In [1]:
```python
# Import the required packages
import pandas as pd
import numpy as np
from igraph import *
import random
random.seed(2021)
```

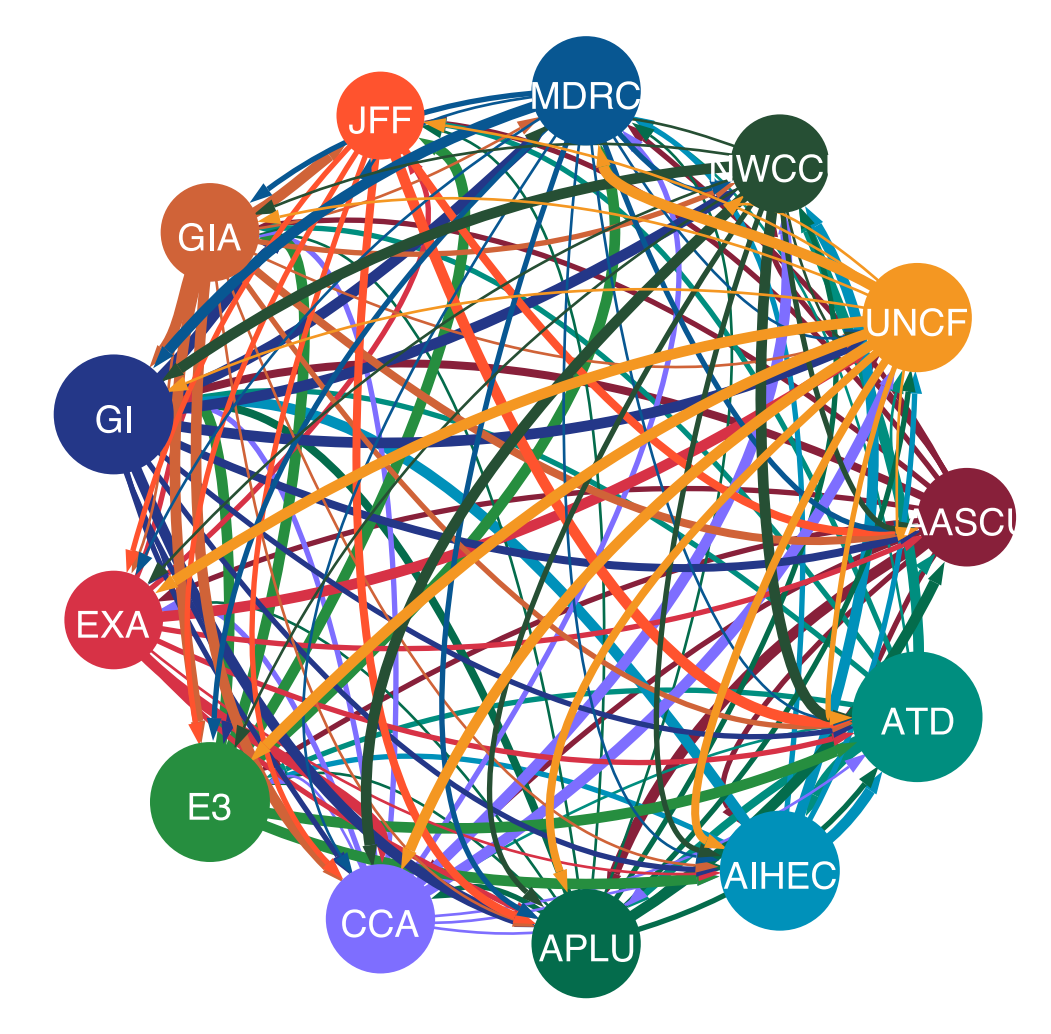
NIC Analysis

Yared Hurisa, Data Scientist, AIR

Network interaction

In [12]:
```python
# import sample clean dataset called attributes and interaction or df
# attribute used to create an empty n by n matrix
attributes = pd.read_excel(open('NIC_Clean_Data.xlsx', 'rb'), sheet_name='interaction')
attributes.rename(columns={'Unnamed: 0':'node_names'}, inplace=True)
attributes['value'] = 0
attributes['index'] = range(13)
attributes['col'] = range(13)

A = pd.pivot_table(attributes, values = 'value', index = 'index', columns =
'col').fillna(0).astype(int)
A = A.reset_index().drop('index', axis=1).astype(int)

# import the main data for processing
df = pd.read_excel(open('NIC_Clean_Data.xlsx', 'rb'), sheet_name='interaction')

df = df.rename(columns={"Unnamed: 0":"node_names"}).set_index(['node_names'])

node_names = df.columns

# standardize the node names with numbers
l=[]
for i in range(13):
    l.append(str(i))
df.columns = l

# get the adjecency matrix

B = df.reset_index().drop('node_names', axis=1)
m = (A.reindex_like(B).fillna(0) + B.fillna(0).fillna(0))
M = m.to_numpy()

# create graph
g = Graph.Adjacency(((M>0).tolist()), mode='directed', weighted=True)

# add edge weights and node labels.
g.es['weight'] = M[M.nonzero()]

# define network measures
network_measures = pd.DataFrame(dict(
    org = node_names,
    weight_in = g.strength(mode='in'),
    weight_out = g.strength(mode='out'),
    in_degree = g.degree(mode='in'),
    out_degree = g.degree(mode='out'),
     betweenness = g.betweenness(weights='weight'),
    edge_count = g.ecount(),
    density = g.density()

))

#  define vertix size
network_measures['weight_in'] = network_measures['weight_in']*(4)

g.vs['size'] = network_measures['weight_in']
vertix_size = g.vs['size']

# set vertix color
color = ["#88203A", "#008E7F", "#0091BA", "#046C4C",
                              "#7E6EFF", "#268E3F", "#D73246", "#243788",
                              "#D06338", "#FF532E", "#085792", "#264F34",
                              "#F49722"]
g.vs['color'] = color

vertix_color = g.vs['color']

#set edge color that match with vertix color

TEMP=[]
for e in g.es:
    elist = list(e.tuple)
    TEMP.append(elist)
g.es["color_assigned"] = TEMP
color_dict = {"0":"#88203A", "1":"#008E7F", "2":"#0091BA", "3":"#046C4C",
                              "4":"#7E6EFF", "5":"#268E3F", "6":"#D73246", "7":"#243788",
                              "8":"#D06338", "9":"#FF532E", "10":"#085792", "11":"#264F34",
                              "12":"#F49722"}
color_assigned = []
for i in range(129):
    color_assigned.append(g.es["color_assigned"][i][0])

edge_color = [color_dict[str(int(col_num))] for col_num in color_assigned]


# set vertix label
g.vs['label'] = node_names


## set layout
g_cr = g.layout_circle()
layout = g_cr
#set visual styel
visual_style = {}
out_name = "background_graph.png"
visual_style["vertex_color"] = g.vs['color']
visual_style["vertex_size"] = g.vs['size']
visual_style["vertex_frame_color"] = g.vs['color']
visual_style["vertex_label_color"] = 'white'
visual_style["vertex_label_cex"] = 0.7
visual_style["vertex_label_family"] = 'sans'
visual_style["vertex_label_font"] = 2
visual_style["edge_color"] = edge_color
visual_style["layout"] = layout
visual_style["edge_lty"] = 1
visual_style["edge_arrow_size"] = 0.6
visual_style["edge_width"] = g.es['weight']
visual_style["edge_curved"] = 0.2
visual_style["bbox"] = (400, 400)
visual_style["margin"] = 40
# export network measures in csv
network_measures.to_csv('network_measures.csv')
# plot the graph showing the entire network interaction
plot(g, out_name, **visual_style)
```

Out[12]:

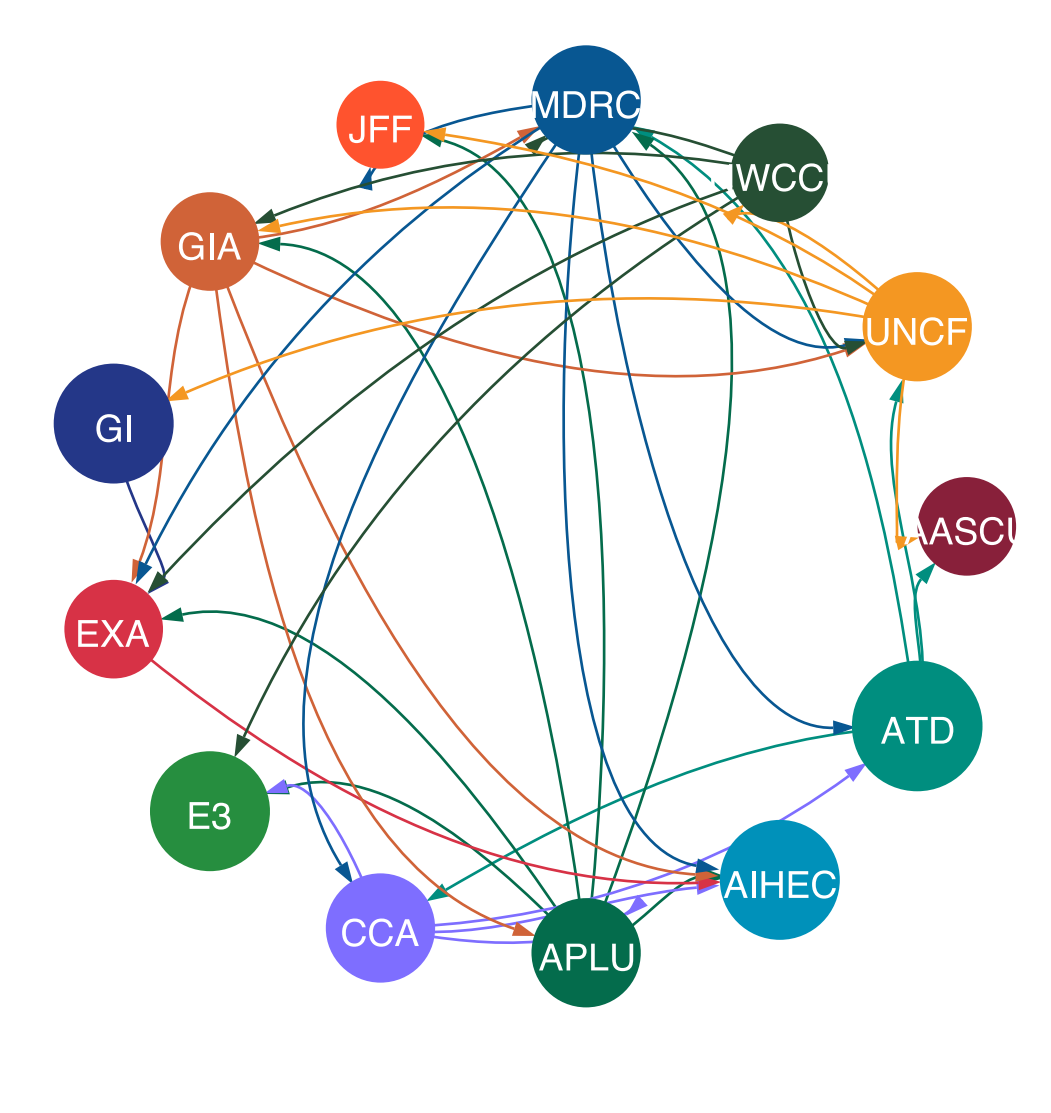


Networking

In [13]:
```python
M[M>1]=0
```

In [14]:
```python
# create graph
g = Graph.Adjacency(M, mode='directed', weighted=True)

# add edge weights and node labels.
g.es['weight'] = M[M.nonzero()]

# define network measures
network_measures = pd.DataFrame(dict(
    org = node_names,
    weight_in = g.strength(mode='in', weights='weight'),
    weight_out = g.strength(mode='out', weights='weight'),
    in_degree = g.vs.indegree(),
    out_degree = g.vs.outdegree(),
     betweenness = g.betweenness(weights='weight'),
    edge_count = g.ecount(),
    density = g.density()

))

#  define vertix size
network_measures['weight_in'] = vertix_size

g.vs['size'] = network_measures['weight_in']

# set vertix color
color = ["#88203A", "#008E7F", "#0091BA", "#046C4C",
                              "#7E6EFF", "#268E3F", "#D73246", "#243788",
                              "#D06338", "#FF532E", "#085792", "#264F34",
                              "#F49722"]
g.vs['color'] = color

vertix_color = g.vs['color']

#set edge color that match with vertix color

TEMP=[]
for e in g.es:
    elist = list(e.tuple)
    TEMP.append(elist)
g.es["color_assigned"] = TEMP
color_dict = {"0":"#88203A", "1":"#008E7F", "2":"#0091BA", "3":"#046C4C",
                              "4":"#7E6EFF", "5":"#268E3F", "6":"#D73246", "7":"#243788",
                              "8":"#D06338", "9":"#FF532E", "10":"#085792", "11":"#264F34",
                              "12":"#F49722"}
color_assigned = []
for i in range(37):
    color_assigned.append(g.es["color_assigned"][i][0])

edge_color = [color_dict[str(int(col_num))] for col_num in color_assigned]

# set vertix label
g.vs['label'] = node_names


## set layout
g_cr = g.layout_circle()

#set visual styel
visual_style = {}
out_name = "int_graph.png"
visual_style["vertex_color"] = g.vs['color']
visual_style["vertex_size"] = g.vs['size']
visual_style["vertex_frame_color"] = g.vs['color']
visual_style["vertex_label_color"] = 'white'
visual_style["vertex_label_cex"] = 0.7
visual_style["vertex_label_family"] = 'sans'
visual_style["vertex_label_font"] = 2
visual_style["edge_color"] = edge_color
visual_style["layout"] = layout
visual_style["edge_lty"] = 1
visual_style["edge_arrow_size"] = 0.6
visual_style["edge_width"] = g.es['weight']
visual_style["edge_curved"] = 0.2
visual_style["bbox"] = (400, 400)
visual_style["margin"] = 40


plot(g, out_name, **visual_style)
```

Out[14]:



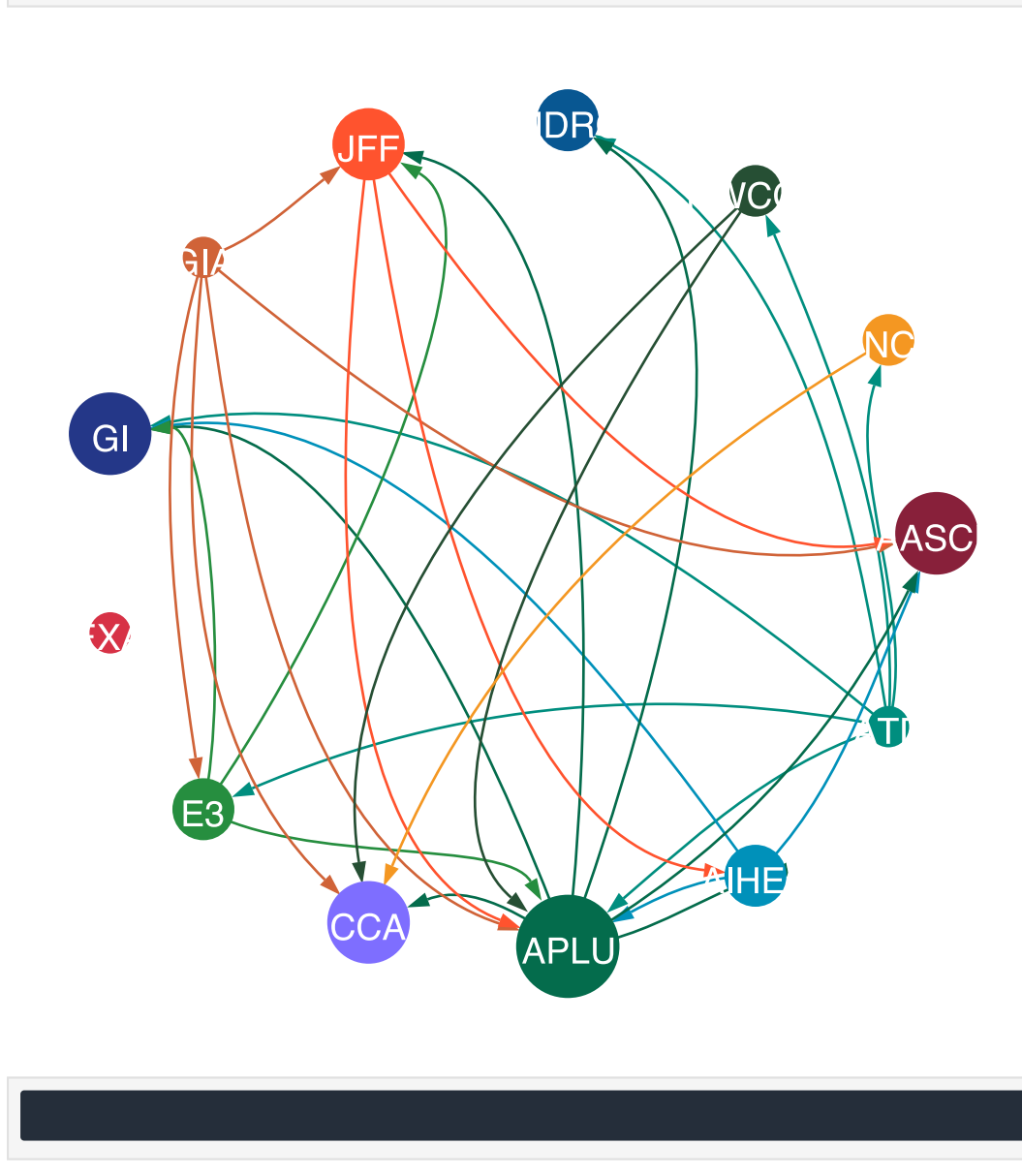
Capacity

In [15]:
```python
df = pd.read_excel(open("NIC_Clean_Data.xlsx",'rb'), sheet_name="advising")
```

In [16]:
```python
attributes = pd.DataFrame(node_names).rename(columns={0:'alter'})
attributes['ego'] = attributes['alter']
attributes['value'] = 0
df = pd.read_excel(open("NIC_Clean_Data.xlsx",'rb'), sheet_name="advising")
df['value'] = 1
# append self_netwok to create nxn matrix
df = df.append(attributes).reset_index().drop('index', axis=1)
df = pd.pivot_table(df, values = 'value', index=['alter'], columns =
'ego').fillna(0).astype(int)
M = df.to_numpy()
```

In [17]:
```python
g = Graph.Adjacency(M, mode='directed', weighted=True)

# add edge weights and node labels.
g.es['weight'] = M[M.nonzero()]

# define network measures
network_measures = pd.DataFrame(dict(
    org = node_names,
    weight_in = g.strength(mode='in', weights='weight'),
    weight_out = g.strength(mode='out', weights='weight'),
    in_degree = g.vs.indegree(),
    out_degree = g.vs.outdegree(),
     betweenness = g.betweenness(weights='weight'),
    edge_count = g.ecount(),
    density = g.density()

))

#  define vertix size
network_measures['weight_in'] = network_measures['weight_in']*4+15

g.vs['size'] = network_measures['weight_in']

# set vertix color
color = ["#88203A", "#008E7F", "#0091BA", "#046C4C",
                              "#7E6EFF", "#268E3F", "#D73246", "#243788",
                              "#D06338", "#FF532E", "#085792", "#264F34",
                              "#F49722"]
g.vs['color'] = color

vertix_color = g.vs['color']

#set edge color that match with vertix color

TEMP=[]
for e in g.es:
    elist = list(e.tuple)
    TEMP.append(elist)
g.es["color_assigned"] = TEMP
color_dict = {"0":"#88203A", "1":"#008E7F", "2":"#0091BA", "3":"#046C4C",
                              "4":"#7E6EFF", "5":"#268E3F", "6":"#D73246", "7":"#243788",
                              "8":"#D06338", "9":"#FF532E", "10":"#085792", "11":"#264F34",
                              "12":"#F49722"}
color_assigned = []
for i in range(29):
    color_assigned.append(g.es["color_assigned"][i][0])

edge_color = [color_dict[str(int(col_num))] for col_num in color_assigned]

# set vertix label
g.vs['label'] = node_names


## set layout
g_cr = g.layout_circle()

#set visual styel
visual_style = {}
out_name = "adv_graph.png"
visual_style["vertex_color"] = g.vs['color']
visual_style["vertex_size"] = g.vs['size']
visual_style["vertex_frame_color"] = g.vs['color']
visual_style["vertex_label_color"] = 'white'
visual_style["vertex_label_cex"] = 0.7
visual_style["vertex_label_family"] = 'sans'
visual_style["vertex_label_font"] = 2
visual_style["edge_color"] = edge_color
visual_style["layout"] = layout
visual_style["edge_lty"] = 1
visual_style["edge_arrow_size"] = 0.6
visual_style["edge_width"] = g.es['weight']
visual_style["edge_curved"] = 0.2
visual_style["bbox"] = (400, 400)
visual_style["margin"] = 40


plot(g, out_name, **visual_style)
```

Out[17]:



In [ ]: