Segmentation function

Felipe Barros (f.barros@iis-rio.org)

8 de março de 2016

A segmentação de imagens visa agrupar pixels com características similares, como por exemplo, nível de cinza, textura e contraste, resultando em áreas com unidades homogêneas. Logo, a segmentação baseia-se em análise estatística espacial por crescimento de regiões, onde tem-se como parâmetros a quantidade mínima de pixel (para criar a região) e um limiar de similaridade (*threshold* usado apra definir quão similar determinado pixel é em relação aos pixels vizinhos). Assim, pode-se dizer que a segmentação diferencia áreas homogêneas com padrões diferentes aos seus vizinhos, criando k regiões diferentes na imagem. porém tal abordagem não classifica as regioes criadas quanto ao pertencimento a um mesmo grupo ou não. Para mais informações sobre segmentação, ver manual *spring*, íten 7.7, página 127.

A abordagem aqui apresentada, **não está baseada no método de crescimento por região**. Mas cria n grupos com áreas homogêneas, a partir dos valores dos pixels da(s) imagem(ns). Portanto, não consideram elementos como, contraste e textura. Em resumo, esta abordagem classifica da imagem em n grupos (definidos pelo usário), que serão classificados estatísticamente através da análise de cluster (K-means) e posteriormente *extrapolado* (usado para classificação) para toda a imagem em questão.

```
source('segmentation.R')
args(segmentation)
```

```
## function (envLayer = envLayer, studyArea = studyArea, projName = "MT",
##     randomforest = TRUE, random.pt = NULL, Kmeans = TRUE, ngroup = NULL,
##     polygonize = FALSE, seed = 123)
## NULL
```

Function arguments:

1. **envLayer**: raster object;
2. **studyArea**: sptail polygon;
3. **projName**: a sufix to be used on output names;
4. **randomForest**: TRUE/FALSE - if the segmentation must be run with randomForest or not;
5. **random.pt**: number of random points to be generated **if** using randomForest classification. IF **NULL**, the function will generate **0.01*ncell(envLayer)** (1% of envLayer's cells) random points.
6. **Kmeans**: TRUE/FALSE - if the segmentation must be run with Kmeans analysis or not;
7. **ngroup**: number of classes to be identifyed. If **NULL**, and running randomForest algorithm, the function will plot the Kmeans variance analysis and ask the number of groups wanted. Thus the decision can be done based on the plot.
8. **polygonize**: TRUE/FALSE - whether or not to the segmentation must be saved **also** in *shapefile* format;

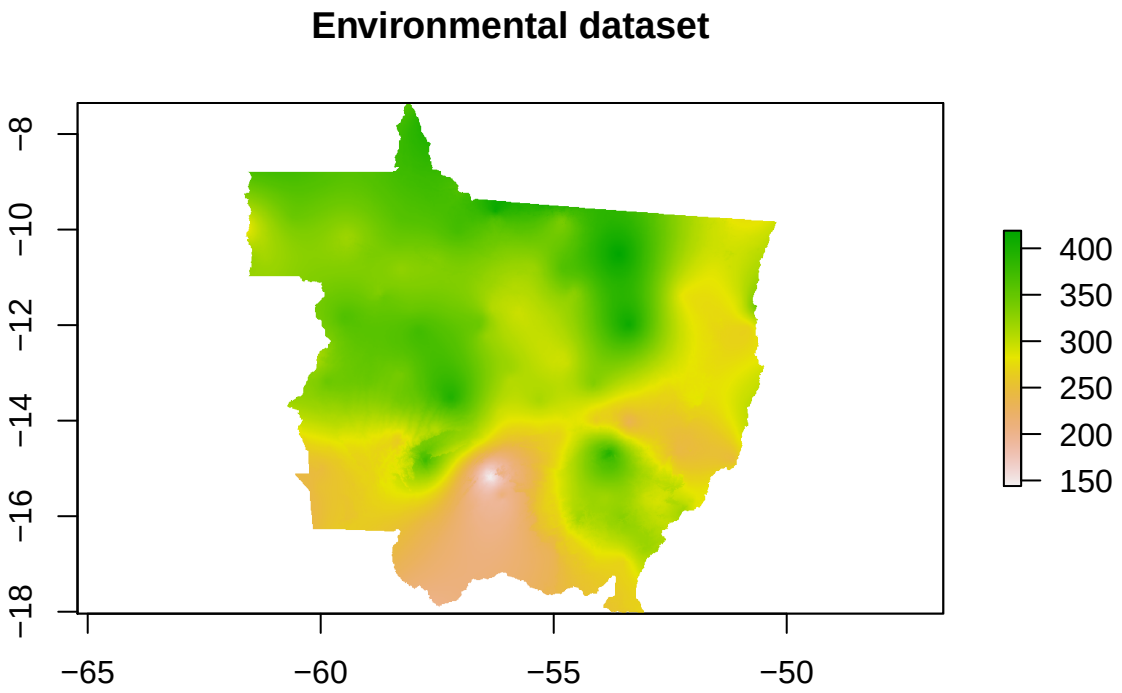
```
# loading libraries
library(rgdal)
library(raster)
library(rgeos)
library(RColorBrewer)
```

Input data:

Raster Layer

A raster Environmental data (or any other spatial variables with continous values). Can be a single raster layer or raster stack.

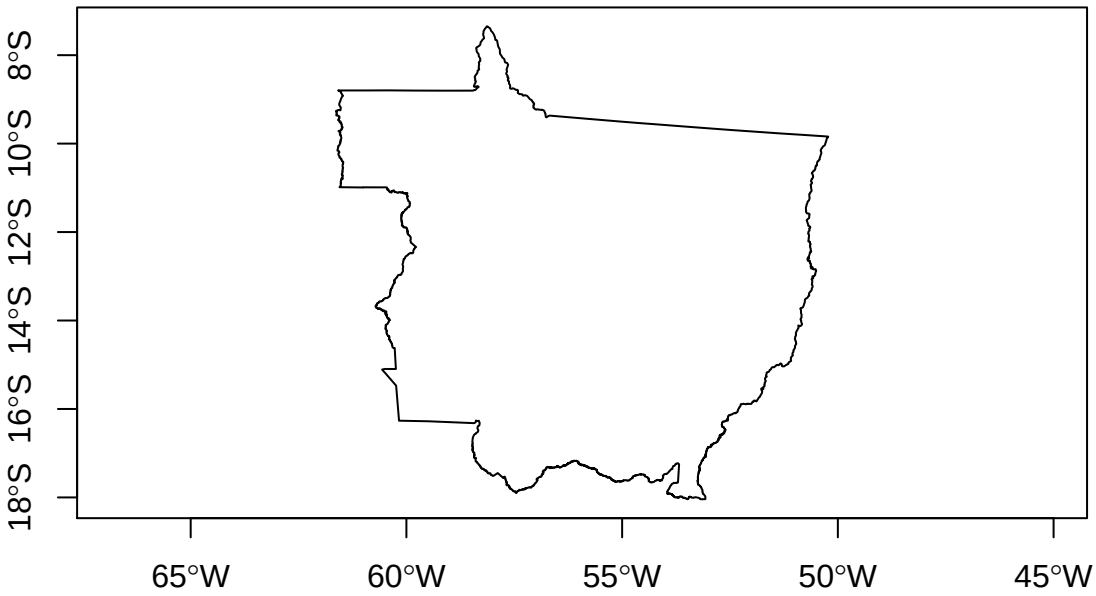
```
# Data set:
plot(MT_prec[[1]], main='Environmental dataset')
```



Vector layer

A study area in polygon format.

```
plot(MT, main='Study area', axes=TRUE)
```



Segmentation algorithms:

The segmentation function is implemented to run with two algorithms: 1. The function can be run interely with Kmeans cluster analysis; **or/and** 1. with randomForest classification after Kmeans group identification.

In any case, the Kmeans is used to identify k classes (clusters) from the data provided. For more information about Kmeans.

Running the segmentation with randomForest

```
segmentation(envLayer = MT_prec, #raster Layer or raster stack
  studyArea = MT, # SpatialPolygonsDataFrame
  randomforest = TRUE,
  projName = 'MT_yr_prec',
  random.pt = NULL, # Number of random points to be genarated to run randomForest
  Kmeans = FALSE,
  ngroup = 6, # Number of classes to be classified
  polygonize = FALSE,
  seed=123)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.

## Generating 17564 random points

## Saving 6.5 x 4.5 in image
```

```
##
  |
  |                                                                      |   0%
  |
  |================                                                      |  25%
  |
  |================================                                      |  50%
  |
  |================================================                      |  75%
  |
  |======================================================================| 100%
##
## Saving raster result
## randomForest segmentation done.
```

By randomForest, a random points is generated in the study area from which the values will be analysed with Kmeans analysis and the k groups (**clusters**) will be created. After that, the pixel values and its group assotiated with be used as a train set to run randomForest classification.

Running the segmentation with Kmeans

The Kmeans cluster analysis and randomForest classification will use all raster layers in the analysis

```
segmentation(envLayer = MT_prec, #raster Layer or raster stack
  studyArea = MT, # SpatialPolygonsDataFrame
  randomforest = FALSE,
  projName = 'MT_yr_prec',
  random.pt = NULL, # Number of random points to be genarated to run randomForest
  Kmeans = TRUE,
  ngroup = 6, # Number of classes to be classified
  polygonize = FALSE,
  seed=123)
```

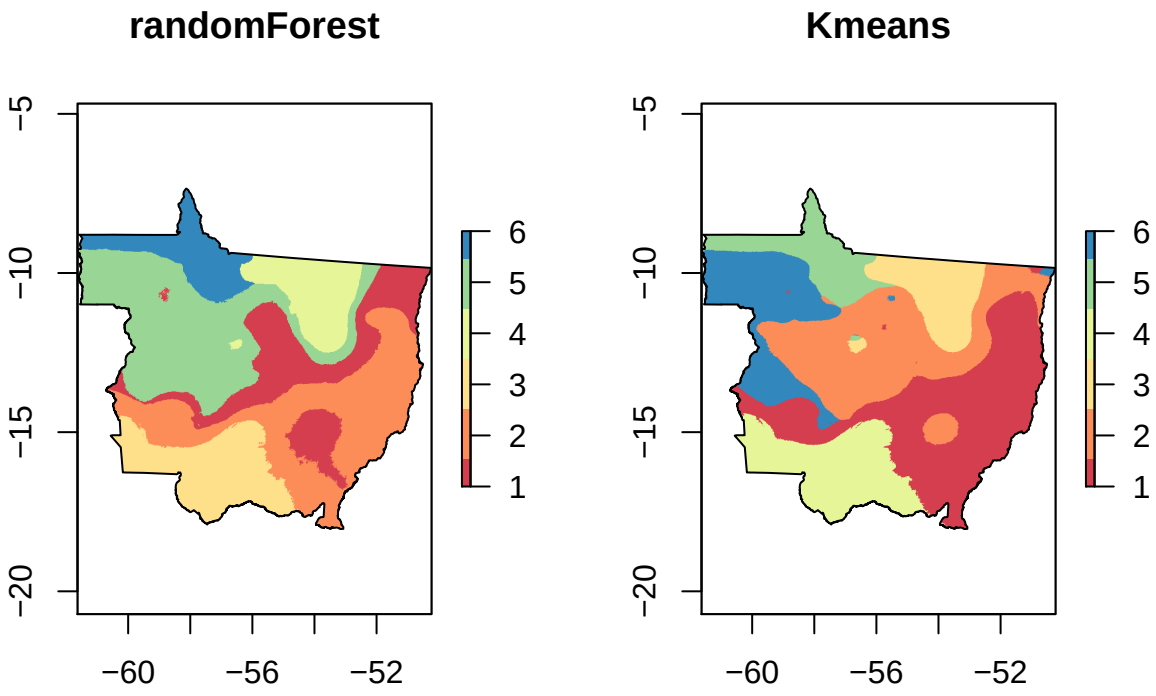
```
## Generating 17564 random points

## Saving 6.5 x 4.5 in image

## Raster data with NA values.
## Saving raster result
## Kmeans segmentation done.
```

When running the segmentations interely with Kmeans, **all the pixels values from all raster layers are used to run the Kmeans cluster analysis**.

Comparing results from both methods

```
rf_result <- raster('./rf_segmentation_MT_yr_prec.tif')
km_result <- raster('./km_segmentation_MT_yr_prec.tif')
```

```
pallete <- brewer.pal(6, 'Spectral')
par(mfrow=c(1,2))
plot(rf_result, col = pallete, main = 'randomForest')
plot(MT, add=TRUE)
plot(km_result, col = pallete,main = 'Kmeans')
plot(MT, add=TRUE)
```



Running the segmentation with both algorithms

```
segmentation(envLayer = MT_prec, #raster Layer or raster stack
  studyArea = MT, # SpatialPolygonsDataFrame
  randomforest = TRUE,
  projName = 'MT_yr_prec',
  random.pt = NULL, # Number of random points to be genarated to run randomForest
  Kmeans = TRUE,
  ngroup = 6, # Number of classes to be classified
  polygonize = FALSE,
  seed=123)
```