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*.

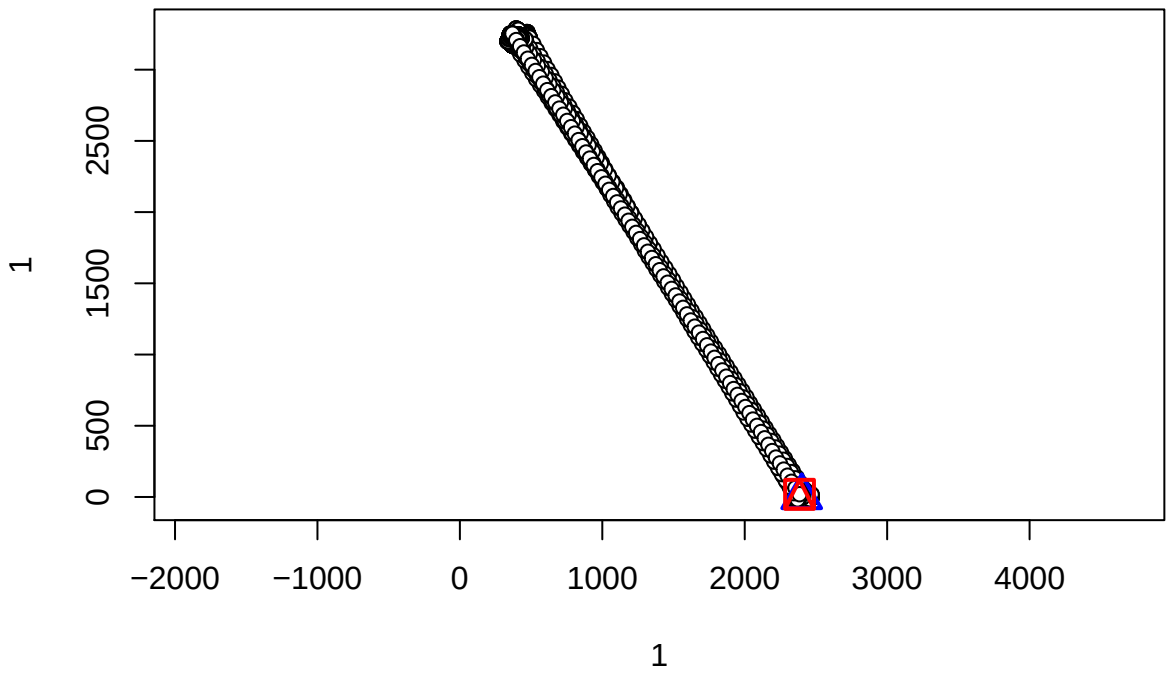
```
# Simulating migration with two-states model
mig<-sim_mov(type="2states", npatches=2, ratio=2, nswitch=25, ncore=150, grph=F)
mig
```

```
##
## ************ 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     4350   0 1960-01-01 00:00:01 1960-01-01 01:12:30
##
##
##  infolocs provided. The following variables are available:
## [1] "out.Corri"
```

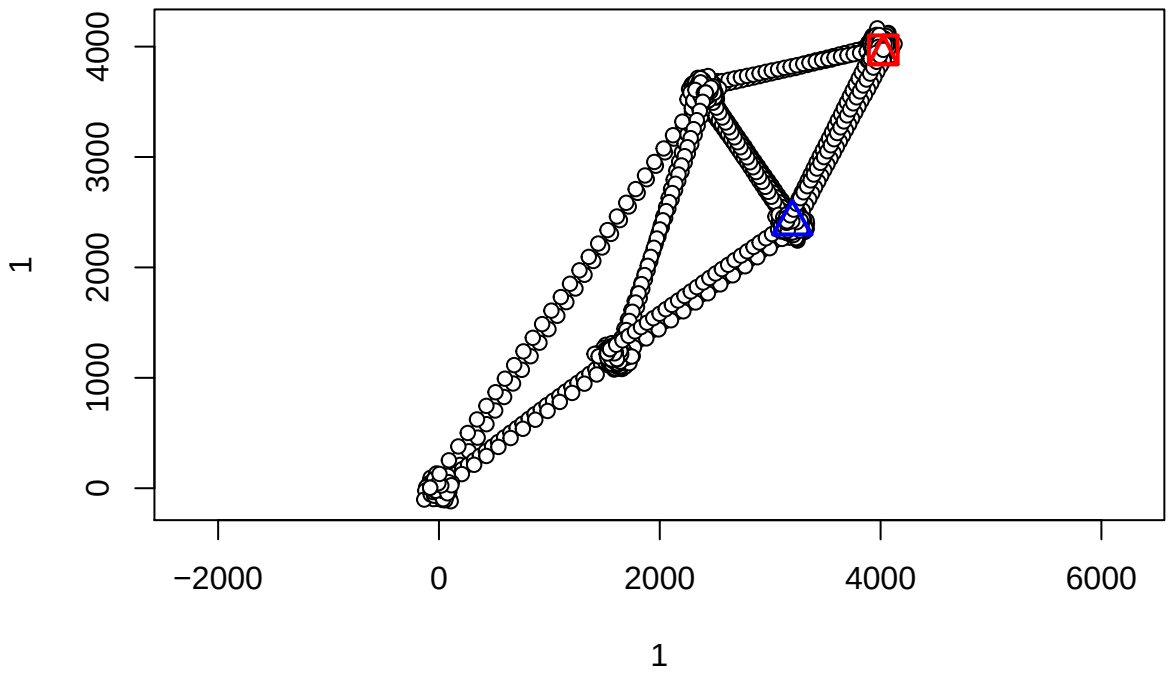
```
head(ld(mig))
```

```
##          x            y                date            dx            dy
## 1 2400.000  0.000000000 1960-01-01 00:00:01  2.652413e-06 -1.340245e-03
## 2 2400.000 -0.001340245 1960-01-01 00:00:02 -2.714569e+00 -8.845423e+00
## 3 2397.285 -8.846763193 1960-01-01 00:00:03 -2.099967e-03 -3.146057e-02
## 4 2397.283 -8.878223765 1960-01-01 00:00:04  1.124943e-04  5.153662e-05
## 5 2397.283 -8.878172228 1960-01-01 00:00:05  1.342770e+01  8.853713e+00
## 6 2410.711 -0.024459302 1960-01-01 00:00:06 -2.486410e-01 -1.546810e-01
##           dist dt          R2n  abs.angle  rel.angle id burst out.Corri
## 1 1.340248e-03  1 0.000000e+00 -1.5688173         NA id    id         2
## 2 9.252588e+00  1 1.796264e-06 -1.8685619 -0.2997446 id    id         2
## 3 3.153058e-02  1 8.563409e+01 -1.6374466  0.2311152 id    id         2
## 4 1.237376e-04  1 8.620313e+01  0.4295913  2.0670380 id    id         2
## 5 1.608389e+01  1 8.620161e+01  0.5829283  0.1533370 id    id         2
## 6 2.928286e-01  1 1.147293e+02 -2.5850774  3.1151796 id    id         2
```

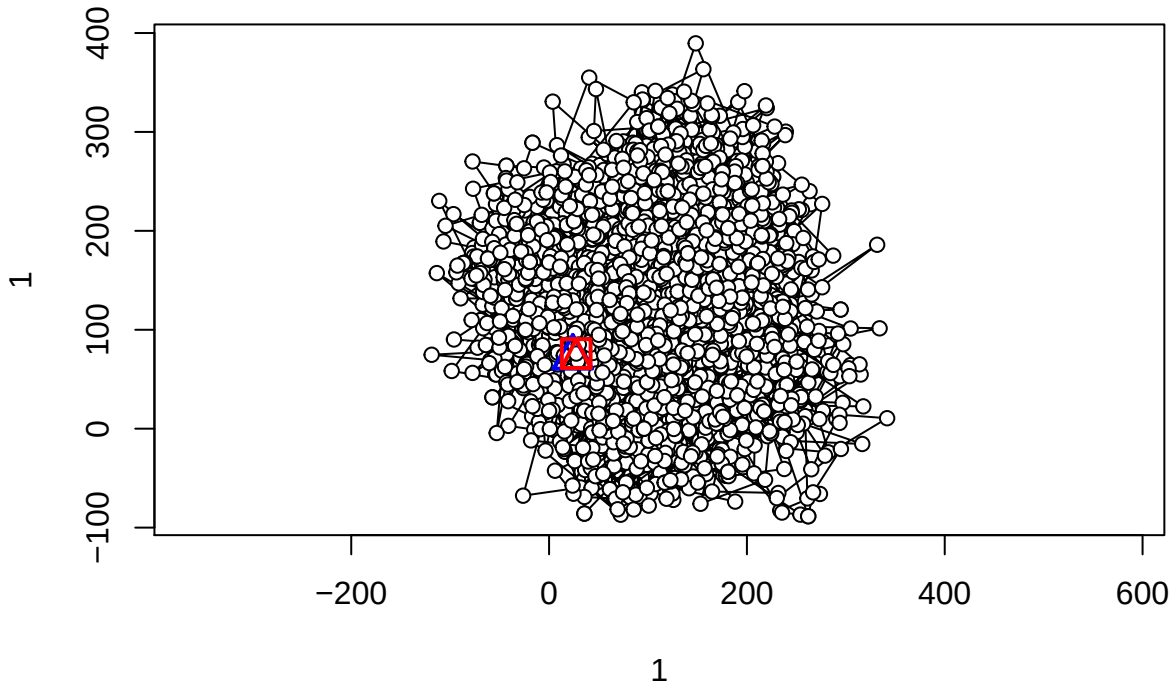
```
plot(mig)
```




```
# 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)
```



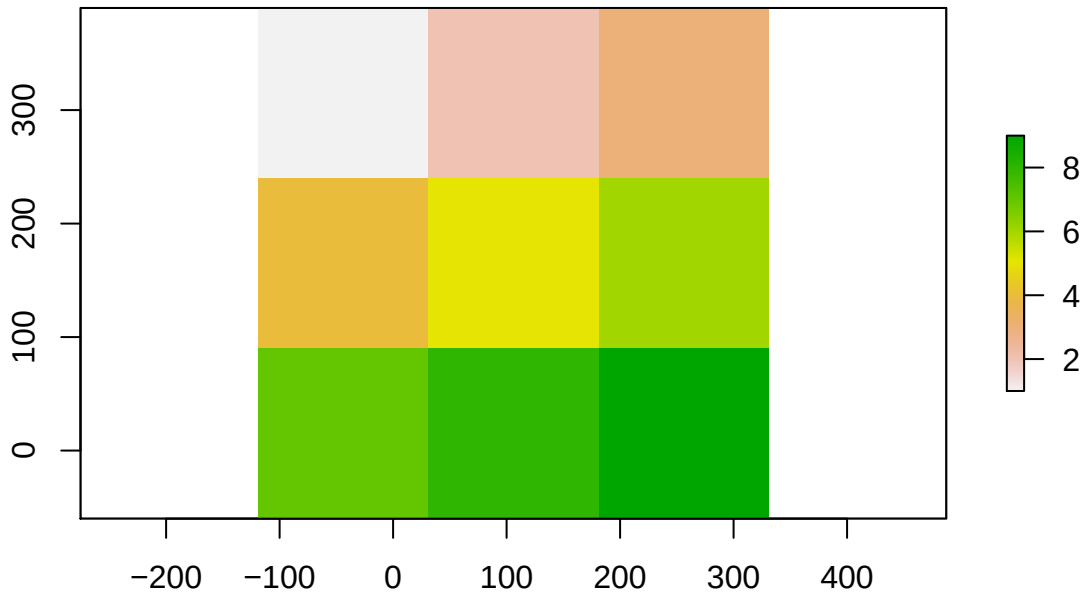
Converting movement to adjacency matrix - *traj2adj*

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,]    8    5    0    9    2    0    0    0    0
##  [2,]    1  163   16    2  112   22    1    0    0
##  [3,]    0   20   17    0   19   13    0    0    0
##  [4,]   11    6    0  349  115    1   33   18    0
##  [5,]    4  102   22  109  980  128   31   92    6
##  [6,]    0   21   14    0  121  203    0    7   16
##  [7,]    0    0    0   44   18    0   98   43    0
##  [8,]    0    0    0   20  102    3   39  388   25
##  [9,]    0    0    0    0    5   13    0   31  112
```

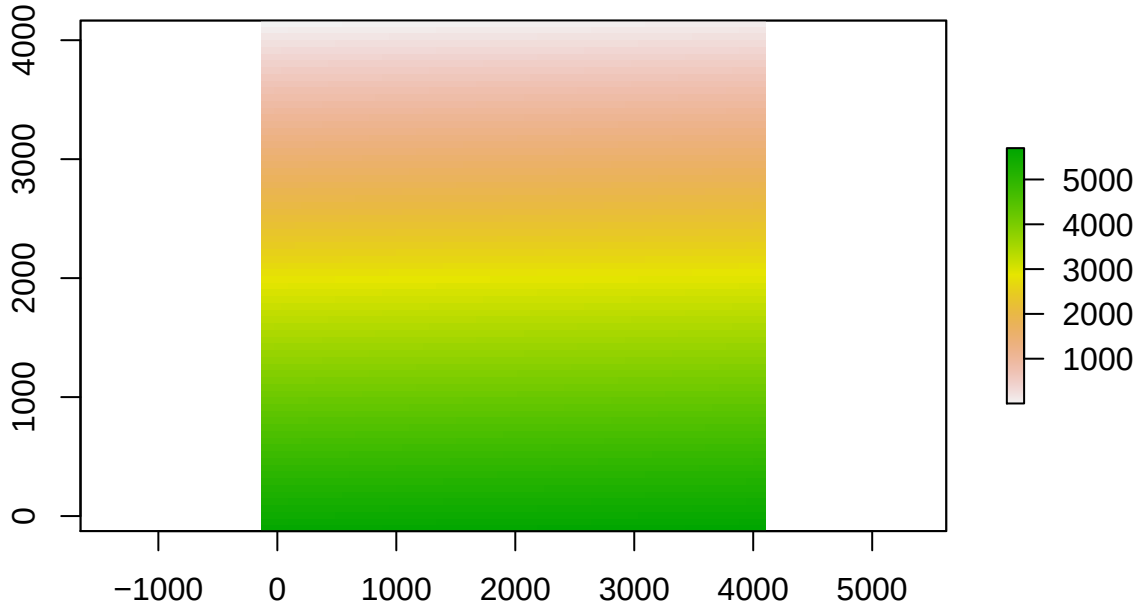
```r
plot(adj_seden[[2]]) #Plot grid used
```



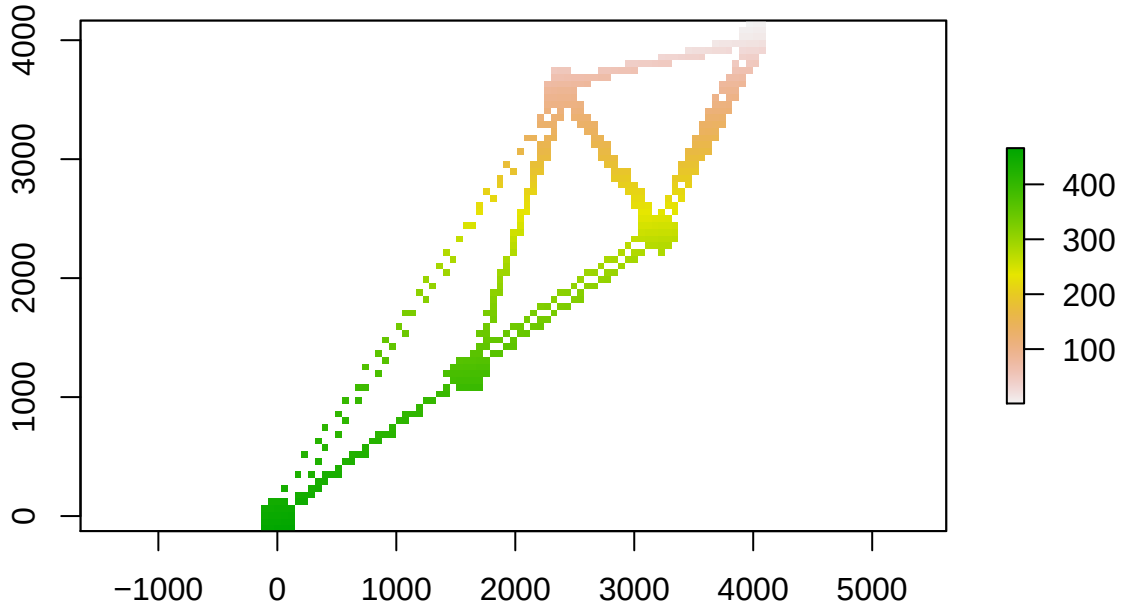

```r
# Using multi-patches movement and median distance travelled
adj_patches<-traj2adj(patches, res=quant(patches, p=0.5)) #Grid size based on median
dim(adj_patches[[1]]) # Size of the adjacency matrix
```

```
## [1] 466 466
```

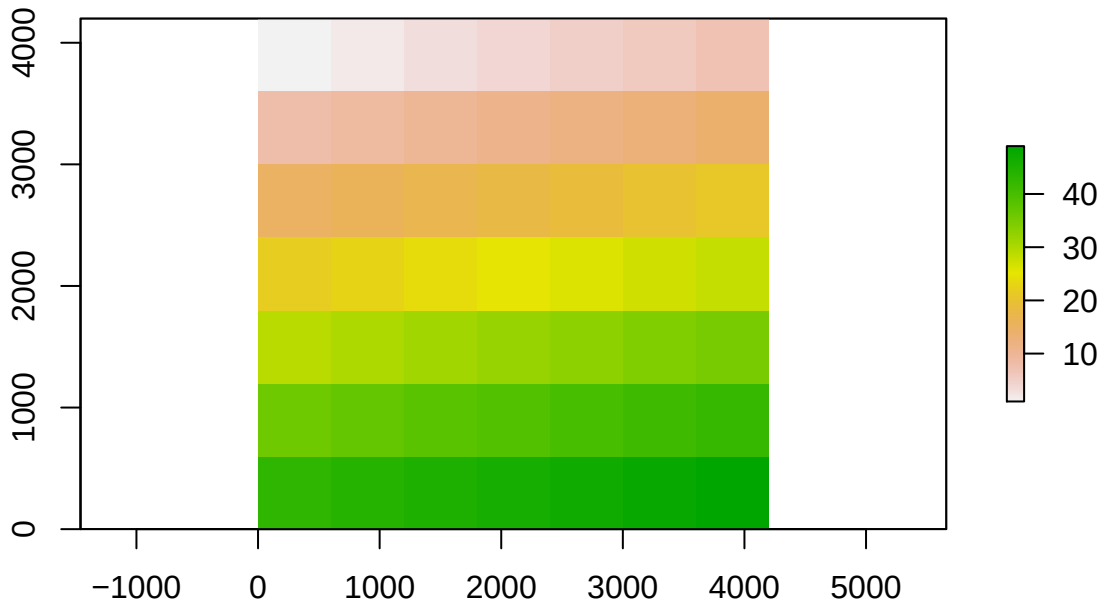
```r
plot(adj_patches[[2]]) #Plot grid used
```




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



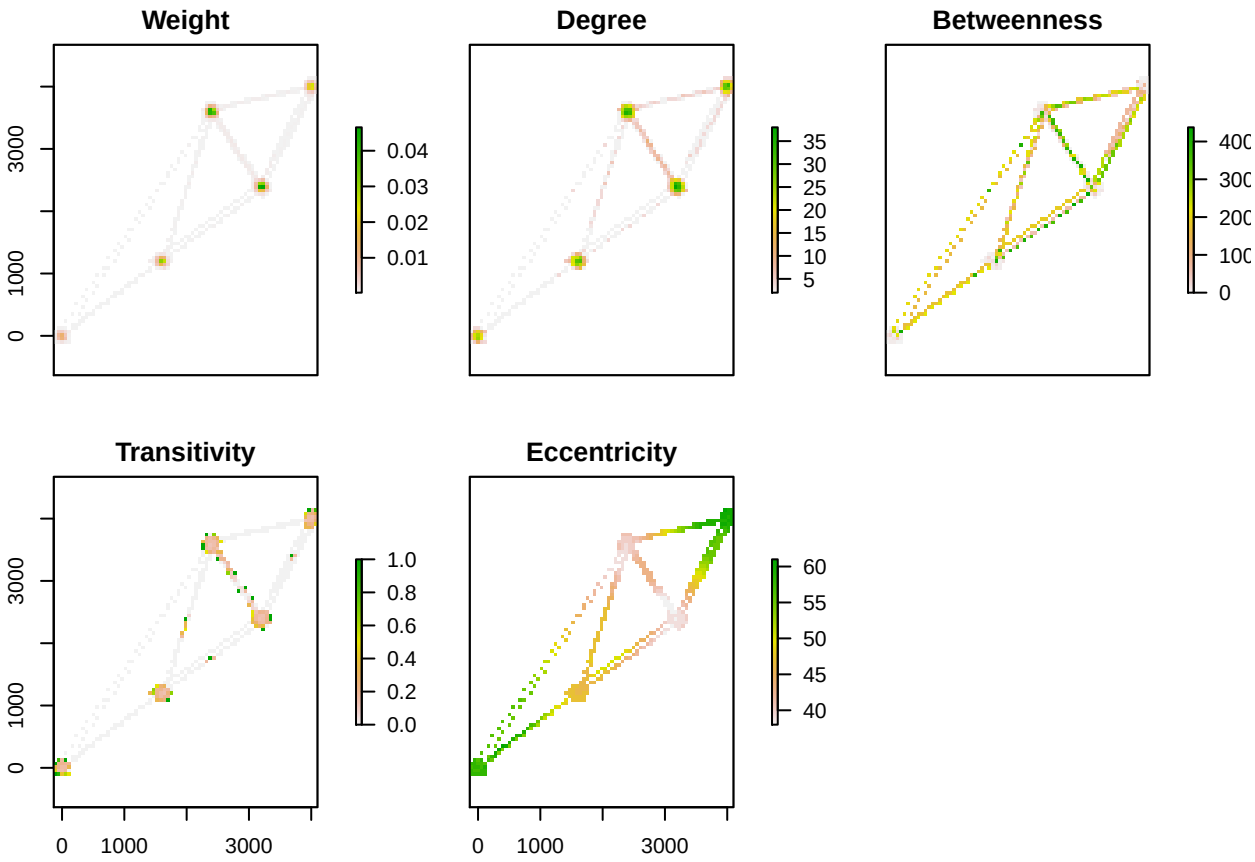

```
# Using user defined grid
ras<-raster(nrows=10, ncols=10, xmn=0, ymn=0, xmx=6000, ymx=6000)
adj_patches2<-traj2adj(patches, res=quant(patches, p=0.5), grid=ras) #Grid size based on median
plot(adj_patches2[[2]]) #Crop version of the grid created
```



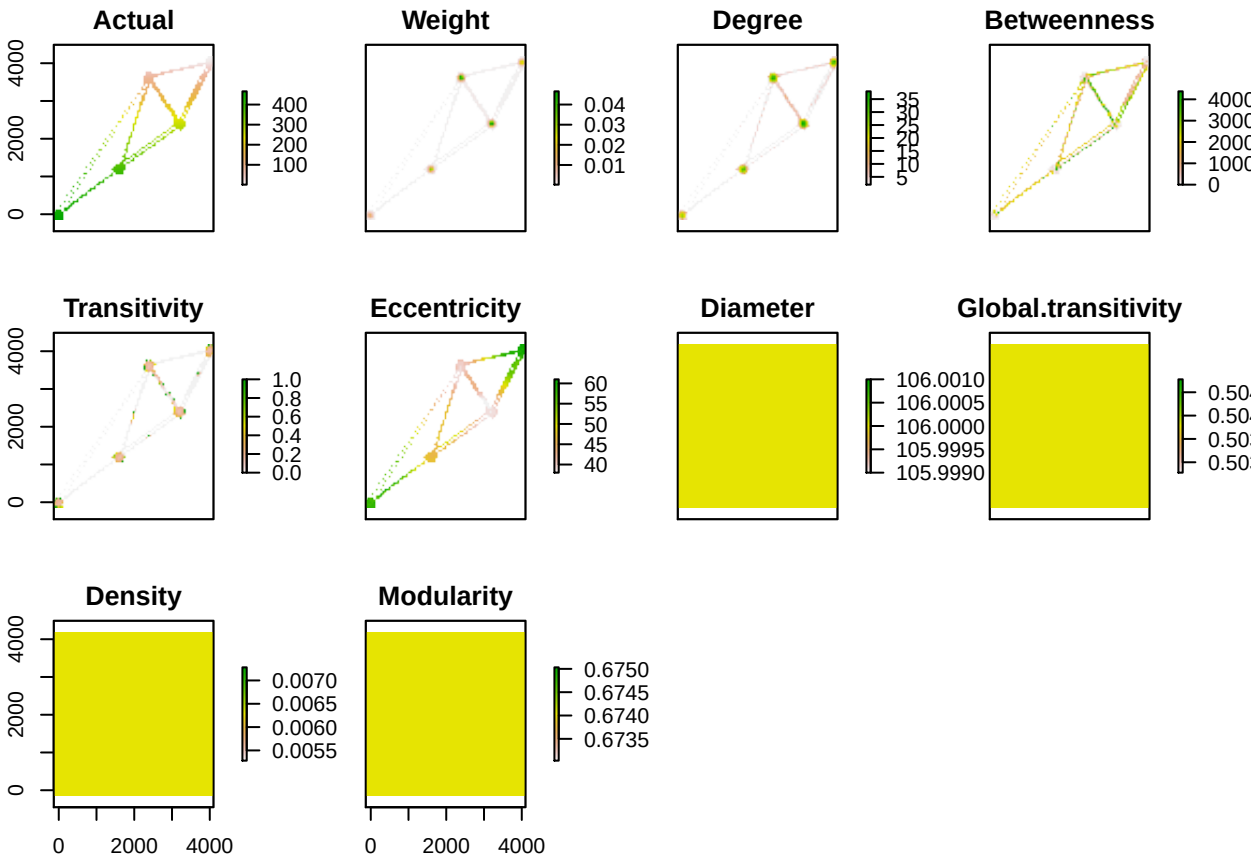
Calculation of network metrics - *adj2stack*

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.

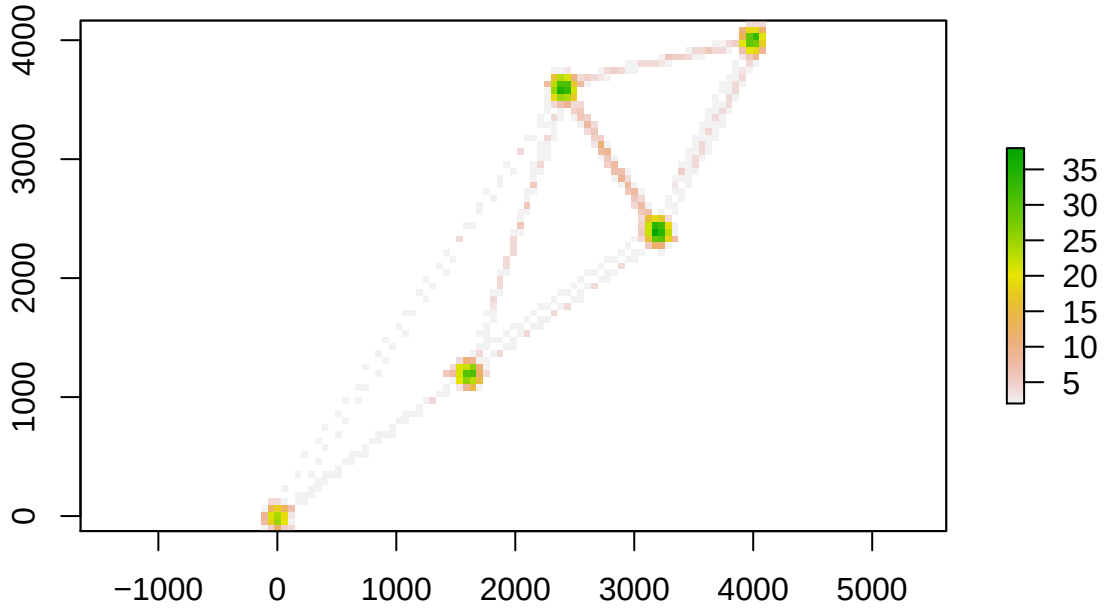
```
# Using multi-patches movement and median distance travelled
stck<-adj2stack(adj_patches,grph=T) #Plot the node-level metrics at the same time
```




```
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
##        1.060000e+02        5.037771e-01        6.280862e-03
##          Modularity
##        6.740327e-01
```

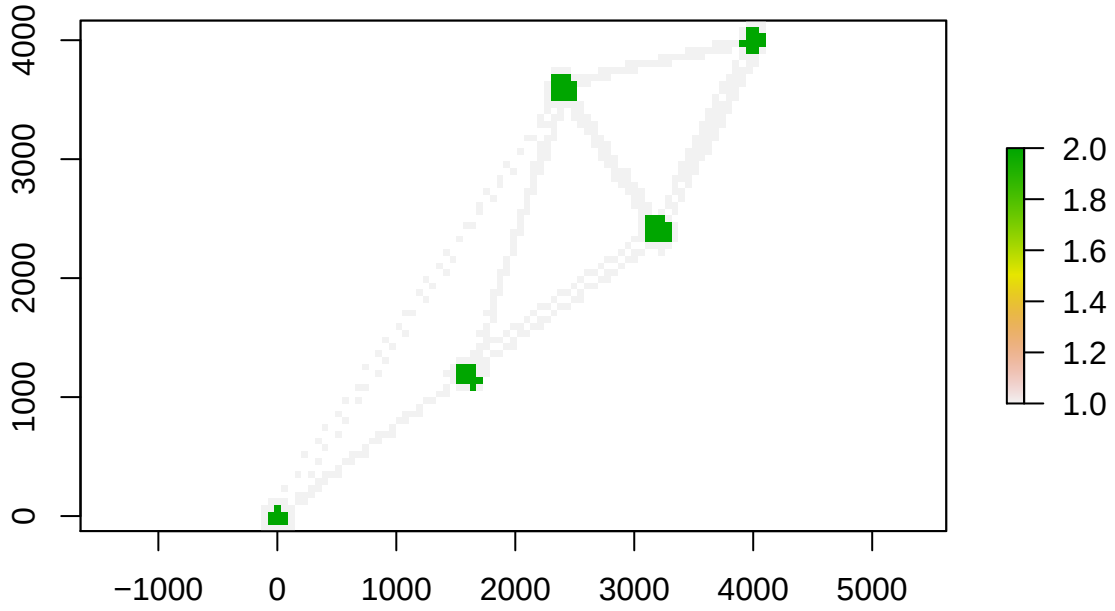
```
cv(val(stck, 4)) #Extract coefficient of variation of node-level betweenness.
```

```
## [1] 75.13711
```

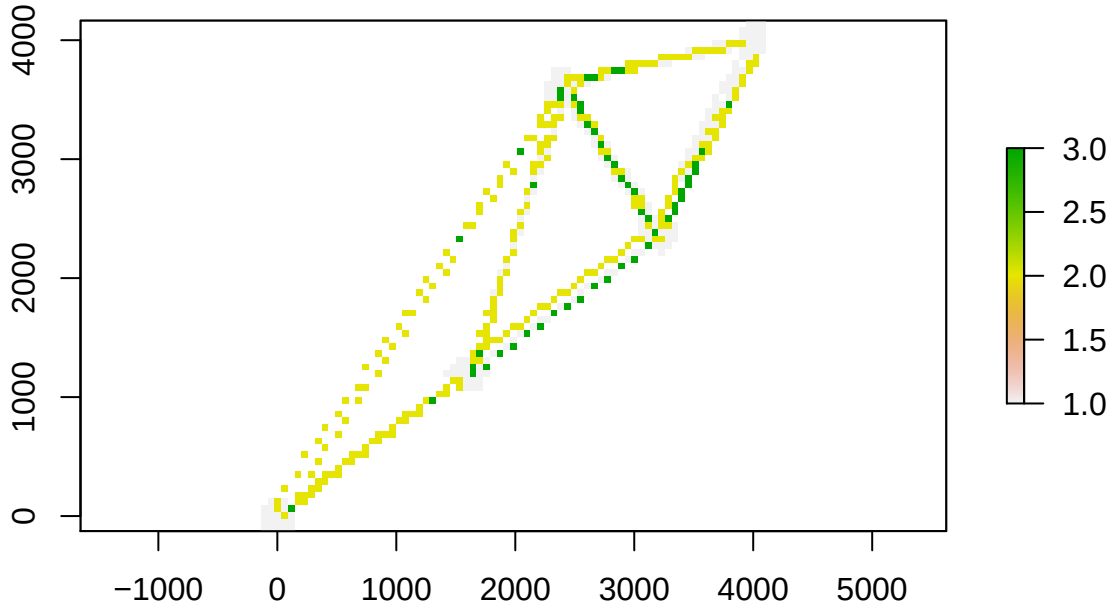
Clustering of node level metrics - *clustnet*

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
```



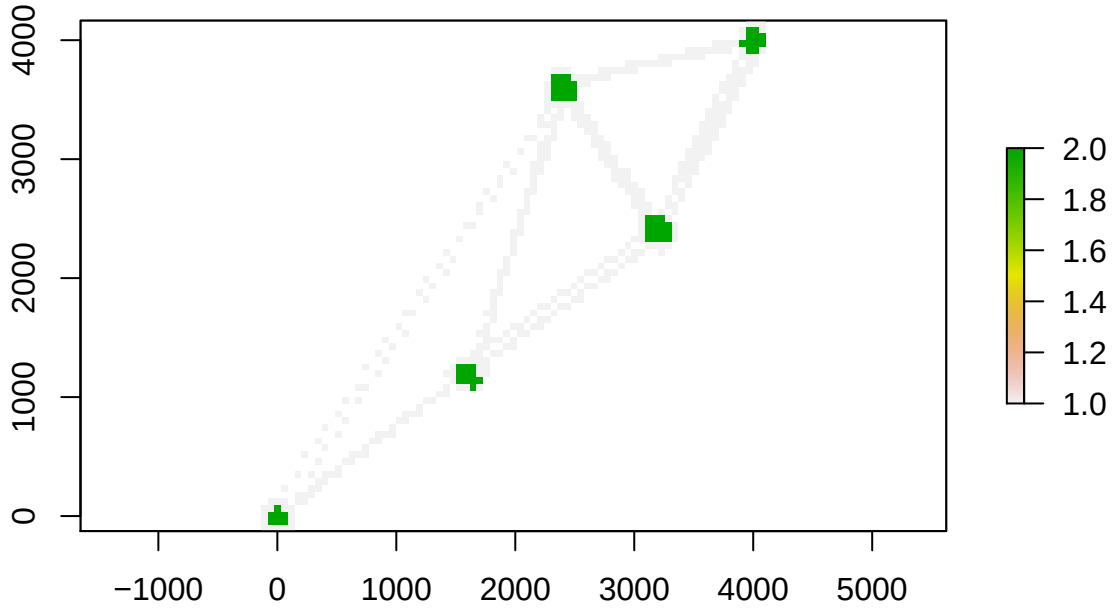

```
clust3<-clustnet(stck, id=4, nclust=3) #Clustering of betweenness in three groups
```




```
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
##       -1347.681 466  4 -2719.938 -2722.035
##
## Clustering table:
##   1   2
## 407  59
```

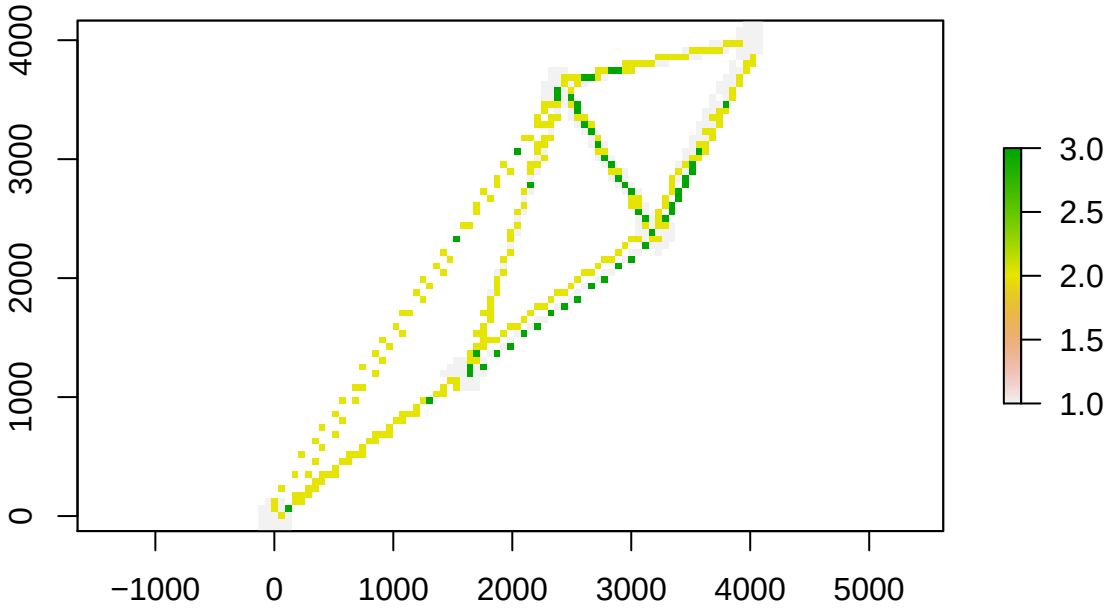
```
plot(clust2[[2]])
```




```
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
##       -4924.952 466  6 -9886.768 -9914.549
##
## Clustering table:
##   1   2   3
## 183 231  52
```

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


4000
3000
2000
1000
0
−1000
0
1000
2000
3000
4000
5000
3.0
2.5
2.0
1.5
1.0