Egonetwork example

Load the data

```
library(igraph)
library(graphlayouts)
library(ggraph)
#remotes::install_github("thomasp85/patchwork)
library(patchwork)
```

```
ego_files <- list.files("../data/egonet/",full.names = T)
egonets <- lapply(ego_files,function(x) read.graph(x,format="graphml"))
```

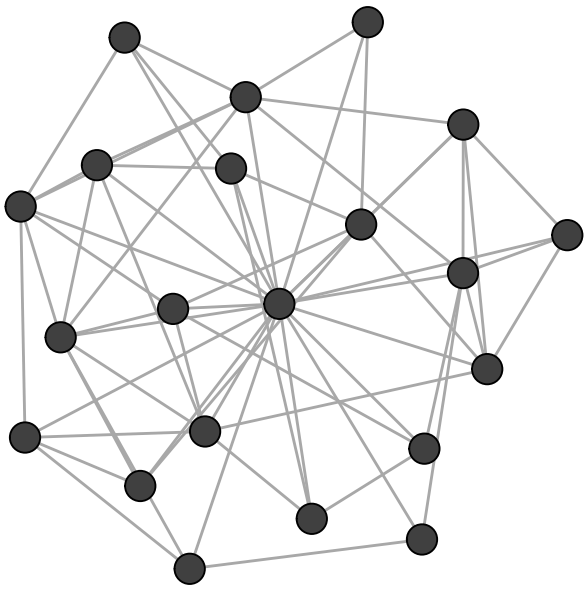
`egonets` is a list of 32 ego networks. You can access them individually as follows.

```
egonets[[1]]
```

```
## IGRAPH 196990a UNW- 21 64 --
## + attr: name (v/c), gender (v/c), age (v/c), rank (v/n), id (v/c),
## | weight (e/n)
## + edges from 196990a (vertex names):
##  [1] 1 --2   1 --8   1 --20  1 --ego 2 --5   2 --9   2 --12  2 --13
##  [9] 2 --14  2 --19  2 --ego 3 --4   3 --6   3 --11  3 --12  3 --15
## [17] 3 --18  3 --ego 4 --14  4 --ego 5 --9   5 --12  5 --13  5 --16
## [25] 5 --ego 6 --17  6 --19  6 --ego 7 --12  7 --13  7 --16  7 --ego
## [33] 8 --10  8 --11  8 --16  8 --18  8 --19  8 --ego 9 --17  9 --18
## [41] 9 --19  9 --20  9 --ego 10--12  10--ego 11--12  11--15  11--18
## [49] 11--ego 12--13  12--ego 13--19  13--20  13--ego 14--20  14--ego
## + ... omitted several edges
```

Each network has three node variables(`age`, `rank` and `gender`) and one edge variable (`weight`). You can get a quick view of the network via `qgraph`.

```
qgraph(egonets[[1]])
```

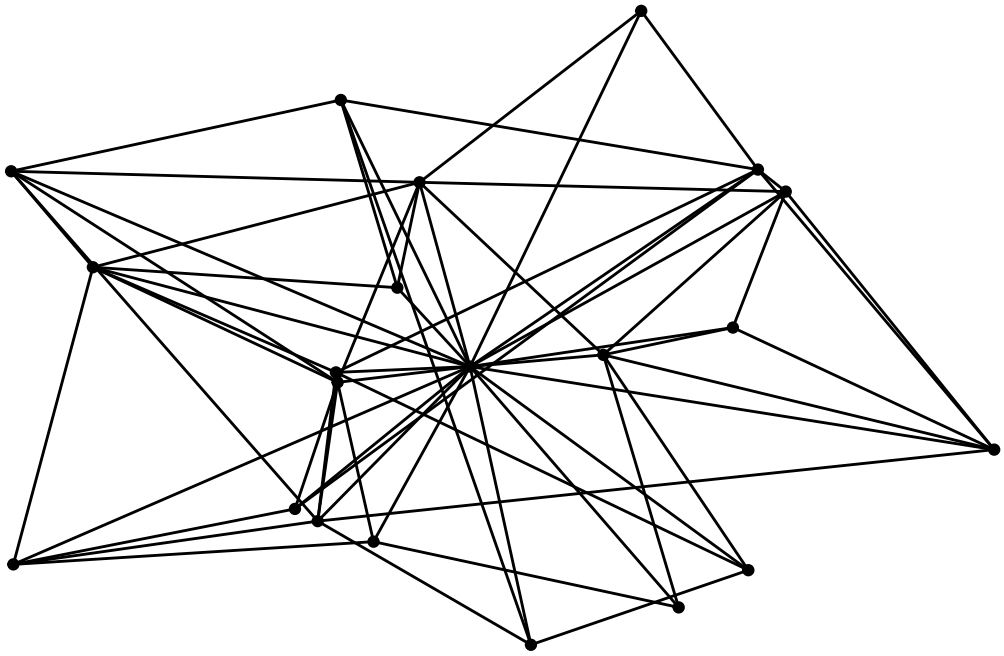


radial layout

For ego networks, a radial layout is appropriate, for instance, when there exists a node attribute that ranks the alters proximity to ego. This attribute can be used with `layout_with_centrality()` to create an ego-centric layout. Note, however, that centraity based layouts put nodes with higher values in the center. We thus need to invert the ranks given in the data (rank 1 means very close to ego).

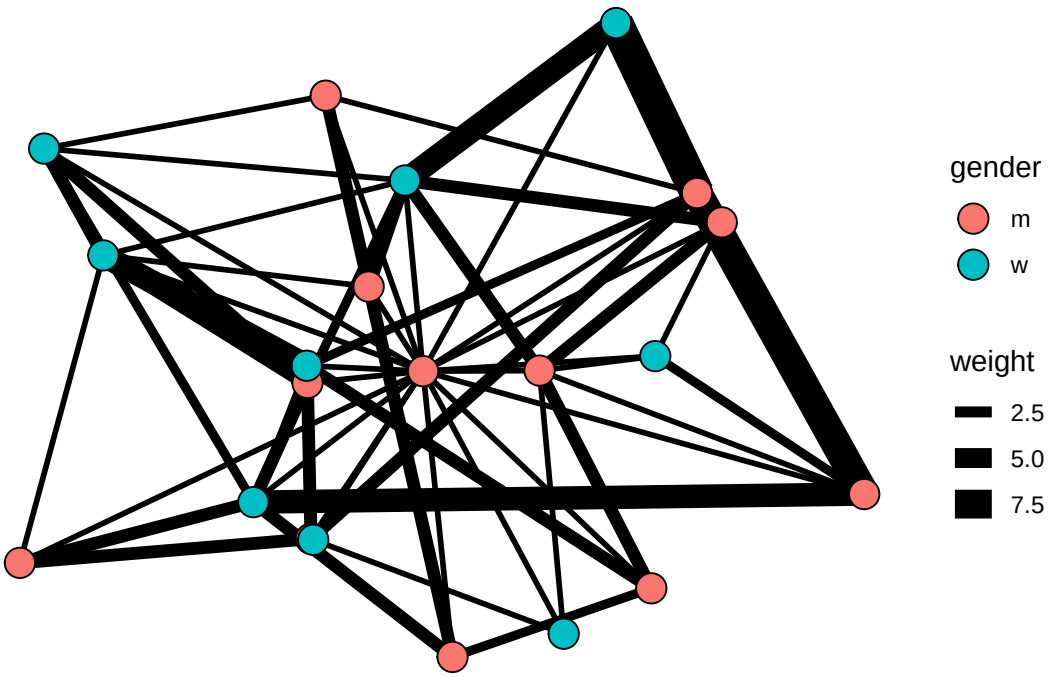
The figure below shows the basic radial layout plot without any additional informations.

```
ggraph(egonets[[1]],layout = "centrality", cent = 5-V(egonets[[1]])$rank)+
  geom_edge_link0()+
  geom_node_point()+
  theme_graph()
```



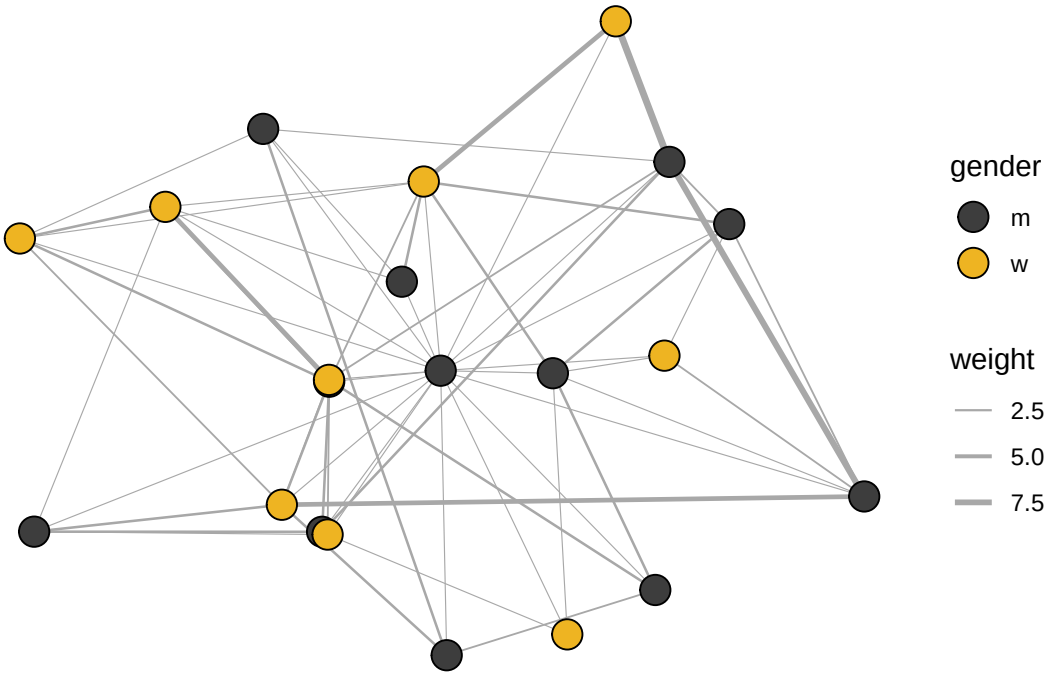
In the next step, we add some more aesthetics to the plot. We map the node fill color to gender and edge width to the weight attribute

```
ggraph(egonets[[1]],layout = "centrality", cent = 5-V(egonets[[1]])$rank)+
  geom_edge_link0(aes(edge_width = weight))+
  geom_node_point(aes(fill=gender),shape = 21,size = 5)+
  theme_graph(base_family = "Helvetica")
```



This is already quite ok, though edge are clearly way to thick. We also change the default colors of the nodes and edges in the code below.

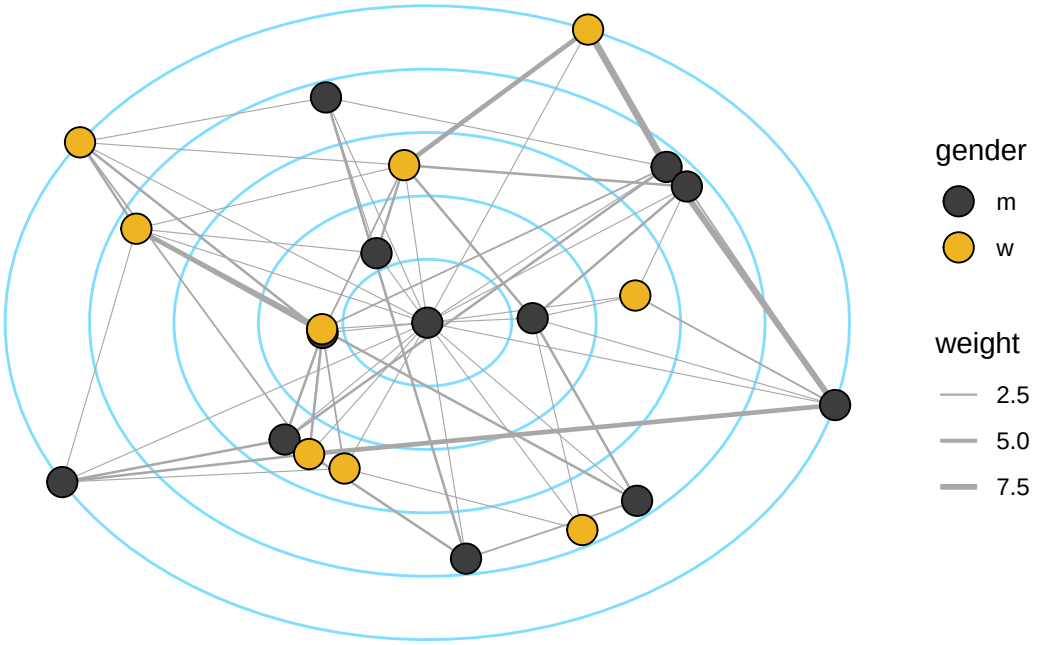
```
ggraph(egonets[[1]],layout = "centrality", cent = 5-V(egonets[[1]])$rank)+
  geom_edge_link0(aes(edge_width = weight),edge_colour="grey66")+
  geom_node_point(aes(fill = gender),shape = 21,size = 5)+
  scale_edge_width_continuous(range = c(0.2,1.2))+
  scale_fill_manual(values = c("w" = "#EEB422", "m" = "#3D3D3D"))+
  theme_graph(base_family = "Helvetica")
```



As a last step, we can add optional circles to the plot to emphasize the position of nodes. Note that we draw

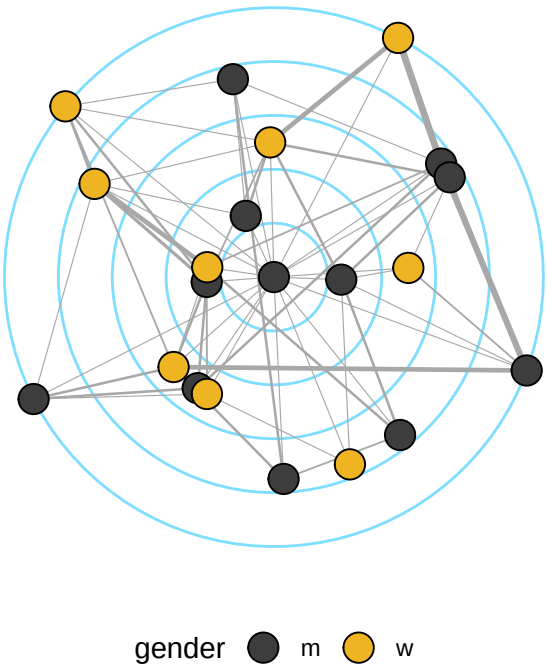
them **before** edges and nodes, so that they are in the background.

```
ggraph(egonets[[1]],layout = "centrality", cent = 5-V(egonets[[1]])$rank)+
  draw_circle(use = "cent")+
  geom_edge_link0(aes(edge_width = weight),edge_colour="grey66")+
  geom_node_point(aes(fill = gender),shape = 21,size = 5)+
  scale_edge_width_continuous(range = c(0.2,1.2))+
  scale_fill_manual(values = c("w" = "#EEB422", "m" = "#3D3D3D"))+
  theme_graph(base_family = "Helvetica")
```



The circles look a little too elliptic. We can fix the aspect ratio with `coord_fixed`. Additionally, we do not need the legend for edge widths and turn it of with `guide = FALSE` in `scale_edge_width_continuous`. The legend for the node colors is placed at the bottom.

```
ggraph(egonets[[1]],layout = "centrality", cent = 5-V(egonets[[1]])$rank)+
  draw_circle(use = "cent")+
  geom_edge_link0(aes(edge_width = weight),edge_colour="grey66")+
  geom_node_point(aes(fill = gender),shape = 21,size = 5)+
  scale_edge_width_continuous(range = c(0.2,1.2),guide = FALSE)+
  scale_fill_manual(values = c("w" = "#EEB422", "m" = "#3D3D3D"))+
  coord_fixed()+
  theme_graph(base_family = "Helvetica")+
  theme(legend.position = "bottom")
```



There are still some flaws in the plot (overlapping nodes and ego might not be relevant to show), but overall it is a nice way of plotting an ego network using the `graphlayouts` package.

Small multiples

In some cases, it might be needed to show a large collection of ego networks in one plot. We can easily reuse the code from above for our entire ego network collection that is stored in `egonets`. We start by turning the ggraph code into a function, which only takes a network as input.

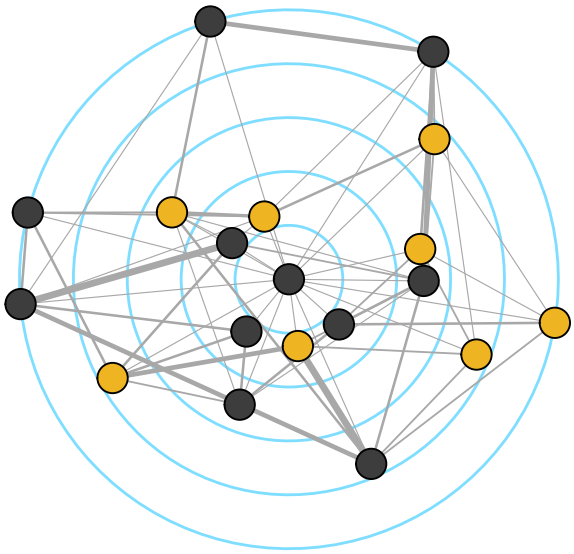
```
plot_ego <- function(net){
  ggraph(net,layout = "centrality", cent = 5-V(net)$rank)+
  draw_circle(use = "cent")+
  geom_edge_link0(aes(edge_width = weight),edge_colour="grey66")+
  geom_node_point(aes(fill = gender),shape = 21,size = 5)+
  scale_edge_width_continuous(range = c(0.2,1.2),guide = FALSE)+
  scale_fill_manual(values = c("w" = "#EEB422", "m" = "#3D3D3D"))+
  coord_fixed()+
  theme_graph(base_family = "Helvetica")+
  theme(legend.position = "bottom")
}
```

Using `lapply`, we apply it to our whole collection.

```
ego_plots <- lapply(egonets,plot_ego)
```

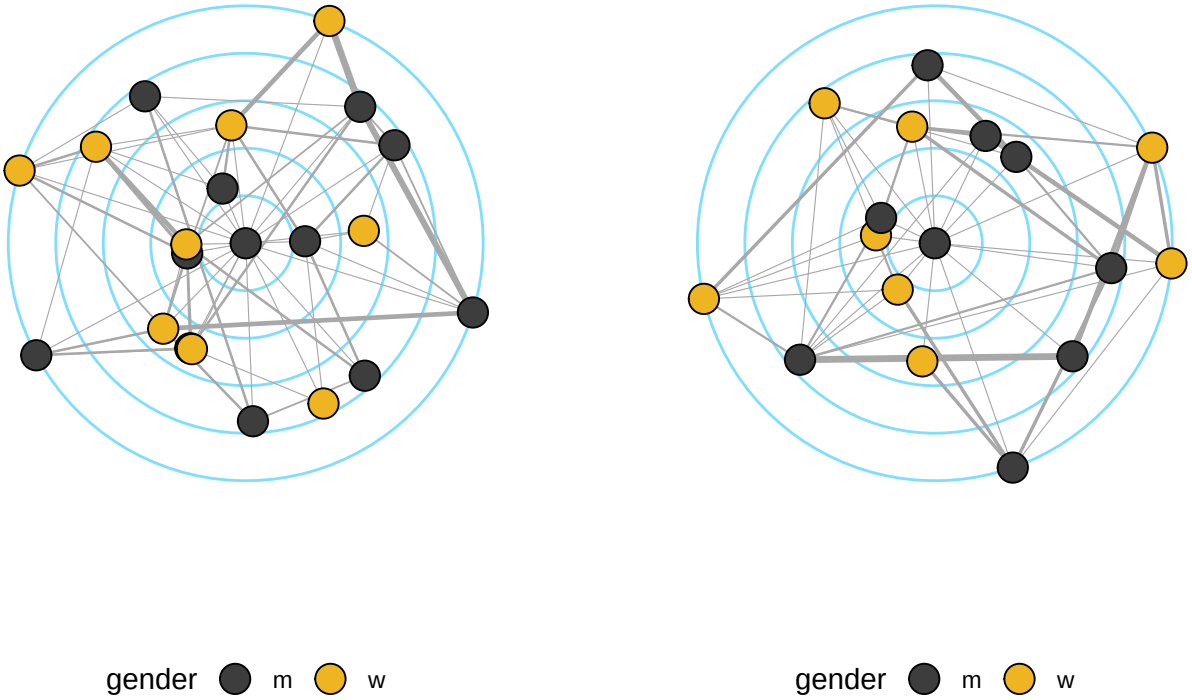
You can now look at specific networks with

```
ego_plots[[12]]
```



or we can stitch them all together with the `patchwork` package. Doing so is pretty straightforward.

```
ego_plots[[1]] + ego_plots[[2]]
```



To put all networks into one final plot, we use a for loop.

```
p <- ego_plots[[1]]
for(i in 2:32){
 p <- p + ego_plots[[i]]
}
```

The patchwork package has some additional functions which allow us to, e.g., control how many single plots are put into a row.

```
p + plot_layout(ncol=8)
```