Adaptive spatio-temporal intensity by partitioning algorithm

First example: Log-Gaussian spatio-temporal point pattern

Generating an observation

First, we want to check the intensity using an artificial point pattern. Therefore, we simulate a spatio-temporal point pattern (in this case, we fix the number of points prior to simulation).

```
# Number of points coming from a Poisson distribution with mean 1000
N <- rpois(1, 1000)
```

Then we set a covariance model for the underlying random field; in our case, we chose a non-separable Geneiting covariance function with its parameters. We set the additional parameters for the random field simulation as the mesh, variance and scales.

```
logGaussianPP <- rlgcp(npoints = N, nx = 128, ny = 128, nt = 64,
                       separable = F, model = "gneiting", scale = c(0.5, 0.8),
                       param = c(1, 1, .1, 0.1, 1, 2), var.grf = 2, mean.grf = 1)
```

We want to plot the point pattern generated above in 2D, so we employ the **spatstat** package where we see our point pattern as a marked spatial point pattern.

```
# Spatstat format
XX <- ppp(x = logGaussianPP$xyt[, 1], y = logGaussianPP$xyt[, 2],
          marks = logGaussianPP$xyt[, 3], window = owin())

# Set the color scheme
colmap <- colourmap(rainbow(512), range = range(marks(XX)))
sy <- symbolmap(pch = 21, bg = colmap, range = range(marks(XX)))

# Plotting
plot(XX, symap = sy, 'log-Gaussian Cox point pattern')
```

Estimating global bandwidths

Before using the adaptive estimator, we should set the global bandwidths, i.e., the bandwidths for estimating the pilot intensities. These pilot intensities will assign a particular bandwidth to each data point.

```
# Global bandwidths
bwS0 <- OS(XX) # The spatial bandwidth will be the oversmoothing version
bwt0 <- bw.SJ(XX$marks) # We employ Sheather & Jones's bandwidth for time

# Spatial and temporal bandwidths based on pilot estimations
bwS <- bw.abram(unmark(XX), h0 = bwS0)
bwt <- bw.abram.temp(t = XX$marks, h0 = bwt0)
```

Intensity estimates

First we use a fixed-bandwidth kernel estimate for the intensity using a function provided by the **sparr** package.

```
# Fixed bandwidth estimate (non-separable)
classic.dens <- spattemp.density(pp = unmark(XX), tt = XX$marks,
                                 sres = 128, tres = 64,
                                 lambda = bwt0, h = bwS0,
                                 sedge = "uniform")
```

Now we compute the adaptive intensity by using our partitioning algorithm. For doing that, we have to set first a fixed number of groups for the spatial and the temporal bandwidths; it could be done manually or let the function decide based on a rule-of-thumb. In addition, we set a temporal resolution of 64 pixels (the default is 128 for space and for time). We can evaluate the intensity at the data points (at = "points"), or obtain snapshots (at = "bins", the default).

```
# In this case we use 8 groups for space and 4 for time
adapt.dens <- dens.par(X = XX,
                       dimt = 64,
                       bw.xy = bwS, bw.t = bwt,
                       ngroups.xy = 8, ngroups.t = 4,
                       at = "bins")
```

We plot some snapshots of the estimates that we have computed.

```
# We select some fixed times for visualisation
I <- c(13, 17, 21, 31, 50)

# We subset the lists
CN <- as.imlist(classic.dens$z[I])
AN <- as.imlist(adapt.dens[I])

# We generate the plots
plot.imlist(CN, ncols = 5, main = 'Classic fixed-bandwidth estimate')
plot.imlist(AN, ncols = 5, main = 'Adaptive non-separable estimate')
```

Second example: Amazonia Data

Loading and visualising data

We want to estimate the intensity of the Amazonia fires.

```
# Loading dataset
data("amazon")
```

Now, given that the number of points is huge, for visualisation we select a sample of 5000.

```
# Extract a sample of 5000 data points
AmazonReduced <- amazon[sample.int(amazon$n, 5000)]

# Set the color scheme
colmap <- colourmap(rainbow(512), range = range(marks(AmazonReduced)))
sy <- symbolmap(pch = 21, bg = colmap, range = range(marks(AmazonReduced)))

# Plotting
plot(AmazonReduced, symap = sy, 'Sample of Amazonia fires')
```

Separability test

Before estimating the intensity, it is very convenient to know whether separability holds. When separability holds, estimating the spatio-temporal intensity becomes easier and faster. So we perform a separability test to have an approximate answer to that question. The test is performed in a Monte Carlo fashion; therefore, we have to fix a number of Monte Carlo samples prior to the test, say 1500.

```
separability.test(X = amazon, nperm = 1500)
```

Given the test result, we should not assume separability.

Estimating global bandwidths

We select the bandwidths for estimating the pilot intensities.

```
# Global bandwidths
bwSO <- OS(amazon) # The spatial bandwidth will be the oversmoothing version
```

```
bwt0 <- bw.nrd(amazon$marks) # We employ Scott bandwidth for time

# Spatial and temporal bandwidths based on pilot estimations
bwS <- bw.abram(amazon, h0 = bwS0)
bwt <- bw.abram.temp(t = amazon$marks, h0 = bwt0)
```

Intensity estimates

We use a fixed-bandwidth kernel estimate for the intensity

```
# Fixed bandwidth estimate (non-separable)
# This could be time consuming
classic.dens <- spattemp.density(pp = unmark(amazon), h = bwS0, tt = amazon$marks,
                                 sres = 64, tres = 64,
                                 lambda = bwt0,
                                 sedge = "uniform")
```

Now we compute the adaptive intensity by using our partitioning algorithm.

```
# In this case we let the algorithm to choose the numbers of groups
# It could be time consuming
adapt.dens <- dens.par(X = amazon,
                       dimyx = 64, dimt = 64,
                       bw.xy = bwS, bw.t = bwt,
                       at = "bins")
```

We plot some snapshots of the estimates that we have computed.

```
# We select some fixed times for visualisation
I <- c(12, 18, 23, 31, 55)

# We subset the lists
CN <- as.imlist(classic.dens$z[I])
AN <- as.imlist(adapt.dens[I])

# We generate the plots
plot.imlist(CN, ncols = 5, main = 'Classic fixed-bandwidth estimate')
plot.imlist(AN, ncols = 5, main = 'Adaptive non-separable estimate')
```

Dynamic intensity estimate

```
animation::saveVideo(
  for(i in 1:length(adapt.dens)){
    plot(adapt.dens[[i]], main = paste("Time",i))
  },
  video.name="amazon.mp4", other.opts = '-b:v 1M -pix_fmt yuv420p',
  ani.width = 640, ani.height = 640, interval = 1 / 12)
```