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

Abstract

By definition, Prehistorical societies are characterised by the absence of a writing system. Prehistorical times cover more than 99% of the human living. Even if it is being discussed, first symbolic manifestations start around 200,000 BC (d'Errico and Nowell 2000). The duration from first symbolic expressions to start of writing represents 97% of the human living. In illiterate societies, testimonies of symbolic systems mostly come from iconography (ceramic decorations, rock-art, statuary, etc.) and signs are displayed mostlty a discontinuous figures which can have different relationships one with another. An graphical composition can be "read" as a spatial distribution of features having intrinsic values possibily having meaningful relationships one with another depending on their pairwise spatial proximities.

To understand meaningful associations of signs, geometric tools, graph analysis and statistical analysis offer great tools to recognize iconographical patterns and to infer collective conventions. We present the **decorr** R package which ground concepts, methods and tools to analyse ancient graphical systems.

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

concordance=TRUE

1. Introduction

For decades, study of ancient iconography was linked to history of religion because closely linked to symbolism, believes and religions. 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 opÃľratoire*, susbsitence strategies, etc. (Renfrew and Bahn 1991), (Leroi-Gourhan 1992). But unlike many aspects of the material culture – 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 study of ancient iconography had undergone significative improvements at the site scale – with GIS, database, paleoclimatic 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 signs organised also in conventional manners. Until our days, formal methods to study ancient iconography Semantics, has 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 – specially in alphabetic and binary writings – and the development of the signified on a horizontal, vertical or boustrophedon axis. Let us take the example of the word `"art"` which contains 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 (GUs) are not necessary 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.

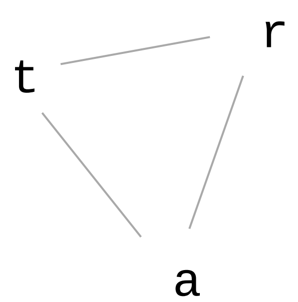


Figure 2: Potential spatial relations between GUs.

Applying the Saussurian model to any prehistorical graphical content had led to tedious division of the iconographical content with, for example, graphical as a relationship of figures grouping GUs, patterns grouping figures, motives grouping patterns, etc., until the entire decorated support is described and can be compared to another decoration (XXX). But during this *decomposition* process, many imprecisions occur:

- groups and relationships are often defined empirically
- their level of significance are often implicit
- the iconographical and spatial proximities between GUs and categories of GUs are not quantified

Furthermore, due to the inherent variability of iconography, most of the studies developp proper descriptive vocabularies, singular relationships of categories, idosyncratic methods in a site-dependend or period-dependend scales. This limits 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 geometric 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. The idea is that a graphical system can be represented by vertices connected (or not) to each other with edges. This package has been grounded on the seminal work of C. Alexander (Alexander 2008) and its first IT implementation by T. Huet (Huet 2018).

2. 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. The spatial levels of the GUs can be retrieve by a planar graph (Graph Theory) and a spatial (GIS) analysis. Nodes and edges – repectively for GUs and their connexion – are created on a GIS interface. In the GIS, the decoration figure is open in the first place in a new project with no projection. The decoration image will be considerated as the basemap of the project and will cover the region of interest of the analysis. The decoration image can be binarized where GUs are considerated active and the undecorated parts of the support, or background, are considerated inactive. After what, the decoration image is tiled. A simplier solution will be to create directly centroids over the GUs. The x and y coordinates of the nodes are relative to the decoration and measured in pixels. Exist a link between a couple of GUs when these graphical units share a border. A planar graph is constructed from grahical units (nodes) and their proximity links (edges). This model is a Voronoi diagram of the support where the Voronoi seeds are the GUs. Its geographical equivalent is a Thiessen polygon.

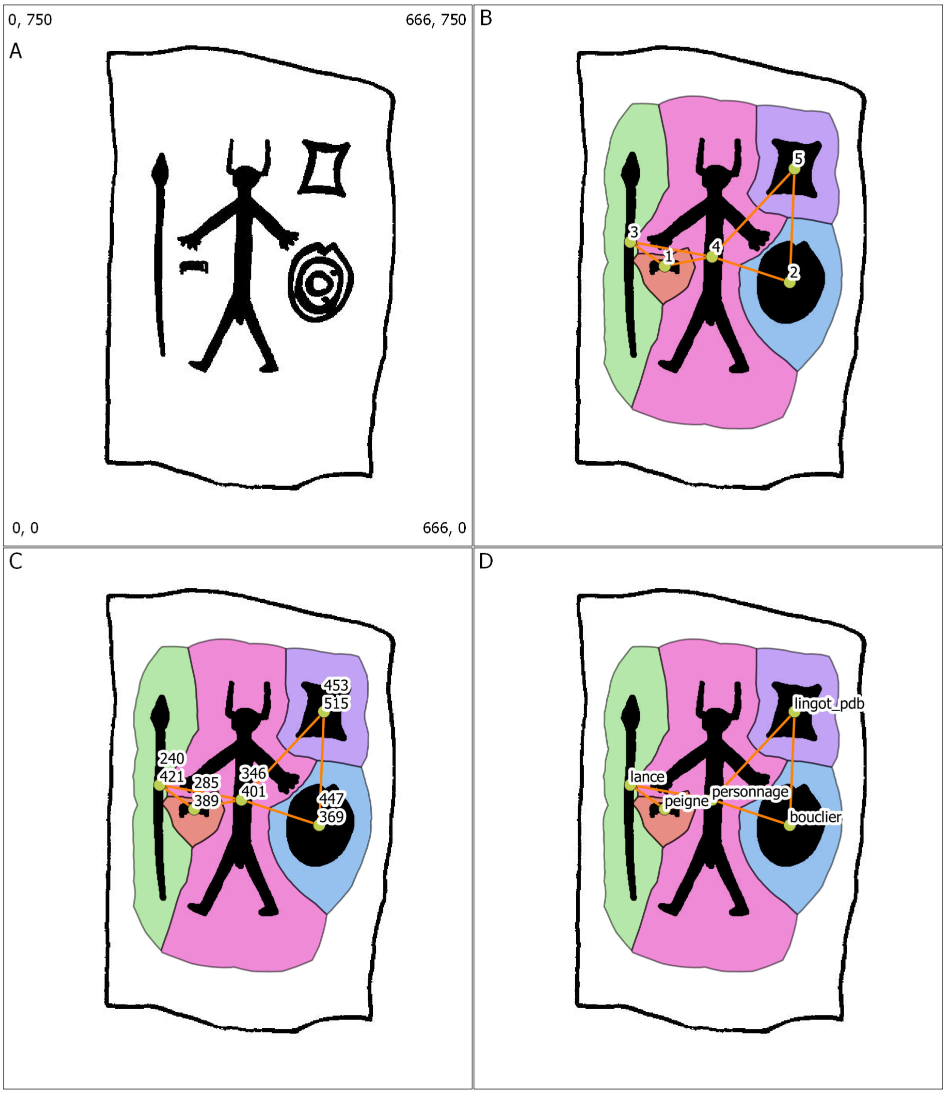


Figure 3: GIS interface. A) Original decoration of the Late Bronze Age Cerro Muriano 1 stele (drawing: Díaz-Guardamino Uribe (2010)) with its extent (`xmin, xmax, ymin, ymax`); B) After the polygonisation of the GUs, including the border of the stelae, the Voronoi cells, the centroid of GUs and the links 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.

This model has a minimal of *a priori* definitions. Those definitions only concern the GUs (type, technology, color, orientation, size, etc.). The plasticity of fraph theory allows to develop conventions in order to quote the different types of relations between GUs.

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

But 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). To record this information, a new type of edge, tagged with `'+'`, is be introduced. This type of edges is be directed and displayed with a dashed line. Its starts from the *main unit* and ends with the *attribute units*. For example `4 -+- 6`) means that the main node `4` has the attribute node `6`.

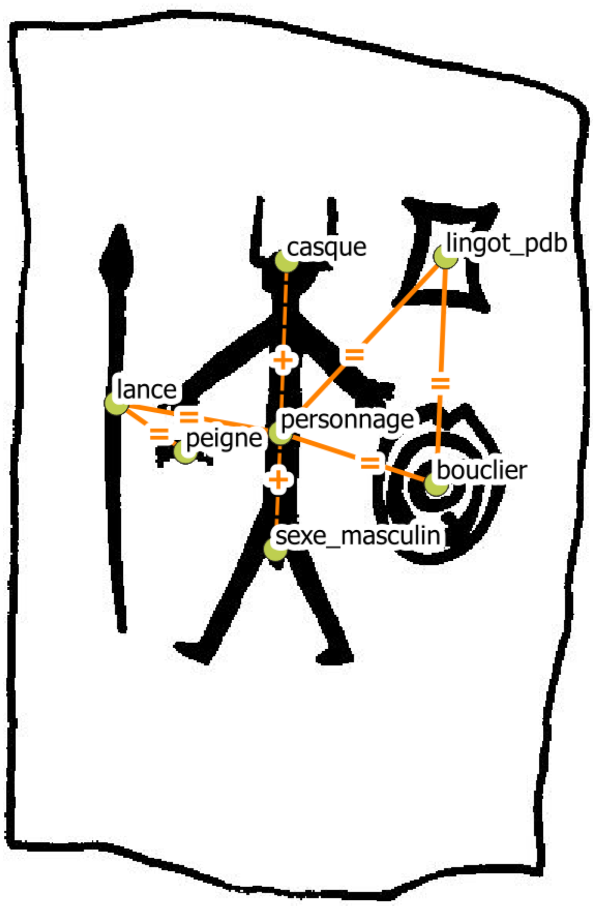


Figure 4: GIS interface. The GUs 'casque' (helmet) and 'sexe_masculin' (male sex) are two attributes of the GU 'personnage' (character).

Finally, it is quite common that a graphical composition shows superimpositions between different UGs. This stratigraphic 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. For example `A ->- B` means that `A` crosses `B`.

node 1	edge type	node 2	(un)directed	*birel*	stratigraphical meaning
`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 1: Synthesis for the different types of relations between GUs

3. The R package decorr

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

```
R>   devtools::install_github("zoometh/iconr")
```

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/or 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

The training dataset is a selection of four drawing of stelae belonging to the Late Bronze age of the SW Iberian peninsula. At the first, the training dataset is in the `extdata` folder of the **decorr**. The dataframe storing the inventory of decorations is `imgs`.

- The `imgs` dataframe structure is

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`.

At first the drawing dataset can be checked by using the `imgs` dataframe and the **magick**
width=16,height=16

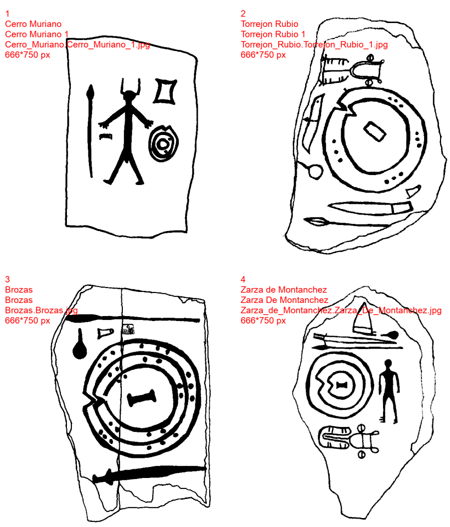
idf	site	decor	img
1	Cerro Muriano	Cerro Muriano 1	Cerro_Muriano.Cerro_Muriano_1.jpg
2	Torrejon Rubio	Torrejon Rubio 1	Torrejon_Rubio.Torrejon_Rubio_1.jpg
3	Brozas	Brozas	Brozas.Brozas.jpg
4	Zarza de Montanchez	Zarza De Montanchez	Zarza_de_Montanchez.Zarza_De_Montanchez.jpg

Table 2: The studied corpus, the imgs.tsv dataframe

```
R> library(magick)

## 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

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)
```



As said, the GIS offers the more suitable interface to register all GUs and to get their coordinates. But the coordinates origin (0,0) in a GIS is the bottom-left corner, while this origin is top-left for any R rasters or matrices: this will affect the y axis. To recover the correct GUs coordinates on this axis, for nodes and edges, between the inputs in the GIS and the decoration image coordinates in R, the **decorr** calculate the absolute y value and used the image height as a constant offset.

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

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

- The `nodes` dataframe structure is

```
R> nodes.cm <- subset(nodes, decor == "Cerro Muriano 1")
R> caption <- "Nodes (\\code{nodes.csv} dataframe) for \\emph{Cerro Muriano 1}"
R> xtable::xtable(nodes.cm,
+        caption = caption)
```

	site	decor	id	type	x	y
1	Cerro Muriano	Cerro Muriano 1	1	personnage	349.81	-298.32
2	Cerro Muriano	Cerro Muriano 1	2	casque	349.81	-243.99
3	Cerro Muriano	Cerro Muriano 1	3	lance	238.46	-298.32
4	Cerro Muriano	Cerro Muriano 1	4	bouclier	446.02	-381.17
5	Cerro Muriano	Cerro Muriano 1	5	peigne	283.00	-358.01
6	Cerro Muriano	Cerro Muriano 1	7	sexe_masculin	342.69	-427.49
7	Cerro Muriano	Cerro Muriano 1	8	lingot_pdb	451.15	-237.48

Table 3: Nodes (`nodes.csv` dataframe) for *Cerro Muriano 1*

- The `edges` dataframe structure is

```
R> edges.cm <- subset(edges, decor == "Cerro Muriano 1")
R> caption <- "Edges (\\code{edges.csv} dataframe) for \\emph{Cerro Muriano 1}"
R> xtable::xtable(edges.cm,
+               caption=caption)
```

For edges, there is no need to get the coordinates of the starting point and the ending point. These coordinates can be calculated from the `nodes` dataframe. For example, the first edge of the *Cerro Muriano 1* decoration connect the nodes `1` and `8`. A way to retrieve coordinates of these two nodes – which are the two end points – will be:

	site	decor	a	b	type
1	Cerro Muriano	Cerro Muriano 1	1	8	=
2	Cerro Muriano	Cerro Muriano 1	4	8	=
3	Cerro Muriano	Cerro Muriano 1	1	4	=
4	Cerro Muriano	Cerro Muriano 1	1	5	=
5	Cerro Muriano	Cerro Muriano 1	3	5	=
6	Cerro Muriano	Cerro Muriano 1	1	2	+
7	Cerro Muriano	Cerro Muriano 1	1	7	+
8	Cerro Muriano	Cerro Muriano 1	3	1	=

Table 4: Edges (`edges.csv` dataframe) for *Cerro Muriano 1*

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

## 349.8148 -298.3244 ; 451.1489 -237.4782
```

Once done, the list of graphs can be stored with the `list_dec()` function.

3.3. `list_dec()` function

The `list_dec()` function allows to store graphs for each decorations stored into `nodes`, `edges` and `images` dataframes and store the graphs in a list. The join between these dataframes is done on the two fields `site` and `decor`. The first graph of can be plotted

```
R> par(mar=c(0.1,0.1,0.1,0.1) )
R> library(decorr)
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")
R> plot(lgrph[[1]],
+       vertex.color = "orange",
+       vertex.frame.color="orange",
+       vertex.label.color = "black",
+       vertex.size = 10,
+       vertex.label.cex =.7,
+       edge.color = "orange"
+       # vertex.label.family="Courier New"
+       )
```

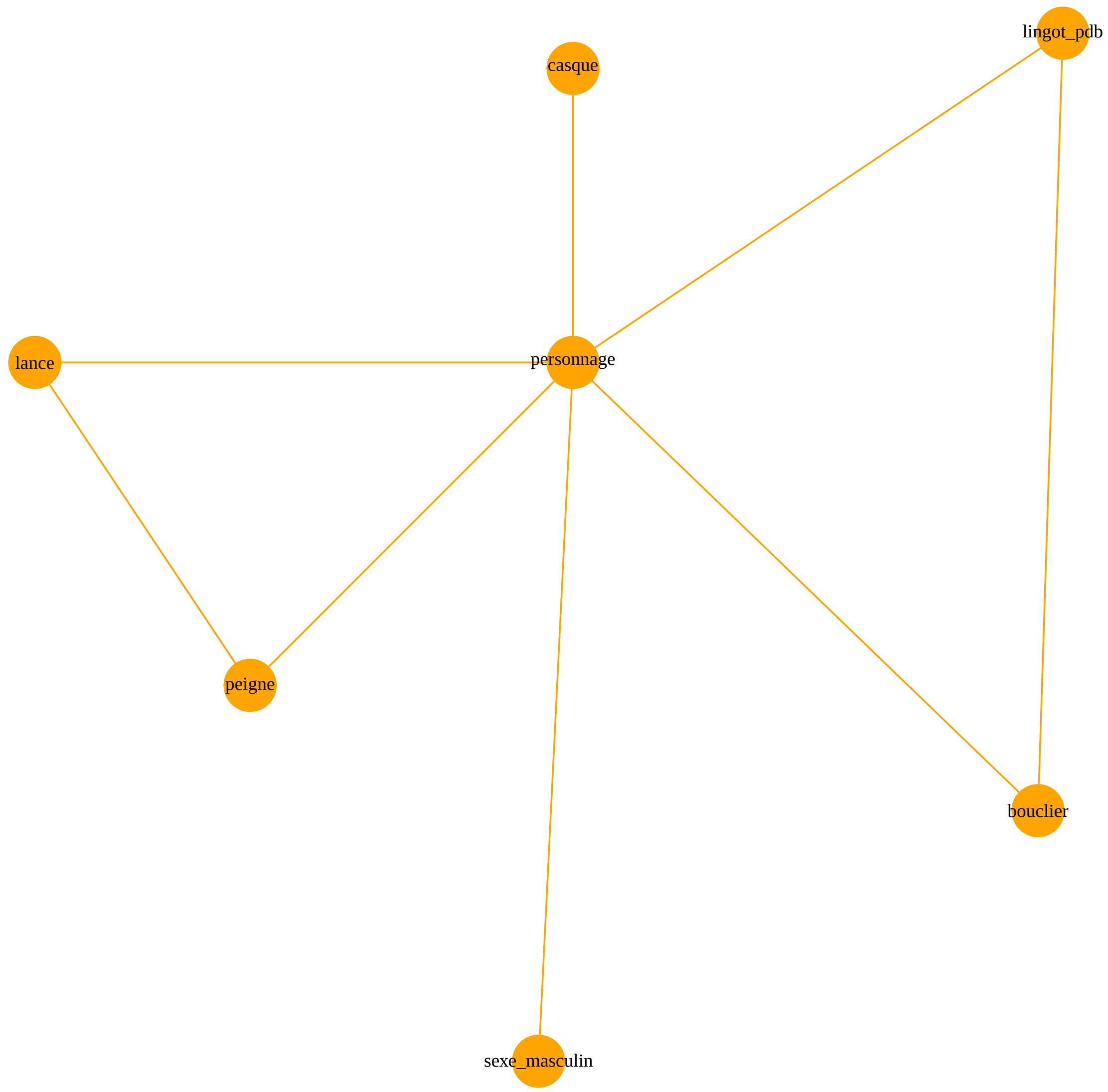


Figure 6: Plot of the first graph of the list

H

```
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> xtable::xtable(eds.df[1,],
+                 caption="first edge of the dataframe",
+                 label="Test_table_1",
+                 size=7)
```

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** has three purely 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 geometric graph 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 native `read.table()` function but allows to read *shapefiles* of nodes.

the `read_eds()` permits to read a *shapefiles* of nodes or to retrieve the coordinates of the ends of 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)

- `plot_dec_grph ()` allows to plot a geometric graph 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 allow to decide which field of the nodes will be displayed as the label, here the column `nodes$type`

width=3,height=3

3.6. Decoration comparisons function

The functions allowing to compare different decorations with geometric graphs 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 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

	decorA	decorB
1	Cerro Muriano 1	Torrejon Rubio 1
2	Cerro Muriano 1	Brozas
3	Cerro Muriano 1	Zarza De Montanchez
4	Torrejon Rubio 1	Brozas
5	Torrejon Rubio 1	Zarza De Montanchez
6	Brozas	Zarza De Montanchez

Table 6: comparison dataframe

- `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*)

H

```
R> library(decorr)
R> par(mar=c(1,1,1,1) )
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(img.graph)
```

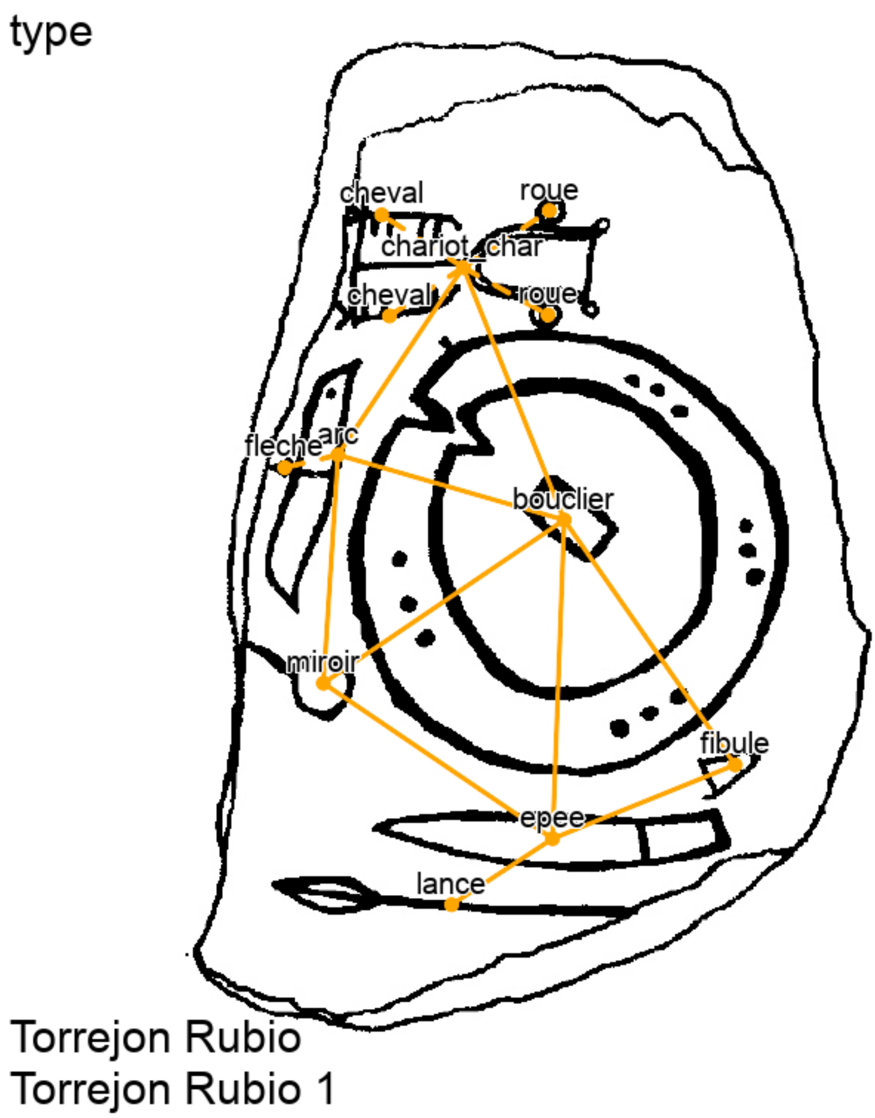


Figure 7: *Torrejon Rubio 1*

width=20,height=10

```
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"))
R> plot(image_read(eds_compar))
```

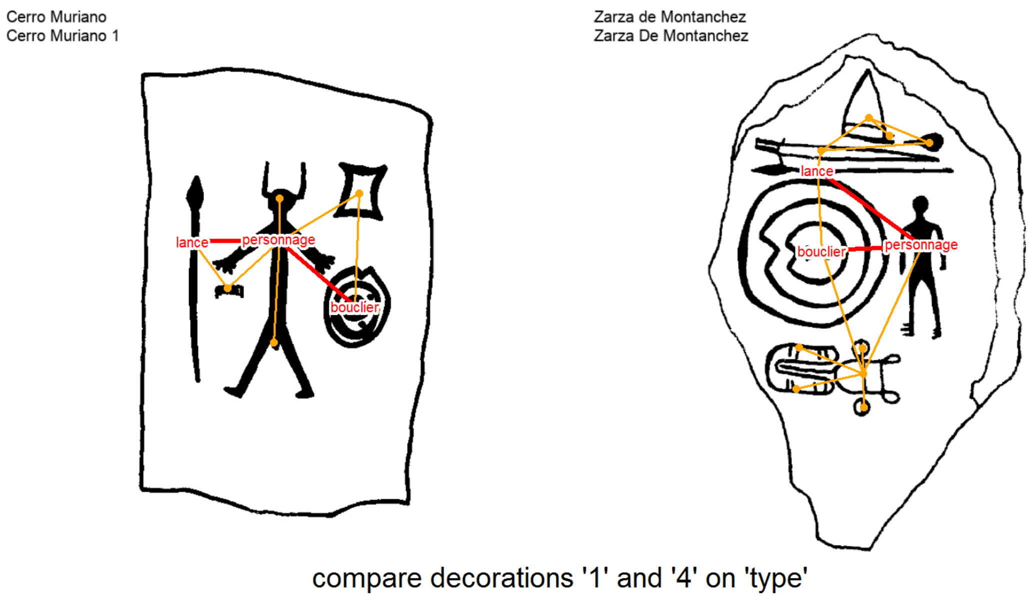


Figure 8: comparisons between `1` (*Cerro Muriano 1*) and `4` (*Zarza de Montsanchez* decorations

The comparison on Figure 8 shows that `1` (*Cerro Muriano 1*) and `4` (*Zarza de Montsanchez* decorations have two (2) common edges: `lance -=- personnage` and `bouclier -=- personnage`

- `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 with all pairwise comparisons and the number of common nodes or edges in the cells.

```
R> df.same_edges <- same_eds(lgrph,"type")
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)),
+       include.rownames=TRUE)
```

	1	2	3	4
1	0	0	1	2
2	0	0	3	7
3	1	3	0	1
4	2	7	1	0

Table 7: Number of same edges between all decoration pairwise comparisons

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

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

	1	2	3	4
1	0	2	3	4
2	2	0	5	7
3	3	5	0	4
4	4	7	4	0

Table 8: Common nodes table

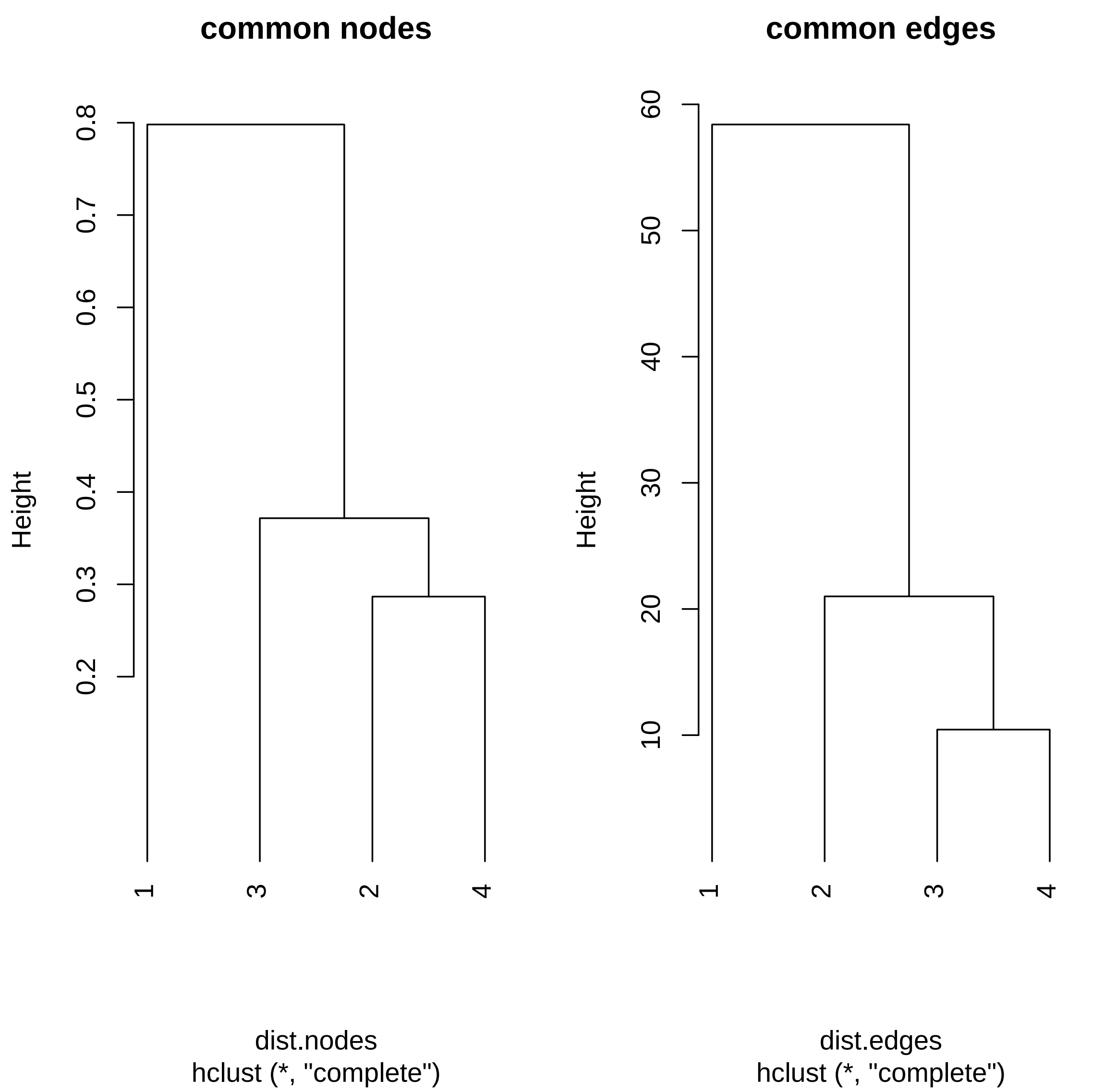
	1	2	3	4
1	0	0	1	2
2	0	0	3	7
3	1	3	0	1
4	2	7	1	0

Table 9: Common edges table

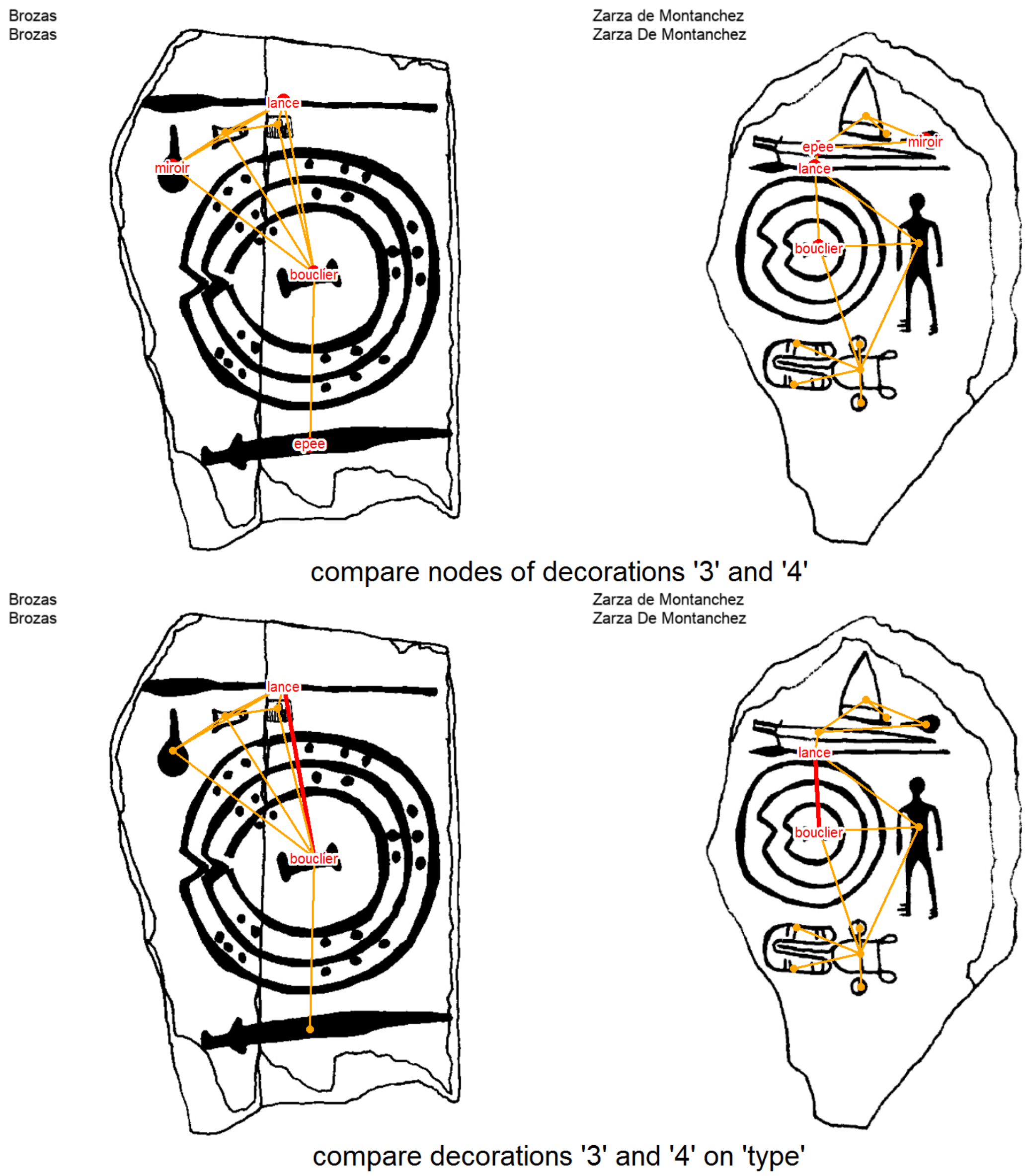
```
R> library(matlib)

## Warning:  package 'matlib' was built under R version 3.6.3

R> par(mfrow=c(1,2))
R> dist.nodes  <- dist(inv(as.matrix(df.same_nodes)))
R> dist.edges  <- dist(inv(as.matrix(df.same_edges)))
R> plot(hclust(dist.nodes), hang = -1, main = "common nodes")
R> plot(hclust(dist.edges), hang = -1, main = "common edges")
```



For both nodes and edges, the most distant decorations are `1` and `4`. These two decorations share four (4) common nodes and, as previously seen, 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. Plotting the comparisons for for `3` and `4`, helps to understand the differences between the two classifications.



width=20,height=20

Decorations `3` and `4` share four (4) common GUs (`bouclier, epee, lance, miroir`) but the spatial organisation of theses GUs are different between the two decorations so their number of common edges is lower with only one common edge (`bouclier -=- lance`)

5. Summary and discussion

- As usual ...

Computational details

> If necessary or useful, information about certain computational details such as version numbers, operating systems, or compilers could be included in an unnumbered section. Also, auxiliary packages (say, for visualizations, maps, tables, ...) that are not cited in the main text can be credited here.

The results in this paper were obtained using R 3.4.1 with the **MASS** 7.3.47 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

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.

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.

Leroi-Gourhan A (1992). *L'art pariétal: langage de la préhistoire*. Editions Jérôme Millon.

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

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`.

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