Vignette moveNT

Guillaume Bastille-Rousseau

April 18, 2017

Simulating movement strategies - *sim_mov*

The function *sim_mov* generates movement trajectories including patches and movement between patches. Movement within patches can follow an Ornstein-Uhlenbeck process (based on *simm.mou* function from package *adehabitatLT*) or two-states movement model (based on *simmData* function from package *moveHMM*). Movement between patches is following a brownian bridge movement model (based on *simm.bb* function from package *adehabitatLT*). Generated outputs are of the class *ltraj* from package *adehabitatlt*.

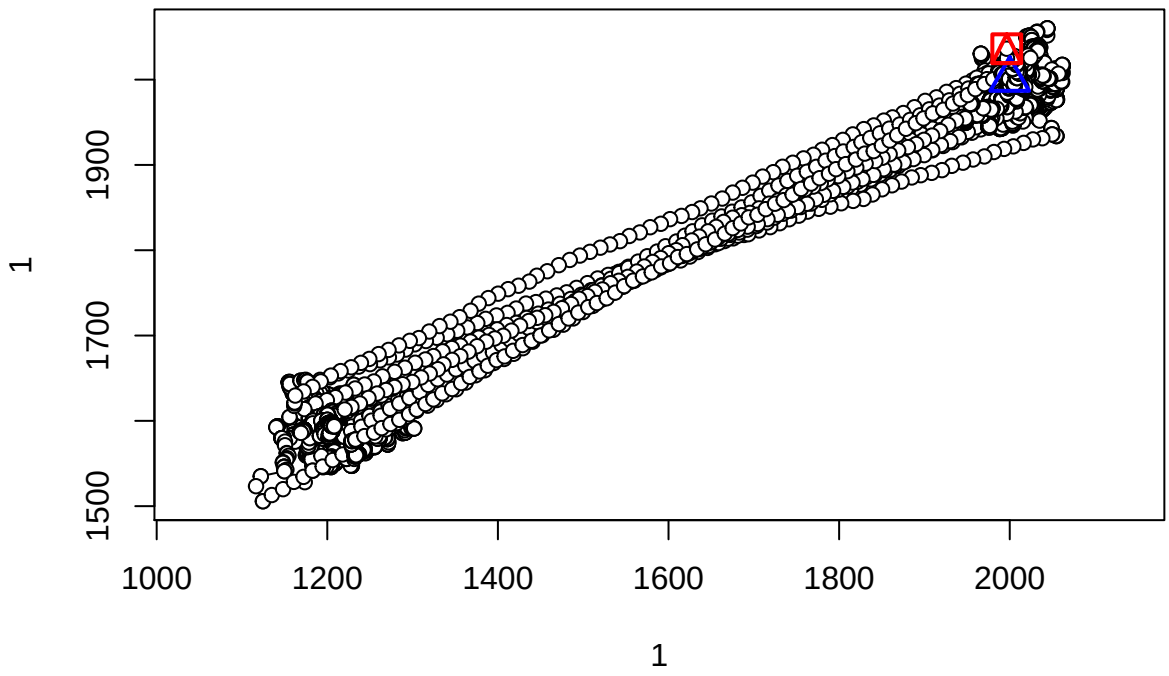
```
# Simulating migration with two-states model
mig<-sim_mov(type="2states", npatches=2, ratio=2, nswitch=25, ncore=150, grph=F)
mig
```

```
##
## ************ List of class ltraj ************
##
## Type of the traject: Type II (time recorded)
## * Time zone: GMT *
## Regular traject. Time lag between two locs: 1 seconds
##
## Characteristics of the bursts:
##   id burst nb.reloc NAs          date.begin            date.end
## 1 id    id     4650   0 1960-01-01 00:00:01 1960-01-01 01:17:30
##
##
##  infolocs provided. The following variables are available:
## [1] "out.Corri"
```

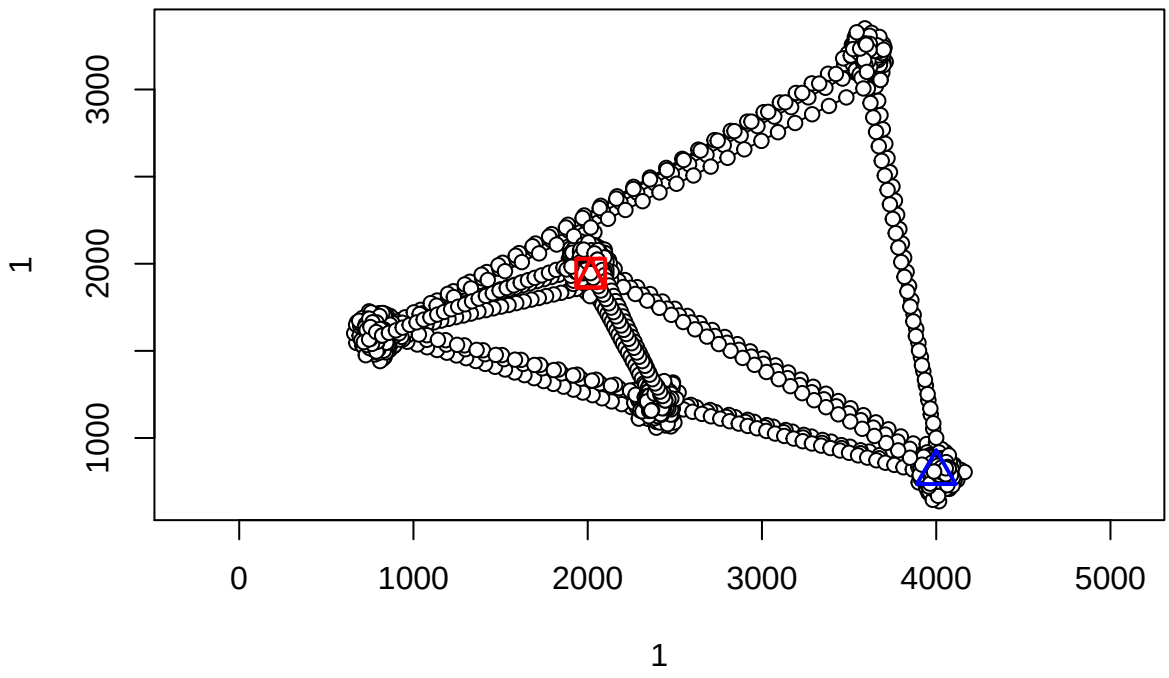
```
head(ld(mig))
```

```
##          x        y                date         dx         dy       dist
## 1 2000.000 2000.000 1960-01-01 00:00:01  8.9854214 -2.3249521  9.2813362
## 2 2008.985 1997.675 1960-01-01 00:00:02  1.8401376  5.4455729  5.7480754
## 3 2010.826 2003.121 1960-01-01 00:00:03  3.7973301  1.9428722  4.2654974
## 4 2014.623 2005.063 1960-01-01 00:00:04  9.9221723 14.9207144 17.9186278
## 5 2024.545 2019.984 1960-01-01 00:00:05 -0.2601030 -0.6208479  0.6731312
## 6 2024.285 2019.363 1960-01-01 00:00:06  0.2583454 -0.4682957  0.5348300
##   dt       R2n  abs.angle  rel.angle id burst out.Corri
## 1  1    0.0000 -0.2531942         NA id    id         2
## 2  1   86.1432  1.2449285  1.4981227 id    id         2
## 3  1  126.9310  0.4729175 -0.7720110 id    id         2
## 4  1  239.4678  0.9839532  0.5110357 id    id         2
## 5  1 1001.8286 -1.9675298 -2.9514831 id    id         2
## 6  1  964.6989 -1.0666707  0.9008591 id    id         2
```

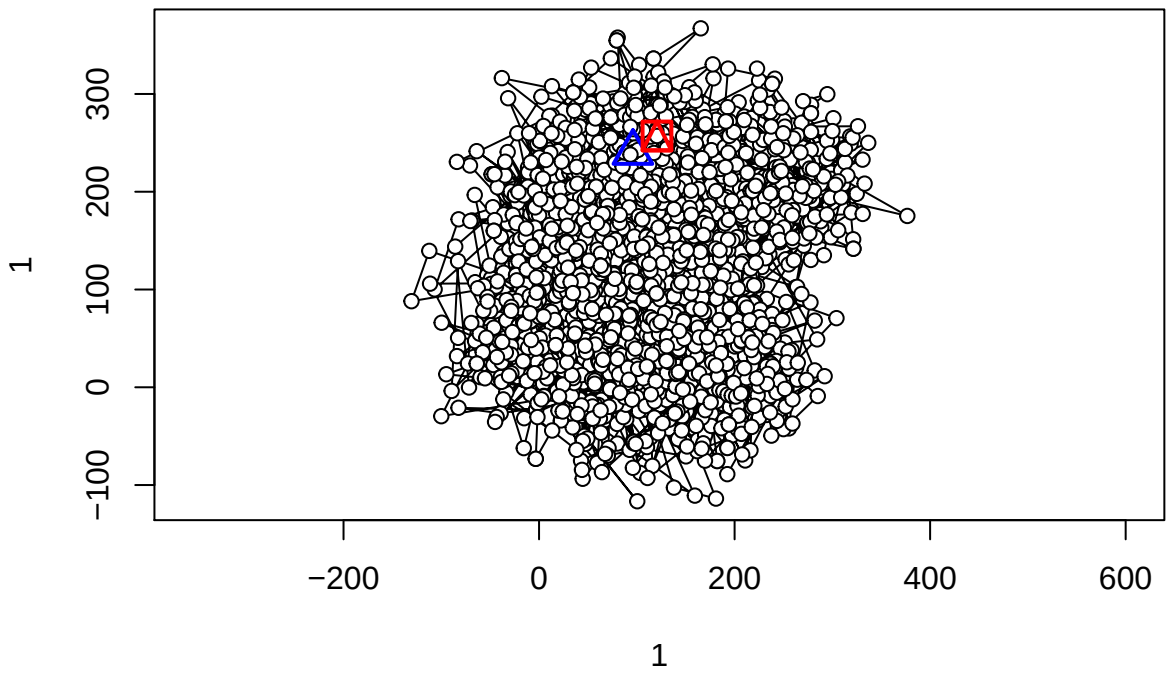
```
plot(mig)
```




```
# Simulating multi-patches movement with Ornstein-Uhlenbeck process
patches<-sim_mov(nswitch=25, ncore=150, ratio=5, type="OU", npatches=5, grph=T)
```




```
# Simulating sedentary movement
seden<-sim_mov(type="OU", npatches=10, spacecore=12, ratio=3, nswitch=150, ncore=20, grph=T)
```



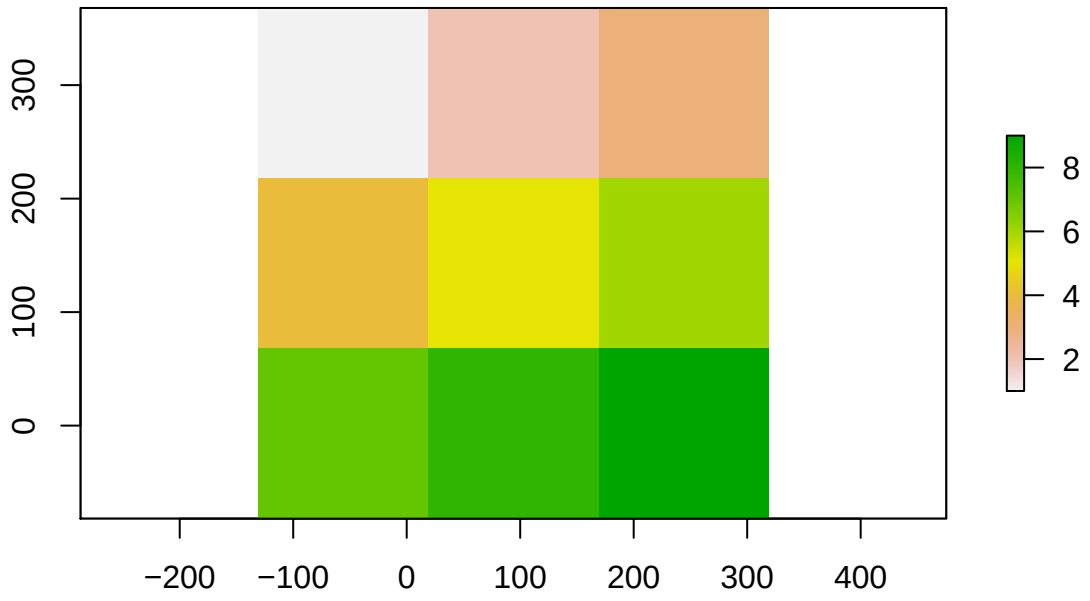
Converting movement to adjacency matrix - *traj2adj*

The function *traj2adj* converts a trajectory object of class *ltraj* to an adjacency matrix. This is done by overlapping a grid over the relocation data and tallying the number of transitions among each pixel. Users need to specify the grid size, which can be based on distance travelled. The function *quant* is a wrapper that allows to sample a quantile of step length distribution from a *ltraj* object. Output produced by *traj2adj* is a list containing the adjacency matrix, the grid used (raster format), and a raster indicating pixel numbers that are occupied. These rasters are used by other functions such as *adj2stack* and *clustnet*.

```
# Using sedentary movement and user specific grid-size
adj_seden<-traj2adj(seden, res=150) #Pixel size of 150m
adj_seden[[1]] # Adjency matrix
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
##  [1,]    9    8    0   10   10    0    0    0    0
##  [2,]    7  142    7   10   77   18    0    0    0
##  [3,]    0   13   80    0    9   54    0    0    0
##  [4,]    6   11    0  173   94    0   27   13    0
##  [5,]   15   78   12   96  866   92    9  115   26
##  [6,]    0    9   59    0  100  286    0   27   35
##  [7,]    0    0    0   25   11    0   69   24    0
##  [8,]    0    0    0   10  121   26   24  421   71
##  [9,]    0    0    0    0   21   38    0   74  196
```

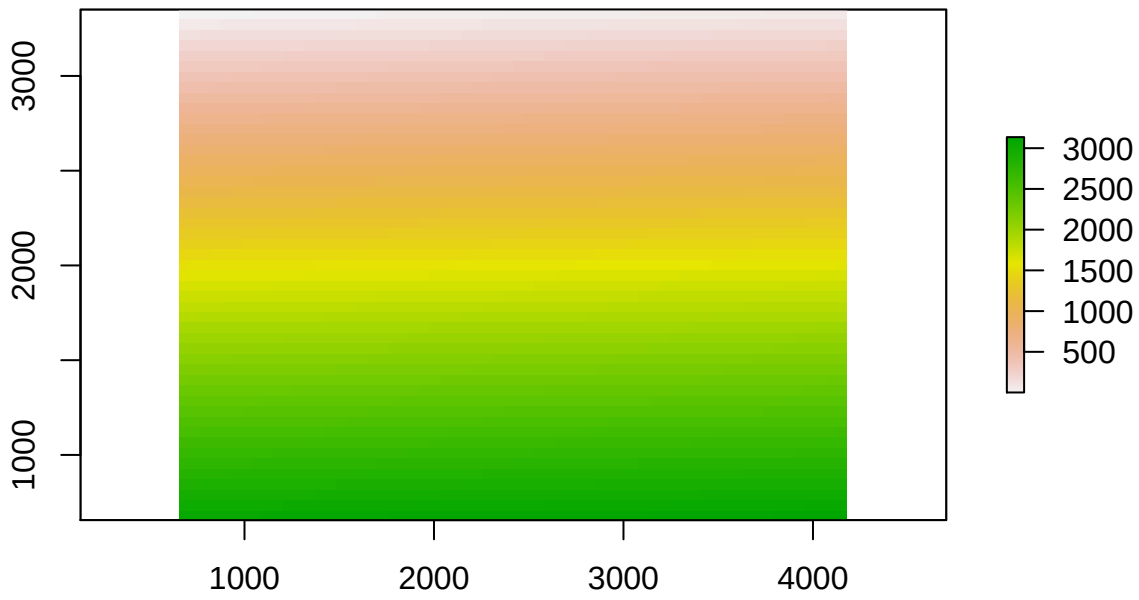
```r
plot(adj_seden[[2]]) #Plot grid used
```



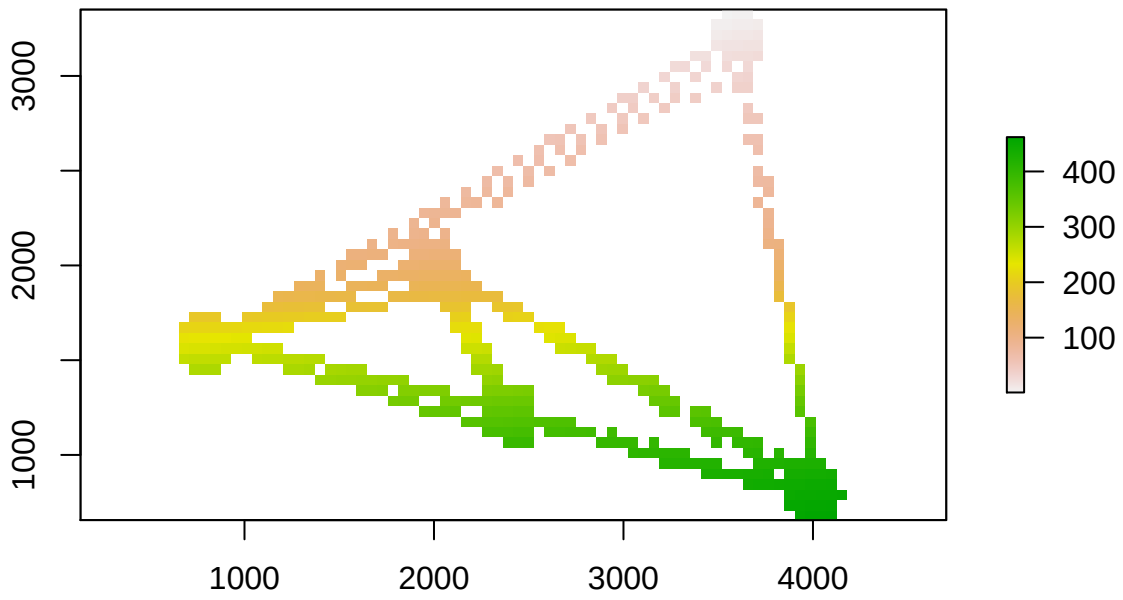

```r
# Using multi-patches movement and median distance travelled
adj_patches<-traj2adj(patches, res=quant(patches, p=0.5)) #Grid size based on median
dim(adj_patches[[1]]) # Size of the adjacency matrix
```

```
## [1] 462 462
```

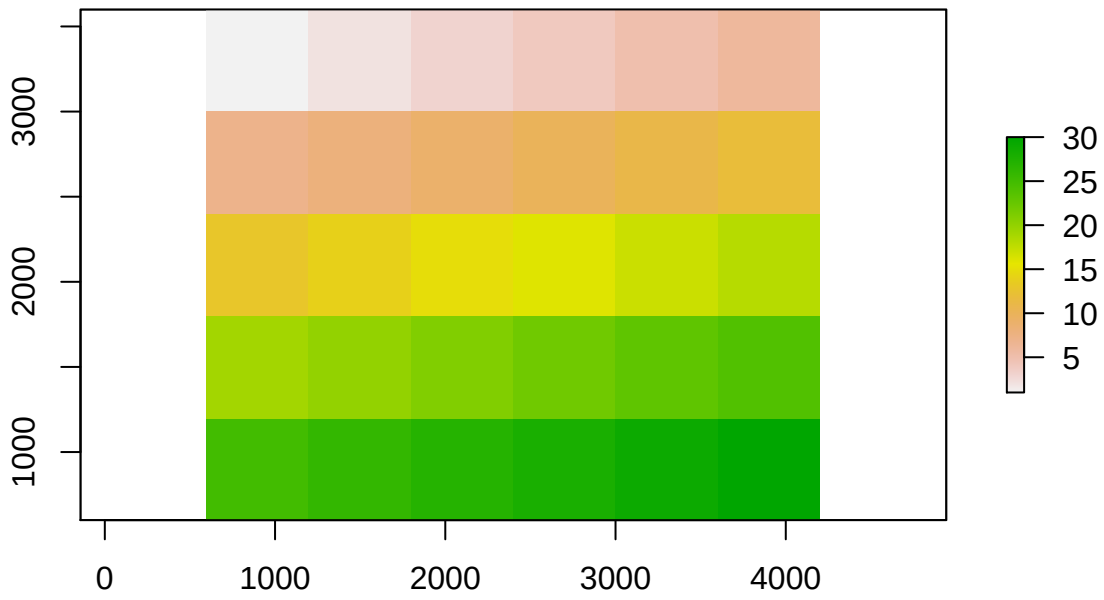
```r
plot(adj_patches[[2]]) #Plot grid used
```




```
plot(adj_patches[[3]]) #Plot occupied pixels
```



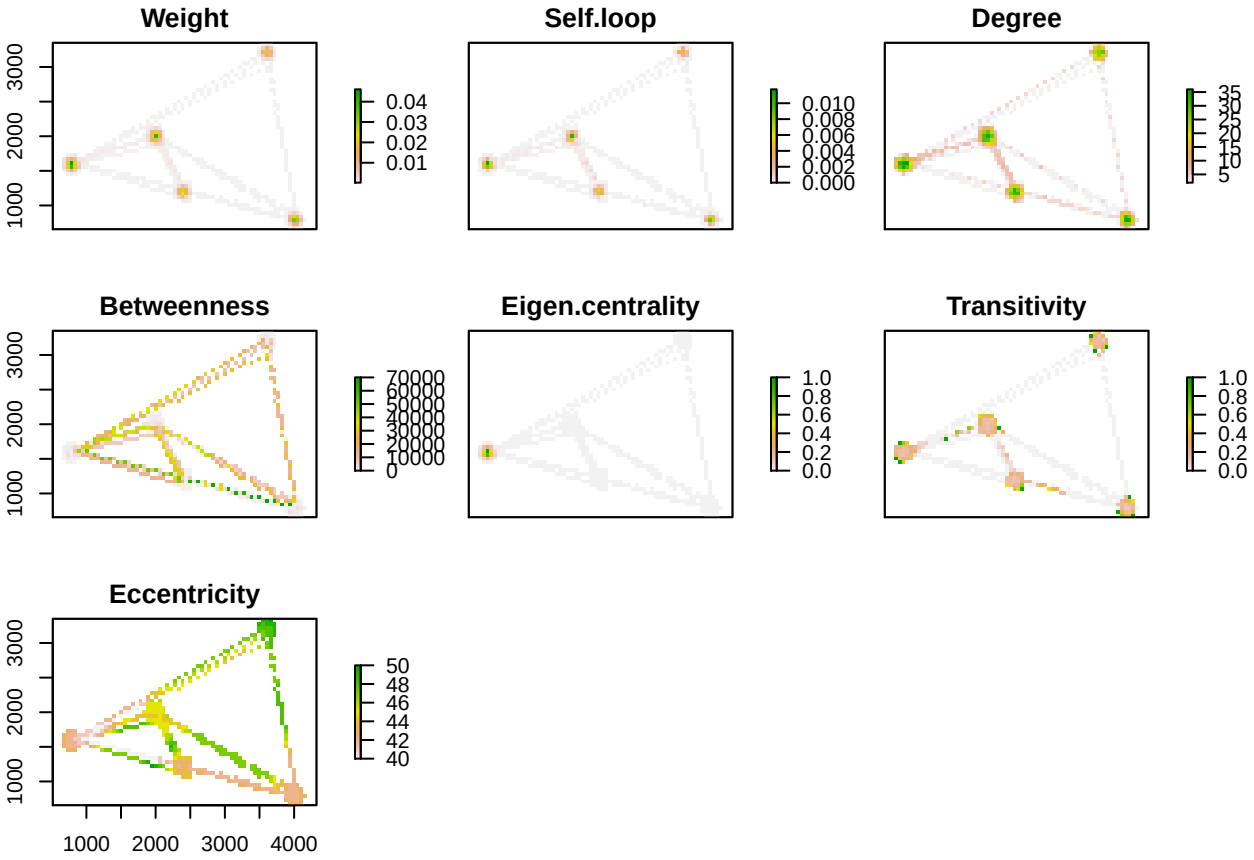

```
# Using user defined grid
ras<-raster(nrows=10, ncols=10, xmn=0, ymn=0, xmx=6000, ymx=6000)
adj_patches2<-traj2adj(patches, res=quant(patches, p=0.5), grid=ras) #Grid size based on median
plot(adj_patches2[[2]]) #Crop version of the grid created
```



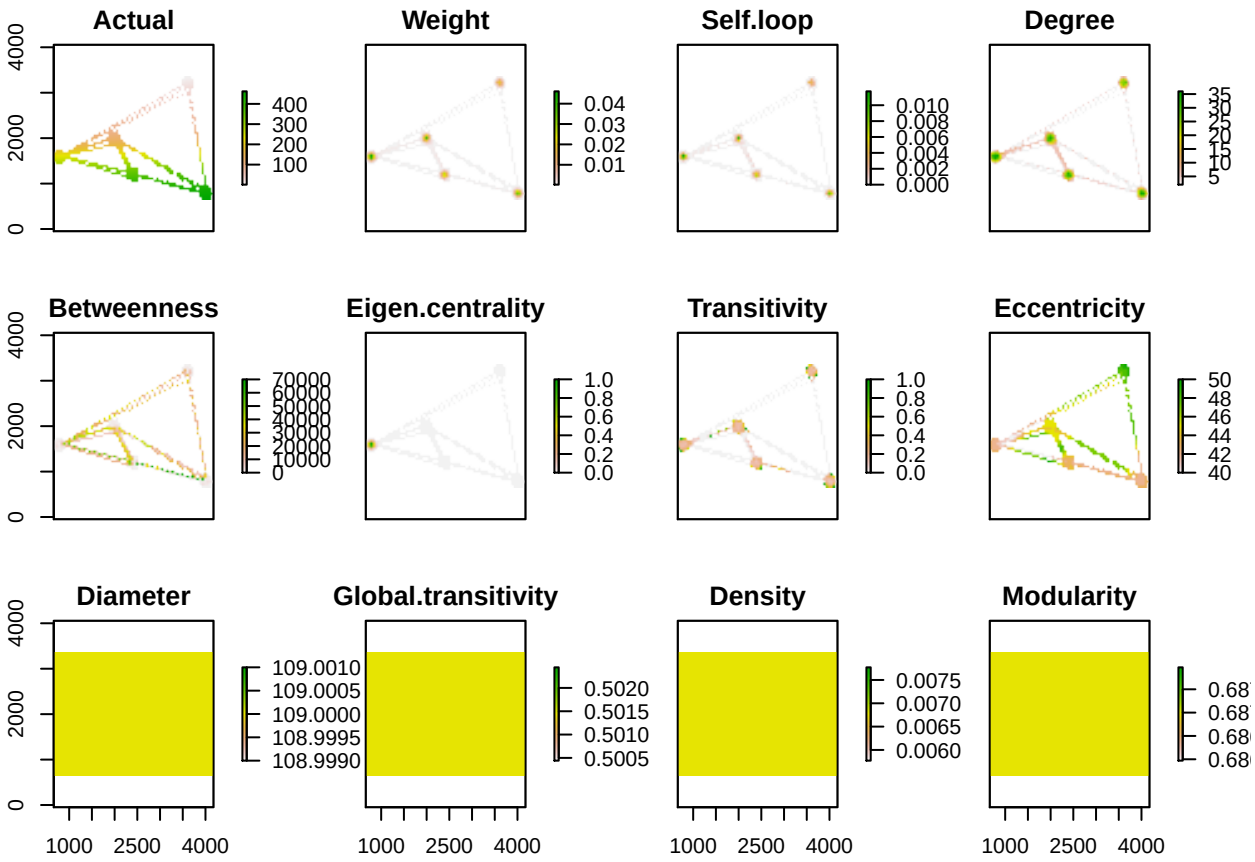
Calculation of network metrics - *adj2stack*

The function *adj2stack* takes the output of function *traj2adj* and calculates a series of node- and graph-level metrics. Each metric is stored as a individual raster and the output is a raster stack combining each metric. Graph-level metrics are also stored as a raster, each containing an unique value. The function *graphmet* extracts graph-level metrics. The function *val* extracts only the occupied cells (remove NA) in a raster and allows the calculation of statistics from node-level metrics.

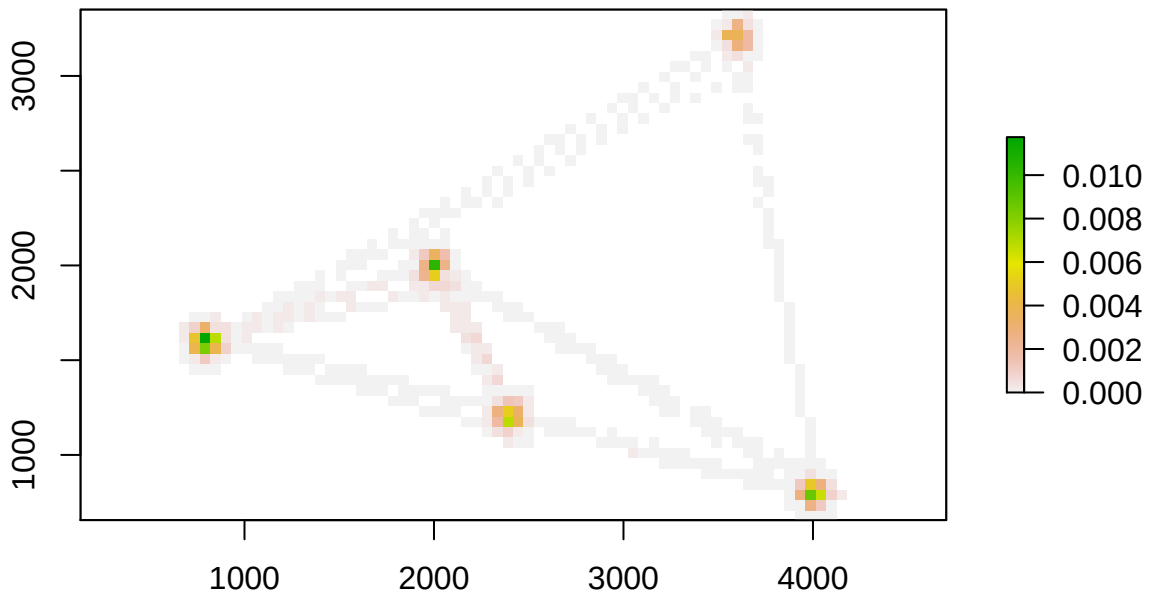
```
# Using multi-patches movement and median distance travelled
stck<-adj2stack(adj_patches,grph=T) #Plot the node-level metrics at the same time
```




```
plot(stck) #Plot also the graph-level metrics (not really useful)
```




```
plot(stck[[3]]) #Plot only one metric (degree)
```




```
graphmet(stck) # Extract graph-level metrics
```

```
##            Diameter Global.transitivity             Density
##        1.090000e+02        5.014409e-01        6.770525e-03
##          Modularity
##        6.869705e-01
```

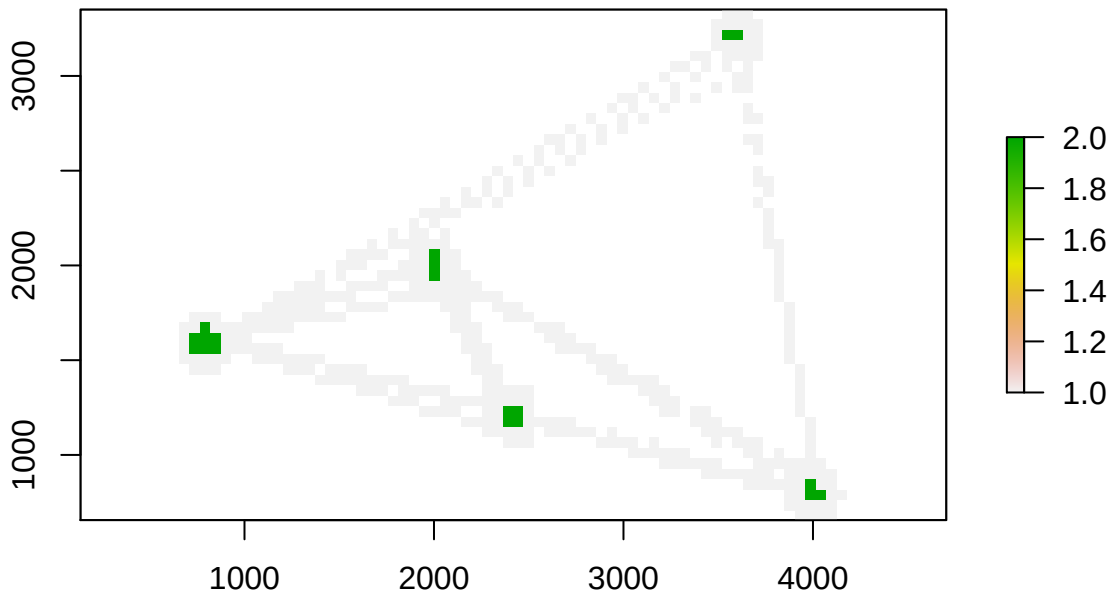
```
cv(val(stck, 4)) #Extract coefficient of variation of node-level betweenness.
```

```
## [1] 120.985
```

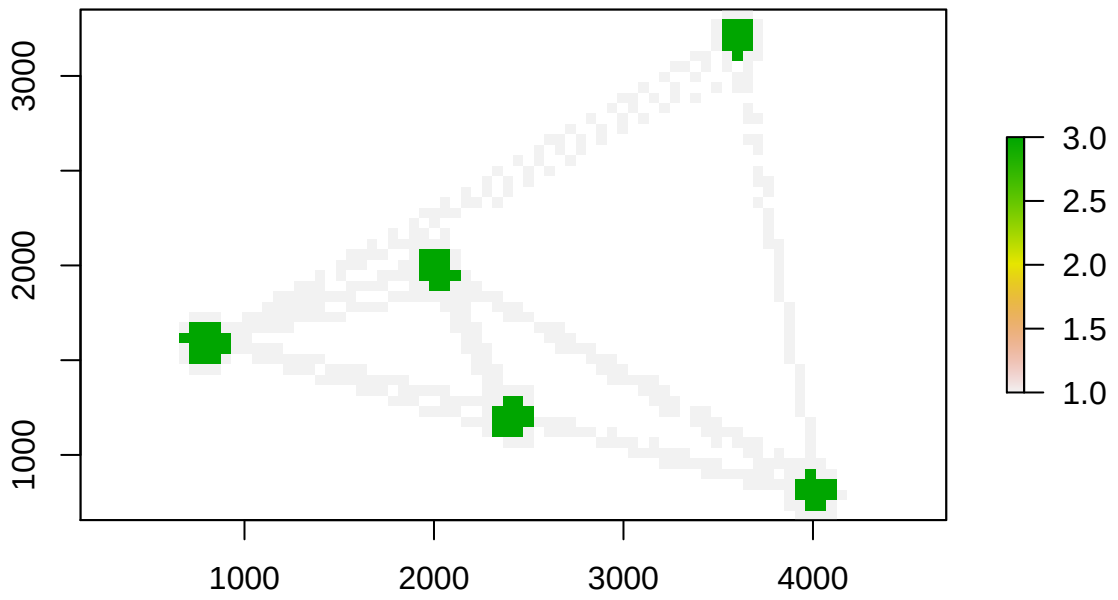
Clustering of node level metrics - *clustnet*

The function *clustnet* applies a normal mixture model to node-level metrics in order to cluster them into separate groups (default = 2). The function takes the output of function *adj2stack* with the user specifying the metric to cluster and the number of groups. Return a list containing output of function *Mclust* from package *mclust* and a raster displaying classification.

```
# Using multi-patches movement and median distance travelled
clust2<-clustnet(stck, id=3, nclust=2) # Clustering of degree in two groups
```



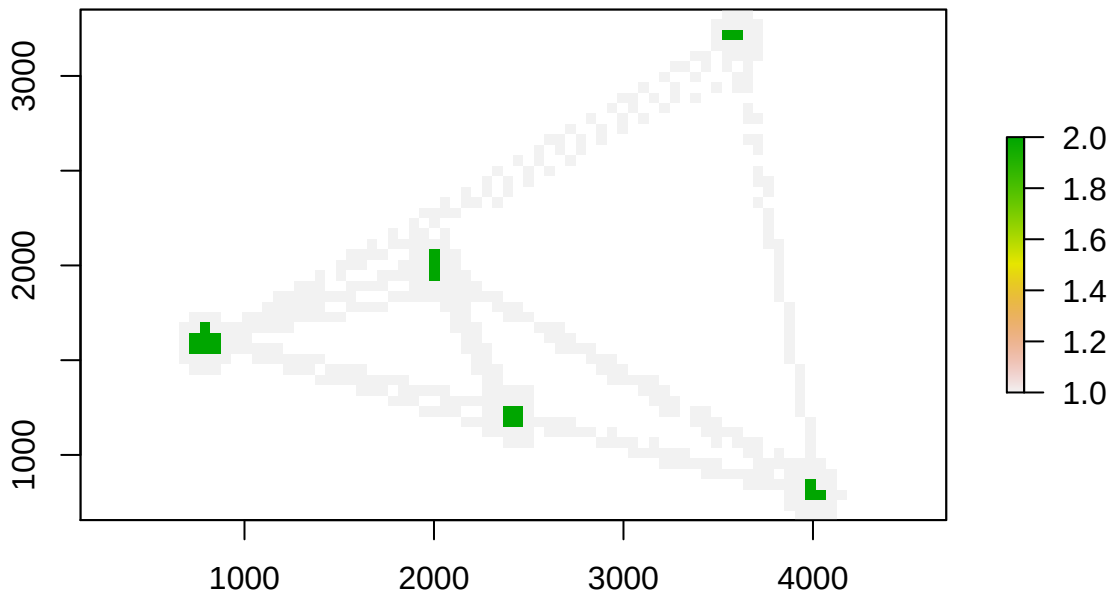

```
clust3<-clustnet(stck, id=4, nclust=3) #Clustering of betweenness in three groups
```




```
summary(clust2[[1]])
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust E (univariate, equal variance) model with 2 components:
##
##  log.likelihood   n df      BIC    ICL
##        2649.259 462  4 5273.976 5272.6
##
## Clustering table:
##   1   2
## 443  19
```

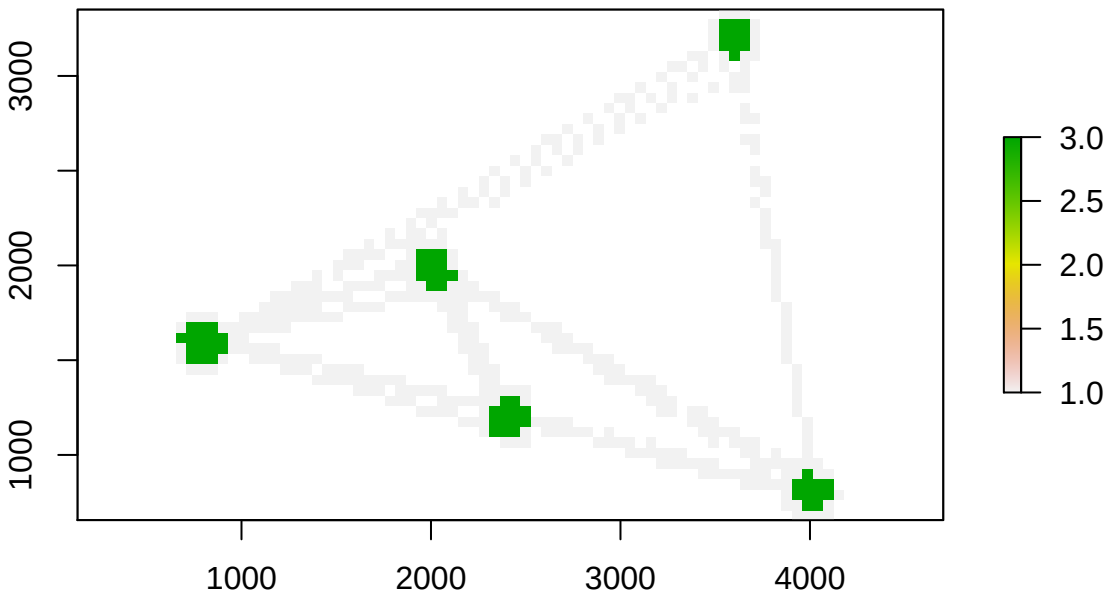
```
plot(clust2[[2]])
```




```
summary(clust3[[1]])
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust E (univariate, equal variance) model with 3 components:
##
##  log.likelihood   n df       BIC       ICL
##       -1374.905 462  6 -2786.624 -3099.297
##
## Clustering table:
##   1   2   3
## 401   0  61
```

```
plot(clust3[[2]])
```


3000
2000
1000
1000
2000
3000
4000
3.0
2.5
2.0
1.5
1.0