

---

---

## The Promise to Partner. Visualization of networks – analyzing and

visualizing connections between (planned) NFDI consortia

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).

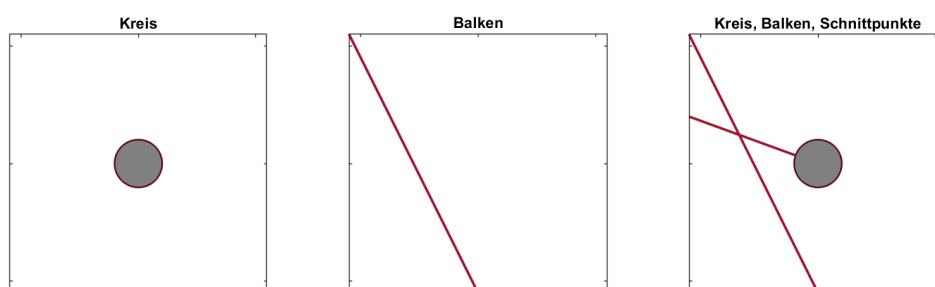
As a data basis, we take the *Letters of Intent* of the respective consortia, in which cooperation partners are named. These mentions are the starting point of our network<sup>1</sup>.

We do the visualization in a JupyterNotebook or R Notebook<sup>2</sup>, 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.



**Figure 1:** Components of a network. Created by ÉD.

---

<sup>1</sup>See also the repository of Dorothea Strecker ([https://github.com/dorothearr/NFDI\\_Netzwerk](https://github.com/dorothearr/NFDI_Netzwerk)), where she has already done a similar visualization and analysis.

<sup>2</sup><https://mybinder.org/v2/gh/jupyterlab/jupyterlab-demo/master?urlpath=lab/tree/demo> cf. <https://bookdown.org/yihui/rmarkdown/notebook.html>

---

R is built in such a way that different libraries can be loaded for different functions. For the network analysis we will use the package `igraph`<sup>3</sup>. 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)
```

## 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`

```
# 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
```

---

<sup>3</sup><https://igraph.org/r/>

---

NFDI4BioDiversity,KonsortSWD  
NFDI4BioDiversity,DataPLANT  
NFDI4Cat,FAIRmat  
NFDI4Cat,NFDI4Chem  
NFDI4Cat,NFDI4Ing  
NFDI4Cat,DAPHNE4NFDI  
NFDI4Chem,FAIRmat  
NFDI4Chem,NFDI4Ing  
NFDI4Chem,NFDI4Cat  
NFDI4Chem,DAPHNE4NFDI  
NFDI4Chem,PUNCH4NFDI  
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,PUNCH4NFDI  
FAIRmat,DAPHNE4NFDI  
FAIRmat,DataPLANT  
FAIRmat,MarDI  
FAIRmat,NFDI-MatWerk  
FAIRmat,NFDI4Cat

---

---

FAIRmat,NFDI4Chem  
FAIRmat,NFDI4DataScience  
FAIRmat,NFDI4Ing  
FAIRmat,PUNCH4NFDI  
MaRDI,BERD@NFDI  
MaRDI,FAIRmat  
MaRDI,NFDI-MatWerk  
MaRDI,NFDI4Cat  
MaRDI,NFDI4Chem  
MaRDI,NFDI4Ing  
MaRDI,PUNCH4NFDI  
NFDI-MatWerk,DAPHNE4NFDI  
NFDI-MatWerk,DataPLANT  
NFDI-MatWerk,FAIRmat  
NFDI-MatWerk,MaRDI  
NFDI-MatWerk,NFDI4Chem  
NFDI-MatWerk,NFDI4DataScience  
NFDI-MatWerk,NFDI4Ing  
NFDI4DataScience,KonsortSWD  
NFDI4DataScience,MaRDI  
NFDI4DataScience,NFDI-MatWerk  
NFDI4DataScience,NFDI4BioDiversity  
NFDI4DataScience,NFDI4Cat  
NFDI4DataScience,NFDI4Chem  
NFDI4DataScience,NFDI4Culture  
NFDI4DataScience,NFDI4Health  
NFDI4DataScience,NFDI4Ing  
NFDI4DataScience,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,NFDI4DataScience  
NFDI4Microbiota,NFDI4Health

---

```
NFDI4Microbiota,NFDI4Ing
PUNCH4NFDI, DAPHNE4NFDI
PUNCH4NFDI, FAIRmat
PUNCH4NFDI, GHGA
PUNCH4NFDI, MaRDI
PUNCH4NFDI, NFDI4Earth
PUNCH4NFDI, NFDI4Ing
Text+,KonsortSWD
Text+,NFDI4BioDiversity
Text+,NFDI4Culture
Text+,NFDI4Earth
Text+,NFDI4Ing
")
```

Now we can check, if the data has been loaded properly into the variable.

```
head(NFDI_edges)
```

A data.frame: 6 × 2

from

to

<fct>

<fct>

1

DataPLANT

NFDI4BioDiversity

2

DataPLANT

NFDI4Chem

3

GHGA

NFDI4Health

4

KonsortSWD

BERD@NFDI

---

5

KonsortSWD

NFDI4BioDiversity

6

KonsortSWD

NFDI4Earth

So that we can create a network from this dataset, we have to prepare it and create a `igraph` graph.<sup>4</sup> 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
                                   )
```

Let us check this again and have a look at the data.

```
head(NFDI_network)
```

```
[[ suppressing 19 column names 'DataPLANT', 'GHGA', 'KonsortSWD' ... ]]
```

6 x 19 sparse Matrix of class "dgCMatrix"

DataPLANT	.	.	.	2	.	1	.	.	.	.	.	1	.	1	.	1	1	.	.	
GHGA	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.	.	1	1	1	.
KonsortSWD	.	.	.	2	.	.	.	2	.	2	.	.	.	.	.	1	2	.	.	2
NFDI4BioDiversity	2	.	2	.	.	2	.	2	.	.	.	.	.	.	.	1	2	1	.	1
NFDI4Cat	.	.	.	.	.	2	.	.	1	.	2	2	1	.	1	1	.	.	.	.
NFDI4Chem	1	.	.	2	2	.	.	2	2	.	2	2	1	1	1	1	1	1	1	.

---

<sup>4</sup>[https://igraph.org/r/doc/graph\\_from\\_data\\_frame.html](https://igraph.org/r/doc/graph_from_data_frame.html)

---

## 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`.

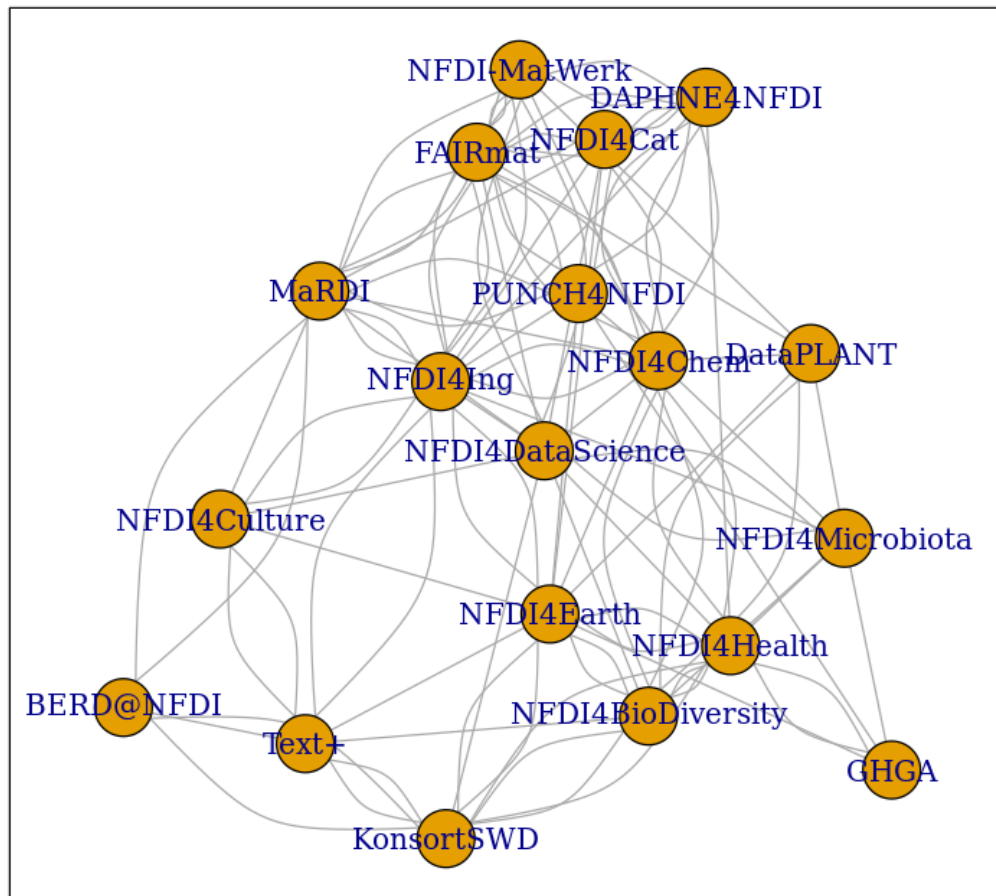
```
set.seed(9876543)
```

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



---

## NFDI Network



**Figure 2:** png

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

### 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<sup>5</sup> you usually get a good result.

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

```
set.seed(9876543)

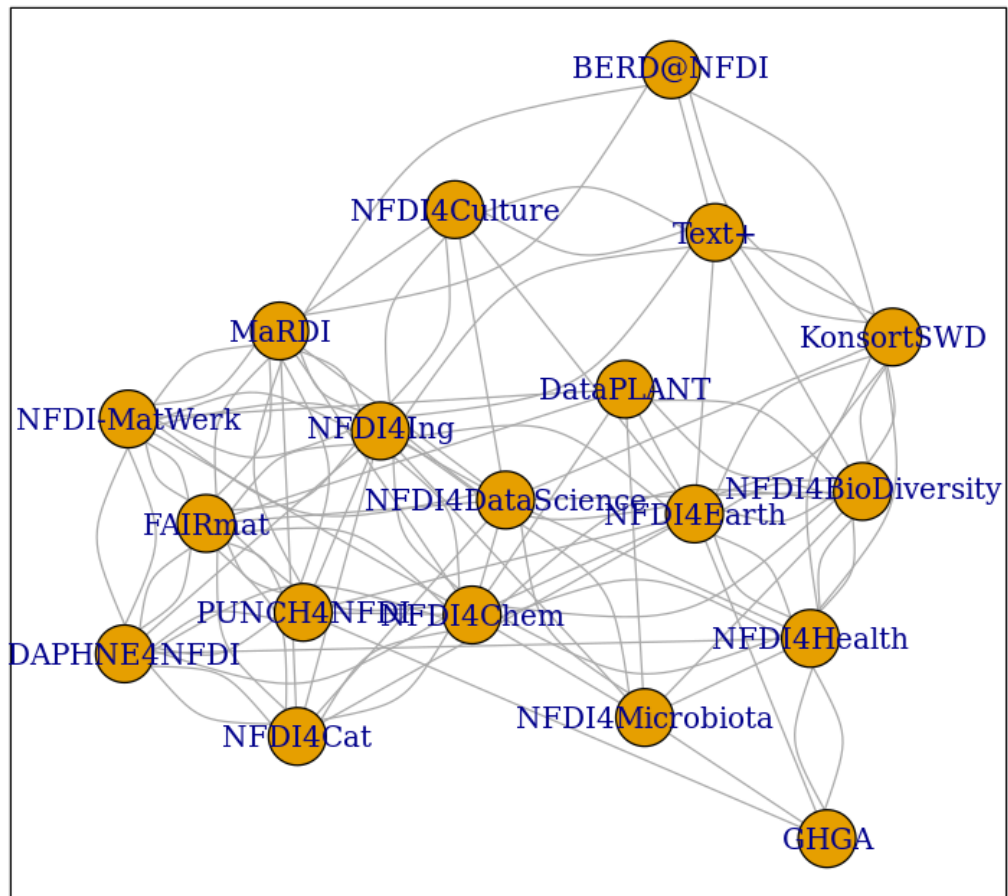
plot(NFDI_network,                # loading data frame
     main = "NFDI Network",       # adding a title
     frame = TRUE,                # making a frame
     layout = layout.graphopt,    /* better layout options
     )
```

---

<sup>5</sup>[https://igraph.org/r/doc/layout\\_with\\_graphopt.html](https://igraph.org/r/doc/layout_with_graphopt.html)

---

## NFDI Network



**Figure 3:** png

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 <sup>6</sup>:

---

<sup>6</sup><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

### 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`).<sup>7</sup> 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"`.

---

<sup>7</sup>[https://www.w3schools.com/colors/colors\\_picker.asp](https://www.w3schools.com/colors/colors_picker.asp)

---

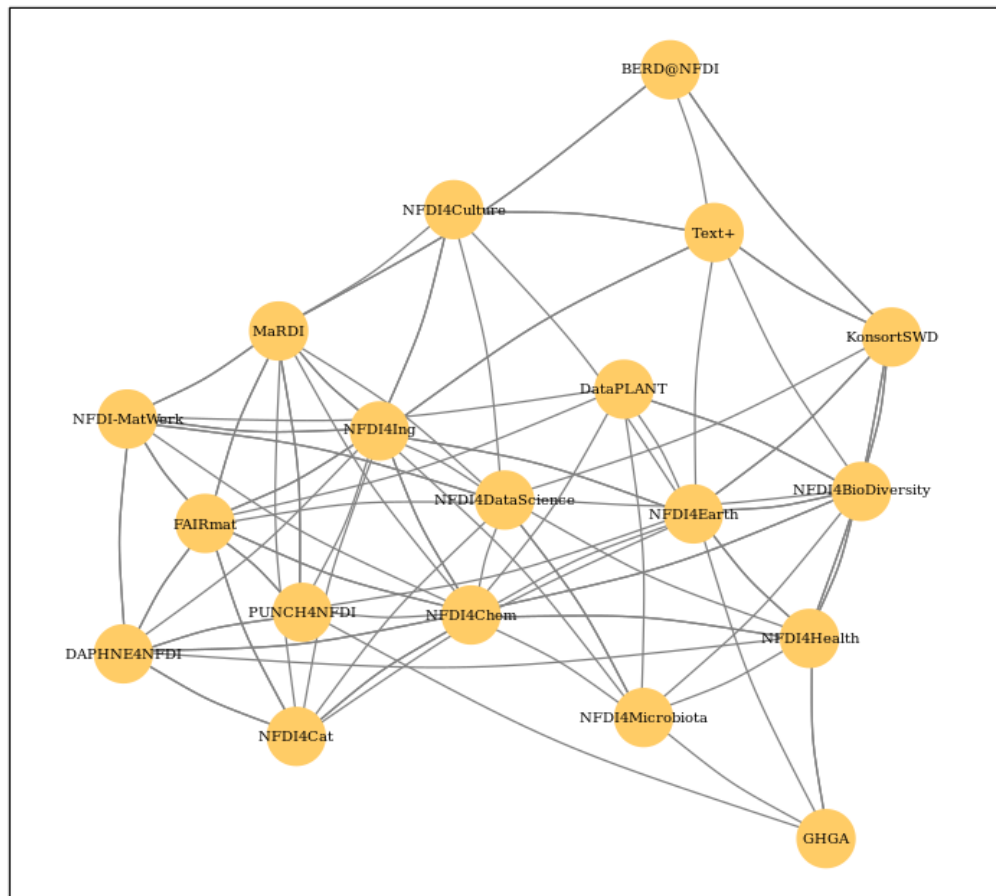
A network consists not only of nodes but also of edges connecting two nodes. For the color change we need the parameter `edge.color`, to which we pass for example `"#808080"`. Besides the color we can also specify the degree of "curvature", which is set with `edge.curved` and the value `0.1`. Again, 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
     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"
     )
```

---

## NFDI Network



**Figure 4:** png

### 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`<sup>8</sup>. 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

---

<sup>8</sup><https://igraph.org/r/doc/degree.html>

---

DAPHNE4NFDI

12

FAIRmat

16

MaRDI

14

NFDI-MatWerk

12

NFDI4DataScience

13

NFDI4Earth

15

NFDI4Microbiota

8

PUNCH4NFDI

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                                       # color of the description
     vertex.label.color = "black",                  # color:
     ↪ https://www.w3schools.com/colors/colors_picke
     ↪
     edge.color = "#808080",                       # color of edges
```



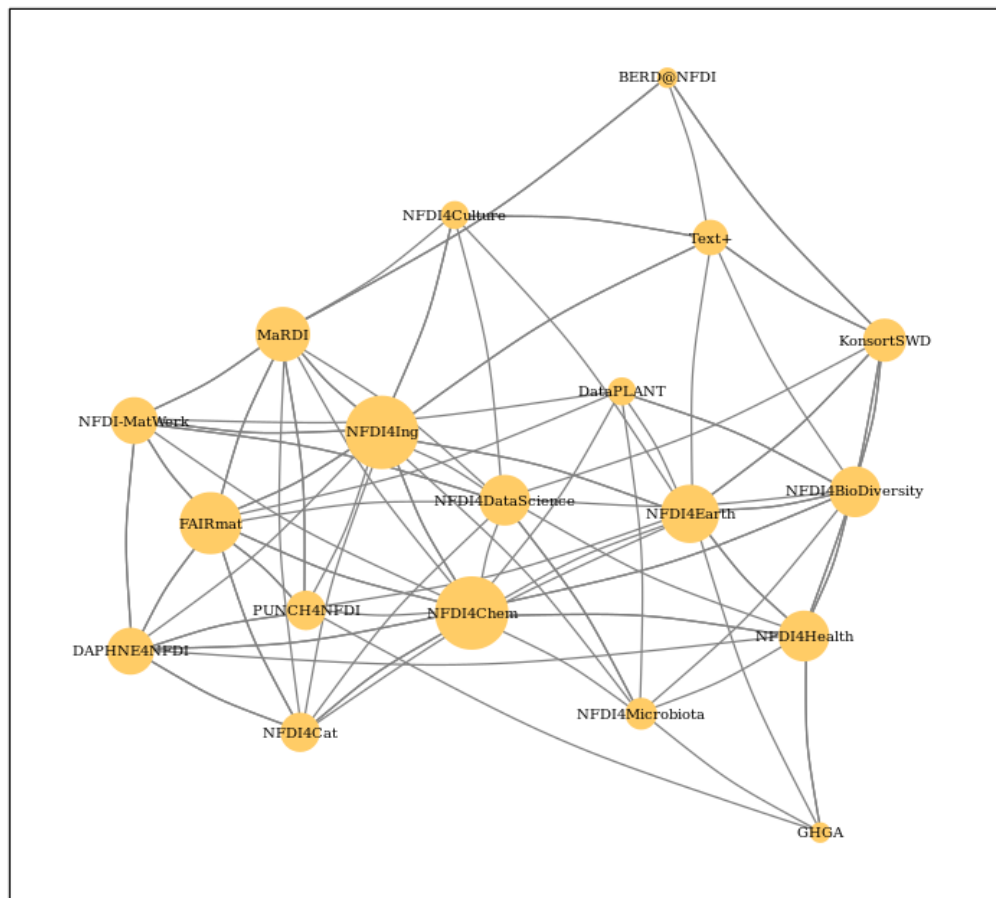
---

```

edge.curved      = 0.1,          # factor of "curvity"
vertex.size      = degree(NFDI_network), #* size of nodes depends on
↳ amount of edges
)

```

## NFDI-Netzwerk



**Figure 5:** png

---

## 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"<sup>9</sup>.

We introduce a new variable (NFDI\_network\_directed), which contains the dataset as a directed graph, which we set with `directed = TRUE`.

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

Ok, let us have a look at the data.

```
head(NFDI_network_directed)
```

```
[[ suppressing 19 column names 'DataPLANT', 'GHGA', 'KonsortSWD' ... ]]
```

6 x 19 sparse Matrix of class "dgCMatrix"

DataPLANT	.	.	.	1	.	1	.	.	.	.	.	.	.	.	.	.	.	.
GHGA	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.
KonsortSWD	.	.	.	1	.	.	.	1	.	1	.	.	.	.	1	.	.	1
NFDI4BioDiversity	1	.	1	.	.	1	.	1	.	.	.	.	.	.	1	.	.	.
NFDI4Cat	.	.	.	.	.	1	.	.	1	.	1	1	.	.	.	.	.	.
NFDI4Chem	.	.	.	1	1	.	.	1	1	.	1	1	.	.	.	.	1	.

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`.

---

<sup>9</sup>[https://en.wikipedia.org/wiki/Directed\\_graph](https://en.wikipedia.org/wiki/Directed_graph)

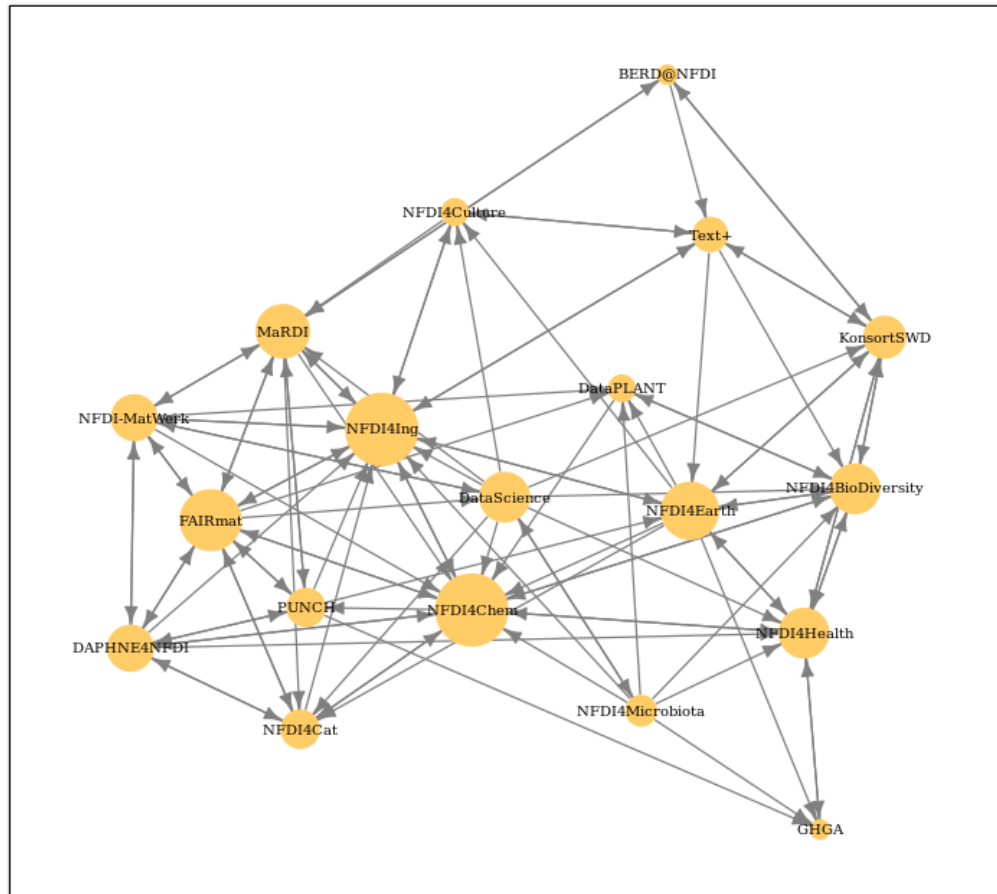
---

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_picke
                                         ↪
     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
)
```

## NFDI-Netzwerk



**Figure 6:** png

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"`<sup>10</sup>.

```
degree(NFDI_network_directed,  
      mode = "in")
```

<sup>10</sup><https://igraph.org/r/doc/degree.html>

---

```
#data.frame(  
  degree(NFDI_network_directed,  
        mode = "in")  
#)
```

DataPLANT

5

GHGA

4

KonsortSWD

6

NFDI4BioDiversity

8

NFDI4Cat

6

NFDI4Chem

12

NFDI4Culture

4

NFDI4Health

8

NFDI4Ing

12

BERD@NFDI

2

DAPHNE4NFDI

5

FAIRmat

7

MaRDI

---

7

NFDI-MatWerk

5

DataScience

3

NFDI4Earth

6

NFDI4Microbiota

1

PUNCH

4

Text+

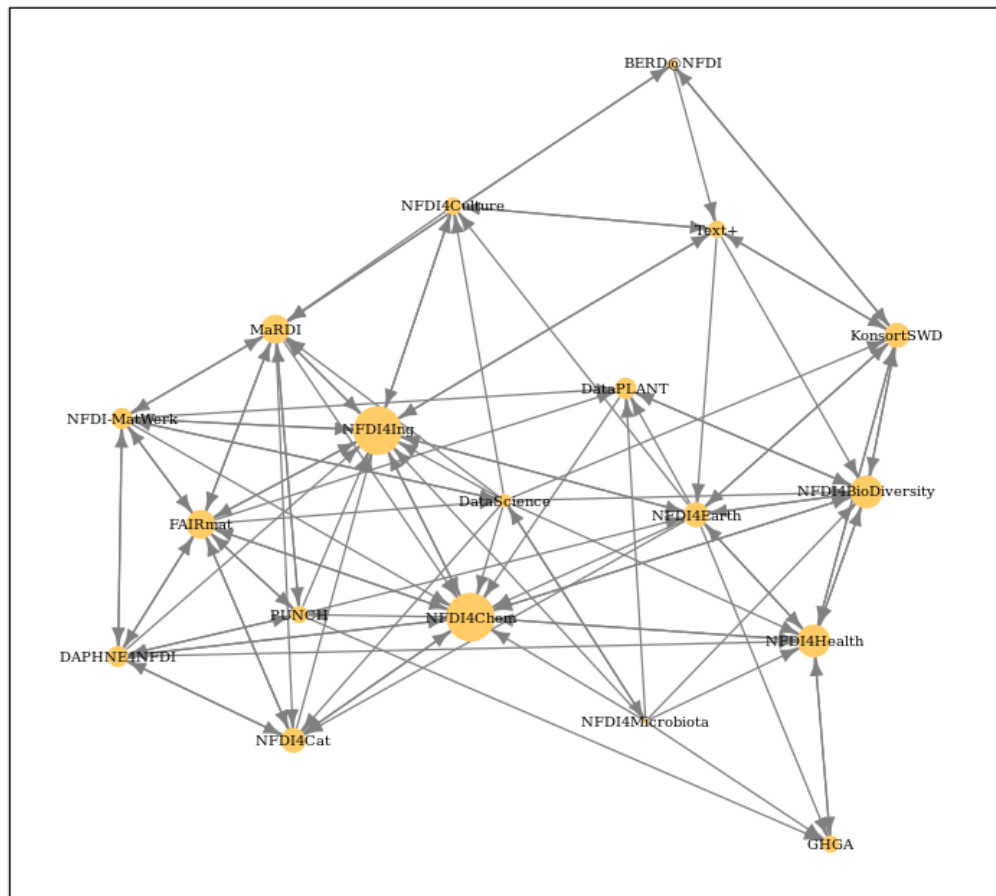
4

`set.seed(9876543)`

```
plot(NFDI_network_directed,           # loading data frame
     main = "NFDI Network (<in>)",    #<<<<<<< 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_picke
                                         ↪
     edge.color = "#808080",           # color of edges
     edge.curved = 0,                  # factor of "curvity"
     vertex.size = degree(NFDI_network_directed,
                           mode = "in"), #<<<<< size of nodes
                                         ↪ depends on amount of edges
     edge.arrow.size = .5,             # arrow size, defaults to 1
)
```

---

## NFDI Network (<in>)



**Figure 7:** png

Likewise, we can now also display the size of the consortia according to their /out/going edges.

We take the complete cell content from before and only change in to out.

```
#data.frame(  
  degree(NFDI_network_directed,  
        mode = "out")  
#)
```

---

DataPLANT

2

GHGA

1

KonsortSWD

5

NFDI4BioDiversity

5

NFDI4Cat

4

NFDI4Chem

7

NFDI4Culture

3

NFDI4Health

5

NFDI4Ing

7

BERD@NFDI

3

DAPHNE4NFDI

7

FAIRmat

9

MaRDI

7

NFDI-MatWerk

7



---

DataScience

10

NFDI4Earth

9

NFDI4Microbiota

7

PUNCH

6

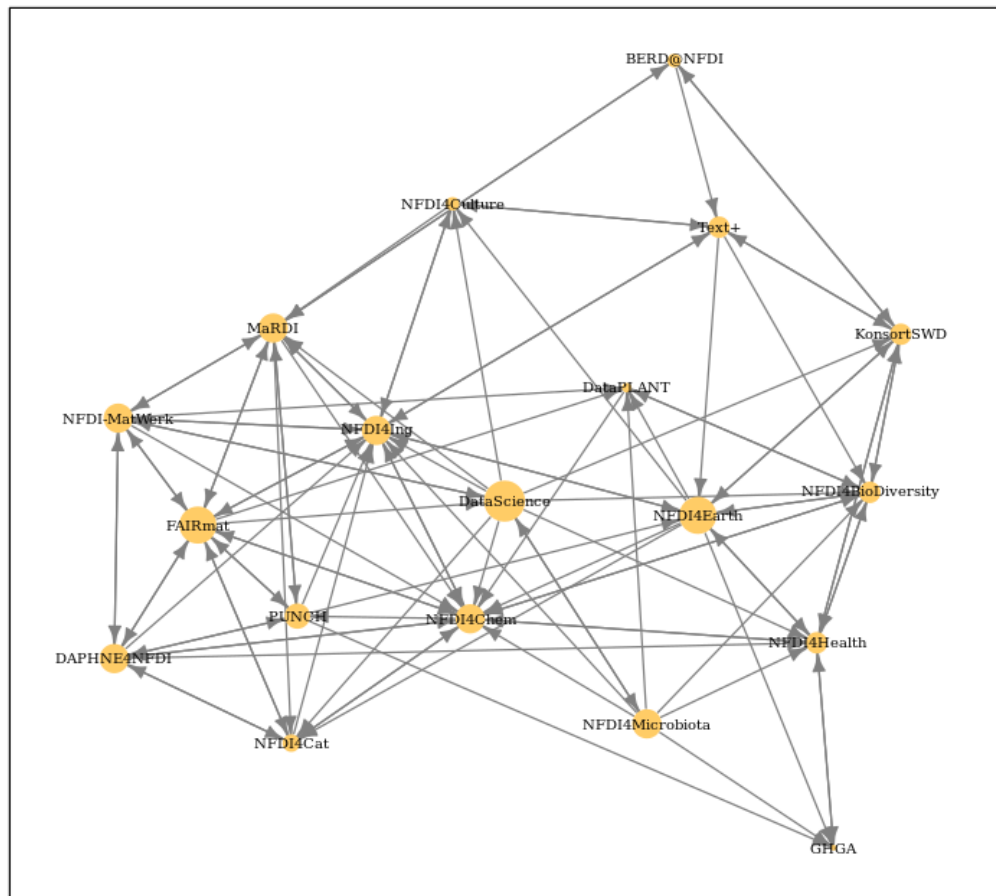
Text+

5

`set.seed(9876543)`

```
plot(NFDI_network_directed,           # loading data frame
     main = "NFDI Network (<out>)",    #<<<<<<< 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,
                           mode = "out"), #<<<<< size of nodes
                                         ↪ depends on amount of edges
     edge.arrow.size = .5,             # arrow size, defaults to 1
)
```

## NFDI Network (<out>)



**Figure 8:** png

It is noticeable that some nodes are shrinking and in the table you can see that they have the value 0 for outgoing edges. This is because these are the consortia that were already approved in the first funding round and therefore did not submit a new Letter of Intent. After all, our dataset only considers the Letters of Intent from the second funding round. The consortia of the first round can therefore only be mentioned as "passive" cooperation partners.

---

## Network analysis

After the previous rounds of network visualization, let's go one step further and analyze the network structure.

### NFDI conference systematics

As a first step, let's color the nodes or consortia in the colors of the NFDI conference systematics.

How does the NFDI conference systematics come about? Five panels have been set up for the presentations:

1. Medicine
2. Life Sciences
3. Humanities
4. Engineering Sciences
5. Chemistry/Physics

The applicant consortia were divided among these five groups:<sup>11</sup>

<b>Uhrzeit</b>	<b>Session 1</b> <b>Medizin</b>	<b>Session 2</b> <b>Lebens- wissenschaften</b>	<b>Session 3</b> <b>Geistes- und Sozialwissen- schaften</b>	<b>Session 4</b> <b>Informatik, Mathematik und Ingenieur- wissenschaften</b>	<b>Session 5</b> <b>Chemie und Physik</b>
11:00 – 11:05	Einführung	Einführung	Einführung	Einführung	Einführung
11:05 – 11:20	<b>NFDI4Health</b>	<b>NFDI4BioDiv.</b>	<b>KonsortSWD</b>	<b>NFDI4Ing</b>	<b>NFDI4cat</b>
11:20 – 11:35	<b>GHGA</b>	<b>DataPLANT</b>	<b>BERD@NFDI</b>	<b>NFDI4MobilTech</b>	<b>NFDI4Chem</b>
11:35 – 11:50	<b>NFDI-Neuro</b>	<b>NFDI4Microbio.</b>	<b>NFDI4Culture</b>	<b>MARDI</b>	<b>FAIRmat</b>
11:50 – 12:10	Fragerunde	Fragerunde	Fragerunde	Fragerunde	Fragerunde

**Figure 9:** NFDI conference systematics

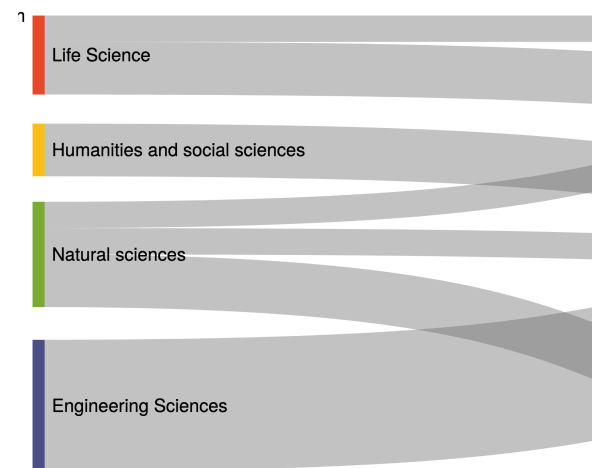
---

<sup>11</sup>[https://www.dfg.de/download/pdf/foerderung/programme/nfdi/nfdi\\_konferenz\\_2020/programm\\_webkonferenz\\_2020.pdf](https://www.dfg.de/download/pdf/foerderung/programme/nfdi/nfdi_konferenz_2020/programm_webkonferenz_2020.pdf)

---

In the following, we abbreviate Group 4 "Computer Science, Mathematics and Engineering" as "Engineering".

It is noticeable that according to the DFG subject classification system, the natural sciences have been divided between the life sciences, engineering sciences and chemistry/physics, as can be seen in the following Sankey (flow chart).



between DFG subject classification and NFDI conference classification.

So all consortia have been assigned to one of these five areas and we now want to show this in the network. We load this classification of the consortia on the conference system in the next cell.

This new record is passed to the variable 'NFDI<sub>nodes</sub>'; the first column contains the consortium names, the second column the number from the NFDI-/conference/systematics. The third column contains the round in which the consortium was approved: 1= 2019, 2= 2020.

The data can be read from the public GitHub gist <https://gist.github.com/LukasCBossert/ce56ebd0059b4879c7d11c1090118c25>.

```
# https://gist.github.com/LukasCBossert/ce56ebd0059b4879c7d11c1090118c25
NFDI_nodes <- read.table(header=TRUE,
                          sep=" ",
                          text="
name,group,round
DataPLANT,2,1
GHGA,1,1
KonsortSWD,3,1
NFDI4BioDiversity,2,1
NFDI4Cat,5,1
NFDI4Chem,5,1
NFDI4Culture,3,1
NFDI4Health,1,1
NFDI4Ing,4,1
BERD@NFDI,3,2
```

---

```

DAPHNE4NFDI,5,2
FAIRmat,5,2
MaRDI,4,2
NFDI-MatWerk,4,2
NFDI4DataScience,4,2
NFDI4Earth,2,2
NFDI4Microbiota,2,2
PUNCH4NFDI,5,2
Text+,3,2
")

```

Now we still have to create a graph dataset from the dataset, which is again done with `graph_from_data_frame`. What is new is that we now differentiate what is our edge data frame and what is the list with the nodes.

```

NFDI_network_directed <- graph_from_data_frame(d = NFDI_edges,      # d =
  ↪ data frame ~ edges

                                vertices = NFDI_nodes,
                                ↪ #nodes
                                directed = TRUE)
                                ↪ #directed

```

## Color nodes according to funding year

We set up the coloring based on the year of the funding. The column `round` is considered and depending on the number (1 or 2) the color is chosen.

```

NFDI_color_year <- c("lightgreen", # 2019 (1)
                    "lightblue"   # 2020 (2)
                    )
NFDI_color_groups <- NFDI_color_year[
  as.numeric(as.factor(
    V(NFDI_network_directed)$round))] # <<<< based on the round

```

Great, so now we can apply the color system on the network.

```

set.seed(9876543)

plot(NFDI_network_directed,      # loading data frame
     main = "NFDI-Network (<Funding year>)", #<<<<<<< adding a title
     frame = TRUE,                # making a frame
     layout = layout.graphopt,    # better layout options
     vertex.color = NFDI_color_groups, #<<<<<<<<< color of nodes

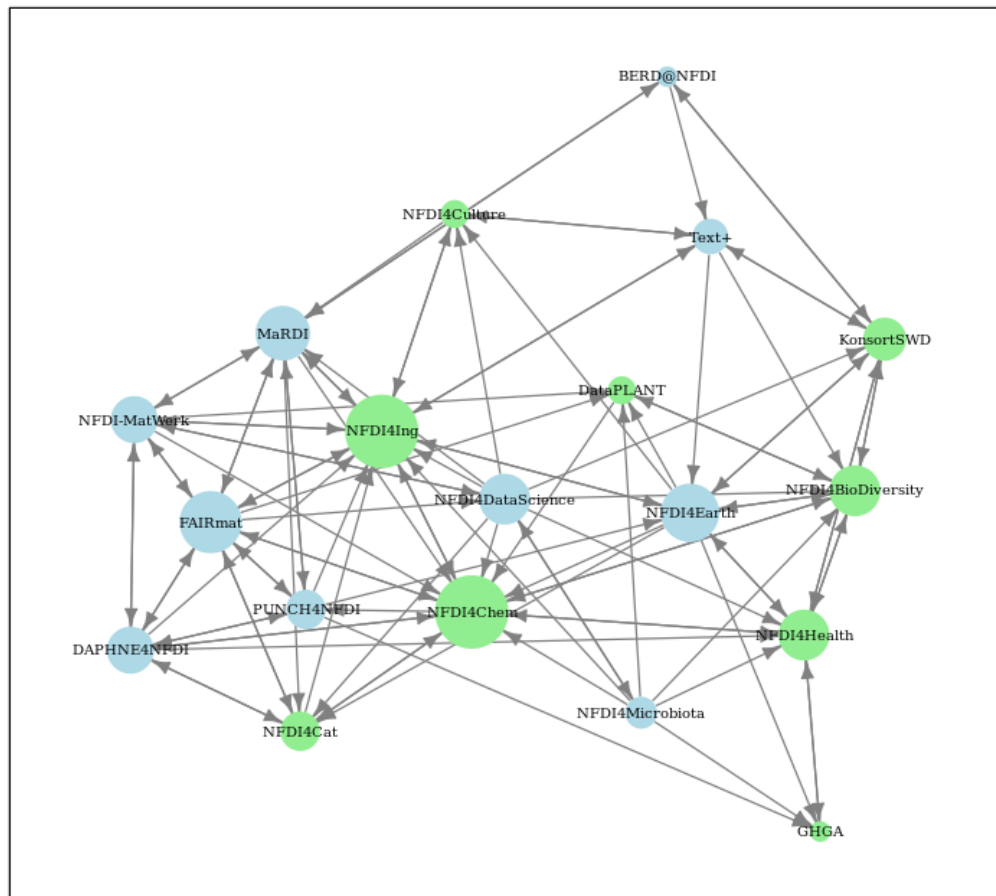
```

---

```
vertex.frame.color = NFDI_color_groups,    #<<<<<<<<< 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_picke
                                          ↳
edge.color          = "#808080",          # color of edges
edge.curved         = 0,                  # factor of "curvity"
vertex.size         = degree(NFDI_network_directed,
                               mode = "total"), #<<<<<<<<<< size of
                               ↳ nodes depends on amount of edges
edge.arrow.size     = .5,                # arrow size, defaults to 1
)
```

---

## NFDI-Network (<Funding year>)



**Figure 10:** png

### DFG/NFDI color coding

In order to better recognize the node classification on the NFDI conference systematics in the network, we choose a color coding according to the DFG subject systematics (slight adjustment if necessary).

The following values apply

---

No.	Designation	HTML color code
(1)	Medicine	#f5ac9f
(2)	Life Sciences	#e43516
(3)	Humanities	#f9b900
(4)	Engineering Sciences	#007aaf
(5)	Chemistry/Physics	#6ca11d

---

We now pass these color values in sequence to the variable 'NFDI<sub>colorcode</sub>', thereby the color values are written into a list. Using the function `c` the values are written into a vector,<sup>12</sup> with which we can continue.

Now we have to establish the link between the color value and the consortia. For this we introduce the variable `NFDI_color_groups`: Each value from `NFDI_color_code` has a position number (1-5), we use this by evaluating the value of the second column of the network graph (`$group`) as a number and thus passing the color value. Simplified and from the result, the NFDI conference system number gets the color value that is in the corresponding position in the list of the variable `NFDI_color_code`.

```
NFDI_color_code <- c("#f5ac9f", # Medicine
                     "#e43516", # Life Sciences
                     "#f9b900", # Humanities
                     "#007aaf", # Engineering Sciences
                     "#6ca11d"  # Chemistry/Physics
                     )
NFDI_color_groups <- NFDI_color_code[
  as.numeric(as.factor(
    V(NFDI_network_directed)$group))]
```

## Network with colored nodes

We can again take the code from the previous cell and adapt it.

It is crucial that we specify the variable `NFDI_color_groups` as value for `vertex.color` and `vertex.frame.color`. We also want to consider and display the entire network with all edges (`mode = "total"`).

What is missing now is a legend so that we can also see what is behind the color coding.

---

<sup>12</sup><https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/c>

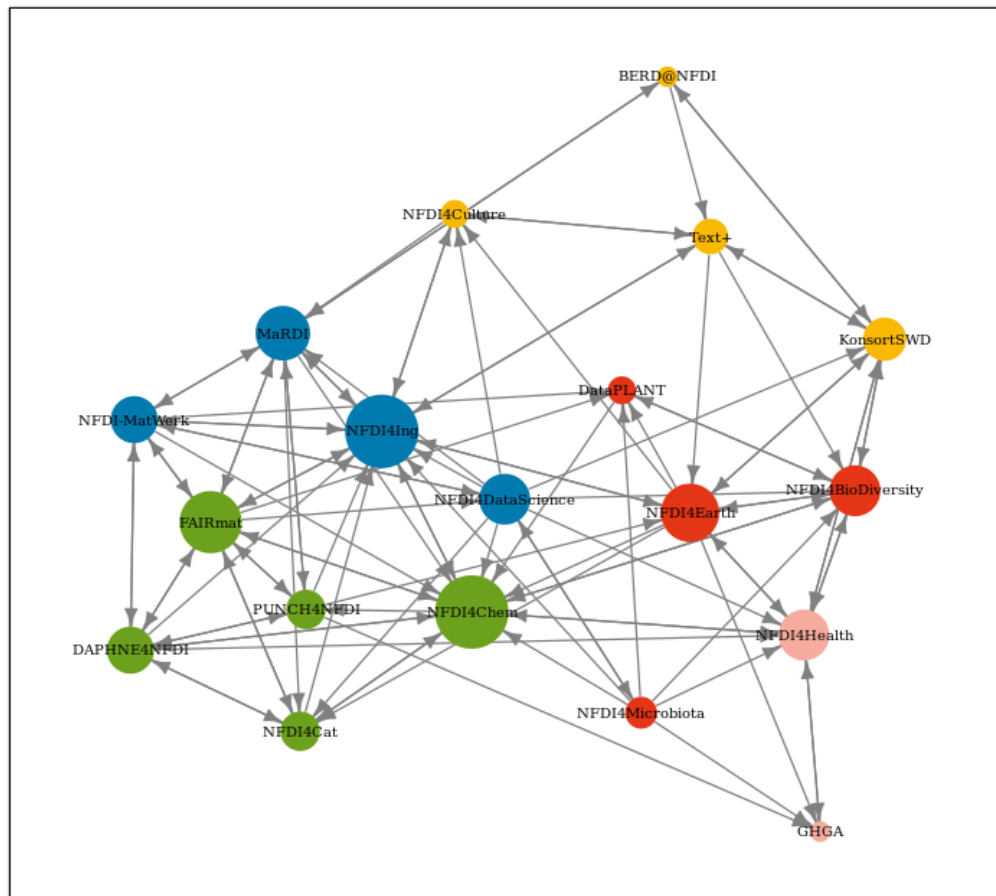


---

```
set.seed(9876543)
```

```
plot(NFDI_network_directed,          # loading data frame
     main = "NFDI-Network (<NFDI conference systematics>)", #<<<<<<<<
     ↪ adding a title
     frame = TRUE,                   # making a frame
     layout = layout.graphopt,      # better layout options
     vertex.color = NFDI_color_groups, #<<<<<<<<< color of nodes
     vertex.frame.color = NFDI_color_groups, #<<<<<<<<< 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_picke
                                     ↪
     edge.color = "#808080",         # color of edges
     edge.curved = 0,                # factor of "curvity"
     vertex.size = degree(NFDI_network_directed,
                           mode = "total"), #<<<<<<<<<< size of
                           ↪ nodes depends on amount of edges
     edge.arrow.size = .5,           # arrow size, defaults to 1
)
```

## NFDI-Network (<NFDI conference systematics>)



**Figure 11:** png

Ok, we want to add a legend now and since we want to define it only once we make it as a function, which we now fill with values:

- First the positioning of the legend, which we want to have bottom right, then the title (`title = "NFDI conference systematics"`), now comes the content of the legend, which is controlled by the `legend` parameter: For this we again build a list (`c()`), in which we enter the desired values.

- `col`: With `col` we set the color scheme and we can directly refer to the NFDI color list via the variable `NFDI_color_code`.
- `pch`: We must not forget the `pch` parameter, because it is used to define the symbol in the legend. With the value 20 we select a filled circle.
- `bty`: With `bty` and the value `n` for no we do without a frame around the legend.
- `cex` (so character expansion) is again a relative value and we can specify the font size; similarly, `pt.cex` works for the legend symbols.

```
nfdi_plot_legend <- function(){

  legend("topleft", # x-position
        title = "NFDI conference systematics", # title
        legend = c(
          "(1) Medicine",
          "(2) Life Sciences",
          "(3) Humanities",
          "(4) Engineering Sciences",
          "(5) Chemistry/Physics"
        ), # the text of the legend
        col = NFDI_color_code , # colors of lines and points beside the
        ↪ legend text
        pch = 20, # the plotting symbols appearing in the legend
        bty = "n", # no frame, the type of box to be drawn around the
        ↪ legend (n=no frame)
        cex = .75, # character expansion factor relative to current
        ↪ par("cex").
        pt.cex = 2 # expansion factor(s) for the points
      )
}
```

Now we add the legend to the plot.

```
set.seed(9876543)

plot(NFDI_network_directed, # loading data frame
     main = "NFDI Network (<NFDI conference systematics>)", #<<<<<<<<
     ↪ adding a title
     frame = TRUE, # making a frame
     layout = layout.graphopt, # better layout options
     vertex.color = NFDI_color_groups, # color of nodes
     vertex.frame.color = NFDI_color_groups, # 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,
    ↪ mode = "total"), #<<<<<<<<<< size of
    ↪ nodes depends on amount of edges
edge.arrow.size     = .5,          # arrow size, defaults to 1
)
nfdi_plot_legend()

```

## NFDI Network (<NFDI conference systematics>)

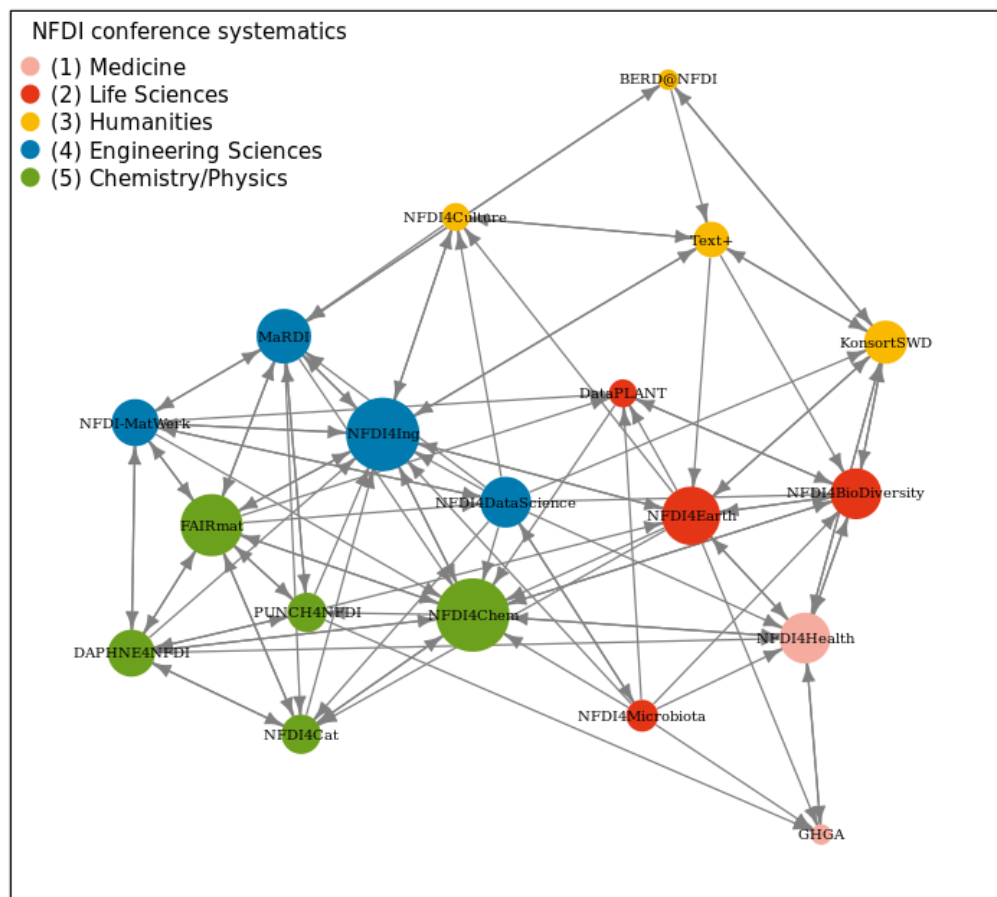


Figure 12: png

### Additional stuff

Let us concentrate on only one consortium and display the connection from or to this consortium.

```
nfdi_plot_group <- function(NFDI_name) {  
  
  set.seed(9876543)  
  nfdi_local_network <- function(NFDI_name) {
```

---

```

plot(NFDI_network_directed,
  main = "NFDI Network (<NFDI conference systematics>)", # adding a
  ↪ title
sub = NFDI_name,
  frame = TRUE, # making a frame
  layout = layout.graphopt, # better layout options
  vertex.color = NFDI_color_groups, # color of nodes
  vertex.frame.color = NFDI_color_groups, # 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.curved = 0.2, # factor of "curvity"
  vertex.size = degree(NFDI_network_directed,
    mode = "total"), #<<<<<<<<<< size of
    ↪ nodes depends on amount of edges
  edge.arrow.size = .5, # arrow size, defaults to 1
  edge.color = with(NFDI_edges,
    ifelse(from %in% NFDI_name,"#808080", # grey
    ifelse(to == NFDI_name,"#000000", # black
    NA)))
)
nfdi_plot_legend()

}

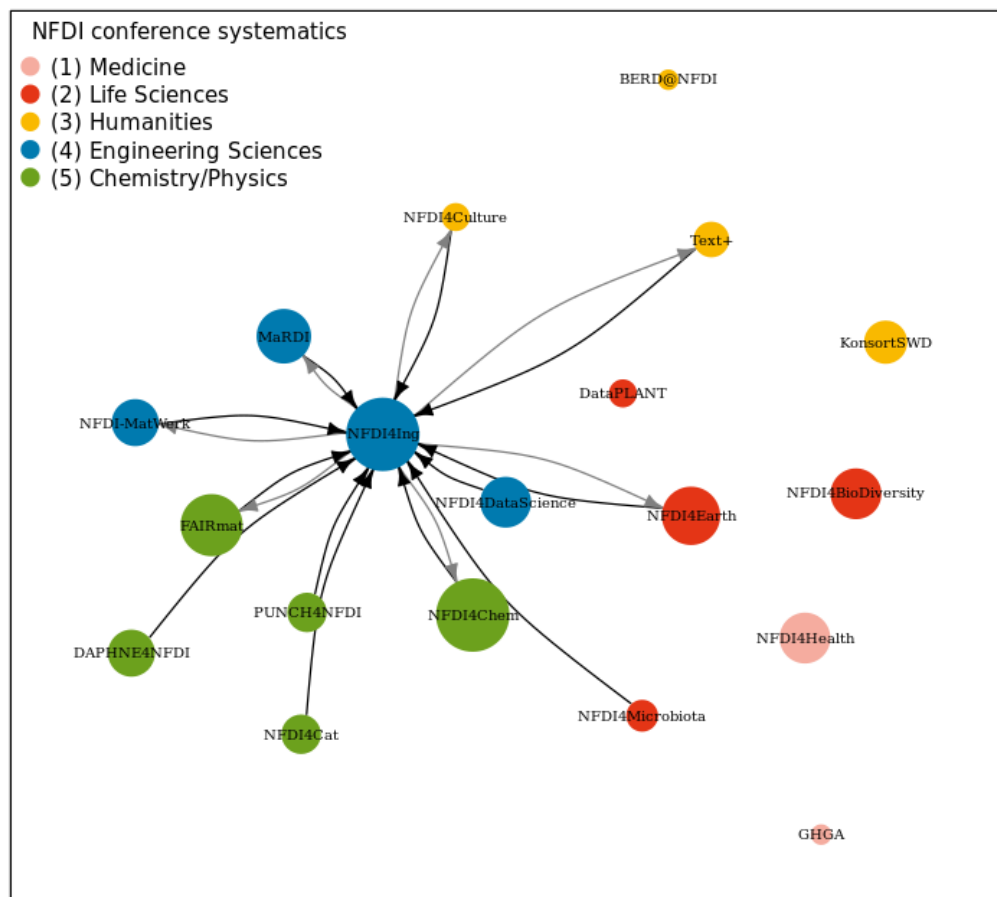
#####
### ! important!
### If you want to export the plots,
### you need to create a folder called "img" first.
#####
# pdf(paste0("img/network_group_",NFDI_name,".pdf")) # save image as
↪ PDF
# nfdi_local_network(NFDI_name) # display image for saving
# dev.off() # close image stream

nfdi_local_network(NFDI_name) # display image in JupyterNotebook
}
nfdi_plot_group("NFDI4Ing")

```

---

## NFDI Network (<NFDI conference systematics>)



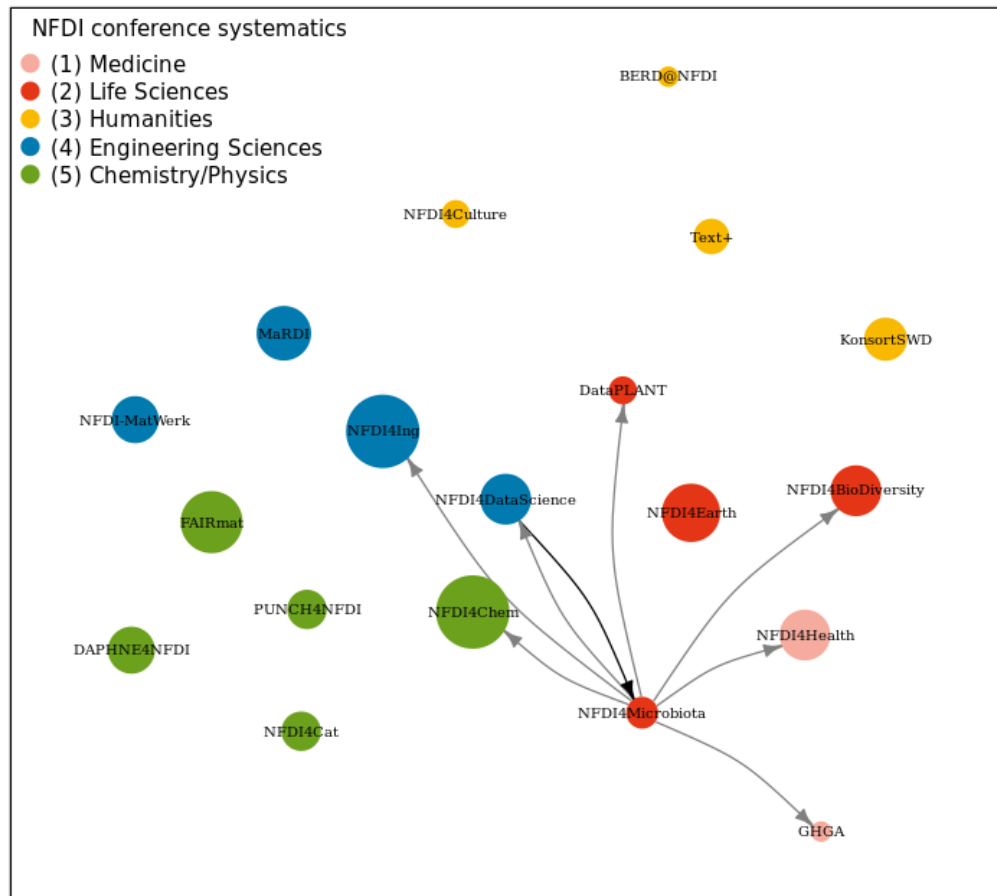
NFDI4Ing

**Figure 13:** png

Here is another consortium and its connections.

```
nfdi_plot_group("NFDI4Microbiota")
```

## NFDI Network (<NFDI conference systematics>)



NFDI4Microbiota

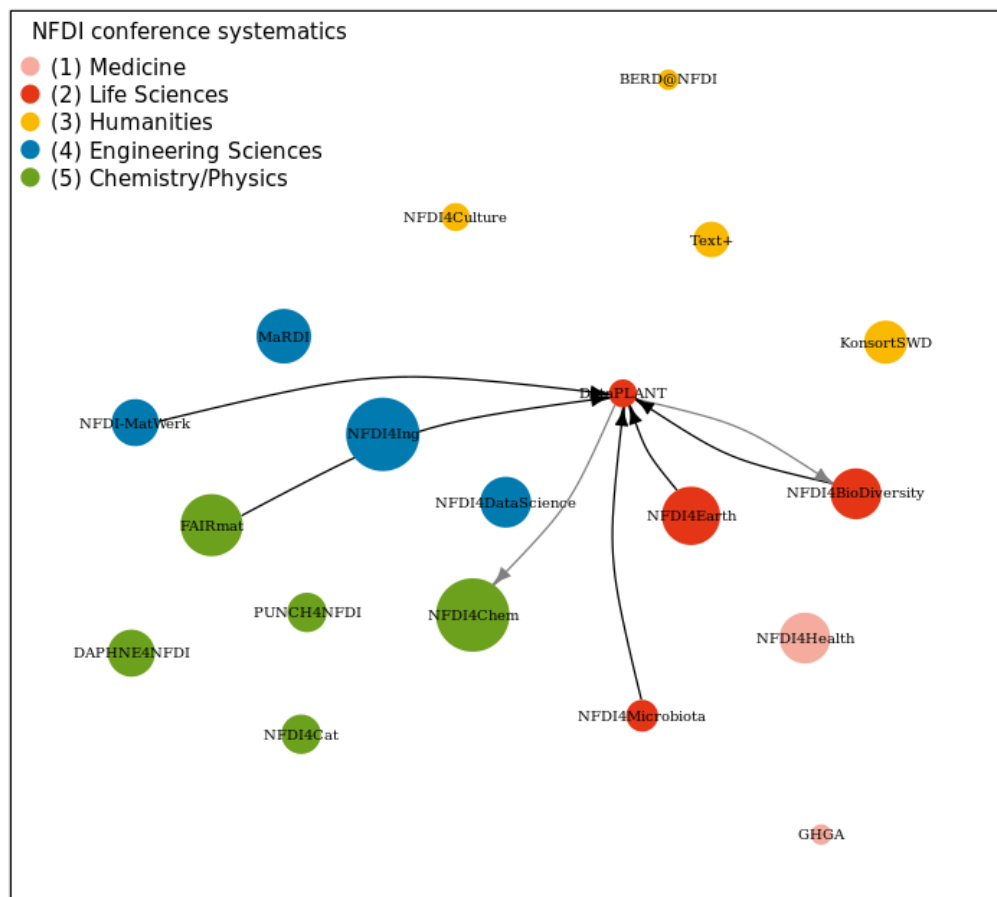
**Figure 14:** png

I love loops....

```
for (name in NFDI_nodes$name){
  nfdi_plot_group(name)
}
```



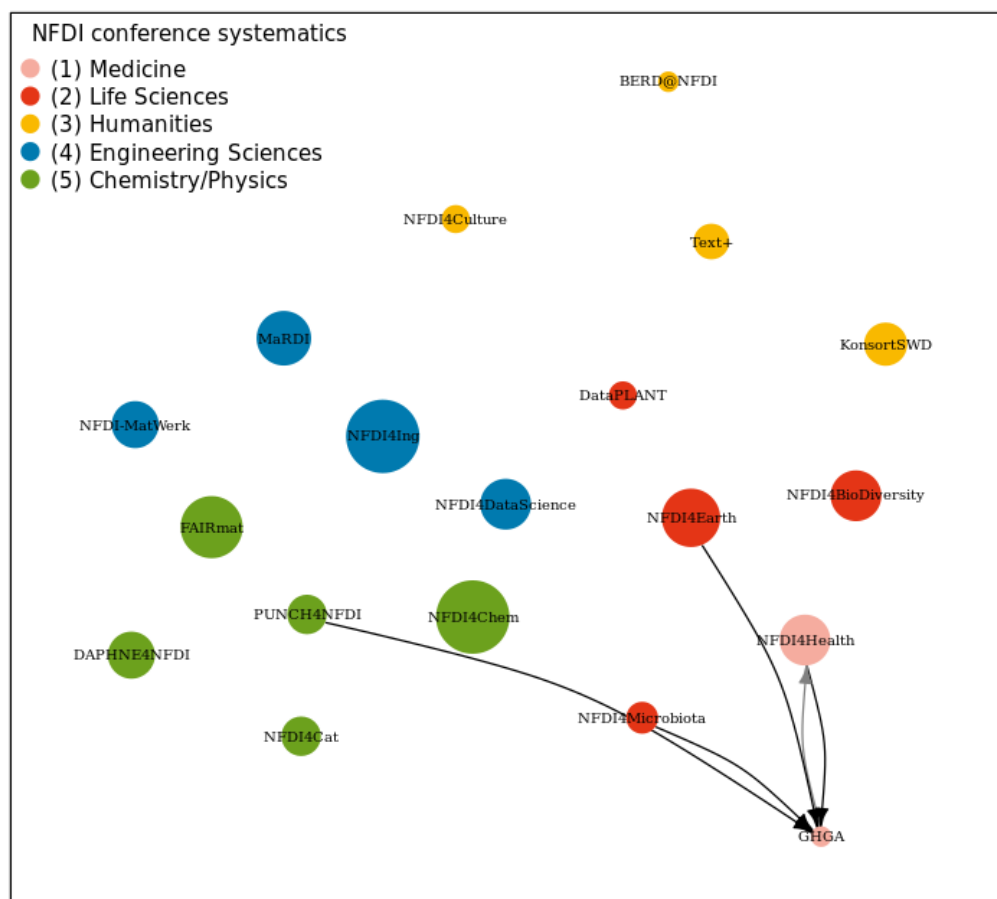
## NFDI Network (<NFDI conference systematics>)



DataPLANT

**Figure 15:** png

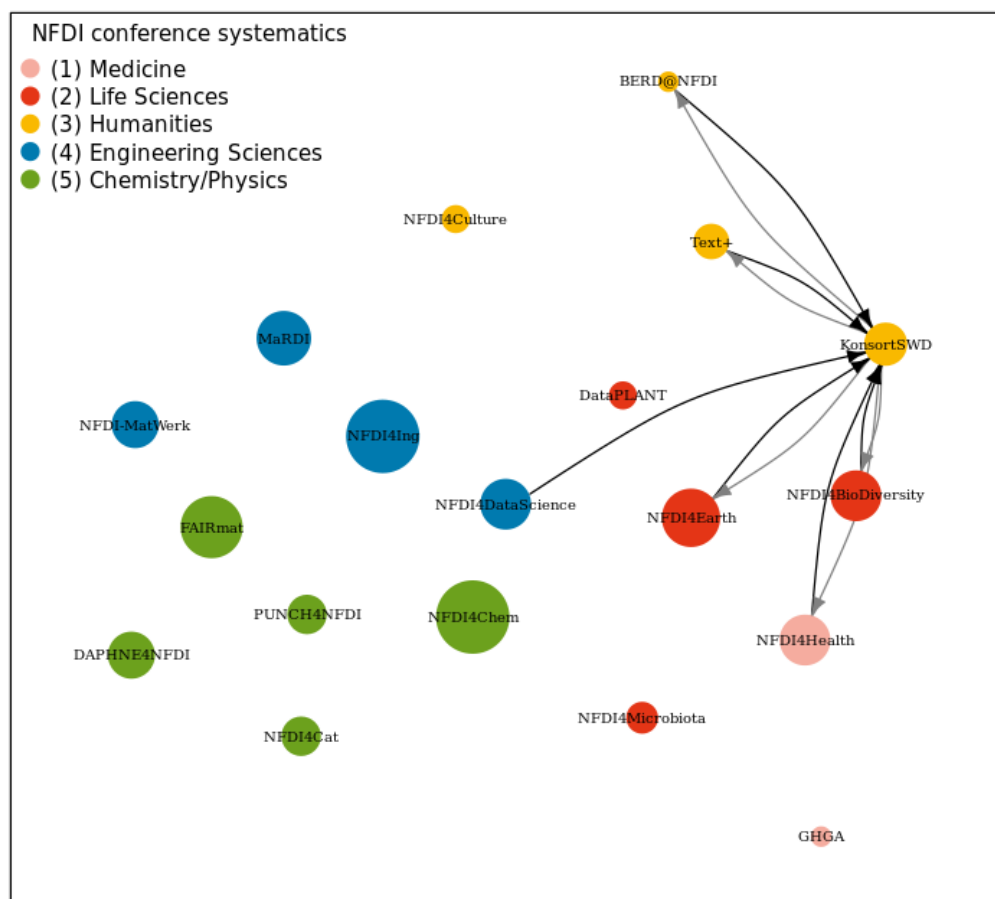
## NFDI Network (<NFDI conference systematics>)



GHGA

**Figure 16:** png

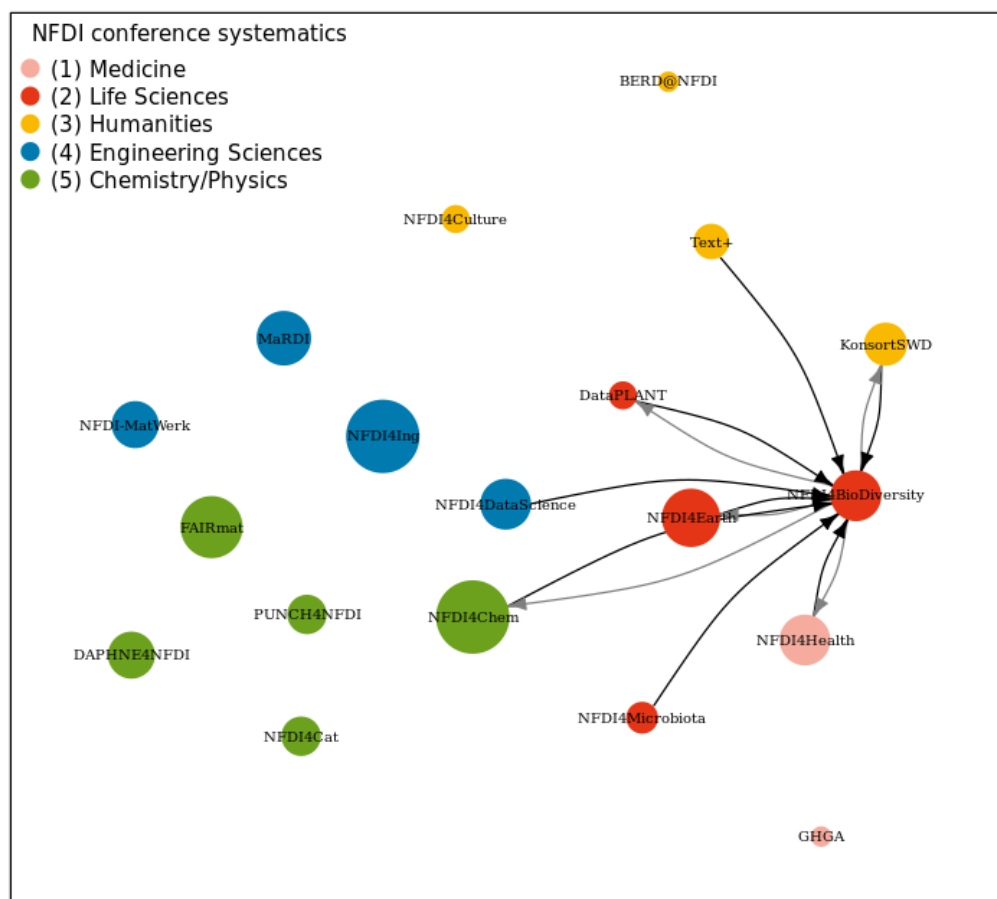
## NFDI Network (<NFDI conference systematics>)



KonsortSWD

**Figure 17:** png

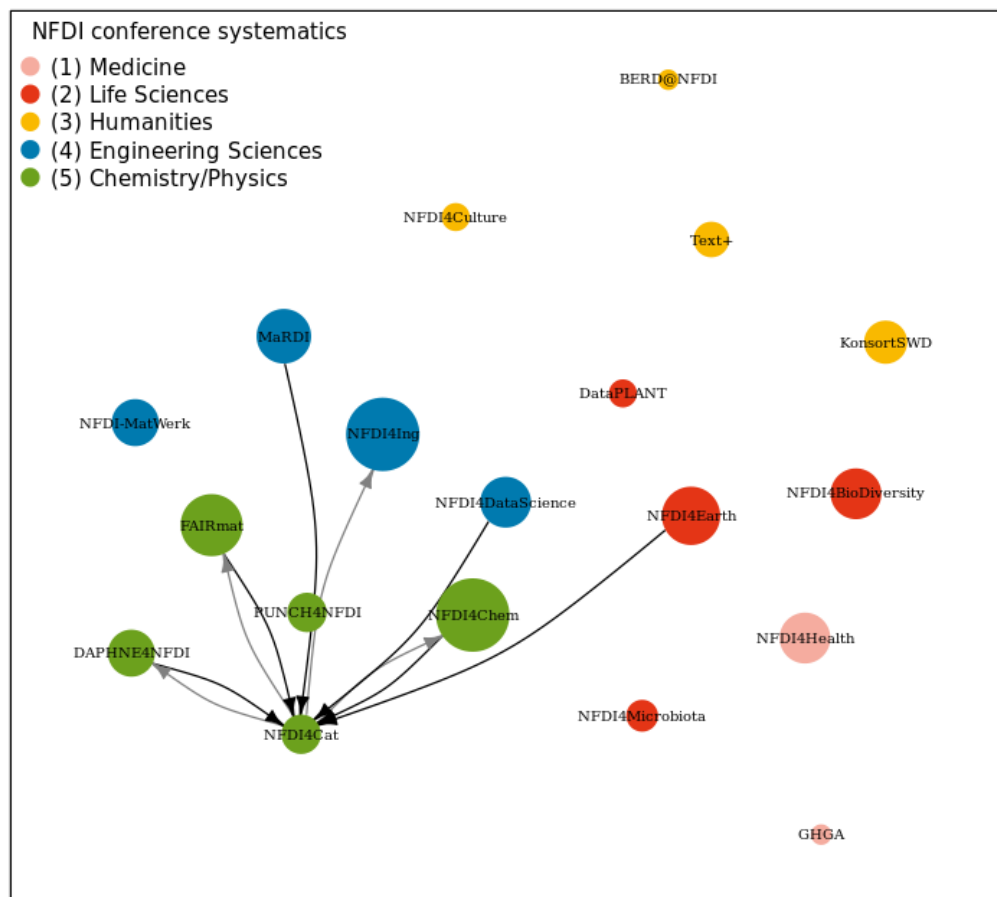
## NFDI Network (<NFDI conference systematics>)



NFDI4BioDiversity

Figure 18: png

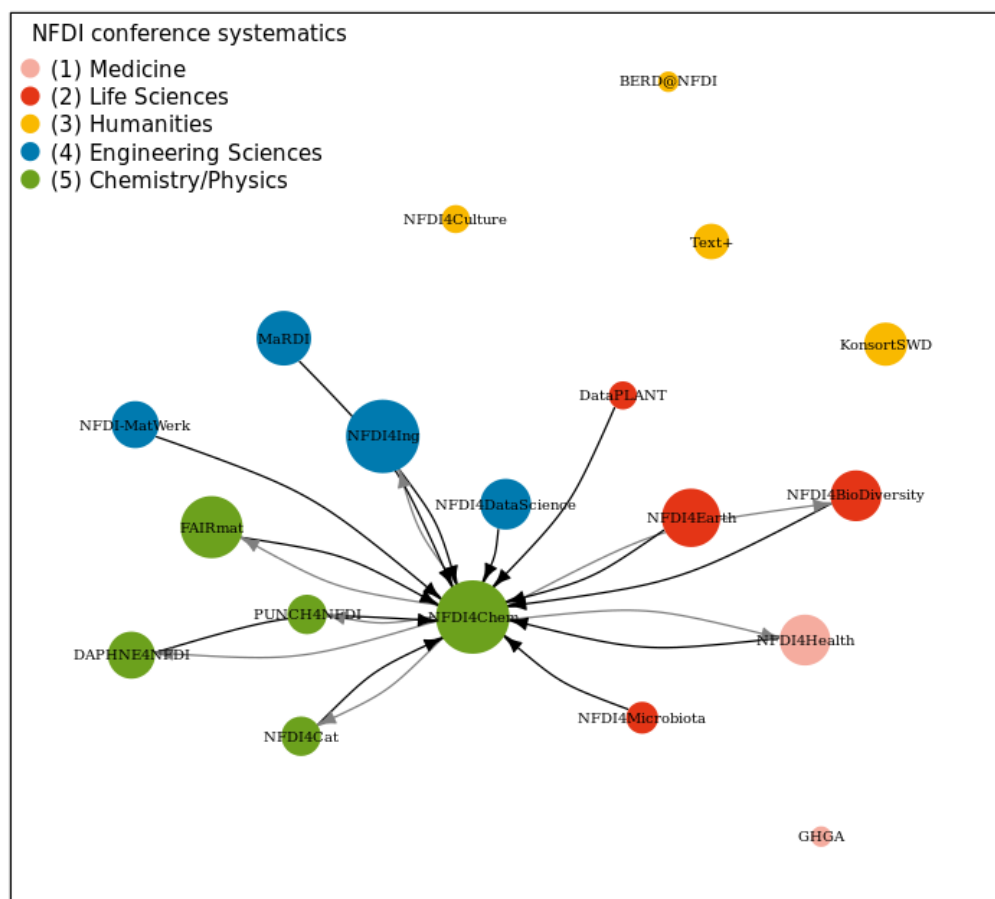
## NFDI Network (<NFDI conference systematics>)



NFDI4Cat

Figure 19: png

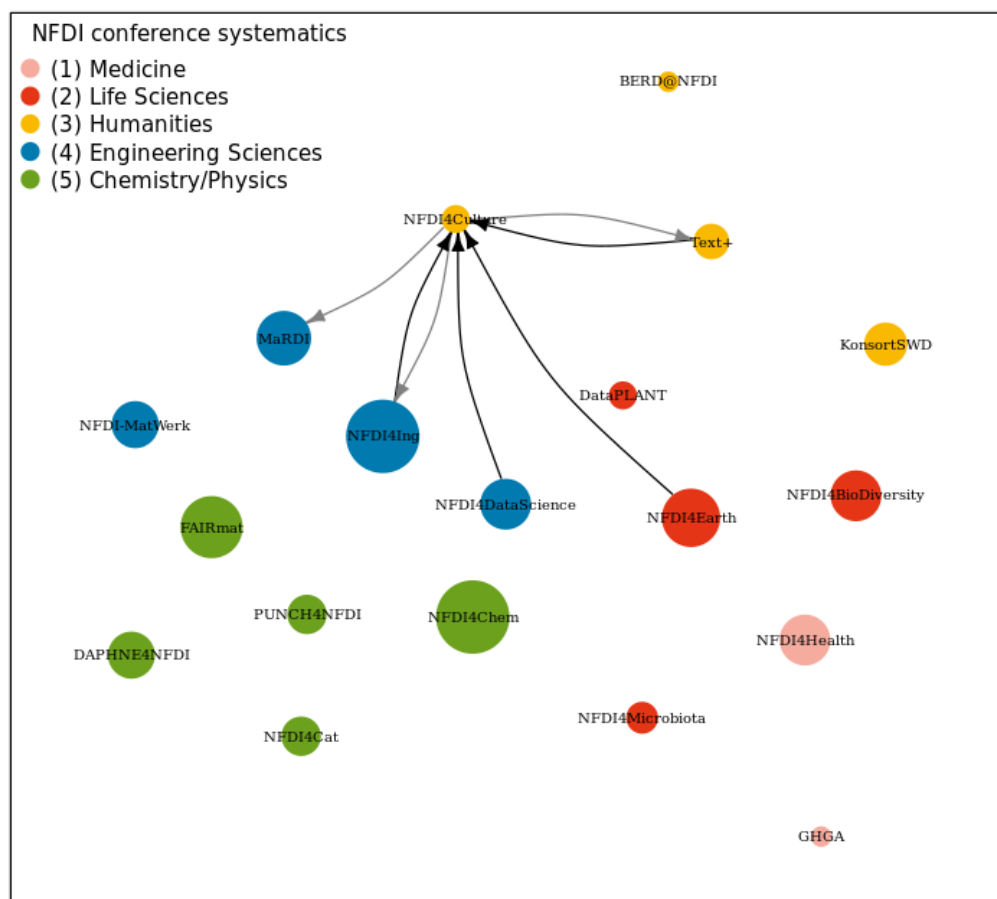
## NFDI Network (<NFDI conference systematics>)



NFDI4Chem

Figure 20: png

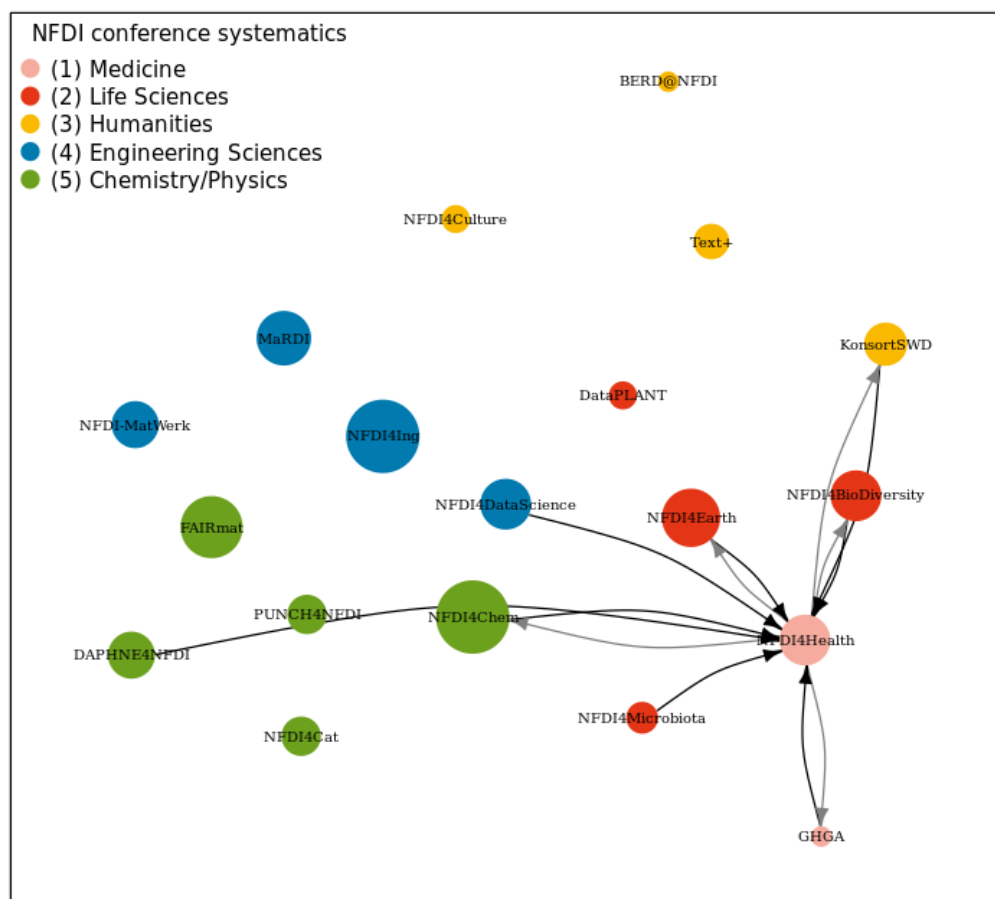
## NFDI Network (<NFDI conference systematics>)



NFDI4Culture

Figure 21: png

## NFDI Network (<NFDI conference systematics>)

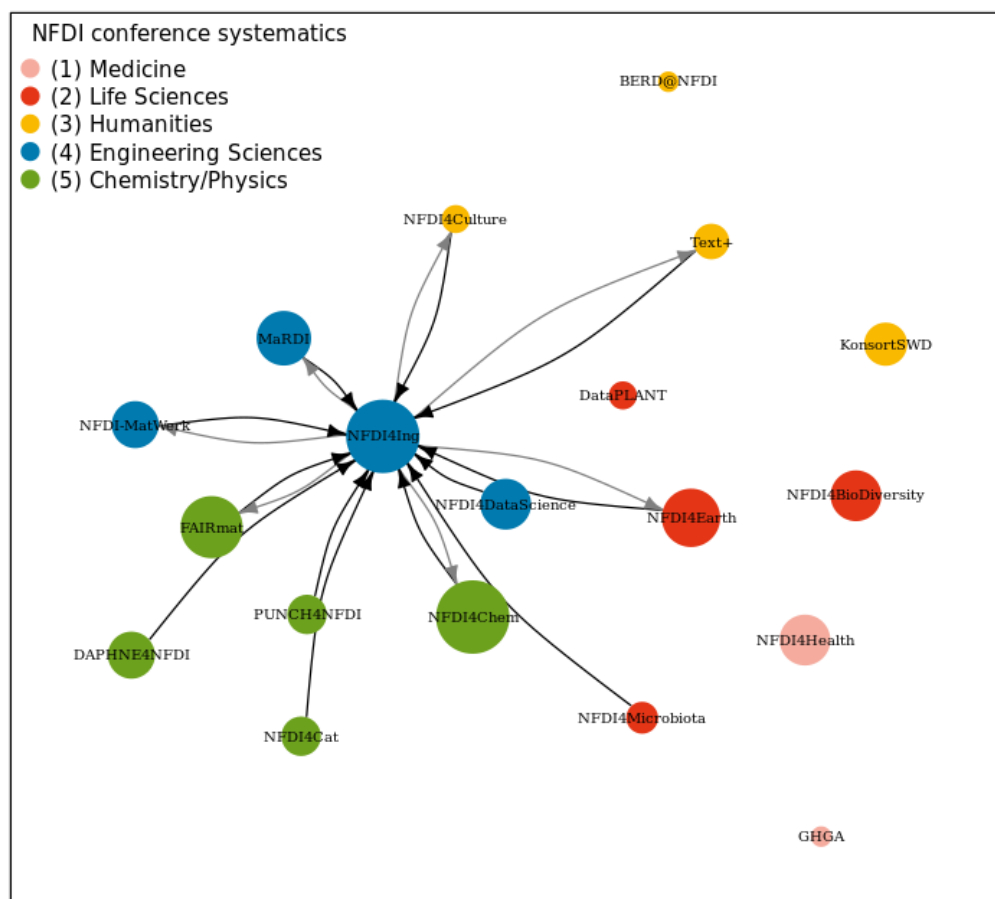


NFDI4Health

Figure 22: png



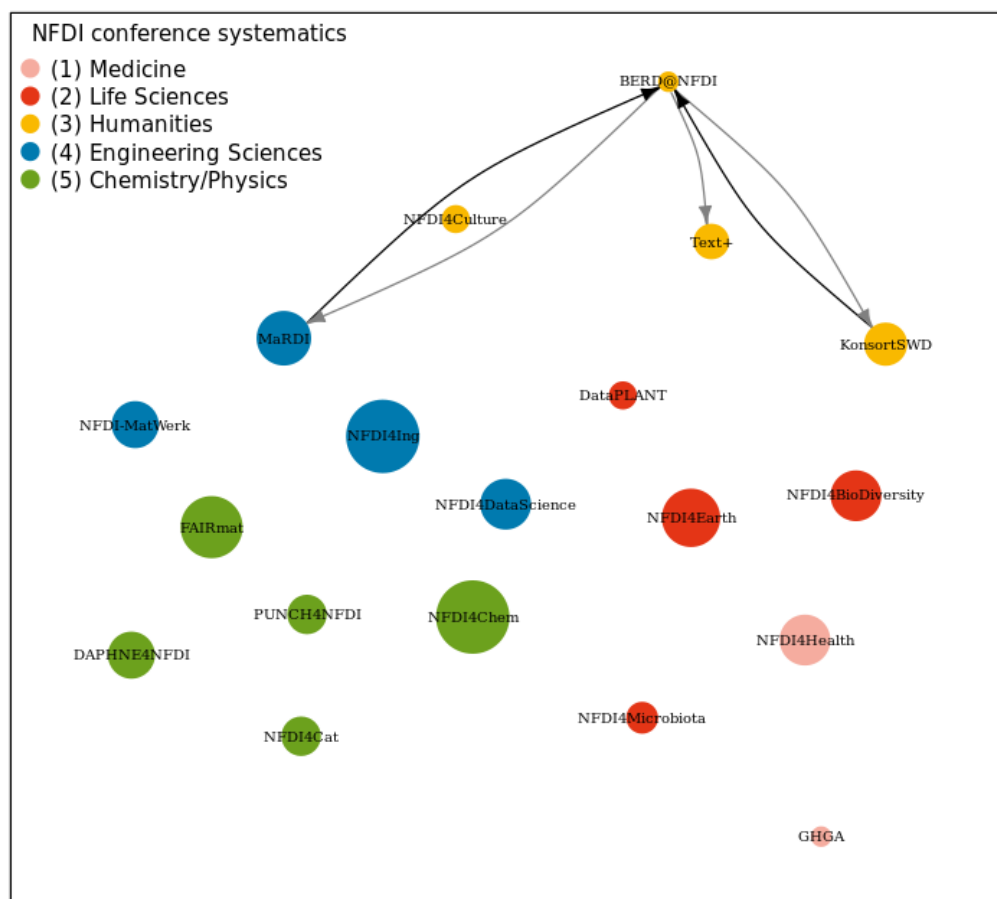
## NFDI Network (<NFDI conference systematics>)



NFDI4Ing

Figure 23: png

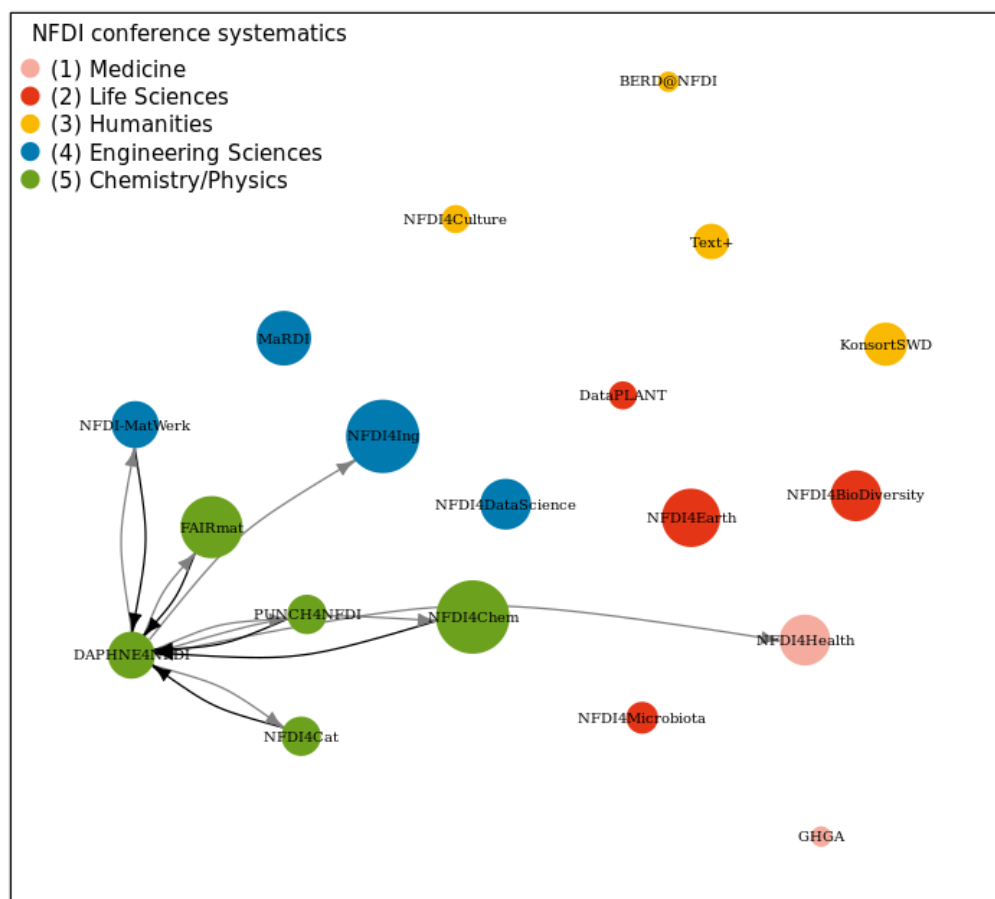
## NFDI Network (<NFDI conference systematics>)



BERD@NFDI

Figure 24: png

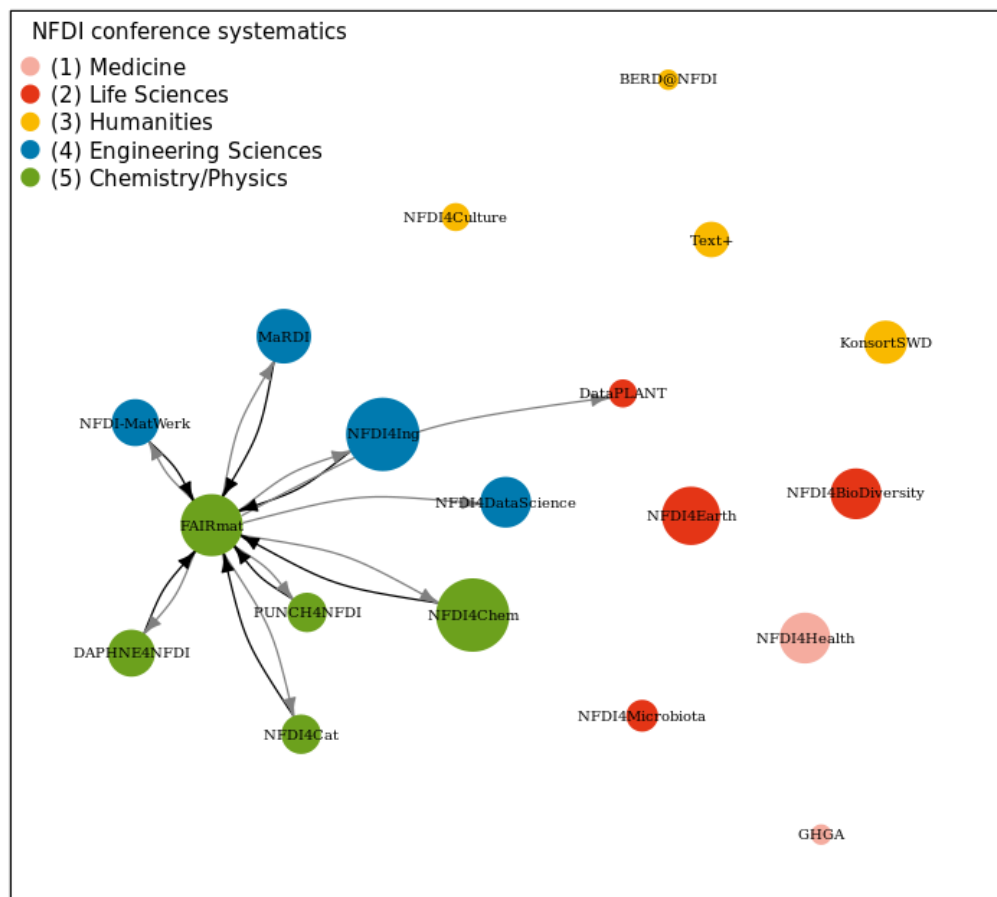
## NFDI Network (<NFDI conference systematics>)



DAPHNE4NFDI

Figure 25: png

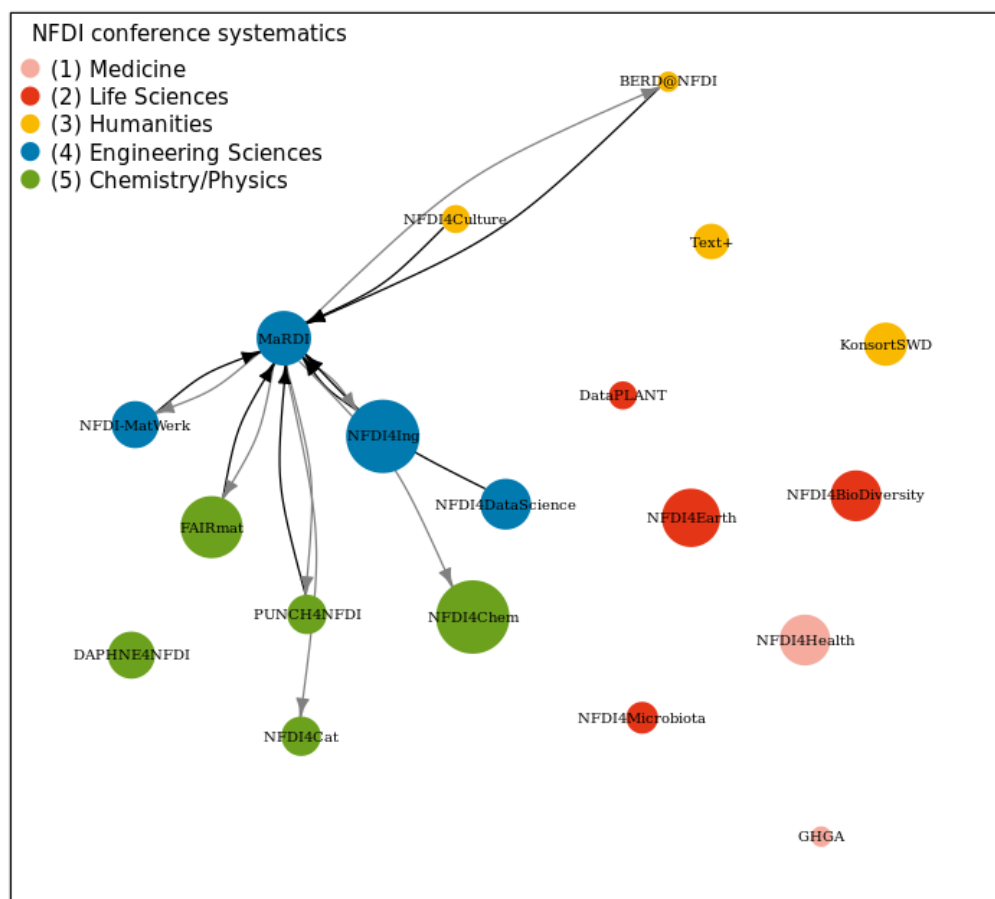
## NFDI Network (<NFDI conference systematics>)



FAIRmat

Figure 26: png

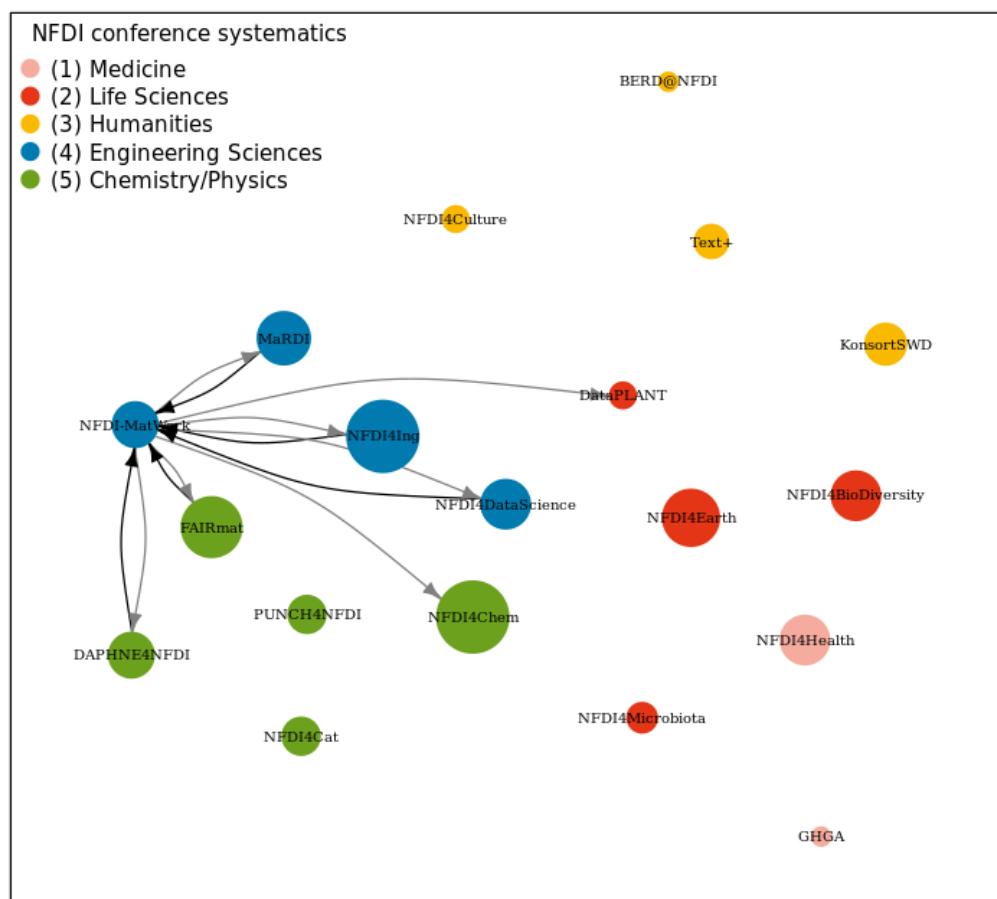
## NFDI Network (<NFDI conference systematics>)



MaRDI

**Figure 27:** png

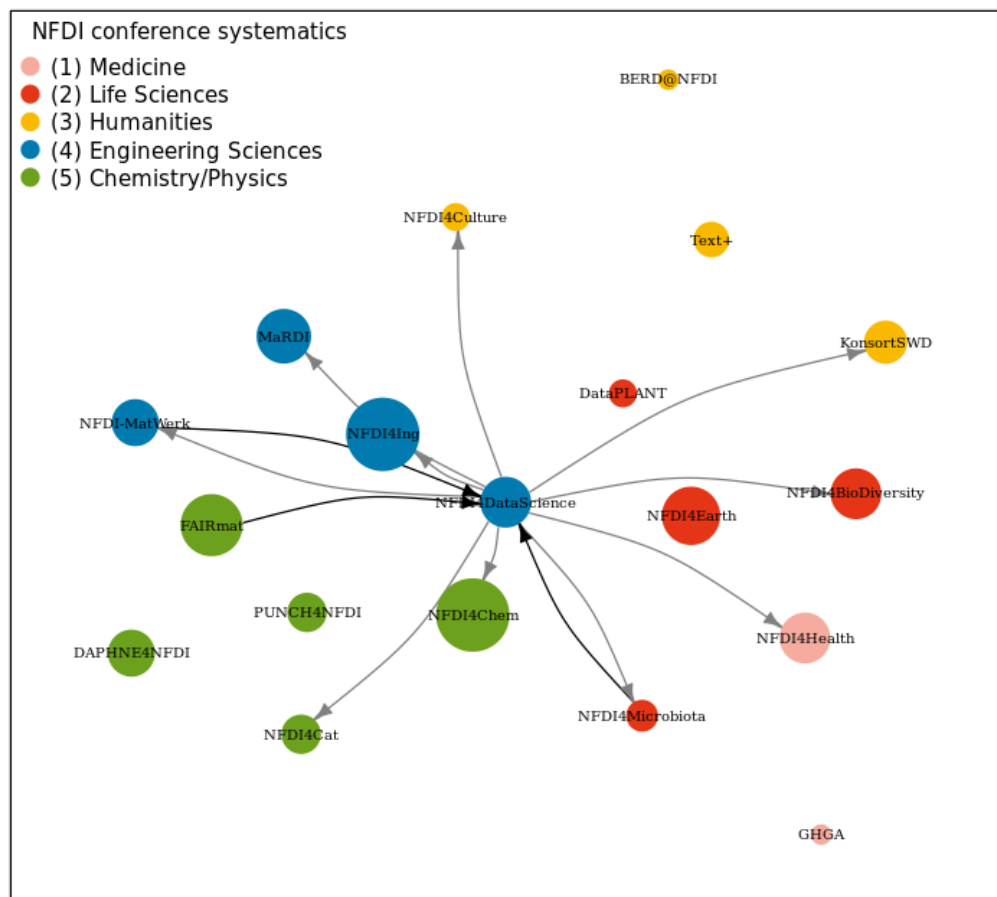
## NFDI Network (<NFDI conference systematics>)



NFDI-MatWerk

Figure 28: png

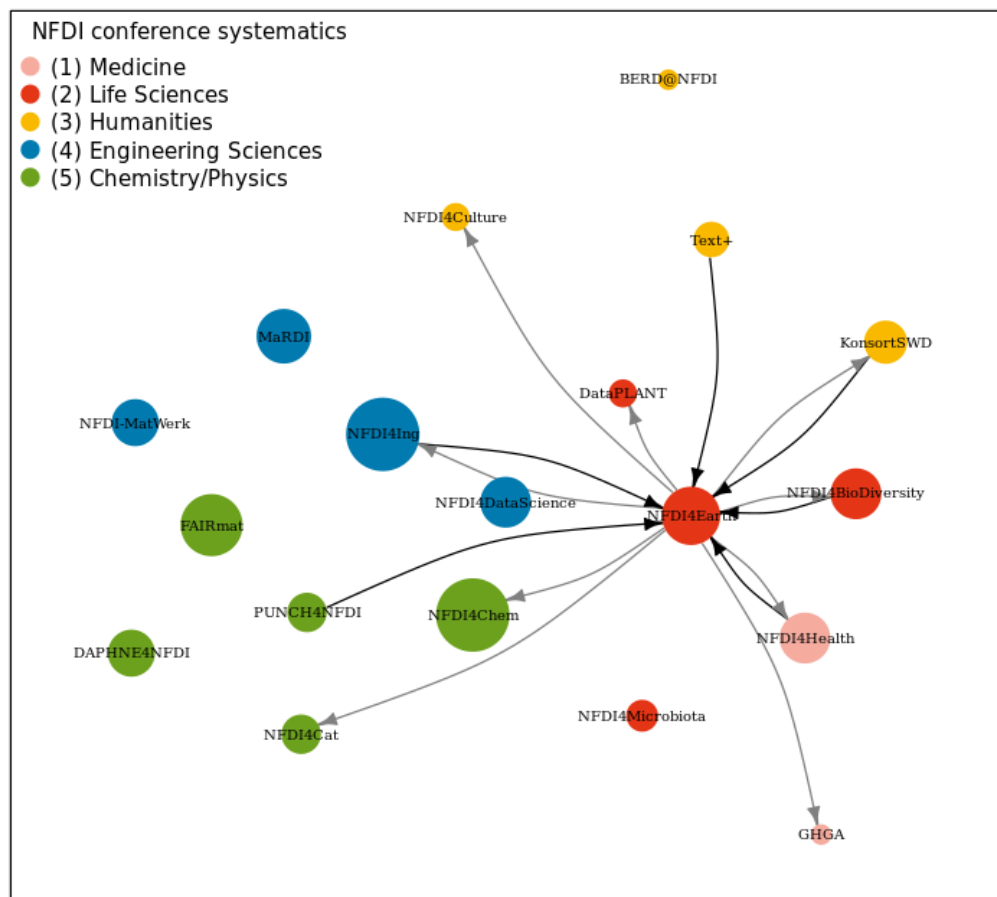
## NFDI Network (<NFDI conference systematics>)



NFDI4DataScience

Figure 29: png

## NFDI Network (<NFDI conference systematics>)

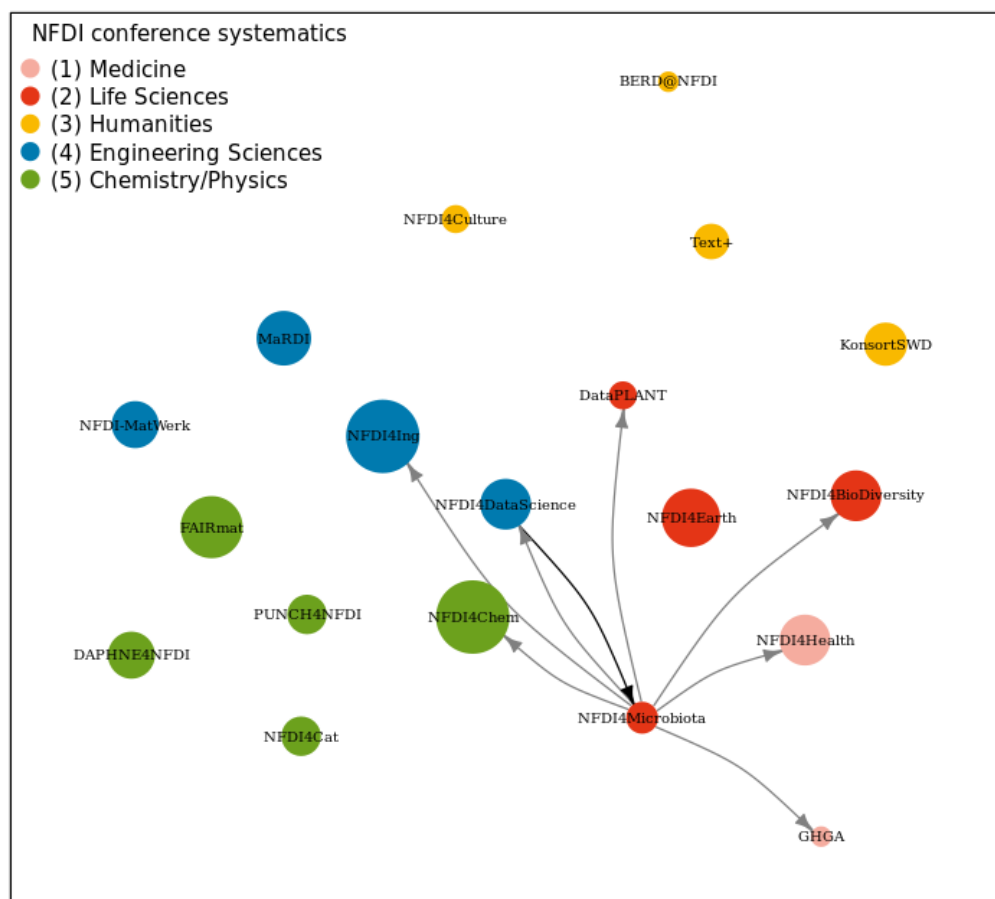


NFDI4Earth

Figure 30: png



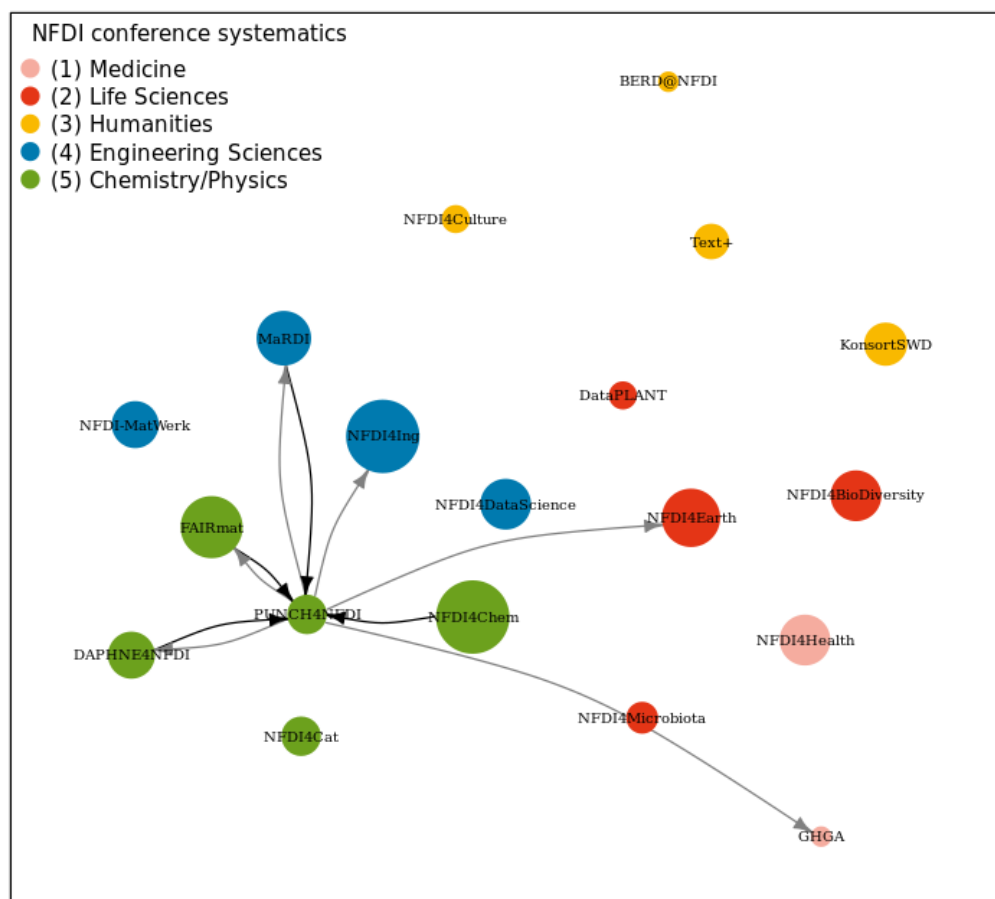
## NFDI Network (<NFDI conference systematics>)



NFDI4Microbiota

**Figure 31:** png

## NFDI Network (<NFDI conference systematics>)

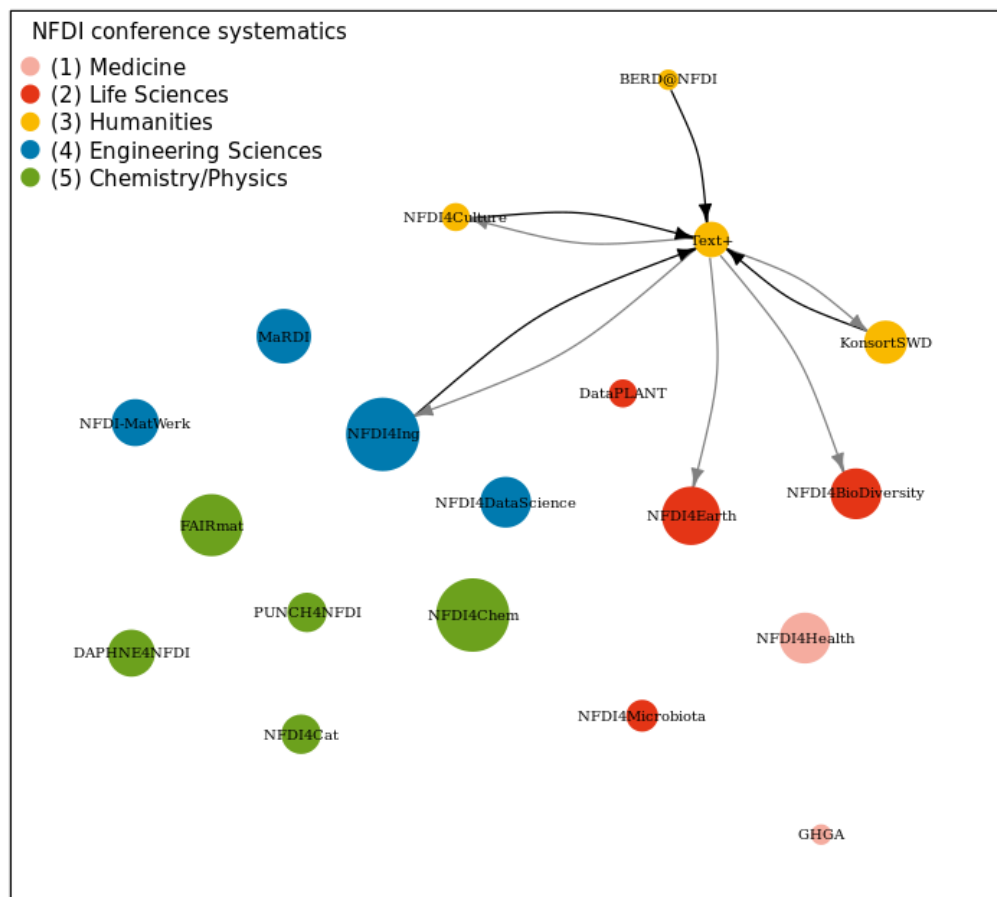


PUNCH4NFDI

Figure 32: png

---

## NFDI Network (<NFDI conference systematics>)



Text+

**Figure 33:** png

## Backup, export and outlook

We have done the network visualization and analysis using only the package 'igraph'. Now you have to save the result, e.g. under "File" -> "Save and Checkpoint". You can also download the JupyterNotebook, there are several formats available.

If you have created the network with the RNoteBook, you can call it up again at any time via the URL

---

and you can make further modifications in the network.

There are other exciting occupations with this network. For example, you can also create an interactive network or display the network as a pie chart. Have a look at the overview on <https://www.r-graph-gallery.com/network.html>.