector(seacoefsi), coor)
```

```r
coordinates(sdfsi) <- ~x + y
sdf1 <- sdfsi
gridded(sdf1) <- TRUE
spplot(sdf1)
```



We could see a spatial pattern in the seasonality coefficient: the cosine term

```r
sdfco <- data.frame(seacoef = as.vector(seacoefco), coor)
coordinates(sdfco) <- ~x + y
sdf1 <- sdfco
gridded(sdf1) <- TRUE
spplot(sdf1)
```

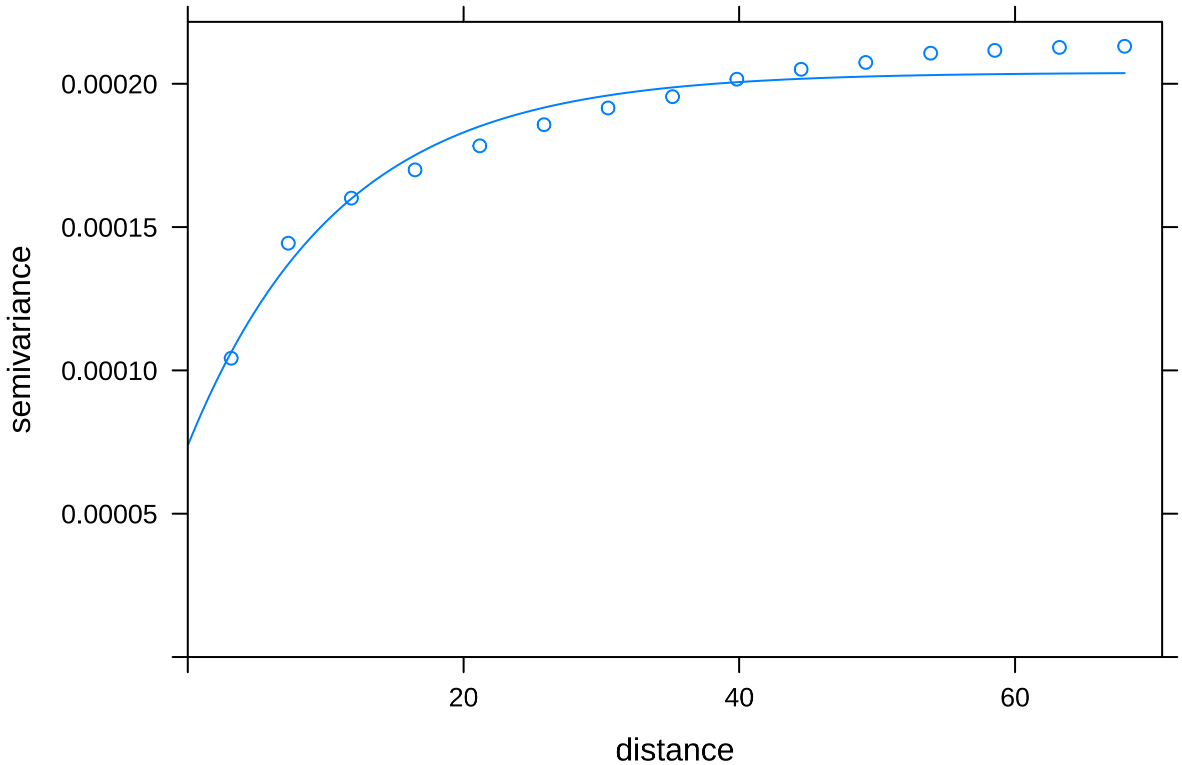


We could also have a look at the variogram and fit a variogram model. Here I regressed on locations to get rid of spatial trend. It is clear that semivariance increase with distance, which indicates spatial correlation. The sine term

```
v <- variogram(seacoef ~ x + y, sdfsi)
model3d <- fit.variogram(v, vgm(c("Exp", "Ste", "Sph", "Mat",
    "Gau")))
model3d
```

```
##   model        psill    range kappa
## 1   Nug 7.378884e-05  0.00000   0.0
## 2   Ste 1.301794e-04 15.48218   0.5
```

```
plot(v, model3d)
```

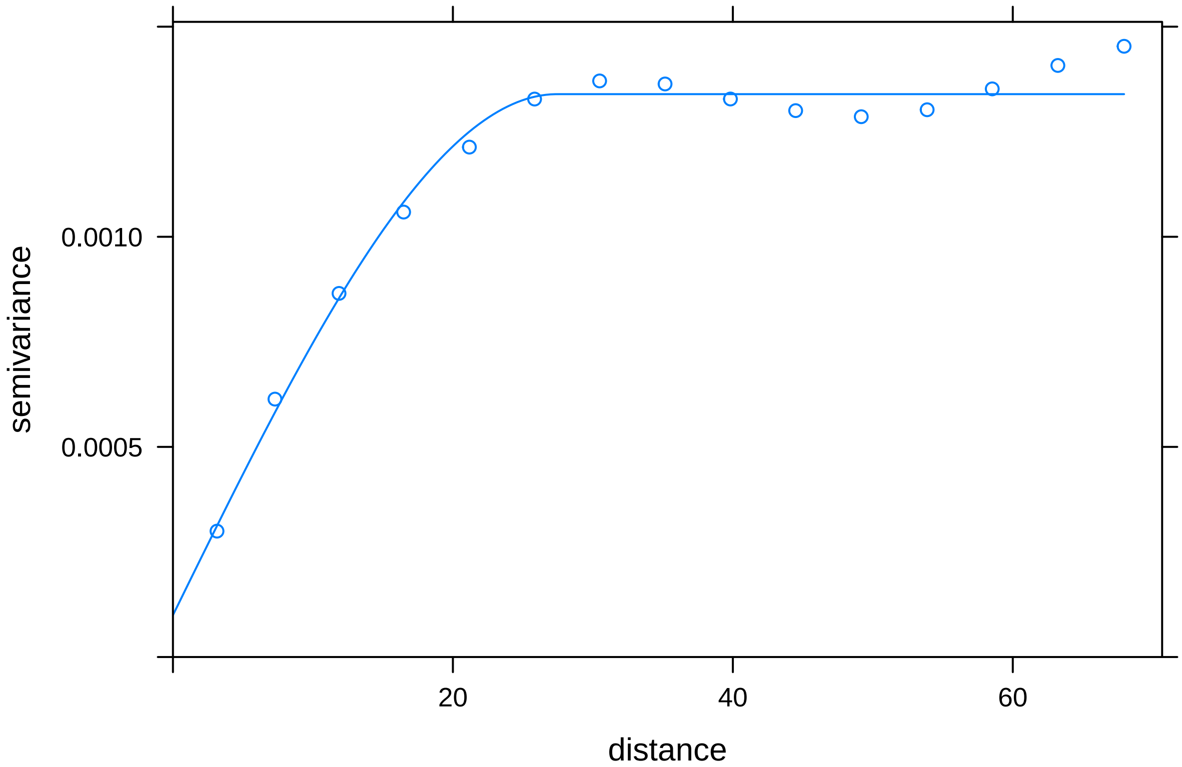


The cosine term:

```
v <- variogram(seacoef ~ x + y, sdfco)
model3d <- fit.variogram(v, vgm(c("Exp", "Ste", "Sph", "Mat",
    "Gau")))
model3d
```

```
##   model        psill    range
## 1   Nug 9.969915e-05  0.00000
## 2   Sph 1.239729e-03 27.42691
```

```
plot(v, model3d)
```



SAR integrated efp model:

Create spatiotemporal cubes and weight matrix.

```
eday <- as.Date("2000-01-30")  # date
e8day <- seq(eday, length.out = 636, by = "8 days")
xyd <- expand.grid(x1 = 1:3, y1 = 1:3)
coordinates(xyd) <- ~x1 + y1
lecube <- 3 * 3 * 636
aa3 <- as.data.frame(c(1:lecube))
stfdf3b3 <- STFDF(xyd, e8day, aa3)  ## for creating neighbors only, aa3 could be any data
cn <- cell2nb(3, 3, type = "queen", torus = FALSE)
neigh1 <- nbMult(cn, stfdf3b3, addT = FALSE, addST = FALSE)  # only spatial neighbours are added for ea
listcn636 <- nb2listw(neigh1)
```

Regressors (trend and seasonality) in a matrix

```
X = matrix(0, 636 * 9, 9 * 8)

for (i in 1:9) {
    X[seq(i, by = 9, length.out = 636), 1 + (i - 1) * 8] = 1
    X[seq(i, by = 9, length.out = 636), 2 + (i - 1) * 8] = tl
    X[seq(i, by = 9, length.out = 636), 3 + (i - 1) * 8] = co
    X[seq(i, by = 9, length.out = 636), 4 + (i - 1) * 8] = co2
    X[seq(i, by = 9, length.out = 636), 5 + (i - 1) * 8] = co3
    X[seq(i, by = 9, length.out = 636), 6 + (i - 1) * 8] = si
    X[seq(i, by = 9, length.out = 636), 7 + (i - 1) * 8] = si2
```

```
    X[seq(i, by = 9, length.out = 636), 8 + (i - 1) * 8] = si3
    colnames(X) = paste0("v", 1:(9 * 8))
    X
}
```

Load the modified version of strucchange, the only difference is the change in the efp function, for OLS-MOSUM and OLS-CUSUM tests. In the modified verson, structural change is analysed directly from the residuals of spatialtemporal model. The function efp() takes a "spatial1" variable (i.e. the modified version of efp: efp <- function(. . . ., spatial1 = list())) and skip the linear regression formula. The "spatial1" contains a list of residuals from SAR integrated time series regression mode.

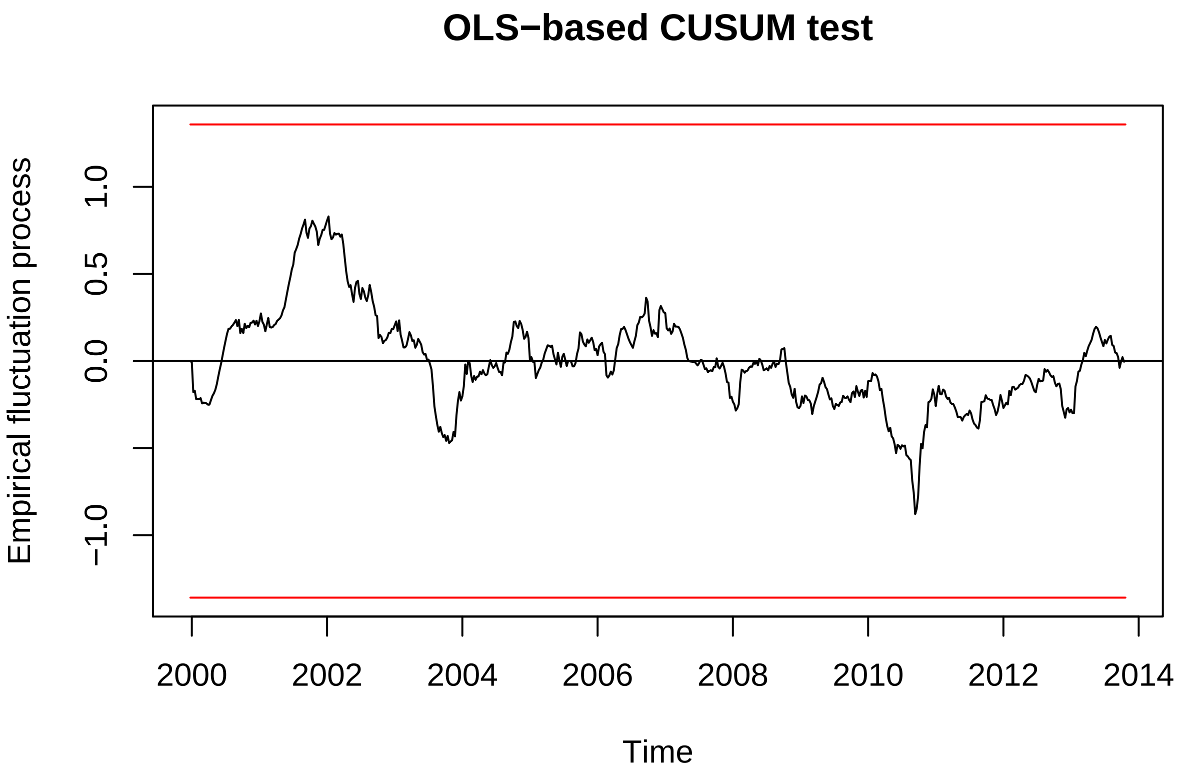
```
library(devtools)
install_github("mengluchu/strucchange", build_vignettes = FALSE)
```

SAR integrated efp. The most time-consuming process is the SAR model (spautolm). It costs 22 seconds to run on my computer.

```
f2 <- fevi8[(lon - 1):(lon + 1), (lat - 1):(lat + 1), ]
fevi3b312t1 <- ts(f2[2, 2, ], start = c(2000, 1), frequency = 46)  # reconstruct the time series
aa2 <- as.vector(f2)
system.time(try2 <- spautolm(aa2 ~ ., data.frame(aa2, X), family = "SAR",
    method = "Matrix", listw = listcn636))
```

```
##    user  system elapsed
##   22.47    0.66   23.12
```

```
rn <- lapply(1:9, function(i) {
    residuals(try2)[seq(i, 636 * 9 - (9 - i), 9)]
})
p.Vt1 <- efp(fevi3b312t1 ~ 1, h = 0.15, type = "OLS-CUSUM", spatial1 = as.numeric(rn[[5]]))
plot(p.Vt1)
```

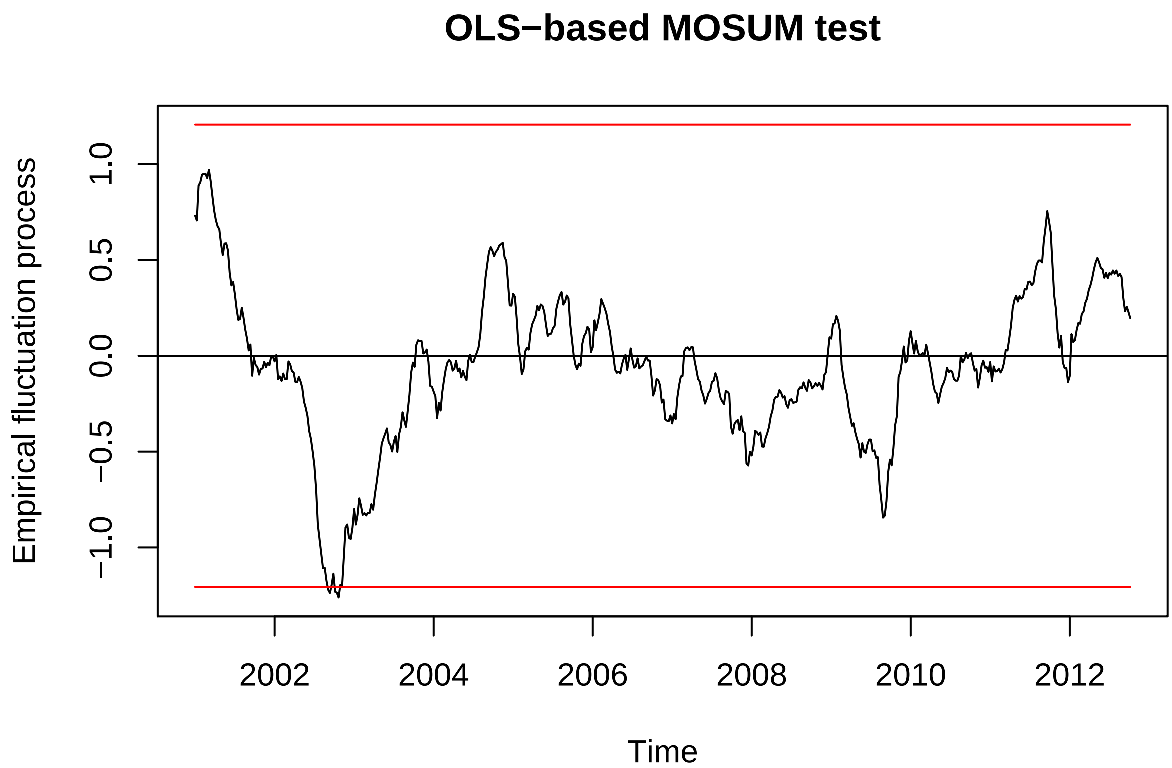



```
sctest(p.Vt1)$p.value
```

```
##        S0
## 0.4229956
```

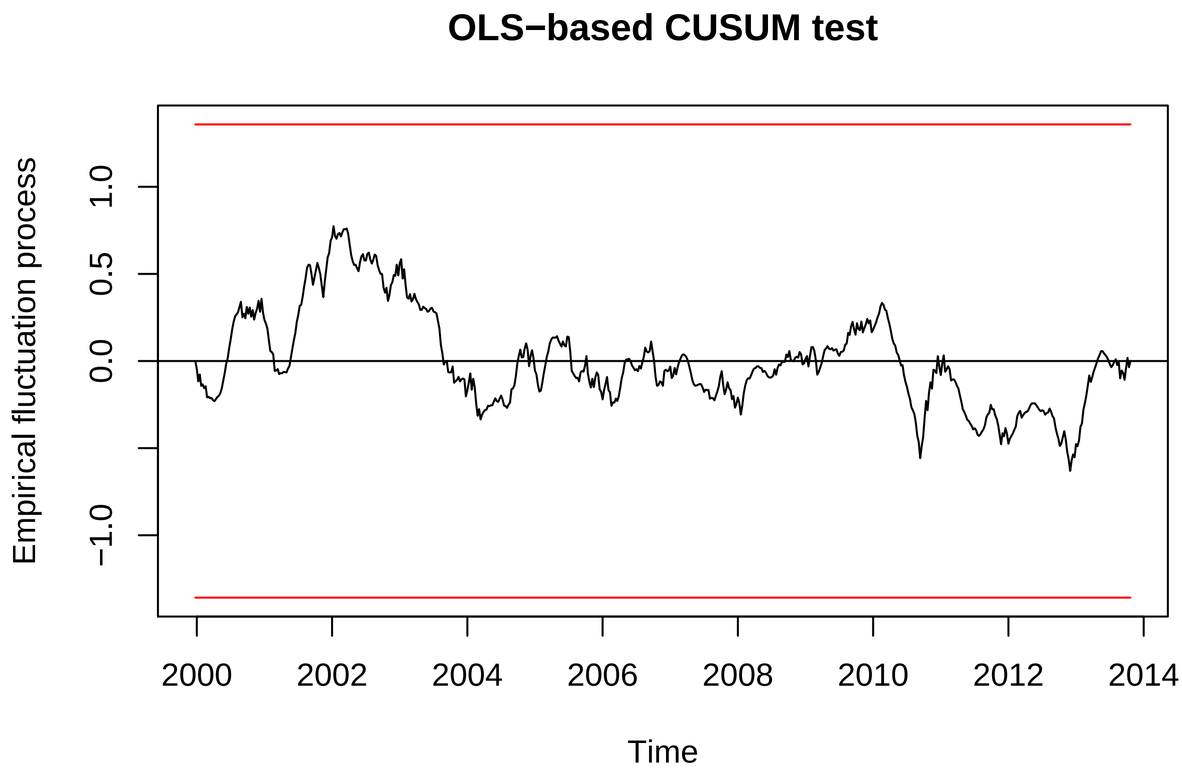
OLS-MOSUM method:

```
p.Vt1 <- efp(fevi3b312t1 ~ 1, h = 0.15, type = "OLS-MOSUM", spatial1 = as.numeric(rn[[5]]))
plot(p.Vt1)
```



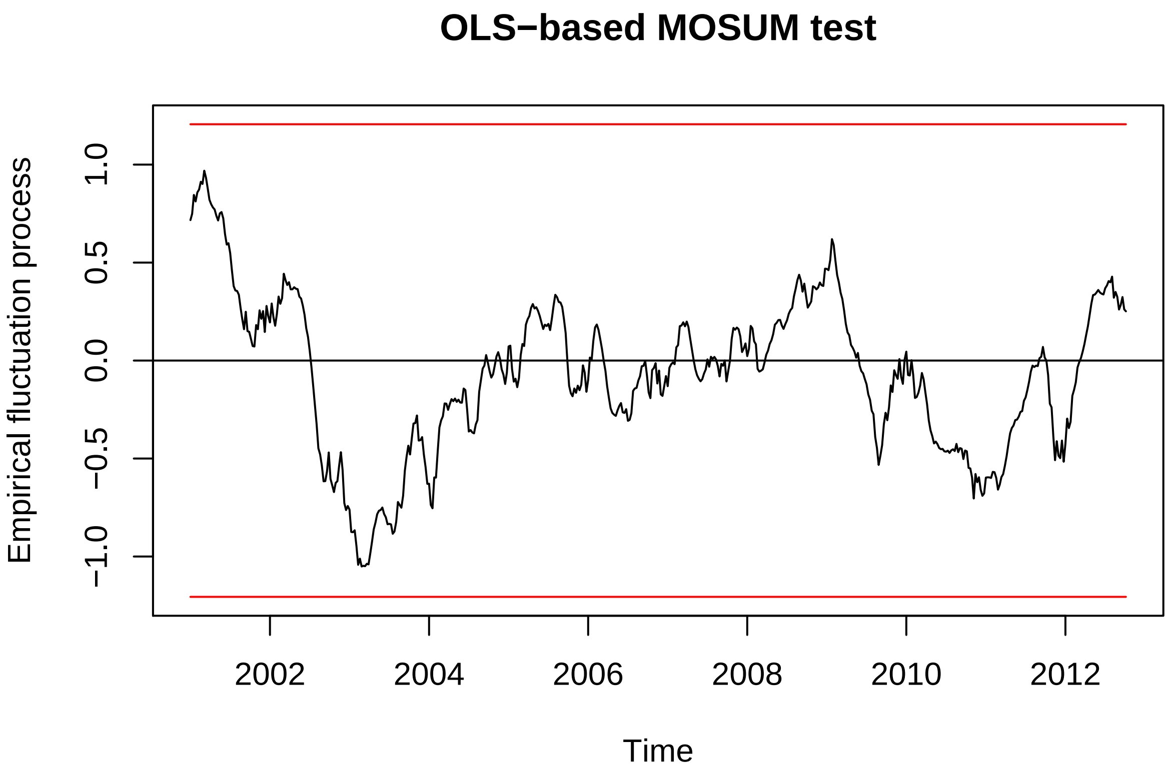
Comparing with the pure time series anaylsis: OLS-CUSUM

```
p.Vt1 <- efp(fevi3b312t1 ~ tl + co + co2 + co3 + si + si2 + si3,
    h = 0.15, type = "OLS-CUSUM")
plot(p.Vt1)
```



OLS-MOSUM

```
p.Vt1 <- efp(fevi3b312t1 ~ tl + co + co2 + co3 + si + si2 + si3,
    h = 0.15, type = "OLS-MOSUM")
plot(p.Vt1)
```



Detect change using structural change test and store the p-value into an array. Here is an example conduct the analysis for 4 $3*3*636$ spatiotemporal cubes.

```
tssar1 <- array(NA, c(2, 2))

for (i in 30:31) {
    for (j in 30:31) {
        f2 <- fevi8[i:(i + 2), j:(j + 2), ]
        fevi3b312t1 <- ts(f2[2, 2, ], start = c(2000, 1), frequency = 46)  # reconstruct the time serie
        aa2 <- as.vector(f2)
        try2 <- spautolm(aa2 ~ ., data.frame(aa2, X), family = "SAR",
            method = "Matrix", listw = listcn636)
        rn <- lapply(1:9, function(i) {
            residuals(try2)[seq(i, 636 * 9 - (9 - i), 9)]
        })
        p.Vt1 <- sctest(efp(fevi3b312t1 ~ 1, h = 0.15, type = "OLS-CUSUM",
            spatial1 = as.numeric(rn[[5]])))
        tssar1[i, j] < -p.Vt1$p.value
    }
}
```

Pure time series analysis:

```
ts1 <- array(NA, c(2, 2))
system.time(for (i in 1:2) {
    for (j in 1:2) {
        f2 <- fevi8[i:(i + 2), j:(j + 2), ]
```

```
        fevi3b312t1 <- ts(f2[2, 2, ], start = c(2000, 1), frequency = 46)  # reconstruct the time series
        p.Vt1 <- sctest(efp(fevi3b312t1 ~ tl + co + co2 + co3 +
            si + si2 + si3, h = 0.15, type = "OLS-CUSUM"))
        ts1[i, j] <- p.Vt1$p.value
    }
})
```

P-values for each pixels.

```
tssar1
ts1
```

Scale the SAR-efp with SciDB and reproduce the results of a study case in "Spatio-Temporal Change Detection from Multidimensional Arrays: detecting deforestation from MODIS time series", ISPRS journal, Mar, 2016:

https://github.com/mengluchu/scalable-spatial-temporal-BFAST