Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Modelling Prehistorical Iconographic Compositions. The R package decorr

Thomas Huet
UMR 5140

Craig Alexander

Jose Pozo

Abstract

By definition, Prehistorical societies are characterised by the absence of a writing system. During, the largest part of human history – from far – symbolic expressions belong to illiterate societies which express themself with rock-art paintings, pottery decorations, figurines and statuary, etc., and a lot of now disappeared carved woods, textil design, etc. At the composition level, recognition of meaningful associations of signs and reccurent patterns indicate clearly the existence of social conventions in the way to display and to read these expressions. We present the **decorr** R package which grounds concepts, methods and tools to analyse any ancient graphical systems. Our asumption is that i) any graphical system is a spatial distribution of features, and ii) these features have possibly any meaningful relationships one with another depending on their pairwise spatial proximities. To model the graphical compositions we employ concepts coming from the Graph Theory. To ensure the feasability of this type of analysis, we propose a GIS-based method for inputs and a serie of functions for data management.

Keywords: Iconography, Prehistory, Graph Theory, Graph Drawing, Spatial Analysis, R.

1. Introduction

Symbolic practices is a characteristic trait of human societies. Even discussed, such practices seems to start between 233,000 to 800,00 BC (d'Errico and Nowell 2000), covering more than 97% of total human societies time span. Symbolism cover a large range of practices, from ochre deposit in a tomb, to menhir alignements, passing through wall fresco. This latter, what might be called "iconographical practices", probably shows the most complex and interesting testimonies of past societies symbolism. For decades, its study was linked to history of religion because commonly seen as closely linked to cultual practices and believes. Since the *New Archaeology* developpement during the 60's (Clarke 2014), symbolic expressions start to be studied with the same formal methods (statistics, seriations, distribution maps, etc.) as any another aspect of social organisation: settlement patterns, tools *chaine opératoire*, subsistence strategies, etc. (Renfrew and Bahn 1991). But unlike many aspects of the material culture where technological requirements and technical efficiency determine the choice of the raw material and the of the object – a flint blade for cutting, a pottery for containing, a house for living –, the function of an iconographic composition cannot be drawn directly from itself. Whether these last decades study of ancient iconography had undergone significative improvements at the site scale – with GIS/database statistics, paleoenvironmental restitutions, etc. – and at the sign scale with the development of archaeological sciences – radiocarbon dating, use-wear analysis, elemental analysis, etc. –, these improvement do not necessarly help to understand the semantic content of the iconography.

Semantics or semiotics can be defined as a system of conventional features – called signs – organised also in conventional manners. Until our days, formal methods to study ancient iconography semantics have been mostlty been grounded – explicitly or not – on the prime principle of Saussurian linguistic: the '*linearity of the signifier*' (De Saussure 1989). Writing is one of the most rational semiographical system with a clear distinction between signified and signifier and the development of the signified on a linear axis. Even if we do not understand the meaning of the signs, writing can easily be modelled with Graph theory and recurent patterns can be identified. For example, the 3-letters word `"art"` can be modelled with three vertices (`a`, `r`, `t`) and two edges (one between `a` and `r`, the other between `r` and `t`). In R, these features, concatenated in this order with a `paste0()`, is `art`, and not `rat`.

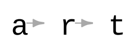

Figure 1: concatenate of graphical units (GUs) is art

But, as stated, in Prehistorical the writing system does not exists. Spatial relationships between graphical features, or graphical units, are not necessarly linear and directed but could most probably be more multi-directional and undirected: the direction of the interactions of pairwise GUs can be in any order.

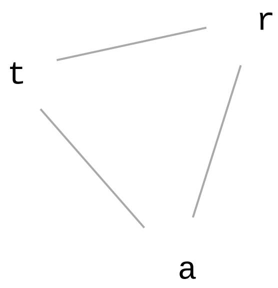

Figure 2: Potential spatial relations between GUs.

Because of the inherent variability of iconography, and because graphical and spatial proximities between GUs are generaly not quantified, applying the Saussurian model to any prehistorical graphical content had led to considerable problems:

- unexplicit groupings of graphical units – like graphical units grouped into figures, figures grouped into patterns, patterns grouped into motives, etc. – with tedious number of groups
- consistency, proximities and relationships between these groups are often implicit and not quantified
- studies develop proper descriptive vocabularies, singular relationships of grouped, idosyncratic methods in a site-dependend or period-dependend scales

These issues limit drastically the possibility to conduct cross-cultural comparisons and to draw a synthesis of humankind's symbolism at a large scale and over the long-term.

In this article we present the R package **decorr**. Its purpose is to formalise a method based on relative neighborhood graphs to analyse any graphical content. As any formal system, iconography can be modelled as spatial features related one with the other depending on rules of spatial proximities, as state by the First Law of Geography: "*everything is related to everything else, but near things are more related than distant things*" (Tobler 1970). The principal idea of our model is that any graphical system can be represented by features connected (or not) to each other depending on their spatial proximity. To map and analyse these proximities, the model uses concepts and methods coming from Graph Theory. This package has been grounded on the seminal work of Alexander (2008) and its first IT implementation by Huet (2018).

2. Graph theory Model

Graph theory offers a conceptual framework and indices (global at the entire graph scale, local at the vertex scale) to deal with notions of networks, relationships and neighbourhoods. Graphical units (GUs) can be modelled as vertices (nodes) and their spatial relations can be modelled as edges. The different levels of GUs spatial organisation can be retrieve by a relative neighborhood graphs analysis (Graph Theory) and a spatial (GIS) analysis.

Nodes and edges – repectively GUs and connexions between GUs – are created on a GIS interface. Indeed, for large series of graphical decorations, GIS offers the most suitable and flexible interface to register all GUs and to get their coordinates. These x and y coordinates, measured in pixels, are relative to the decoration figure which is open in the first place in a new GIS project without any projection system. The decoration image is considerated as the basemap of the project and will cover the region of interest of the analysis. The decoration image can be binarized: GUs are considerated active, the undecorated parts of the support – the background – are considerated inactive. The entire decoration image is tiled into different GUs' area of influence like a Voronoi diagram of the support where the Voronoi seeds are the GUs. Exist a link between a pair of GUs when their area of influence share a common border. A RNG – also called a geometric graph, or a planar graph – is constructed from GUs (nodes) and their proximity links (edges). The graph itself is as a subgraph of a Delaunay triangulation.

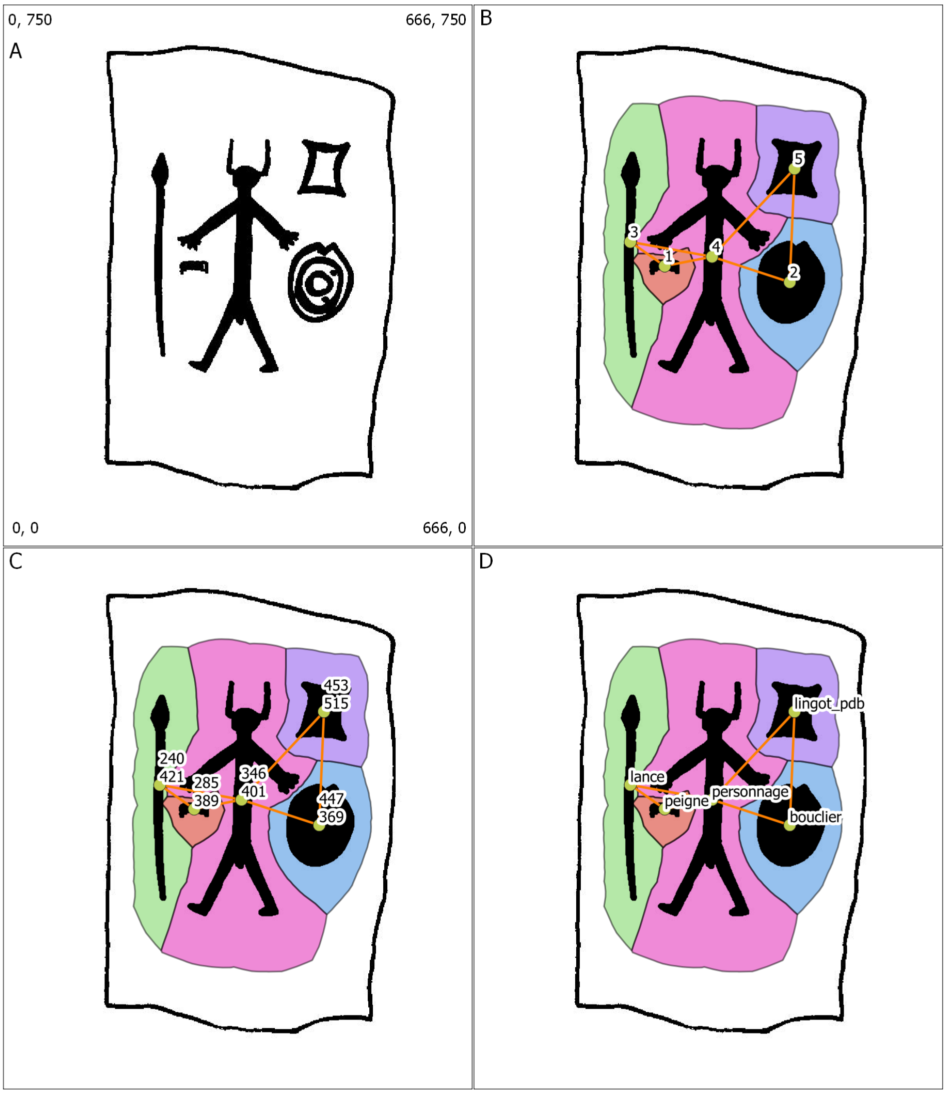

Figure 3: GIS interface. A) Original decoration of the Late Bronze Age Cerro Muriano 1 stele (drawing: Díaz-Guardamino Uribe (2010)) with its extent (`xmin, xmax, ymin, ymax`); B) After the polygonisation of the GUs, including the border of the stelae, the Voronoi cells, the centroid of GUs and the links between GUs having adjacent cells (ie, sharing a border) are calculated; C) For each GUs, x and y are calculated; D) At least one variable, like the `type` of the GUs is defined in order to compute composition analysis. A simplier solution will be to create directly centroids (`POINTS`) on the GUs and to draw the edges manually

This model has a minimal of *a priori* definitions. Those definitions only concern the intrinsic properties of GUs (type, technology, color, orientation, size, etc.) and the types of relations they share. Here, we will only consider one property for the nodes, its type (column `type`), and the most common types of relations between GUs: *normal*, *attribute* and *overlapping* edges.

- *normal* edges

By convention, two different GUs having a Voronoi cell sharing a border, have a common edge tagged `'='` and represented with a plain line. The textual notation of such an edge is `'-=-'`. For example: `1 -=- 4` means that the nodes `1` and `4` have a common border.

- *attribute* edges

It occurs frequently that a GU can be divided into a *main unit* (eg, a character) and one or various *attribute units* (eg, a helmet, male sex). Broadly speaking, for further statistical analysis, it is better to use this *attribute* method than to multiply the categories of GUs. To record this information, a new type of edge, tagged with `'+'`, is be introduced. This type of edges is directed and, by convention, displayed with a dashed line. Its starts from the *main unit* and ends with the *attribute units*. The textual notation of such an edge is `'-+-'`. For example `4 -+- 6` means that the main node `4` has the attribute node `6`.

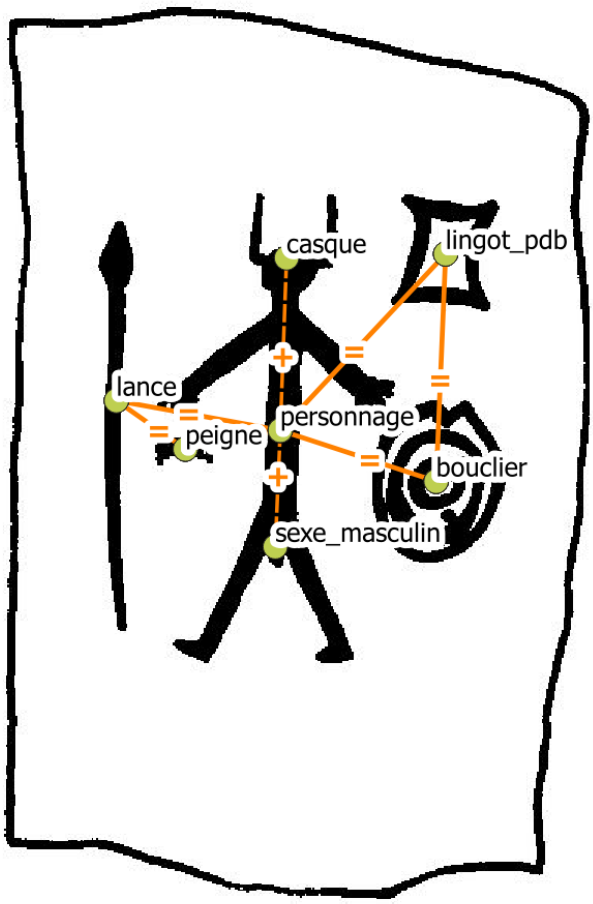

Figure 4: GIS interface. The GUs `casque` (helmet) and `sexe_masculin` (male sex) are two nodes attributes of the GU `personnage` (character).

- *overlapping* edges

Finally, it is quite common that a graphical composition shows superimpositions between different UGs permit to distinguish different decoration phases for a single support. So, at first, the analyse must be performed on each different phases of decoration separately. This stratigraphical information (`A` *over* `B`, or `B` *under* `A`) helps to understand the relative chronology between GUs and must be recorded. A simple way to achieve this is to introduce the new tag `'>'` for the for the type of edge. This type of edges is directed. The textual notation of such an edge is `'->-'`. For example `A ->- B` means that `A` crosses `B` (ie, `A` overlaps `B` in the stratigraphical sense).

3. The R package decorr

The **decorr** package can be downloaded from GitHub

```
R>   devtools::install_github("zoometh/iconr", build_vignettes=TRUE)
```

3.1. External package

The **decorr** package imports the following packages:

- **magick** for image manipulation (Ooms 2018)
- **igraph** for graph and network analysis (Csardi and Nepusz 2006)
- **rgdal** to read *shapefiles* of nodes and edges (Bivand, Keitt, and Rowlingson 2019)
- **grDevices** for colors and font plotting, **graphics** for graphics, **utils** and **methods** for formally defined methods and *varia* methods (all combinations, etc.) (R Core Team 2019)

3.2. Data

A training dataset (nodes and edges coordinates, decoration images) is stored in the `extdata` folder of the **decorr** package

- The `imgs` dataframe

The dataframe storing the inventory of decorations is `imgs`. The field `imgs$idf` is the short name of the decoration, useful during statistical analysis. The primary key of each decoration is the concatenate of `imgs$site` and `imgs$decor`. These keys will allow joints with the other dataframes (`nodes` and `edges`)

```
## Warning:  package 'magick' was built under R version 3.6.3
## Linking to ImageMagick 6.9.9.14
## Enabled features:  cairo, freetype, fftw, ghostscript, lcms, pango, rsvg, webp
## Disabled features:  fontconfig, x11
```

idf	site	decor	img
1	Cerro Muriano	Cerro Muriano 1	Cerro_Muriano.Cerro_Muriano_1.jpg
2	Torrejon Rubio	Torrejon Rubio 1	Torrejon_Rubio.Torrejon_Rubio_1.jpg
3	Brozas	Brozas	Brozas.Brozas.jpg
4	Zarza de Montanchez	Zarza De Montanchez	Zarza_de_Montanchez.Zarza_De_Montanchez.jpg
5	Ibahernando	Ibahernando	Ibahernando.Ibahernando.jpg

Table 1: The studied corpus, the `imgs.tsv` dataframe

The **decorr** package training dataset is composed by four stelaes decorations drawings (Díaz-Guardamino Uribe 2010) belonging to the so-called 'Warrior stelae' family – with about 140 stelae – dated to the Late Bronze Age of SW Iberian peninsula (Pérez 2001). At first the drawings dataset can be checked by using the `imgs` dataframe and the **magick** package.

```
R> # library("magick")
R> pth <- system.file("extdata", package = "decorr")
R> imgs <- read.table(system.file("extdata", "imgs.tsv", package = "decorr"),
+                     sep="\t", stringsAsFactors = FALSE)
R> lims <- list()
R> for(i in 1:nrow(imgs)){
+   i1 <- image_read(paste0(pth,"\\",imgs[i,"img"]))
+   lbl.txt <- paste0(imgs[i,"idf"],"\n",
+                     imgs[i,"site"],"\n",
+                     imgs[i,"decor"],"\n",
+                     imgs[i,"img"],"\n",
+                     image_info(i1)$width,"*",image_info(i1)$height," px")
+   i1 <- image_annotate(i1,lbl.txt,location = "northwest",
+                        size = 25, color = "red")
+   lims[[length(lims)+1]]<- i1
+ }
R> out.img <- image_append(c(image_append(c(lims[[1]],lims[[2]])),
+                           image_append(c(lims[[3]],lims[[4]]))),
+                         stack = TRUE)
R> plot(out.img)
```

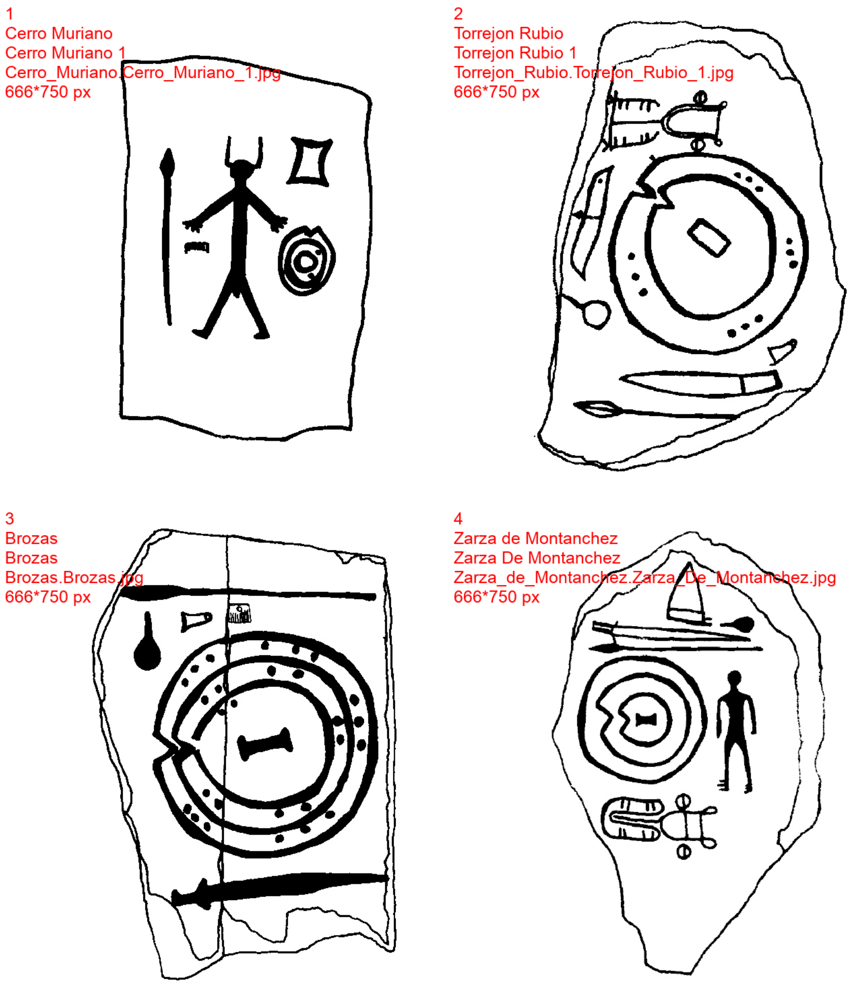

Figure 5: Decoration images of the training dataset

To construct a graph overlapping the decoration images listed in the `images` dataframe, the first step is to load `nodes` and `edges` dataframes.

```
R> nodes <- read.table(system.file("extdata", "nodes.csv", package = "decorr"),
+                       sep=";",stringsAsFactors = FALSE)
R> edges <- read.table(system.file("extdata", "edges.csv", package = "decorr"),
+                       sep=";",stringsAsFactors = FALSE)
```

- The `nodes` dataframe

It contains the required minimum variables for the analysis.

```
R> caption <- "Nodes (from \\code{nodes.csv} dataframe)}"
R> print(xtable::xtable(head(nodes),
+                        caption = caption),
+       table.placement="H")
```

	V1
1	site decor id type x y
2	1 Cerro Muriano Cerro Muriano 1 1 personnage 349.814824824793 -298.324408661478
3	2 Cerro Muriano Cerro Muriano 1 2 casque 349.814824824793 -243.98505280916
4	3 Cerro Muriano Cerro Muriano 1 3 lance 238.463685783159 -298.324408661478
5	4 Cerro Muriano Cerro Muriano 1 4 bouclier 446.022208956765 -381.169656108454
6	5 Cerro Muriano Cerro Muriano 1 5 peigne 283.004141399813 -358.008619187794

Table 2: Nodes (from `nodes.csv` dataframe)

The primary key of the decoration is based on two fields: `nodes$site` and `nodes$decor`. The site is the current unit of analysis in Prehistory and Archaeology, but since a site can have various decorated objects, a primary key on two fields is necessary. The `nodes$id` is the identifier of the node. The `nodes$type` field is the default variable for further statistical analysis. Here, `nodes$type` refers to the typology of the GUs (anthropomorph, weapons, etc.). The `nodes$x` and `nodes$y` columns refer to the x and y coordinates of the nodes. As said, in the first place theses coordinates come from the GIS. But, in a GIS, the coordinates origin (0, 0) is the bottom-left corner and exist negative values, while this origin is top-left for any R matrices (rasters, grids, dataframes, etc.) with only positive values. To recover the correct the y value of GUs nodes and edges, that is to say the y value on the decoration image, the **decorr** calculate the absolute y value and used the image height as a constant offset.

- The `edges` dataframe

The `edges` dataframe is quite similar to the `nodes` dataframe.

```
R> caption <- "Edges (from \\code{edges.csv} dataframe)}"
R> print(xtable::xtable(head(edges),
+                 caption=caption,
+                 label="edgesdf"),
+       table.placement="H")
```

	V1
1	site decor a b type
2	1 Cerro Muriano Cerro Muriano 1 1 4 =
3	2 Cerro Muriano Cerro Muriano 1 1 5 =
4	3 Cerro Muriano Cerro Muriano 1 3 5 =
5	4 Cerro Muriano Cerro Muriano 1 1 2 +
6	5 Cerro Muriano Cerro Muriano 1 1 7 +

Table 3: Edges (from `edges.csv` dataframe)

Fields edges$site and edges$decor are the primary key of decoration. The fields edges$a and edges$b are the equivalent to columns *from* and *to* in Graph theory. Even if undirected graphs will the most common models in further studies, this direction helps to distinguish between nodes. The first column edges$a is the identifier of starting node or *main node* or *overlapping node*. The second column edges$b is the identifier of the ending node or *attribue node* or *overlapped node*. The edges$type is the type of relation (normal, attribute, overlapping, etc.) between the starting node and the ending node. There is no need to get the coordinates of the edges, these coordinates are calculated from the nodes dataframe. For example, Table 3.2 shows that the first edge of the *Cerro Muriano 1* decoration connects the nodes 1 and 8 (respectively in column edges$a and edges$b). A way to retrieve these connected nodes' coordinates will be:

```
R> cm.1 <- subset(nodes, decor == "Cerro Muriano 1" & id == 1)[,c("x","y")]

## Error in eval(e, x, parent.frame()):  object 'decor' not found

R> cm.8 <- subset(nodes, decor == "Cerro Muriano 1" & id == 8)[,c("x","y")]

## Error in eval(e, x, parent.frame()):  object 'decor' not found

R> cat(as.numeric(cm.1),";",as.numeric(cm.8))

## Error in cat(as.numeric(cm.1), ";", as.numeric(cm.8)):  object 'cm.1' not found
```

Once these three dataframes loaded, the list of decoration graphs can be calculated with the list_dec() function.

3.3. list_dec() function

The list_dec() function allows to calculate graphs for all decorations stored into nodes, edges and images. The result is a list of decoration graph. The first graph of can be plotted

```
R> # par(mar=c(0.1,0.1,0.1,0.1))
R> # library("decorr")
R> # library("magick")
R> # imgs <- read.table(system.file("extdata", "imgs.tsv", package = "decorr"),
R> #                           sep="\t", stringsAsFactors = FALSE)
R> # nodes <- read.table(system.file("extdata", "nodes.csv", package = "decorr"),
R> #                         sep="\t",stringsAsFactors = FALSE)
R> # edges <- read.table(system.file("extdata", "edges.csv", package = "decorr"),
R> #                         sep="\t",stringsAsFactors = FALSE)
R> lgrph <- list_dec(imgs,nodes,edges,var="type")

## Error in igraph::graph_from_data_frame(g.edges, directed = FALSE, vertices =
g.nodes):  the data frame should contain at least two columns

R> plot(lgrph[[1]],
```

```
+       vertex.color = "orange",
+       vertex.frame.color="orange",
+       vertex.label.color = "black",
+       vertex.size = 8,
+       vertex.label.cex = 0.6,
+       edge.color = "orange",
+       vertex.label.family="Helvetica"
+  )

## Error in plot(lgrph[[1]], vertex.color = "orange", vertex.frame.color = "orange",
:  object 'lgrph' not found
```

The others **decorr** package functions can be divided into:

1. graphical functions
2. single decoration functions
3. comparisons between different decorations functions

3.4. Graphical functions

The **decorr** package has three graphical functions

- `labels_shadow()` function is a re-use of the `shadowtext()` function from the **TeachingDemos** package (Snow 2020).
- `side_plot_nds()` and `side_plot_eds()` allow to plot figures side-by-side for nodes or edges comparisons

3.5. Single decoration functions

Functions allowing to create a RNG for a single decoration are:

- `read_nds()` and `read_eds()` functions allow to read respectively a file of nodes and a file of edges (`.tsv` or `.shp` files)

The `read_nds()` function is close to the R native `read.table()` function, but allows to read *shapefiles* of nodes.

The `read_eds()` permits to read a *shapefiles* of edges or to retrieve the coordinates of the the edges from the `nodes` dataframe. For example, the first *Torrejon Rubio 1* edge, between the nodes `6` and `5`, has the starting point (xa = 366.7001, ya = -563.1358) and the ending point (xb = 490.1195, yb = -513.2428)

```
R> # library(decorr)
R> sit <- "Torrejon Rubio" ; dec <- "Torrejon Rubio 1"
R> nds.df <- read_nds(site = sit, decor = dec, dev = ".tsv",
+                     doss = system.file("extdata", package = "decorr"))
R> eds.df <- read_eds(site = sit, decor = dec, dev = ".tsv",
+                     doss = system.file("extdata", package = "decorr"))
R> print(xtable::xtable(eds.df[1,],
+                       caption="first edge of the dataframe",
+                       label="Test_table_1",
+                       size=7),
+       table.placement="H")
```

	site	decor	a	b	type	xa	ya	xb	yb
9	Torrejon Rubio	Torrejon Rubio 1	6	5	=	366.70	-563.14	490.12	-513.24

Table 4: first edge of the dataframe

- `plot_dec_grph ()` allows to plot a RNG over a decoration image

Once, the `imgs`, `nodes` and `edges` dataframes have been read, the decoration graph is build and can be plotted, here for the *Torrejon Rubio 1* decoration. The `lbl.txt` parameter allows to decide which field of the nodes will be displayed as the label, by default this is the `nodes$id` field, but here it is the `nodes$type` field.

```
R> # library("decorr")
R> par(mar=c(0,0,0,0))
R> sit <- "Torrejon Rubio" ; dec <- "Torrejon Rubio 1"
R> nds.df <- read_nds(site = sit, decor = dec, dev = ".tsv",
+                     doss = system.file("extdata", package = "decorr"))
R> eds.df <- read_eds(site = sit, decor = dec, dev = ".tsv",
+                     doss = system.file("extdata", package = "decorr"))
R> img.graph <- plot_dec_grph(nds.df = nds.df,
+                            eds.df = eds.df,
+                            site = sit,
+                            decor = dec,
+                            doss = system.file("extdata", package = "decorr"),
+                            lbl.txt = "type",
+                            lbl.size=1.7,
+                            shw = c("nodes","edges"))
R> plot(image_trim(img.graph))
```

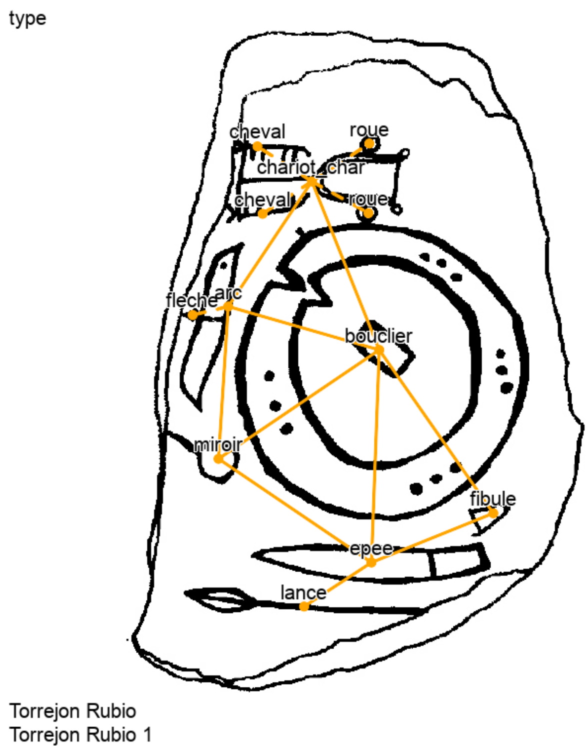

Figure 6: Torrejon Rubio 1 decoration

3.6. Decoration comparison functions

Functions allowing to compare different decorations with RNG are

- `list_nds_compar()` and `list_eds_compar()` functions allow to compare respectively the common nodes and the common edges between two decorations

Comparisons between pairwise of decorations are first stored into a list. These comparisons are performed for nodes and/or edges. There are four (4) decorations in the default dataset, so there is $\frac{4!}{(4-2)!2!} = 6$ pairwise comparisons

```
R> # set wd to data folder
R> # setwd(system.file("extdata", package = "decorr"))
R> # library("decorr")
R> g.compar <- list_eds_compar(lgrph,"type")

## Error in lapply(lgrph, function(x) x$name):  object 'lgrph' not found

R> df.edges.compar <- data.frame(decor.A=c(g.compar[[1]][[1]]$decor,
+                                          g.compar[[2]][[1]]$decor,
+                                          g.compar[[3]][[1]]$decor,
+                                          g.compar[[4]][[1]]$decor,
+                                          g.compar[[5]][[1]]$decor,
+                                          g.compar[[6]][[1]]$decor),
+                                 decor.B=c(g.compar[[1]][[2]]$decor,
+                                          g.compar[[2]][[2]]$decor,
+                                          g.compar[[3]][[2]]$decor,
+                                          g.compar[[4]][[2]]$decor,
+                                          g.compar[[5]][[2]]$decor,
+                                          g.compar[[6]][[2]]$decor))

## Error in data.frame(decor.A = c(g.compar[[1]][[1]]$decor, g.compar[[2]][[1]]$decor,
:  object 'g.compar' not found

R> print(xtable::xtable(df.edges.compar,
+                      caption="Pairwise comparisons dataframe between decor.A and decor.
+                      label="Test_table_1",
+                      size=7),
+       table.placement="H")

## Error in xtable::xtable(df.edges.compar, caption = "Pairwise comparisons dataframe
between decor.A and decor.B", :  object 'df.edges.compar' not found
```

- `plot_nds_compar()` and `plot_eds_compar()` functions allow to plot and save two figures side-by-side for a decorations pairwise with, respectively, common nodes and common edges identified

The `plot_nds_compar()`and `plot_eds_compar()` functions create a `.png` image of two decorations plotted side-by-side with common nodes or edges identified. Functions returns also the name of the image. The common edges or nodes are displayed in red by default. Let us choose the decorations `1` (*Cerro Muriano 1*) and `4` (*Zarza de Montsanchez*) and identify common edges.

```
R> par(mar=c(0,0,0,0))
R> eds_compar <- plot_eds_compar(g.compar, c(1,4),
+                                doss = system.file("extdata", package = "decorr"))

## Error in unlist(listg, recursive = FALSE): object 'g.compar' not found

R> plot(image_trim(image_read(eds_compar)))

## Error in image_read(eds_compar):  object 'eds_compar' not found
```

The comparison shows that `1` (*Cerro Muriano 1*) and `4` (*Zarza de Montsanchez*) decorations have two (2) common edges: `lance -=- personnage` and `personnage -=- bouclier`.

- `same_nds()` and `same_eds()` functions allow to repectively count matching nodes and matching edges between decoration pairwises

`same_nds()` and `same_eds()` allow to repectively count matching nodes and matching edges between decoration pairwises. The result is a square matrix between all pairwise comparisons with the number of common nodes or edges in the cells. For example, we can compute the matrix of common edges.

```
R> df.same_edges <- same_eds(lgrph,"type")

## Error in lapply(lgrph, function(x) x$name):  object 'lgrph' not found

R> caption <- "Number of same edges between all decoration pairwise comparisons"
R> print(xtable::xtable(df.same_edges,
+                       caption = caption,
+                       label = "Test_table_2",
+                       size = 8,
+                       digits = c(0)),
+       table.placement="H",
+       include.rownames = TRUE)

## Error in xtable::xtable(df.same_edges, caption = caption, label = "Test_table_2",
:  object 'df.same_edges' not found
```

For these two last exemples, the edges comparisons between the decoration `1` and the decoration `4` show that they have two (2) common edges.

4. Illustrations

As said, since the precise location of the GUs is usualy not registred, the most commonly used method in statistical analysis on prehistorical iconography is the presence of common nodes. In order to demonstrate the first insight of a graph-based analysis of the decorations, we will compare two classifications, the first one based on the "classic" presence of common nodes, the second one based on the presence of common edges.

```
## Error in lapply(lgrph,
function(x) x$name):  object
'lgrph' not found
## Error in lapply(lgrph,
function(x) x$name):  object
'lgrph' not found
## Error in xtable(df.same_nodes,
digits = c(0)):  object
'df.same_nodes' not found
```

Table 5: Common nodes table

```
## Error in xtable(df.same_edges,
digits = c(0)):  object
'df.same_edges' not found
```

Table 6: Common edges table

Once the heatmap matrices calculated, the native `dist()` and `hclust()` functions (R Core Team 2019) are calculated from the inverse matrices with the function `inv()` of the **matlib** package (Friendly, Fox, and Chalmers 2020)

```
R> library("matlib")
R> par(mfrow=c(1,2))
R> dist.nodes  <- dist(inv(as.matrix(df.same_nodes)))

## Error in as.matrix(df.same_nodes):  object 'df.same_nodes' not found

R> dist.edges  <- dist(inv(as.matrix(df.same_edges)))

## Error in as.matrix(df.same_edges):  object 'df.same_edges' not found

R> plot(hclust(dist.nodes), hang = -1, main = "common nodes")

## Error in hclust(dist.nodes):  object 'dist.nodes' not found

R> plot(hclust(dist.edges), hang = -1, main = "common edges")

## Error in hclust(dist.edges):  object 'dist.edges' not found
```

Results of classifications show that for both common nodes and common edges, the most different decorations are `1` and `4`. These two decorations share four (4) common nodes and, as previously seen, only two (2) common edges. In any cases decorations `2` and `3` are closer to decoration `4` than to decoration `1`, but their classifications changes depending on counting of common nodes or common edges. While decorations `2` and `4` have 7 common edges and 7 common nodes, plotting the comparisons for `3` and `4`, helps to understand the differences between the two classifications.

```
R> par(mar=c(0,0,0,0))
R> par(mfrow=c(2,1))
```

```
R> g.compar <- list_nds_compar(lgrph,"type")

## Error in utils::combn(1:length(lgrph), 2):  object 'lgrph' not found

R> nds_compar.3.4 <- plot_nds_compar(g.compar, c(3,4),
+                                    doss = system.file("extdata", package = "decorr"))

## Error in plot_nds_compar(g.compar, c(3, 4), doss = system.file("extdata", :
object 'g.compar' not found

R> plot(image_read(nds_compar.3.4))

## Error in image_read(nds_compar.3.4):  object 'nds_compar.3.4' not found

R> g.compar <- list_eds_compar(lgrph,"type")

## Error in lapply(lgrph, function(x) x$name):  object 'lgrph' not found

R> eds_compar.3.4 <- plot_eds_compar(g.compar, c(3,4),
+                                doss = system.file("extdata", package = "decorr"))

## Error in unlist(listg, recursive = FALSE): object 'g.compar' not found

R> plot(image_read(eds_compar.3.4))

## Error in image_read(eds_compar.3.4):  object 'eds_compar.3.4' not found
```

Decorations 3 and 4 share four (4) common GUs (`bouclier`, `epee`, `lance`, `miroir`) but these GUs have different spatial organisations with only one common edge (`bouclier -=- lance`). At the opposite, decorations 2 and 4 show more properly the same compositions.

5. Summary and discussion

In this example we propose the iconographical `nodes$type` (character, weapon, etc.) GUs as the studied variable, but the user of the package can create and choose any other study variable: color for a painting, technique of realisation, size, etc. `edges$type` can also be extended to other types than normal, attribute, overlapping. The background is considered as homogeneous but a crack, a pit, a something can also be considered The plasticity of Graph Theory allows to develop conventions in order to quote the different types of relations s. Its geographical equivalent is a Thiessen polygon

node 1	edge type	node 2	(un)directed	*birel*	description
`A`	`=`	`B`	undirected	$A \cap B = \emptyset$	`A` and `B` are disjoint, `A` and `B` can be contemporaneous
`A`	`+`	`B`	directed	$A \cap B = A$	`A` and `B` are contemporaneous, `B` is an attribute of `A`
`A`	`>`	`B`	directed	$A \cap B = \exists$	`A` overlaps `B`, `A` can be more recent than `B`

Table 7: Synthesis for the different types of relations between GUs

Acknowledgments

All acknowledgments (note the AE spelling) should be collected in this unnumbered section before the references. It may contain the usual information about funding and feedback from colleagues/reviewers/etc. Furthermore, information such as relative contributions of the authors may be added here (if any).

References

Alexander C (2008). "The Bedolina map – an exploratory network analysis." In A Posluschny, K Lambers, I Herzog (eds.), *Layers of Perception. Proceedings of the 35th International Conference on Computer Applications and Quantitative Methods in Archaeology (CAA), Berlin, 2.-6. April 2007*, pp. 366–371. Koll. Vor- u. FrÃijhgesch. doi:https://doi.org/10.11588/propylaeumdok.00000512.

Bivand R, Keitt T, Rowlingson B (2019). *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*. R package version 1.4-7, URL https://CRAN.R-project.org/package=rgdal.

Clarke DL (2014). *Analytical archaeology*. Routledge.

Csardi G, Nepusz T (2006). "The igraph software package for complex network research." *InterJournal*, **Complex Systems**, 1695. URL http://igraph.org.

De Saussure F (1989). *Cours de linguistique générale*, volume 1. Otto Harrassowitz Verlag.

d'Errico F, Nowell A (2000). "A new look at the Berekhat Ram figurine: implications for the origins of symbolism." *Cambridge Archaeological Journal*, **10**(1), 123–167.

Díaz-Guardamino Uribe M (2010). *Las estelas decoradas en la Prehistoria de la Península Ibérica*. Ph.D. thesis, Universidad Complutense de Madrid, Servicio de Publicaciones.

Friendly M, Fox J, Chalmers P (2020). *matlib: Matrix Functions for Teaching and Learning Linear Algebra and Multivariate Statistics*. R package version 0.9.3, URL https://CRAN.R-project.org/package=matlib.

Huet T (2018). "Geometric graphs to study ceramic decoration." In M Matsumoto, E Uleberg (eds.), *Exploring Oceans of Data, proceedings of the 44th Conference on Computer Applications and Quantitative Methods in Archaeology, CAA 2016*, pp. 311–324. Archaeopress.

Ooms J (2018). "Magick: advanced graphics and image-processing in R." *CRAN. R package version*, **1**.

Pérez SC (2001). *Estelas de guerrero y estelas diademadas: la precolonialzación y formación del mundo tartésico*. Ediciones Bellaterra.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Renfrew C, Bahn PG (1991). *Archaeology: theories, methods and practice*, volume 2. Thames and Hudson London.

Snow G (2020). *TeachingDemos: Demonstrations for Teaching and Learning*. R package version 2.12, URL https://CRAN.R-project.org/package=TeachingDemos.

Tobler WR (1970). "A computer movie simulating urban growth in the Detroit region." *Economic geography*, **46**(sup1), 234–240.

A. More technical details

Appendices can be included after the bibliography (with a page break). Each section within the appendix should have a proper section title (rather than just *Appendix*).

For more technical style details, please check out JSS's style FAQ at `https://www.jstatsoft.org/pages/view/style#frequently-asked-questions` which includes the following topics:

- Title vs. sentence case.
- Graphics formatting.
- Naming conventions.
- Turning JSS manuscripts into R package vignettes.
- Trouble shooting.
- Many other potentially helpful details…

B. Using BibTEX

References need to be provided in a BibTEX file (`.bib`). All references should be made with `\cite`, `\citet`, `\citep`, `\citealp` etc. (and never hard-coded). This commands yield different formats of author-year citations and allow to include additional details (e.g., pages, chapters, …) in brackets. In case you are not familiar with these commands see the JSS style FAQ for details.

Cleaning up BibTEX files is a somewhat tedious task – especially when acquiring the entries automatically from mixed online sources. However, it is important that informations are complete and presented in a consistent style to avoid confusions. JSS requires the following format.

- JSS-specific markup (`\proglang`, `\pkg`, `\code`) should be used in the references.
- Titles should be in title case.
- Journal titles should not be abbreviated and in title case.
- DOIs should be included where available.
- Software should be properly cited as well. For R packages `citation("pkgname")` typically provides a good starting point.

Affiliation:

Thomas Huet
CNRS-UMR 5140
Archeologie des Societes Mediterraneennes
Universite Paul Valery
route de Mende
Montpellier 34199, France
E-mail: thomashuet7@gmail.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd