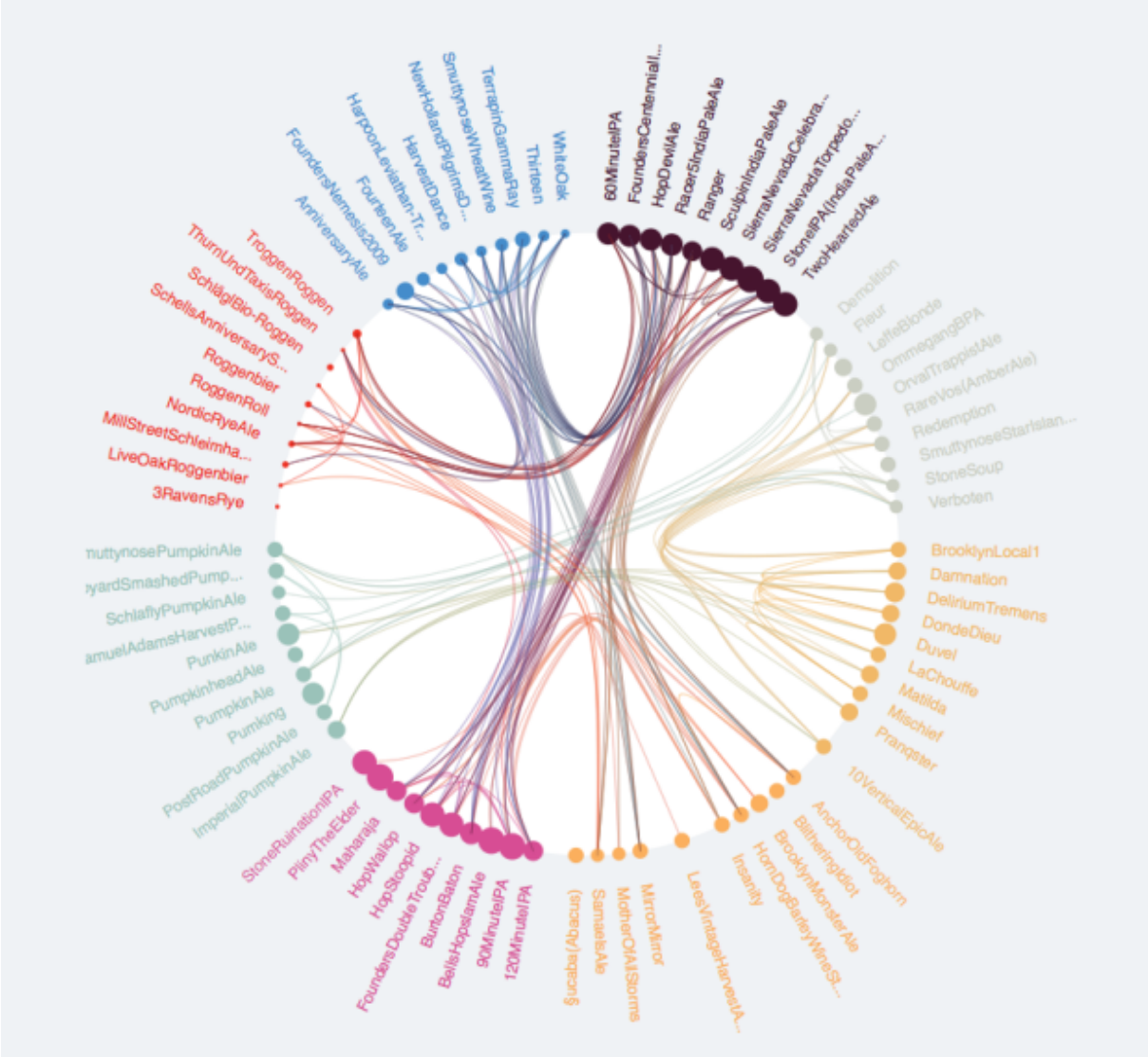
Network chord diagram woes in R

I have some data similar to the `data.frame d` as follows.

```
d <- structure(list(ID = c("KP1009", "GP3040", "KP1757", "GP2243",
                           "KP682", "KP1789", "KP1933", "KP1662", "KP1718", "GP3339",
"GP4007",
                           "GP3398", "GP6720", "KP808", "KP1154", "KP748", "GP4263",
"GP1132",
                           "GP5881", "GP6291", "KP1004", "KP1998", "GP4123", "GP5930",
"KP1070",
                           "KP905", "KP579", "KP1100", "KP587", "GP913", "GP4864",
"KP1513",
                           "GP5979", "KP730", "KP1412", "KP615", "KP1315", "KP993",
"GP1521",
                           "KP1034", "KP651", "GP2876", "GP4715", "GP5056", "GP555",
"GP408",
                           "GP4217", "GP641"),
                    Type = c("B", "A", "B", "A", "B", "B", "B",
                             "B", "B", "A", "A", "A", "A", "B", "B", "B", "A", "A", "A",
"A",
                             "B", "B", "A", "A", "B", "B", "B", "B", "B", "A", "A", "B",
"A",
                             "B", "B", "B", "B", "B", "A", "B", "B", "A", "A", "A", "A",
"A",
                             "A", "A"),
                    Set = c(15L, 1L, 10L, 21L, 5L, 9L, 12L, 15L, 16L,
                            19L, 22L, 3L, 12L, 22L, 15L, 25L, 10L, 25L, 12L, 3L, 10L, 8L,
                            8L, 20L, 20L, 19L, 25L, 15L, 6L, 21L, 9L, 5L, 24L, 9L, 20L,
5L,
                            2L, 2L, 11L, 9L, 16L, 10L, 21L, 4L, 1L, 8L, 5L, 11L), Loc =
c(3L,

2L, 3L, 1L, 3L, 3L, 3L, 1L, 2L, 1L, 3L, 1L, 1L, 2L, 2L, 1L, 3L,

2L, 2L, 2L, 3L, 2L, 3L, 2L, 1L, 3L, 3L, 3L, 2L, 3L, 1L, 3L, 3L,

1L, 3L, 2L, 3L, 1L, 1L, 1L, 2L, 3L, 3L, 3L, 2L, 2L, 3L, 3L)),
               .Names = c("ID", "Type", "Set", "Loc"), class = "data.frame",
               row.names = c(NA, -48L))
```

I want to explore the relationships between members of `d$ID` using a chord diagram similar to the one below.



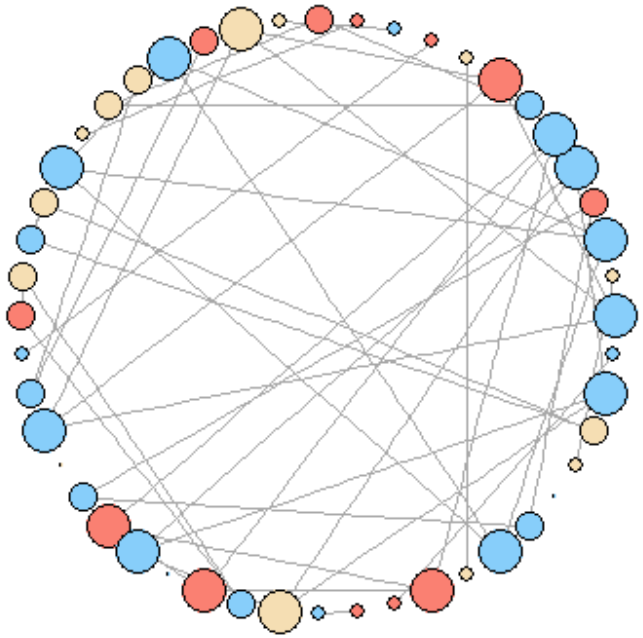
It seesms there ar several options to do so in `R`. (Chord diagram in R).

In my data the relationships are according to `d$Set` (not directional) and the grouping is according to `d$Loc`. The following are my attempts to map theser relationships as a chord diagram.

Attempt 1: Using `igraph`

I have tried `igraph` as follows with node size according to degree.

```
# Get vertex relationships
sets <- unique(d$Set[duplicated(d$Set)])
rel <-  vector("list", length(sets))
for (i in 1:length(sets)) {
  rel[[i]] <- as.data.frame(t(combn(subset(d, d$Set ==sets[i])$ID, 2)))
}
library(data.table)
rel <- rbindlist(rel)

# Get the graph
g <- graph.data.frame(rel, directed=F, vertices=d)
clr <- as.factor(V(g)$Loc)
levels(clr) <- c("salmon", "wheat", "lightskyblue")
V(g)$color <- as.character(clr)

# Plot
plot(g, layout = layout.circle, vertex.size=degree(g)*5, vertex.label=NA)
```

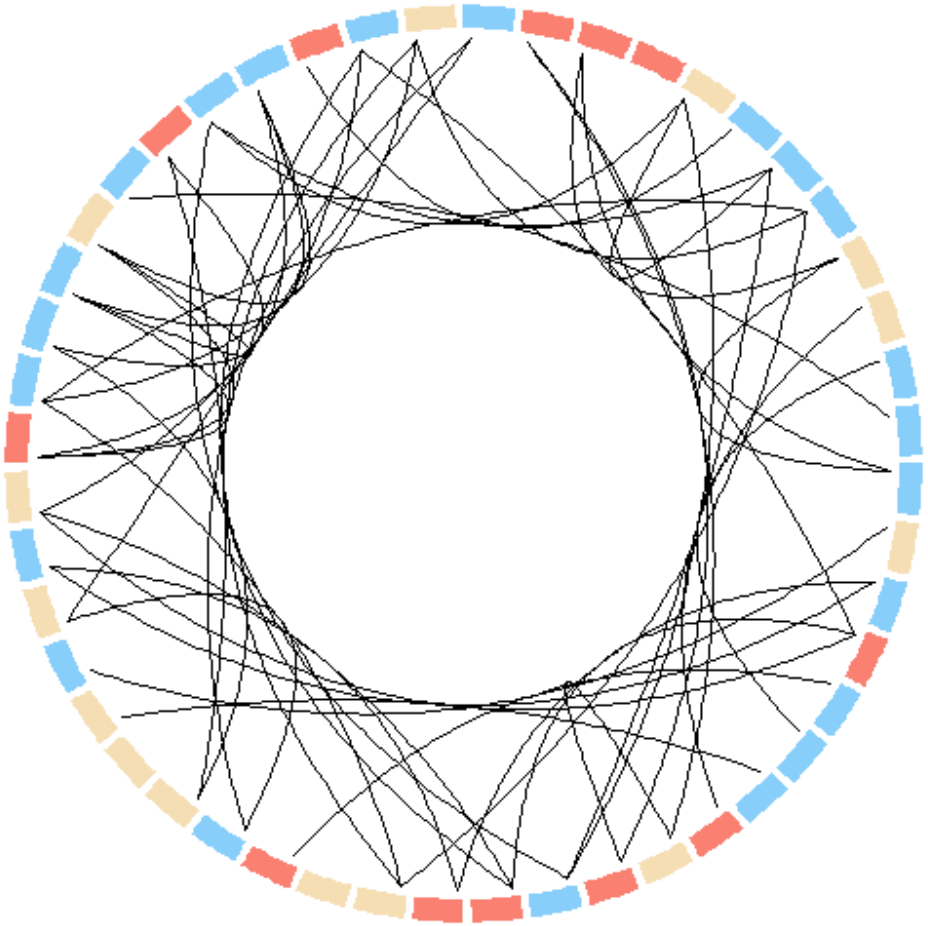


How to modify the plot to look like the first figure? It seems that there are no options to modify `igraph` `layout.circle` .

Attempt 2: Using `Circlize`

It seems smoother bezier curves and grouping are possible in the `R` package `circlize`. But here I am not able to group the nodes as well as adjust their size according to degree as they are plotted as sectors.

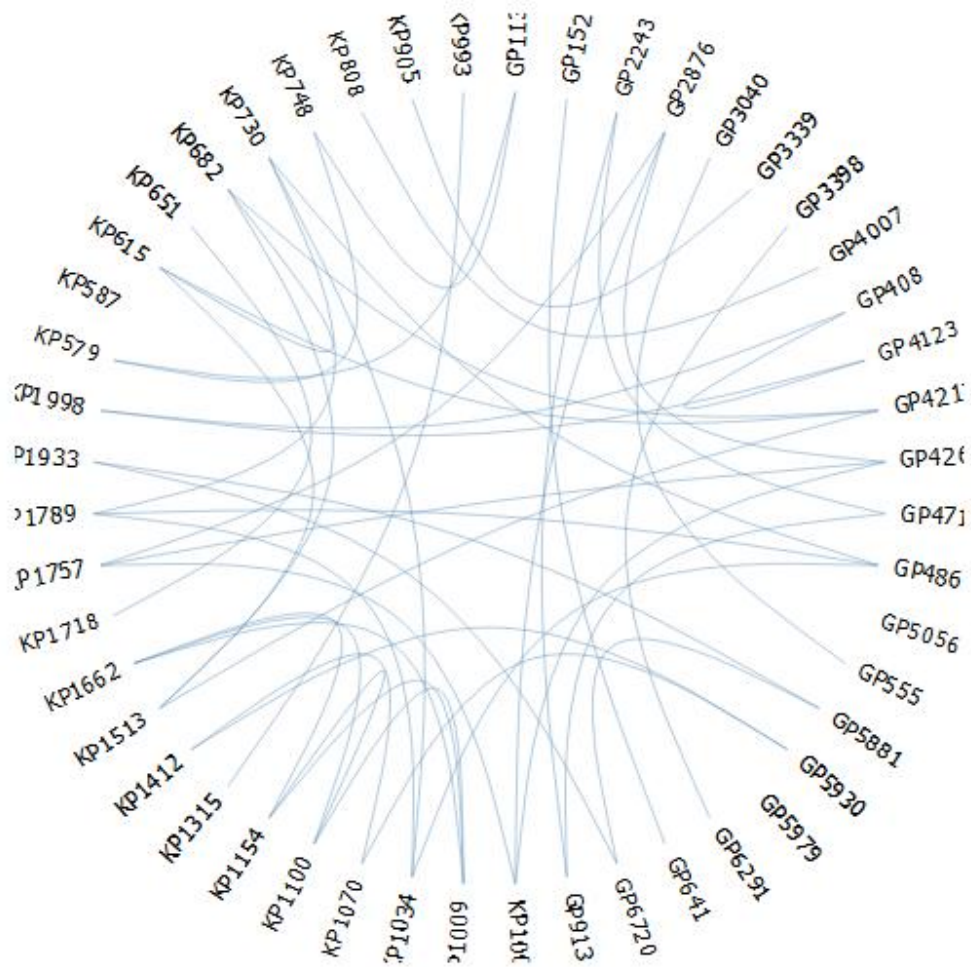
```
par(mar = c(1, 1, 1, 1), lwd = 0.1, cex = 0.7)
circos.initialize(factors = as.factor(d$ID), xlim = c(0, 10))
circos.trackPlotRegion(factors = as.factor(d$ID), ylim = c(0, 0.5), bg.col = V(g)$color,
                       bg.border = NA, track.height = 0.05)
for(i in 1:nrow(rel)) {
  circos.link(rel[i,1], 0, rel[i,2],0, h = 0.4)

}
```



Here however there are no options to modify the nodes. In fact they can be only plotted as sectors? In this case is there any way to modify the sectors into circular nodes of size according to the degree?

Attempt 3: Using `edgebundleR` (https://github.com/garthtarr/edgebundleR)

```
require(edgebundleR)
edgebundle(g,tension = 0.1,cutoff = 0.5, fontsize = 18,padding=40)
```



It seems here there are limited options to modify the aesthetics.

r data-visualization igraph graph-visualization circos

edited Jun 9 '15 at 4:24

asked Jun 8 '15 at 12:02

Crops
1,307 10 28

2 What about christophergandrud.github.io/d3Network ? – Roman Luštrik Jun 8 '15 at 12:46

1 You can group the variables by ordering the adjacency matrix and add some curve to the edges with edge.curve argument. Apologies code dump: `m <- tcrossprod(table(d[c(1,3)])) ; grp <- d[order(d$ID), "Loc"] ; m2 <- m[order(grp), order(grp)] ; diag(m2) <- 0 ; g <- graph.adjacency(m2, mode="undirected"); clr <- as.factor(sort(grp)); levels(clr) <- c("salmon", "wheat", "lightskyblue"); V(g)$color <- as.character(clr); par(mar=rep(0,4)); plot(g, layout = layout.circle, vertex.size=degree(g)*5, vertex.label=NA, edge.curved=seq(-0.5, 0.5, length = ecount(g)))` – user20650 Jun 8 '15 at 14:25

1 Hi Crops; yup almost there, but not quite. I cant post an answer as the question has been closed as a dup (hence code dump above). – user20650 Jun 8 '15 at 14:53

1 @RomanLuštrik `networkD3` (christophergandrud.github.io/networkD3) looks great. But currently this `R` interface supports only *Force directed networks*, *Sankey diagrams* and *Reingold-Tilford Tree graphs*. Not circular layout – Crops Jun 8 '15 at 14:55

2 I understand you are using R, but why not to try circos (circos.ca)? One alternative to use R + circos idea is bioconductor.org/packages/release/bioc/html/OmicCircos.html. – AndreiR Aug 9 '15 at 23:57

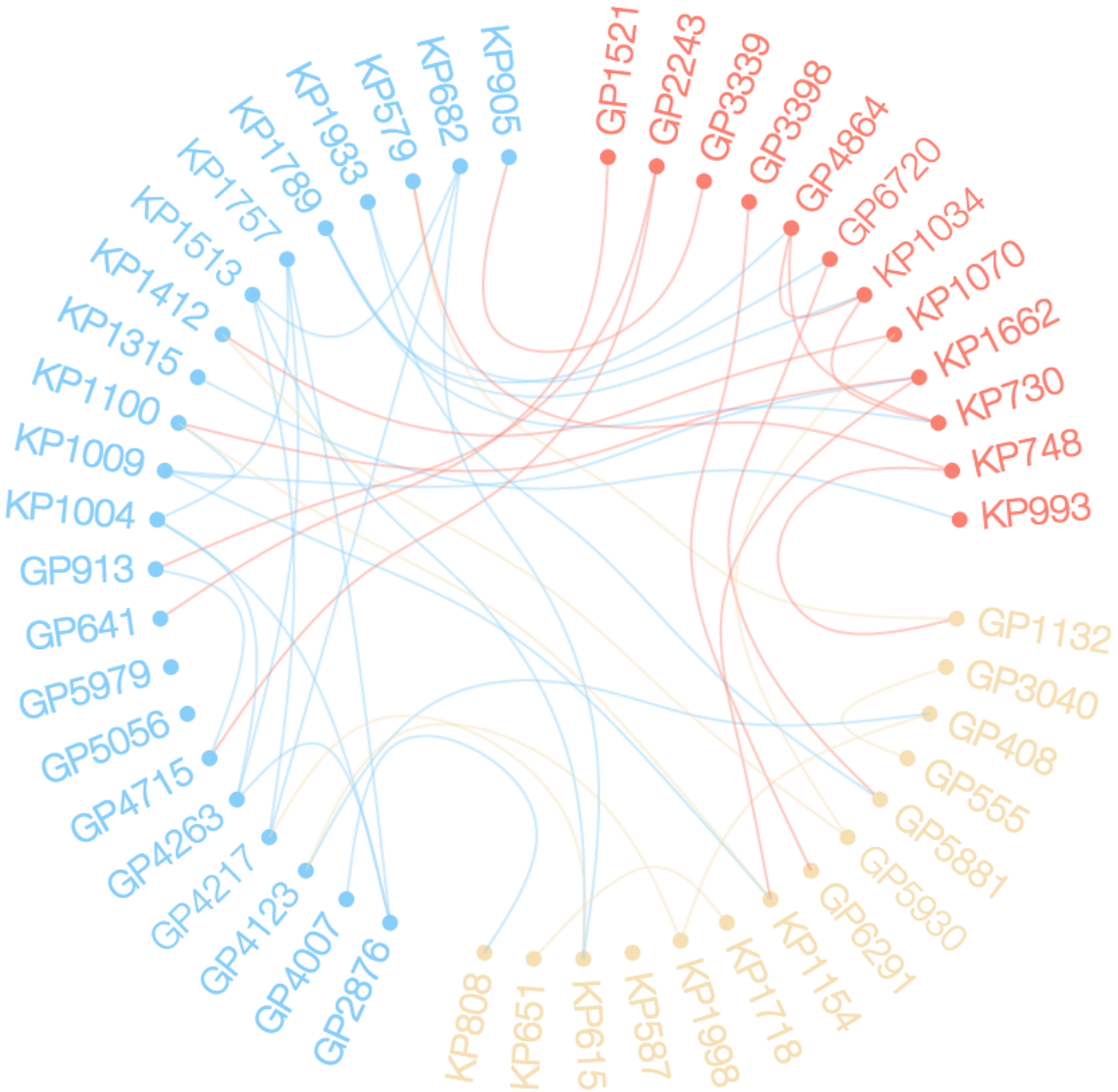
2 Answers

I made a bunch of changes to `edgebundleR` . These are now in the main repo. The following code should get you close to the desired result. live example

```
# devtools::install_github("garthtarr/edgebundleR")

library(edgebundleR)
library(igraph)
library(data.table)

d <- structure(list(ID = c("KP1009", "GP3040", "KP1757", "GP2243",
                           "KP682", "KP1789", "KP1933", "KP1662", "KP1718", "GP3339",
"GP4007",
                           "GP3398", "GP6720", "KP808", "KP1154", "KP748", "GP4263",
"GP1132",
                           "GP5881", "GP6291", "KP1004", "KP1998", "GP4123", "GP5930",
"KP1070",
                           "KP905", "KP579", "KP1100", "KP587", "GP913", "GP4864",
"KP1513",
                           "GP5979", "KP730", "KP1412", "KP615", "KP1315", "KP993",
"GP1521",
```

```
                            "KP1034", "KP651", "GP2876", "GP4715", "GP5056", "GP555",
"GP408",
                            "GP4217", "GP641"),
                    Type = c("B", "A", "B", "A", "B", "B", "B",
                             "B", "B", "A", "A", "A", "A", "B", "B", "B", "A", "A", "A",
"A",
                             "B", "B", "A", "A", "B", "B", "B", "B", "B", "A", "A", "B",
"A",
                             "B", "B", "B", "B", "B", "A", "B", "B", "A", "A", "A", "A",
"A",
                             "A", "A"),
                    Set = c(15L, 1L, 10L, 21L, 5L, 9L, 12L, 15L, 16L,
                            19L, 22L, 3L, 12L, 22L, 15L, 25L, 10L, 25L, 12L, 3L, 10L, 8L,
                            8L, 20L, 20L, 19L, 25L, 15L, 6L, 21L, 9L, 5L, 24L, 9L, 20L,
5L,
                            2L, 2L, 11L, 9L, 16L, 10L, 21L, 4L, 1L, 8L, 5L, 11L), Loc =
c(3L,

2L, 3L, 1L, 3L, 3L, 3L, 1L, 2L, 1L, 3L, 1L, 1L, 2L, 2L, 1L, 3L,

2L, 2L, 2L, 3L, 2L, 3L, 2L, 1L, 3L, 3L, 3L, 2L, 3L, 1L, 3L, 3L,

1L, 3L, 2L, 3L, 1L, 1L, 1L, 2L, 3L, 3L, 3L, 2L, 2L, 3L, 3L)),
               .Names = c("ID", "Type", "Set", "Loc"), class = "data.frame",
               row.names = c(NA, -48L))

# let's add Loc to our ID
d$key <- d$ID
d$ID <- paste0(d$Loc,".",d$ID)

# Get vertex relationships
sets <- unique(d$Set[duplicated(d$Set)])
rel <-  vector("list", length(sets))
for (i in 1:length(sets)) {
  rel[[i]] <- as.data.frame(t(combn(subset(d, d$Set ==sets[i])$ID, 2)))
}

rel <- rbindlist(rel)

# Get the graph
g <- graph.data.frame(rel, directed=F, vertices=d)
clr <- as.factor(V(g)$Loc)
levels(clr) <- c("salmon", "wheat", "lightskyblue")
V(g)$color <- as.character(clr)
V(g)$size = degree(g)*5
# Plot
plot(g, layout = layout.circle, vertex.label=NA)
```

```
edgebundle( g )->eb

eb
```



edited May 18 at 11:33

answered Aug 27 '15 at 23:21

timelyportfolio

3,917 7 17

How do you change the color of the edges? – rrs Feb 5 at 15:38

These three lines clr <- as.factor(V(g)$Loc) levels(clr) <- c("salmon", "wheat", "lightskyblue") V(g)$color <- as.character(clr) is how colors were defined. There are other methods. For instance just doing `V(g)$color <- "red"` would make everything red. – timelyportfolio Feb 5 at 19:11

this doesn't work if you want to color each edge according to some other parameter. E.g., in igraph you can color the edges via `E(g)$color` but the `edgebundleR` package is only coloring edges using the source node's color. so all outgoing edges have to be the same. – rrs Feb 5 at 19:31

I see what should have been obvious. Sorry it took me a while to understand. These lines github.com/garthtarr/edgebundleR/blob/master/inst/htmlwidgets/… demonstrate the problem you mention. Let me play a bit and try to come up with an answer. – timelyportfolio Feb 5 at 20:00

1 After refamiliarizing myself with the code, I realize this will require a rewrite of much of the code or will require a hack. I'll post the hack in an answer below. – timelyportfolio Feb 5 at 20:52

|

I hate to add another answer for a different problem, but I don't know of any way to handle the additional question posed in the comment. The comment asked how might we color the edges. Generally, the response would be easy, but in this case, the answer requires a rewrite of much of the code in `edgebundleR` or requires a hack. I'll go with the hack below.

```
library(edgebundleR)
library(igraph)
library(data.table)

d <- structure(list(ID = c("KP1009", "GP3040", "KP1757", "GP2243",
                           "KP682", "KP1789", "KP1933", "KP1662", "KP1718", "GP3339",
"GP4007",
                           "GP3398", "GP6720", "KP808", "KP1154", "KP748", "GP4263",
"GP1132",
                           "GP5881", "GP6291", "KP1004", "KP1998", "GP4123", "GP5930",
"KP1070",
                           "KP905", "KP579", "KP1100", "KP587", "GP913", "GP4864",
"KP1513",
                           "GP5979", "KP730", "KP1412", "KP615", "KP1315", "KP993",
"GP1521",
                           "KP1034", "KP651", "GP2876", "GP4715", "GP5056", "GP555",
"GP408",
```

```
                          "GP4217", "GP641"),
                Type = c("B", "A", "B", "A", "B", "B", "B",
                         "B", "B", "A", "A", "A", "A", "B", "B", "B", "A", "A", "A",
"A",
                         "B", "B", "A", "A", "B", "B", "B", "B", "B", "A", "A", "B",
"A",
                         "B", "B", "B", "B", "B", "A", "B", "B", "A", "A", "A", "A",
"A",
                         "A", "A"),
                Set = c(15L, 1L, 10L, 21L, 5L, 9L, 12L, 15L, 16L,
                        19L, 22L, 3L, 12L, 22L, 15L, 25L, 10L, 25L, 12L, 3L, 10L, 8L,
                        8L, 20L, 20L, 19L, 25L, 15L, 6L, 21L, 9L, 5L, 24L, 9L, 20L,
5L,
                        2L, 2L, 11L, 9L, 16L, 10L, 21L, 4L, 1L, 8L, 5L, 11L), Loc =
c(3L,

2L, 3L, 1L, 3L, 3L, 3L, 1L, 2L, 1L, 3L, 1L, 1L, 2L, 2L, 1L, 3L,

2L, 2L, 2L, 3L, 2L, 3L, 2L, 1L, 3L, 3L, 3L, 2L, 3L, 1L, 3L, 3L,

1L, 3L, 2L, 3L, 1L, 1L, 1L, 2L, 3L, 3L, 3L, 2L, 2L, 3L, 3L)),
               .Names = c("ID", "Type", "Set", "Loc"), class = "data.frame",
               row.names = c(NA, -48L))

# let's add Loc to our ID
d$key <- d$ID
d$ID <- paste0(d$Loc,".",d$ID)

# Get vertex relationships
sets <- unique(d$Set[duplicated(d$Set)])
rel <-  vector("list", length(sets))
for (i in 1:length(sets)) {
  rel[[i]] <- as.data.frame(t(combn(subset(d, d$Set ==sets[i])$ID, 2)))
}

rel <- rbindlist(rel)

# Get the graph
g <- graph.data.frame(rel, directed=F, vertices=d)
clr <- as.factor(V(g)$Loc)
levels(clr) <- c("salmon", "wheat", "lightskyblue")
V(g)$color <- as.character(clr)

# Plot
plot(g, layout = layout.circle, vertex.size=degree(g)*5, vertex.label=NA)


edgebundle( g )->eb
```

```
eb

# temporary hack to accomplish edge coloring
# requires newest Github version of htmlwidgets
# devtools::install_github("ramnathv/htmlwidgets")

# add some imaginary colors
E(g)$color <- c("purple","green","black")[floor(runif(length(E(g)),1,4))]
# now append these edge attributes to our htmlwidget x
eb$x$edges <- jsonlite::toJSON(get.data.frame(g,what="edges"))

eb <- htmlwidgets::onRender(
  eb,
'
function(el,x){
  // loop through each of our edges supplied
  //  and change the color
  x.edges.map(function(edge){
    var source = edge.from.split(".")[1];
    var target = edge.to.split(".")[1];
    d3.select(el).select(".link.source-" + source + ".target-" + target)
      .style("stroke",edge.color);
  })
}
'
)
eb
```

answered Feb 5 at 20:54

timelyportfolio
3,917 7 17

For some reason this isn't working. I can update the edges and see `"color":"green"` for example in the big JSON goblty-gook, but when I run the code from `onRender` and below the graph ends up looking the same. – rrs Feb 5 at 21:49

any way to use `saveWidget` and post to a gist? did you install the newest `htmlwidgets` from Github? – timelyportfolio Feb 5 at 22:05

I just tried this also - after installing most recent `htmlwidgets` and `edgebundler` . This is the error message: `Error: 'onRender' is not an exported object from 'namespace:htmlwidgets'` – jalapic Feb 7 at 2:58

this functionality introduced here github.com/ramnathv/htmlwidgets/pull/172. Wonder why you are getting that error if you installed from Github. hmmmmmm.... I could add `tasks` functionality, but this should eliminate the need for this. – timelyportfolio Feb 8 at 1:11

@timelyportfolio it's mostly working. turns out you can't have spaces in your node name/ID. but the colors don't look right. it's almost as though the new colors are mixing with the original edge colors. – rrs Feb 10 at 21:26