# Package 'iconr'

February 16, 2021

**Title** Graphical and Spatial Analysis for Prehistoric Iconography

**Version** 0.1.0

**Description** Set of formal methods for studying archaeological iconographic datasets (rock-art, pottery decoration, stelae, etc.) using network and spatial analysis (Alexander 2008 <doi:10.11588/propylaeumdok.00000512>; Huet 2018 <https://hal.archives-ouvertes.fr/hal-02913656>).

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Imports** igraph, magick, rgdal, grDevices, graphics, utils

**Suggests** ggplot2, knitr, rmarkdown, dplyr, kableExtra, data.tree, dendextend

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**URL** https://zoometh.github.io/iconr/

**BugReports** https://github.com/zoometh/iconr/issues

**NeedsCompilation** no

**Author** Thomas Huet [aut, cre] (<https://orcid.org/0000-0002-1112-6122>), Jose Pozo [aut] (<https://orcid.org/0000-0002-0759-3510>), Craig Alexander [ctb] (<https://orcid.org/0000-0001-7539-6415>)

**Maintainer** Thomas Huet <thomashuet7@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-02-16 09:10:05 UTC

## R topics documented:

---

`contemp_nds` *Select Contemporaneous Nodes*

---

## Description

Find the connected component, or subgraph, of contemporaneous nodes (connected by *normal* and *attribute* edges) given a selected node and remove the other components

## Usage

```
contemp_nds(nds.df, eds.df, selected.nd)
```

## Arguments

| | |
|---|---|
| `nds.df` | Dataframe of the nodes as the one obtained by the function `read_nds`. |
| `eds.df` | Dataframe of the edges as the one obtained by the function `read_eds`. |
| `selected.nd` | The node of the decoration graph for which to extract the connected component. It can be either the node order (numeric) or the node name/id (character). |

## Value

A named list of two dataframes: `list(nodes,edges)`, collecting the contemporaneous nodes and edges, respectivelly.

## Examples

```
# Set data folder
dataDir <- system.file("extdata", package = "iconr")

# Read a decoration
nds.df <- read_nds(site = "Ibahernando",
                   decor = "Ibahernando",
                   dir = dataDir)
eds.df <- read_eds(site = "Ibahernando",
                   decor = "Ibahernando",
                   dir = dataDir)

# Extract the subgraph contemporaneous to the node 2
l_dec_df <- contemp_nds(nds.df, eds.df, selected.nd = 2)
```

```
## It returns a list of two dataframes, one for nodes and one for edges:
l_dec_df
```

---

`labels_shadow` *Plot Labels with Contrasting Shadow*

---

## Description

Plot labels (text) with a contrasting buffer to make them more visible when located on a similar color background. This function is the `shadowtext()` function developed by Greg Snow. Called by plot functions: `plot_dec_grph`, `plot_compar`

## Usage

```
labels_shadow(x, y = NULL, labels,
              col = "black", bg = "white",
              theta = seq(0, 2 * pi, length.out = 50),
              r = 0.1,
              cex = 1, ...)
```

## Arguments

| | |
|---|---|
| `x, y` | Numeric vector of coordinates where the labels should be plotted. Alternatively, a single argument `x` can be provided with the same syntax as in `xy.coords`. |
| `labels` | Set of labels provided as a character vector. |
| `col, bg` | Graphical parameters for the label color and background (buffer) color. |
| `theta` | Angles for generating the buffer with possible anisotropy along one direction (default is isotropic) and controlling buffer smoothness (angular resolution). |
| `r` | Thickness of the buffer relative to the size of the used font, by default 0.1. |
| `cex` | Size of the label, by default 1. |
| `...` | Further graphical parameter accepted by `text`, such as `pos`, `offset`, or `family`. |

## Value

No return value. It creates a contrasting buffer to make labels more visible.

## References

https://rdrr.io/cran/TeachingDemos/man/shadowtext.html

`list_compar` *Graph Pairwise Comparison on Common Elements*

## Description

`nds_compar` identifies **common nodes** in a pair of graphs.

`eds_compar` identifies **common edges** in a pair of graphs.

Given a list of graphs, `list_compar` extract all combinations of graph pairs and compare them on common elements (nodes and edges).

## Usage

```
nds_compar(grphs, nd.var = "type")

eds_compar(grphs, nd.var = "type")

list_compar(lgrph, nd.var = "type",
            verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `grphs` | A list of two graphs (pair of graphs) to be compared. |
| `lgrph` | A list of any number of graphs to be pairwise compared. The list can be typically obtained with the function `list_dec` |
| `nd.var` | An attribute of the graph nodes containing the node variable (ie, field) on which the comparison will be done. By default `nd.var = "type"`. |
| `verbose` | Logical. If TRUE, the names of each graph pair combination are listed on the screen. By default `verbose = FALSE`. |

## Details

`list_compar()` calls the functions: `nds_compar()` and `eds_compar()` which return respectively the **common nodes** and the **common edges** of a graph pairwise.

**Nodes** are common when they have the same value for a given variable, for example `horse`, `sword`, etc., for the variable `type` (`nd.var = "type"`).

**Edges** are common when they have the same value for *starting* and *ending* nodes (`horse`, `sword`, etc.) and the same type of edge (`'='`, `'+'`, etc.). For example, `a -=-b` in graph 1 is equal to `a -=-b` in graph 2, but not equal to `a -+-b`. Edges of type `=` (*normal* edges) are undirected, so that `a -=-b` is equal to `b -=-a`. But edges of types `+` (*attribute* edges) or `>` (*diachronic* edges) are directed, so: `a ->-b` is not equal to `b ->-a`.

If any of the graphs has multiple nodes/edges with the same value, it is considered to count for as many coincidences as the smaller multiplicity. For instance, if there are 2 nodes with value `epee` in graph 1, and 3 nodes with value `epee` in graph 2, their number of common nodes is `min(2,3) = 2`.

## Value

`nds_compar()` returns the input pair of graphs, each complemented with a new node attribute named `comm` with value 1 for common nodes and 0 for non-common nodes.

`eds_compar()` returns the input pair of graphs, each complemented with a new edge attribute named `comm` with value 1 for common edges and 0 for non-common edges.

`list_compar()` returns a list of all combinations of graph pairs. For each pair, both graphs are complemented with the node attribute (`comm`) identifying common nodes and the edge attribute (`comm`) identifying common edges. Each pair is also complemented with an attribute named `nd.var` recording the compared node variable.

## See Also

`list_dec`, `plot_compar`, `same_elements`

## Examples

```
# Read data
imgs <- read.table(system.file("extdata", "imgs.tsv", package = "iconr"),
                   sep="\t",stringsAsFactors = FALSE)
nodes <- read.table(system.file("extdata", "nodes.tsv", package = "iconr"),
                    sep="\t",stringsAsFactors = FALSE)
edges <- read.table(system.file("extdata", "edges.tsv", package = "iconr"),
                    sep="\t",stringsAsFactors = FALSE)
# Generate list of graphs from the three data.frames
lgrph <- list_dec(imgs, nodes, edges)

# Generate list of all graph comparisons depending on the node "type" variable
g.compar <- list_compar(lgrph, nd.var = "type")

length(g.compar)
## Ten pairwise comparisons

# Inspect the second pairwise comparison of the list
g.compar[[2]]
## The two compared graphs with the name of the comparison variable

# Inspecting nodes:
igraph::as_data_frame(g.compar[[2]][[1]], "vertices")
## Vertices from the first decoration graph

igraph::as_data_frame(g.compar[[2]][[2]], "vertices")
## Vertices from the second decoration graph

# Inspecting edges:
igraph::as_data_frame(g.compar[[2]][[1]])
## Edges of the first decoration graph

igraph::as_data_frame(g.compar[[2]][[2]])
## Edges of the second decoration graph
```

list_dec *Create Decoration's Graphs and Store them in a List*

## Description

Create undirected graphs for each decoration from `nodes`, `edges` and `imgs` dataframes and store the graphs in a list. The join between these dataframes is done on the two fields `site` and `decor`. Graph names refer to `imgs$idf`.

## Usage

```
list_dec(imgs,
         nodes,
         edges)
```

## Arguments

| | |
|---|---|
| `imgs` | Dataframe of decorations |
| `nodes` | Dataframe of nodes |
| `edges` | Dataframe of edges |

## Value

A list of `igraph` graphs.

## See Also

`graph_from_data_frame`

## Examples

```
# Read imgs, nodes and edges dataframes
imgs <- read.table(system.file("extdata", "imgs.csv", package = "iconr"),
                   sep=";", stringsAsFactors = FALSE)
nodes <- read.table(system.file("extdata", "nodes.csv", package = "iconr"),
                    sep=";", stringsAsFactors = FALSE)
edges <- read.table(system.file("extdata", "edges.csv", package = "iconr"),
                    sep=";", stringsAsFactors = FALSE)
# Create the list of graphs
lgrph <- list_dec(imgs, nodes, edges)

# Get the first graph
g <- lgrph[[1]]
g

# Graph name
g$name
```

```
# Graph label
g$lbl

# Graph number of nodes
igraph::gorder(g)

# Graph number of edges
igraph::gsize(g)
```

---

named_elements *Textual Notation of Graph Elements*

---

## Description

Create a textual notation for nodes or edges.

## Usage

```
named_elements(grph,
               focus = "edges",
               nd.var = "type",
               disamb.marker = "#")
```

## Arguments

| | |
|---|---|
| grph | A decoration graph (object of class igraph). |
| focus | Textual notation of edges (focus = "edges") or nodes (focus = "nodes"). By default focus = "edges". |
| nd.var | The attribute of the graph nodes containing the node variable (ie, field) for the textual annotation. By default nd.var = "type". |
| disamb.marker | Marker used to disambiguated repeated elements. By default disamb.marker = "#". |

## Details

Edges of type '=' (*normal* edges) are **undirected**, so that the order of their nodes is irrelevant and they are presented in alphabetical order. Conversely, edges of types '+' (*attribute* edges) and '>' (*diachronic* edges) are **directed**, so that the given order of nodes is preserved.

Repeated node or edge names are disambiguated by appending the symbol disamb.marker ('#' by default) at the end of the second appearance (suffix). Subsequent appearances are marked by additional disamb.markers.

## Value

A character vector of named nodes or edges.

## See Also

list_compar, same_elements

## Examples

```
# Read data
imgs <- read.table(system.file("extdata", "imgs.tsv", package = "iconr"),
                   sep="\t", stringsAsFactors = FALSE)
nodes <- read.table(system.file("extdata", "nodes.tsv", package = "iconr"),
                    sep="\t", stringsAsFactors = FALSE)
edges <- read.table(system.file("extdata", "edges.tsv", package = "iconr"),
                    sep="\t", stringsAsFactors = FALSE)

# Generate list of graphs from the three data.frames
lgrph <- list_dec(imgs, nodes, edges)

# Textual notation of disambiguated edges
named_elements(lgrph[[2]], focus = "edges", nd.var="type")

# Textual notation of disambiguated nodes
named_elements(lgrph[[2]], focus = "nodes", nd.var="type")
```

---

# plot_compar *Plot and Save Comparison Figures Between Pairs of Graphs*

---

## Description

Given a list of pairwise graph comparisons, the function plots any given subset selected by graph name, displaying side-by-side pairs of graphs and highlighting common nodes or common edges with a choice of several graphical parameters.

## Usage

```
plot_compar(listg, dec2comp = NULL, focus = "nodes",
            dir = getwd(),
            nd.color = c("orange", "red"), nd.size = c(0.5, 1),
            ed.color = c("orange", "red"), ed.width = c(1, 2),
            lbl.size = 0.5,
            dir.out = dir, out.file.name = NULL,
            img.format = NULL, res = 300)
```

## Arguments

| | |
|---|---|
| listg | A list of graph pairwise comparisons as returned by list_compar. |
| dec2comp | A vector with the names of the graphs for which comparisons are to be plotted. The user can select to plot all pairwise combinations (by default), all combinations of a subset, or a single pair. |

`focus` Either `"nodes"` (default) or `"edges"`. It selects the type of comparison to be plotted, highlighting common nodes or common edges, respectively.

`dir` Data folder including the decoration images. By default the working directory.

`nd.color, nd.size, ed.color, ed.width`
Graphical parameters for color and size/widths of nodes and edges. Each of them is a vector with two values for different and common nodes/edges, respectively. If only one value is provided, this unique value is taken for both different and common elements. Labels are displayed with the same color as common nodes. For `focus = "nodes"` all edges are plotted with the first value of `ed.color` and `ed.width`.

`lbl.size` Graphical parameter for the size of the labels with the node names. The default is 0.5.

`dir.out` Folder for the output image. By default, it coincides with the input `dir`.

`out.file.name` Name of the output image, including path from current directory and extension. By default the name is automatically generated including `site`, `decor`, `nd.var`, and the extension from `img.format`.

If set, `out.file.name` overrides `dir.out` and `img.format`.

`img.format, res`
Format and resolution of the saved images. The handled formats are `"png"`, `"bmp"`, `"tiff"/"tif"`, `"jpeg"/"jpg"`, and `"pdf"`. The default resolution is 300 (ppi). The resolution does not applies to format pdf.

if `img.format=NULL` (default), the plot is sent to the active device.

## Details

To highlight common elements between a list of graphs, the user can focus on nodes (`focus = "nodes"`) or edges (`focus = "edges"`). As stated in the function `list_compar`, for a given comparison variable (eg. `nd.var="type"`) if there is multiple nodes/edges with the same value, it is considered to count for as many coincidences as the smaller multiplicity.

`img.format=NULL` (plot to the active device) does not make sense for more than one comparison.

## Value

Generates graph decoration images, for pairwise comparisons between two or more decorations, comparing graphs elements (nodes or edges).

If `img.format=NULL`, the plot is sent to the active device and no value is returned.

If `img.format= "png"` or `"bmp"` or `"tiff"/"tif"` or `"jpeg"/"jpg"` or `"pdf"`, the return value is a character vector with the dir/name of every saved image in the indicated format.

## See Also

`list_compar` `plot_dec_grph`

## Examples

```
# Read data
imgs <- read.table(system.file("extdata", "imgs.tsv", package = "iconr"),
                   sep="\t",stringsAsFactors = FALSE)
nodes <- read.table(system.file("extdata", "nodes.tsv", package = "iconr"),
                    sep="\t",stringsAsFactors = FALSE)
edges <- read.table(system.file("extdata", "edges.tsv", package = "iconr"),
                    sep="\t",stringsAsFactors = FALSE)

# Generate list of graphs from the three dataframes
lgrph <- list_dec(imgs, nodes, edges)

# Generate all pairwise comparisons of the graphs with respect to nodes "type"
g.compar <- list_compar(lgrph, nd.var="type")

# Generate the image showing the comparison on common nodes of graphs
# '1' and '4', save it in png format, and return its path.
dataDir <- system.file("extdata", package = "iconr")
outDir <- tempdir()
plot_compar(g.compar, c(1,4), focus = "nodes",
            dir = dataDir,
            dir.out = outDir,
            img.format = "png")

# Generate the image showing the comparison on common edges of all pairwise
# combinations of graphs '1','3', and '4', save them in pdf format, and return
# their path.
# Plot nodes involved in non-common edges in orange and
# nodes involved in common edges and the corresponding labels in brown.
plot_compar(g.compar, c(1, 3, 4), focus = "edges",
            dir = dataDir,
            nd.color = c("orange", "brown"),
            dir.out = outDir,
            img.format = "pdf")

# Save the png image showing the comparison on common nodes of graphs
# '1' and '4'.
# Then read and plot the image.
img.filename <- plot_compar(g.compar, c(1, 4), focus = "nodes",
                            dir = dataDir,
                            dir.out = outDir,
                            img.format = "png")
plot(magick::image_read(img.filename))

# Plot directly on the active device (default) the comparison on common nodes
# of graphs '1' and '4'.
plot_compar(g.compar, c(1, 4), focus = "nodes",
            dir = dataDir)
```

---

plot_dec_grph *Plot a Graph on a Decoration*

---

## Description

Plot with nodes only, edges only, or both (geometric graph) over a decoration image.

## Usage

```
plot_dec_grph(nodes = NULL,
              edges = NULL,
              imgs,
              site,
              decor,
              dir = getwd(),
              nd.var = 'id',
              nd.color = 'orange',
              nd.size = 0.5,
              lbl.color = 'black',
              lbl.size = 0.5,
              ed.color = c("orange", "blue"),
              ed.lwd = 1,
              dir.out = dir,
              out.file.name = NULL,
              img.format = NULL,
              res = 300)
```

## Arguments

| | |
|---|---|
| `nodes` | Dataframe of nodes |
| `edges` | Dataframe of edges |
| `imgs` | Dataframe of decorations |
| `site` | Name of the site |
| `decor` | Name of the decoration |
| `dir` | Data folder including the decoration images. By default the working directory. |
| `nd.var` | Field name in the nodes data frame to be displayed as node labels. By default the identifier `nodes$id`. |
| `nd.color, nd.size, lbl.color, lbl.size, ed.color, ed.lwd` | Graphical parameters for color and size/widths of nodes, edges, and labels. `ed.color` is a vector with two values (the second value is used for diachronic edges). |
| `dir.out` | Folder for the output image. By default, it coincides with the input `dir`. |
| `out.file.name` | Name of the output image, including path from current directory and extension. By default the name is automatically generated including `site`, `decor`, `nd.var`, and the extension from `img.format`.<br>If set, `out.file.name` overrides `dir.out` and `img.format`. |
| `img.format, res` | Format and resolution of the saved images. The handled formats are `"png"`, `"bmp"`, `"tiff"/"tif"`, `"jpeg"/"jpg"`, and `"pdf"`. The default resolution is 300 (ppi). The resolution does not applies to format pdf.<br>if `img.format=NULL` (default), the plot is sent to the active device. |

## Details

Plot **nodes only** (if `edges = NULL`), **edges only** (if `nodes = NULL`), or both (graph) over a decoration image.

## Value

Generates graph decoration images with nodes, edges, or both, overlapping the decoration image.

If `img.format=NULL`, the plot is sent to the active device and no value is returned.

If `img.format=` `"png"` or `"bmp"` or `"tiff"`/`"tif"` or `"jpeg"`/`"jpg"` or `"pdf"`, the return value is a character vector with the dir/name of the saved image in the indicated format.

## Examples

```
## Set data folder
dataDir <- system.file("extdata", package = "iconr")
## Decoration to be plotted
site <- "Brozas"
decor <- "Brozas"
## Read nodes, edges, and decorations
nds.df <- read_nds(site, decor, dataDir)
eds.df <- read_eds(site, decor, dataDir)
imgs <- read.table(paste0(dataDir, "/imgs.tsv"),
                   sep="\t", stringsAsFactors = FALSE)

## Plot 'Brozas' nodes and edges on the active device
## with node variable "type" as labels
plot_dec_grph(nds.df, eds.df, imgs,
              site, decor,
              dir = dataDir,
              lbl.size = 0.4,
              nd.var = "type")

## Save only edges of 'Brozas' with bigger widths and in image format jpg.
outDir <- tempdir()
img.filename <- plot_dec_grph(nodes = NULL, eds.df, imgs,
                              site, decor,
                              dir = dataDir,
                              ed.lwd = 2,
                              dir.out = outDir,
                              img.format = "jpg")
## Then read and plot the image.
a.dec <- magick::image_read(img.filename)

## Inspect the output image
magick::image_info(a.dec)

## Plot the output image
plot(a.dec)
```

read_eds *Read Edges of a Decoration*

## Description

Read edges' information from a file including all edges and extract edges of one decoration. Accepted formats are tab separated values ('tsv'), semicolon separated values ('csv'), or shapefile ('shp').

## Usage

```
read_eds(site,
         decor,
         dir = getwd(),
         edges = "edges",
         nodes = "nodes",
         format = "tsv")
```

## Arguments

| | |
|---|---|
| site | Name of the site. |
| decor | Name of the decoration. |
| dir | Path to the working folder, by default it is the working directory. |
| edges | Name of the edges file (a dataframe or a shapefile). |
| nodes | Name of the nodes file (a dataframe or a shapefile). |
| format | File extension indicating a file format from 'tsv' (tab separated values), 'csv' (semicolon separated values) or 'shp' (shapefile). For 'tsv' and 'csv' the coordinates of the edges will be calculated from the same decoration's node dataframe. |

## Details

Subset the dataframe of edges depending on 'site' and 'decor'.

## Value

Dataframe of graph edges, including at least the columns "site", "decor", "a", "b", "xa", "ya", "xb", "yb", with values for each edge (row).

## Examples

```
# Set data folder
dataDir <- system.file("extdata", package = "iconr")

# Read .tsv file
eds.df <- read_eds(site = "Cerro Muriano", decor = "Cerro Muriano 1",
                   dir = dataDir, edges = "edges", format = "tsv")
eds.df
```

```
## Dataframe of edges

# Read shapefile
eds.df <- read_eds(site = "Cerro Muriano", decor = "Cerro Muriano 1",
                   dir = dataDir, edges = "edges", format = "shp")
eds.df
## Dataframe of edges
```

---

`read_nds` *Read Nodes of a Decoration*

---

## Description

Read nodes' information from a file including all nodes and extract nodes of one decoration. Accepted formats are tab separated values ('tsv'), semicolon separated values ('csv'), or shapefile ('shp').

## Usage

```
read_nds(site,
        decor,
        dir = getwd(),
        nodes = "nodes",
        format = "tsv")
```

## Arguments

| | |
|---|---|
| `site` | Name of the site |
| `decor` | Name of the decoration |
| `dir` | Path to the working folder, by default it is the working directory |
| `nodes` | Name of the nodes file (a dataframe or a shapefile) |
| `format` | File extension indicating a file format from 'tsv' (tab separated values), 'csv' (semicolon separated values) or 'shp' (shapefile). For 'tsv' and 'csv' the files must include node coordinates (`nodes$x`, `nodes$y`). |

## Value

Dataframe of graph nodes, including at least the columns "site", "decor", "id", "x", "y", with values for each node (row).

## Examples

```
# Set data folder
dataDir <- system.file("extdata", package = "iconr")

# Read dataframe of nodes
nds.df <- read_nds(site = "Cerro Muriano", decor = "Cerro Muriano 1",
                   dir = dataDir, format = "tsv")
nds.df
## Dataframe of nodes

# Read shapefile of nodes
nds.df <- read_nds(site = "Cerro Muriano", decor = "Cerro Muriano 1",
                   dir = dataDir, format = "shp")
nds.df
## Dataframe of nodes
```

---

# same_elements *Number of Equal Elements Between Each Decoration Pair*

---

## Description

Create the (symmetric) dataframe with the count of **common nodes** or **common edges** (see list_compar for comparison criteria) for each pair of decorations (graphs) from a list. The diagonal of the symmetric dataframe is filled with counts of nodes/edges for each decoration.

## Usage

```
same_elements(lgrph, nd.var = "type",
              focus = "nodes")
```

## Arguments

| | |
|---|---|
| lgrph | A list of any number of graphs to be pairwise compared. The list can be typically obtained with the function list_dec |
| nd.var | An attribute of the graph vertices containing the node variable (ie, field) on which the comparison will be done. By default nd.var = "type". |
| focus | Either "nodes" (default) or "edges" to select the type of elements to be compared for the count. |

## Value

A symmetric matrix with the counts of the pairwise coincidences of nodes or edges. The matrix has as row and column names the names of the corresponding graphs in the input list.

## See Also

list_dec, list_compar, plot_compar

## Examples

```
# read imgs, nodes and edges dataframes
imgs <- read.table(system.file("extdata", "imgs.tsv", package = "iconr"),
                   sep="\t",stringsAsFactors = FALSE)
nodes <- read.table(system.file("extdata", "nodes.tsv", package = "iconr"),
                    sep="\t",stringsAsFactors = FALSE)
edges <- read.table(system.file("extdata", "edges.tsv", package = "iconr"),
                    sep="\t",stringsAsFactors = FALSE)
lgrph <- list_dec(imgs,nodes,edges)

# Counting same nodes
df.same_nodes <- same_elements(lgrph, nd.var = "type",
                               focus = "nodes")
df.same_nodes
## a symmetric matrix of nodes comparisons

# same edges
df.same_edges <- same_elements(lgrph, nd.var = "type",
                               focus = "edges")
df.same_edges
## a symmetric matrix of edges comparisons
```

---

`side_plot` *Plot Two Figures Side-by-Side Identifying Common Elements*

---

## Description

Plot two decoration graphs side-by-side identifying common nodes and common edges. This function is called by the function `plot_compar`.

## Usage

```
side_plot(grph, dir, nd.var, focus = "nodes",
          nd.color = c("orange", "red"),
          nd.size = c(0.5, 1),
          ed.color = c("orange", "red"),
          ed.width = c(1, 2),
          lbl.size = 0.5)
```

## Arguments

| | |
|---|---|
| `grph` | List of two or more 'igraph' graphs created with the `list_compar` function. |
| `dir` | Working directory which contains the imgs, nodes, edges dataframes and the decoration images. |
| `nd.var` | Field of nodes on which the comparison will be done. |
| `focus` | Focus on nodes or on edges, by default `focus = "nodes"`. |

`nd.color, nd.size, ed.color, ed.width`
: Graphical parameters for the nodes and edges. The **different** nodes/edges will be displayed with the first values of the vectors (eg, "orange") while the **common** nodes/edges will be displayed with the second values of the vectors (eg, "red").

`lbl.size`
: Size of the labels

## Value

No return value, group images side-by-side

## See Also

`plot_compar`

# Index