The Promise to Partner

Lukas C. Bossert | Sama Majidian | Évariste Demandt

September 23, 2021*

1 The Promise to Partner

In this JupyterNotebook we show you how to visualize and analyze a network. We do this using the example of the consortia that are participating in or have applied to the National Research Data Infrastructure Initiative (NFDI).

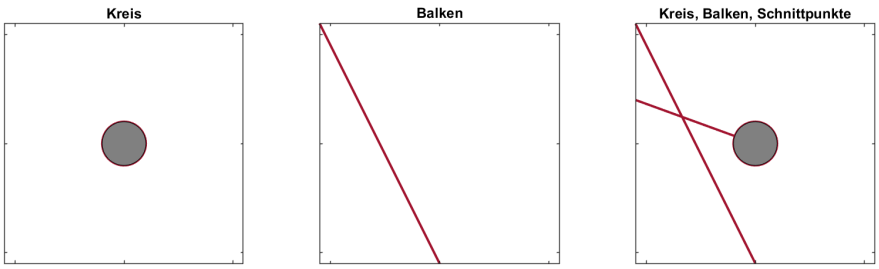
As a data basis, we take the *Letters of Intent* of the respective consortia, in which cooperation partners are named. These mentions are the starting point of our network[1].

We do the visualization in a JupyterNotebook or R Notebook[2], so no local installation of R is necessary. JupyterNotebooks are built in such a way that you have different cells in which you write code (in our case R code). To run the cell with the code, we can click on "*Cell*" and "*Run Cells*" in the menu. Or click with the cursor in the cell and then press *SHIFT*" and "*ENTER*" at the same time. You will then see the result of the code displayed directly below the cell.

Before we get started, let's clarify a few terms. A network consists of two components:

- Nodes (circle)
- Edges (line)

Nodes (*nodes* or *vertices*) are represented as circles and represent consortia. Edges (*edges*) are represented as more or less curved lines and emanate from the nodes. They indicate a connection between two nodes.



R is built in such a way that different libraries can be loaded for different functions. For the network analysis

*The automated conversion of the R notebook was created with LuaHBTeX, Version 1.13.2 (TeX Live 2021).

[1] See also the repository of Dorothea Strecker (https://github.com/dorothearrr/NFDI_Netzwerk), where she has already done a similar visualization and analysis.

[2] https://rnotebook.io cf. https://bookdown.org/yihui/rmarkdown/notebook.html

we will use the package `igraph`[3]. With `library('igraph')` we load the package.

With `if (!require("igraph")) install.packages("igraph")` we install the package in case it is not available on the current system.

```
if (!require("igraph")) install.packages("igraph")
library('igraph')
```

1.1 The Dataset

The data basis is a two-column listing of the consortia. The first column (`from`) contains the consortium whose *letter of intent* is evaluated. The second column (`to`) contains the consortium which is named as cooperation partner.

This data is read in by means of the function `read.table`. There are three parameters:

- `header=TRUE` (there is a header line in the dataset).
- `sep=","` (the values are separated by a comma)
- `text=""` (the values themselves are between the quotes)

We pass these values to the self-selected variable `NFDI_edges` , which is done with the arrow symbol pointing to the left.

The data itself comes from the GitHub gist nfdi-collaborations.csv

```
# Dataset:
# https://gist.github.com/LukasCBossert/9bd04115db3aa9ed974fdc69d3ff227c
NFDI_edges <- read.table(header=TRUE,
                         sep=",",
                         text="
from,to
DataPLANT,NFDI4BioDiversity
DataPLANT,NFDI4Chem
GHGA,NFDI4Health
KonsortSWD,BERD@NFDI
KonsortSWD,NFDI4BioDiversity
KonsortSWD,NFDI4Earth
KonsortSWD,NFDI4Health
KonsortSWD,Text+
NFDI4BioDiversity,NFDI4Earth
NFDI4BioDiversity,NFDI4Chem
NFDI4BioDiversity,NFDI4Health
NFDI4BioDiversity,KonsortSWD
NFDI4BioDiversity,DataPLANT
NFDI4Cat,FAIRmat
NFDI4Cat,NFDI4Chem
NFDI4Cat,NFDI4Ing
NFDI4Cat,DAPHNE4NFDI
NFDI4Chem,FAIRmat
```

[3] https://igraph.org/r/

```
NFDI4Chem,NFDI4Ing
NFDI4Chem,NFDI4Cat
NFDI4Chem,DAPHNE4NFDI
NFDI4Chem,PUNCH
NFDI4Chem,NFDI4Health
NFDI4Chem,NFDI4BioDiversity
NFDI4Culture,Text+
NFDI4Culture,MaRDI
NFDI4Culture,NFDI4Ing
NFDI4Health,GHGA
NFDI4Health,KonsortSWD
NFDI4Health,NFDI4Chem
NFDI4Health,NFDI4Earth
NFDI4Health,NFDI4BioDiversity
NFDI4Ing,NFDI-MatWerk
NFDI4Ing,FAIRmat
NFDI4Ing,NFDI4Chem
NFDI4Ing,NFDI4Earth
NFDI4Ing,MaRDI
NFDI4Ing,Text+
NFDI4Ing,NFDI4Culture
BERD@NFDI,KonsortSWD
BERD@NFDI,MaRDI
BERD@NFDI,Text+
DAPHNE4NFDI,FAIRmat
DAPHNE4NFDI,NFDI-MatWerk
DAPHNE4NFDI,NFDI4Cat
DAPHNE4NFDI,NFDI4Chem
DAPHNE4NFDI,NFDI4Health
DAPHNE4NFDI,NFDI4Ing
DAPHNE4NFDI,PUNCH
FAIRmat,DAPHNE4NFDI
FAIRmat,DataPLANT
FAIRmat,MaRDI
FAIRmat,NFDI-MatWerk
FAIRmat,NFDI4Cat
FAIRmat,NFDI4Chem
FAIRmat,DataScience
FAIRmat,NFDI4Ing
FAIRmat,PUNCH
MaRDI,BERD@NFDI
MaRDI,FAIRmat
MaRDI,NFDI-MatWerk
MaRDI,NFDI4Cat
MaRDI,NFDI4Chem
MaRDI,NFDI4Ing
MaRDI,PUNCH
```

```
NFDI-MatWerk,DAPHNE4NFDI
NFDI-MatWerk,DataPLANT
NFDI-MatWerk,FAIRmat
NFDI-MatWerk,MaRDI
NFDI-MatWerk,NFDI4Chem
NFDI-MatWerk,DataScience
NFDI-MatWerk,NFDI4Ing
DataScience,KonsortSWD
DataScience,MaRDI
DataScience,NFDI-MatWerk
DataScience,NFDI4BioDiversity
DataScience,NFDI4Cat
DataScience,NFDI4Chem
DataScience,NFDI4Culture
DataScience,NFDI4Health
DataScience,NFDI4Ing
DataScience,NFDI4Microbiota
NFDI4Earth,DataPLANT
NFDI4Earth,GHGA
NFDI4Earth,KonsortSWD
NFDI4Earth,NFDI4BioDiversity
NFDI4Earth,NFDI4Cat
NFDI4Earth,NFDI4Chem
NFDI4Earth,NFDI4Culture
NFDI4Earth,NFDI4Health
NFDI4Earth,NFDI4Ing
NFDI4Microbiota,DataPLANT
NFDI4Microbiota,GHGA
NFDI4Microbiota,NFDI4BioDiversity
NFDI4Microbiota,NFDI4Chem
NFDI4Microbiota,DataScience
NFDI4Microbiota,NFDI4Health
NFDI4Microbiota,NFDI4Ing
PUNCH,DAPHNE4NFDI
PUNCH,FAIRmat
PUNCH,GHGA
PUNCH,MaRDI
PUNCH,NFDI4Earth
PUNCH,NFDI4Ing
Text+,KonsortSWD
Text+,NFDI4BioDiversity
Text+,NFDI4Culture
Text+,NFDI4Earth
Text+,NFDI4Ing
")
```

So that we can create a network from this dataset, we have to prepare it and create a `igraph graph`.[4] This is done with the function `graph_from_data_frame`, to which we pass our dataset.

We also specify that our dataset or network is undirected (`directed=FALSE`), that means that the direction as specified by `from,to` in the dataset does not matter. All we care about now is that two consortia are linked.

We pass this information to the variable `NFDI_network`.

```
NFDI_network <- graph_from_data_frame(NFDI_edges,
                                      directed = FALSE
                                     )
```

1.2 Basic setting

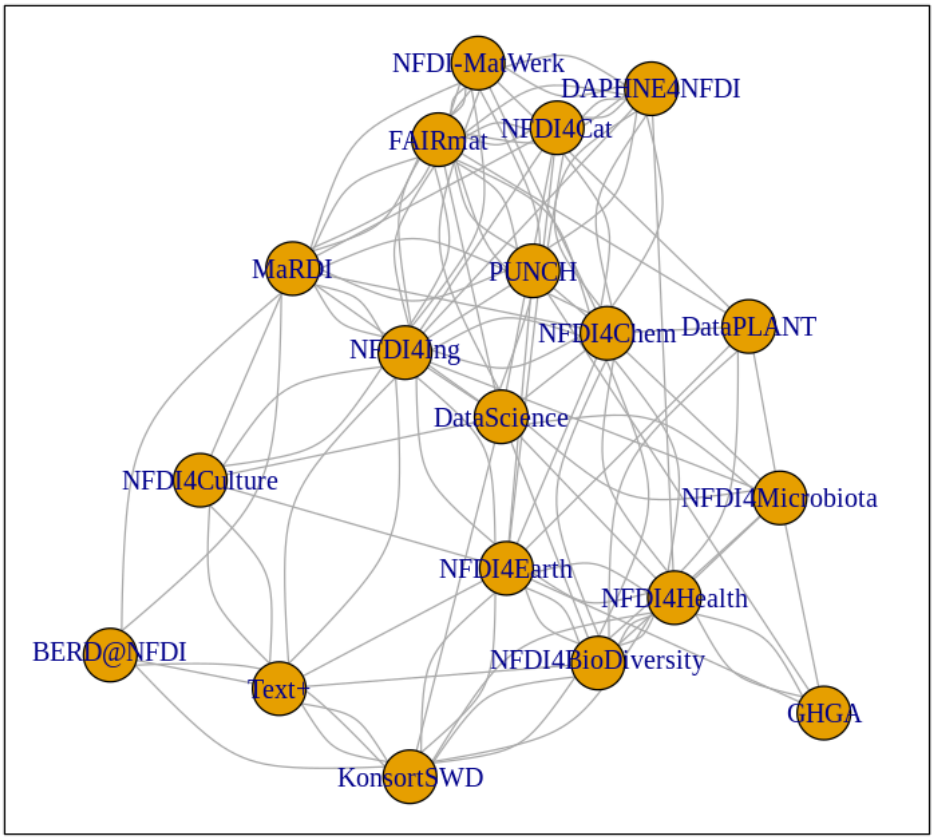
First, we will set a parameter so that our network always looks the same when the data is the same. This parameter is `seed`. We choose an arbitrary number, which may be large.

After that we come to the actual plot. For this we call the function `plot` and pass it the variable of our network graph `NFDI_network`. For a title we can still specify the parameter `main` and also we can specify if we want to have a frame around the network with `frame=TRUE`.

```
set.seed(9876543)

plot(NFDI_network,                    # loading data frame
     main  = "NFDI Network",          # adding a title
     frame = TRUE                     # making a frame
     )
```

[4] https://igraph.org/r/doc/graph_from_data_frame.html



We see the network of NFDI consortia without any other explicit settings.

1.3 Layout settings

The next step we want to do is optimize the layout of the network. Instead of retyping the code for the plot, we will select the content of the last cell, copy and paste it into the next cell.

We'll expand the code this way and work on the network step by step.

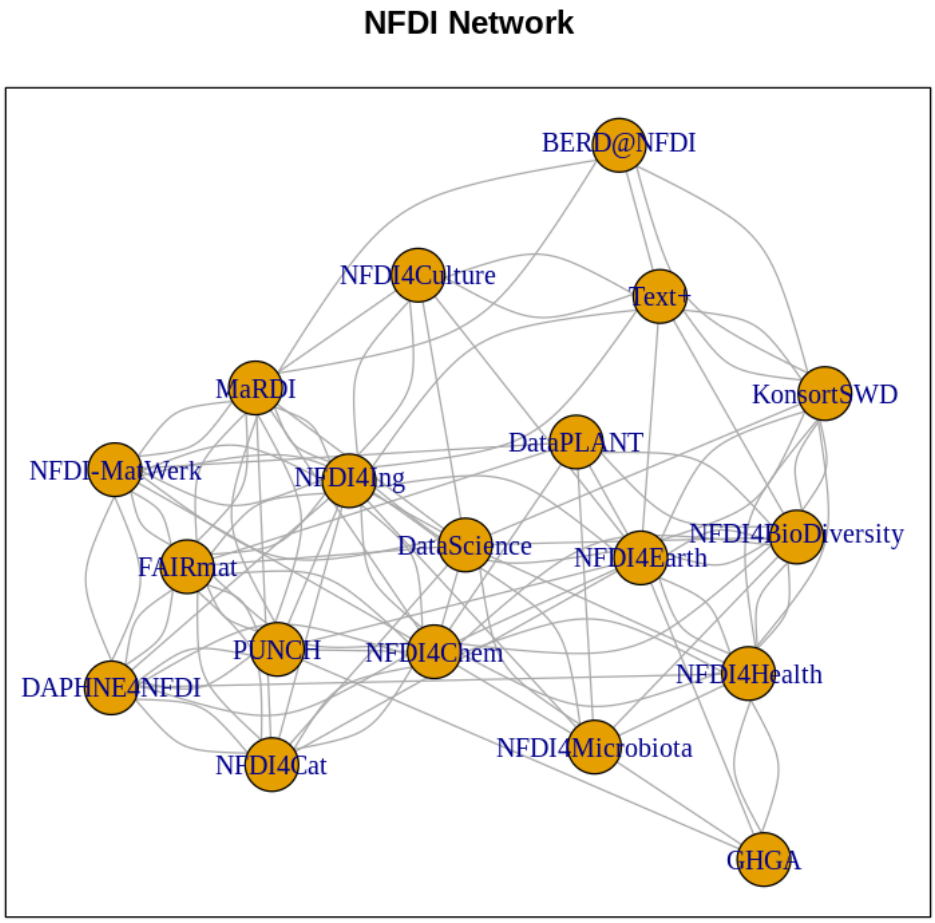
There are different algorithms for the layout of networks. Depending on the data set, sometimes one layout, sometimes the other may be more suitable. With the layout `graphopt`[5] you usually get a good result.

We pass this value `layout.graphopt` to the parameter `layout`.

[5] https://igraph.org/r/doc/layout_with_graphopt.html

```r
set.seed(9876543)

plot(NFDI_network,                    # loading data frame
     main  = "NFDI Network",          # adding a title
     frame  = TRUE,                   # making a frame
     layout = layout.graphopt,        #* better layout options
     )
```



We see the network of NFDI consortia without any other explicit settings.

The network is now already better structured and the distances between the nodes are more harmonious.

If you like, you can try out further layout settings [6]:

[6] https://igraph.org/python/doc/tutorial/tutorial.html#layout-algorithms

- `layout_circle` (`circle`,`circular`): Deterministic layout that places the vertices on a circle
- `layout_drl` (`drl`): The Distributed Recursive Layout algorithm for large graphs
- `layout_fruchterman_reingold` (`fr`): Fruchterman-Reingold force-directed algorithm
- `layout_fruchterman_reingold_3d` (`fr3d, fr_3d`): Fruchterman-Reingold force-directed algorithm in three dimensions
- `layout_grid_fruchterman_reingold` (`grid_fr`): Fruchterman-Reingold force-directed algorithm with grid heuristics for large graphs
- `layout_kamada_kawai` (`kk`): Kamada-Kawai force-directed algorithm
- `layout_kamada_kawai_3d` (`kk3d, kk_3d`): Kamada-Kawai force-directed algorithm in three dimensions
- `layout_lgl` (`large, lgl, large_graph`): The Large Graph Layout algorithm for large graphs
- `layout_random` (`random`): Places the vertices completely randomly
- `layout_random_3d` (`random_3d`): Places the vertices completely randomly in 3D
- `layout_reingold_tilford` (`rt, tree`): Reingold-Tilford tree layout, useful for (almost) tree-like graphs
- `layout_reingold_tilford_circular` (`rt_circular, tree`): Reingold-Tilford tree layout with a polar coordinate post-transformation, useful for (almost) tree-like graphs
- `layout_sphere` (`sphere`,`spherical`,`circular_3d`): Deterministic layout that places the vertices evenly on the surface of a sphere

1.3.1 Color, Size, Curvature (Nodes and Edges)

After we have optimized the arrangement of the nodes, let's tackle the representation of the nodes and edges in the next step.

Various parameters can be adjusted according to your own wishes.

First we want to tackle the color of the nodes. The parameter is `vertex.color` and we can specify an HTML color value (for example `#ffcc66`).[7] For the border of the nodes we choose the same color code. The parameter is `vertex.frame.color`.

The labels of the nodes can also be modified. The change of the font size is done by the parameter `vertex.label.cex`, to which we pass the value `0.5`. It is important here that the value is *not* written in quotes. This is a relative size and we want the labels to be half the size they were in the previous network. The color of the label can also be changed. Quite analogously, the parameter is called `vertex.label.color`, to which we can also pass the color value as a string, such as `"black"`.

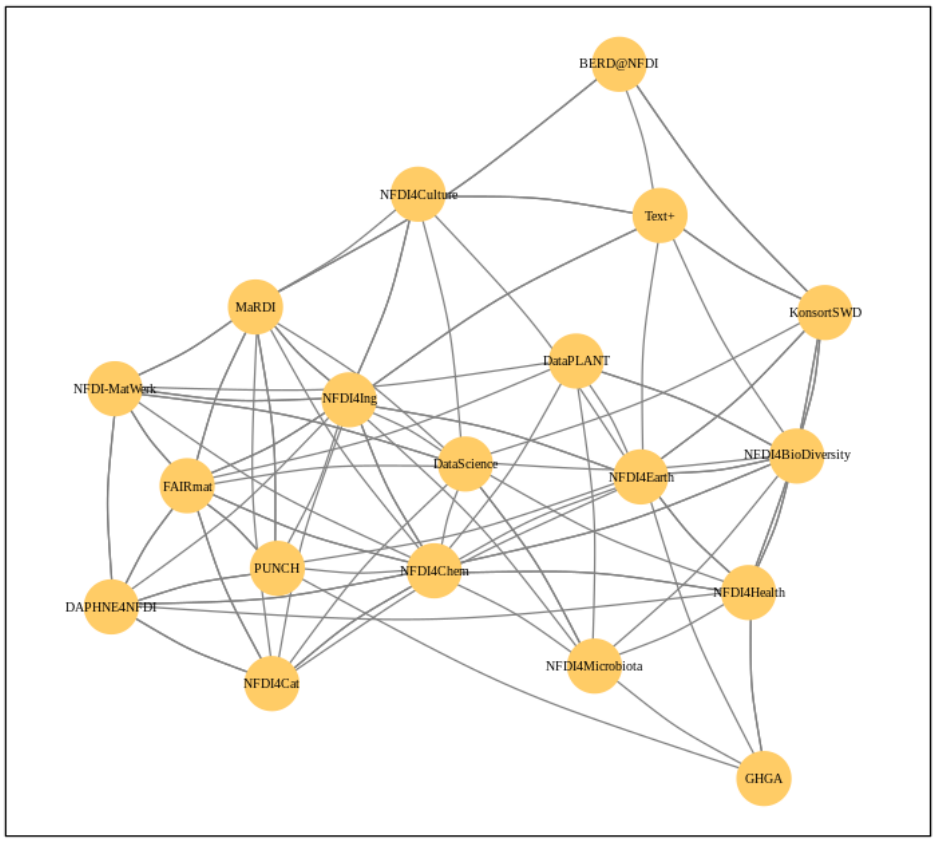
A network consists not only of nodes but also of edges connecting two nodes. For the color change we need the parameter `edge.color`, to which we pass for example `"#808080"`. Besides the color we can also specify the degree of "curvature", which is set with `edge.curved` and the value `0.1`. Again, it is important that *no* quotes are set.

```
set.seed(9876543)


plot(NFDI_network,                    # loading data frame
     main   = "NFDI Network",         # adding a title
     frame  = TRUE,                   # making a frame
```

[7] https://www.w3schools.com/colors/colors_picker.asp

```
    layout = layout.graphopt,          # better layout options
    vertex.color       = "#ffcc66",    #* color of nodes
    vertex.frame.color = "#ffcc66",    #* color of the frame of nodes
    vertex.label.cex   = 0.5,          #* size of the description of the labels
    vertex.label.color = "black",      #* color of the description
    edge.color         = "#808080",    #* color of edges
    edge.curved        = 0.1,          #* factor of "curvity"
    )
```



1.4 Node size as a function of the number of edges

In the previous network representations, all nodes are the same size.

Now we want to add another layer of information and output the node size according to the number of its edges.

We can determine the number of edges per node with the function `degree`[8]. If we pass this function the dataset of the network (`degree(NFDI_network)`), then we get the number of edges per node. We take these values as the size specification for the nodes.

We thus extend the previous code by one line. The node size is hidden behind the parameter `vertex.size` and as value we pass the function `degree(NFDI_network)`.

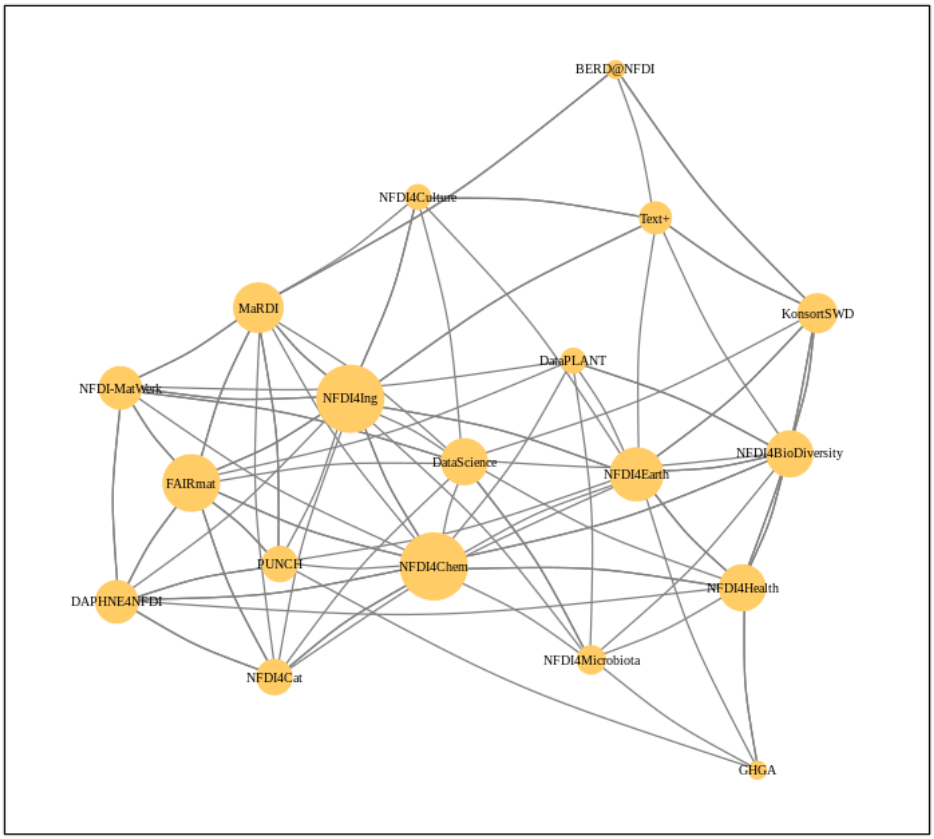
```
#data.frame(
    degree(NFDI_network) #* calculate number of edges
#)
```

DataPLANT 7 **GHGA** 5 **KonsortSWD** 11 **NFDI4BioDiversity** 13 **NFDI4Cat** 10 **NFDI4Chem** 19 **NFDI4Culture** 7 **NFDI4Health** 13 **NFDI4Ing** 19 **BERD@NFDI** 5 **DAPHNE4NFDI** 12 **FAIRmat** 16 **MaRDI** 14 **NFDI-MatWerk** 12 **DataScience** 13 **NFDI4Earth** 15 **NFDI4Microbiota** 8 **PUNCH** 10 **Text+** 9

```
set.seed(9876543)

plot(NFDI_network,                      # loading data frame
     main   = "NFDI-Netzwerk",          # adding a title
     frame  = TRUE,                     # making a frame
     layout = layout.graphopt,          # better layout options
     vertex.color       = "#ffcc66",    # color of nodes
     vertex.frame.color = "#ffcc66",    # color of the frame of nodes
     vertex.label.cex   = 0.5,          # size of the description of the labels
     vertex.label.color = "black",      # color of the description
                                        # color: https://www.w3schools.com/
 ↪colors/colors_picker.asp
     edge.color         = "#808080",    # color of edges
     edge.curved        = 0.1,          # factor of "curvity"
     vertex.size        = degree(NFDI_network), #* size of nodes depends on␣
 ↪amount of edges
     )
```

[8] https://igraph.org/r/doc/degree.html



1.5 Node size as a function of the number of incoming and outgoing edges.

We have now introduced a second layer of information into our network and can display the node size in relation to the number of edges.

In the next step, we would like to introduce another component. Until now, it was irrelevant whether a consortium was named first or second in the dataset, i.e., it was irrelevant whether it was the active or the passive collaborator.

Now we would like to consider the distinction in the network. To do this, our graph (network) must be "directed"[9].

We introduce a new variable (`NFDI_network_directed`), which contains the dataset as a directed graph,

[9] https://en.wikipedia.org/wiki/Directed_graph

which we set with `directed = TRUE`.

```
NFDI_network_directed <- graph_from_data_frame(NFDI_edges,
                                               directed = TRUE
                                              )
```

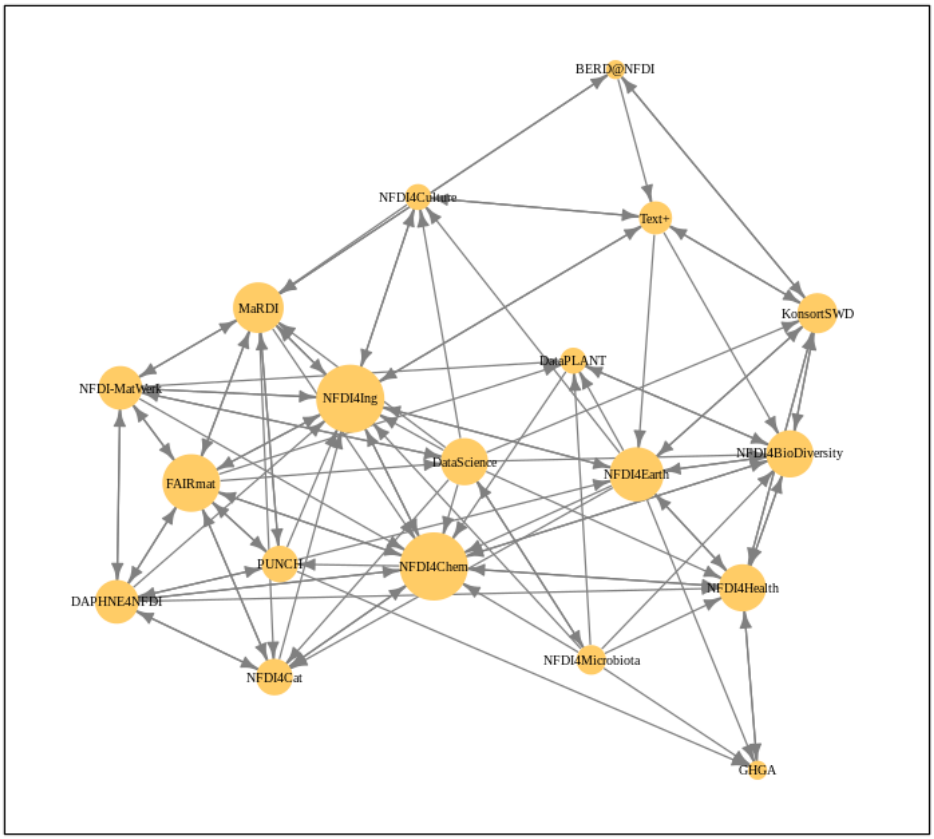
We transfer the remaining plot data from the previous cell. It is now crucial that we pass the new variable with the directed graph to the plot function. In addition, we also pass the new variable to the `degree` function.

In the directed network, the curvature of the edges makes it difficult to read. Therefore we choose the value 0 for `edge.curved`.

Likewise, the arrowheads should become smaller, which is possible with `edge.arrow.size` and the relative value `0.5`.

```
set.seed(9876543)

plot(NFDI_network_directed,            #<<<<<<< loading data frame
     main   = "NFDI-Netzwerk",         # adding a title
     frame  = TRUE,                    # making a frame
     layout = layout.graphopt,         # better layout options
     vertex.color       = "#ffcc66",   # color of nodes
     vertex.frame.color = "#ffcc66",   # color of the frame of nodes
     vertex.label.cex   = 0.5,         # size of the description of the labels
     vertex.label.color = "black",     # color of the description
                                       # color: https://www.w3schools.com/
 ↪colors/colors_picker.asp
     edge.color         = "#808080",   # color of edges
     edge.curved        = 0,           #<<<<<<<<<< factor of "curvity"
     vertex.size        = degree(NFDI_network_directed), #<<<<<< size of nodes␣
 ↪depends on amount of edges
     edge.arrow.size    = .5,          #* arrow size,  defaults to 1
    )
```



In the next step, we want to scale the node size according to the *in*bound edges. The more often a consortium is named as a collaborator, the larger its node will be.

We can modify the function `degree` for this by adding `mode = "in"`[10].

=2=2

[10] https://igraph.org/r/doc/degree.html