UNITED STATES MILITARY ACADEMY

FINAL ASSESSMENT PART 2

MA 394

SECTION A1

MAJ CHARLES LEVINE

BY

CADET JOSEPH N. ZUCCARELLI '21, CO F1

WEST POINT, NEW YORK

06 DECEMBER 2020

____ MY DOCUMENTATION IDENTIFIES ALL SOURCES USED AND ASSISTANCE RECEIVED IN COMPLETING THIS ASSIGNMENT.

JZ I DID NOT USE ANY SOURCES OR ASSISTANCE REQUIRING DOCUMENTATION IN COMPLETING THIS ASSIGNMENT.

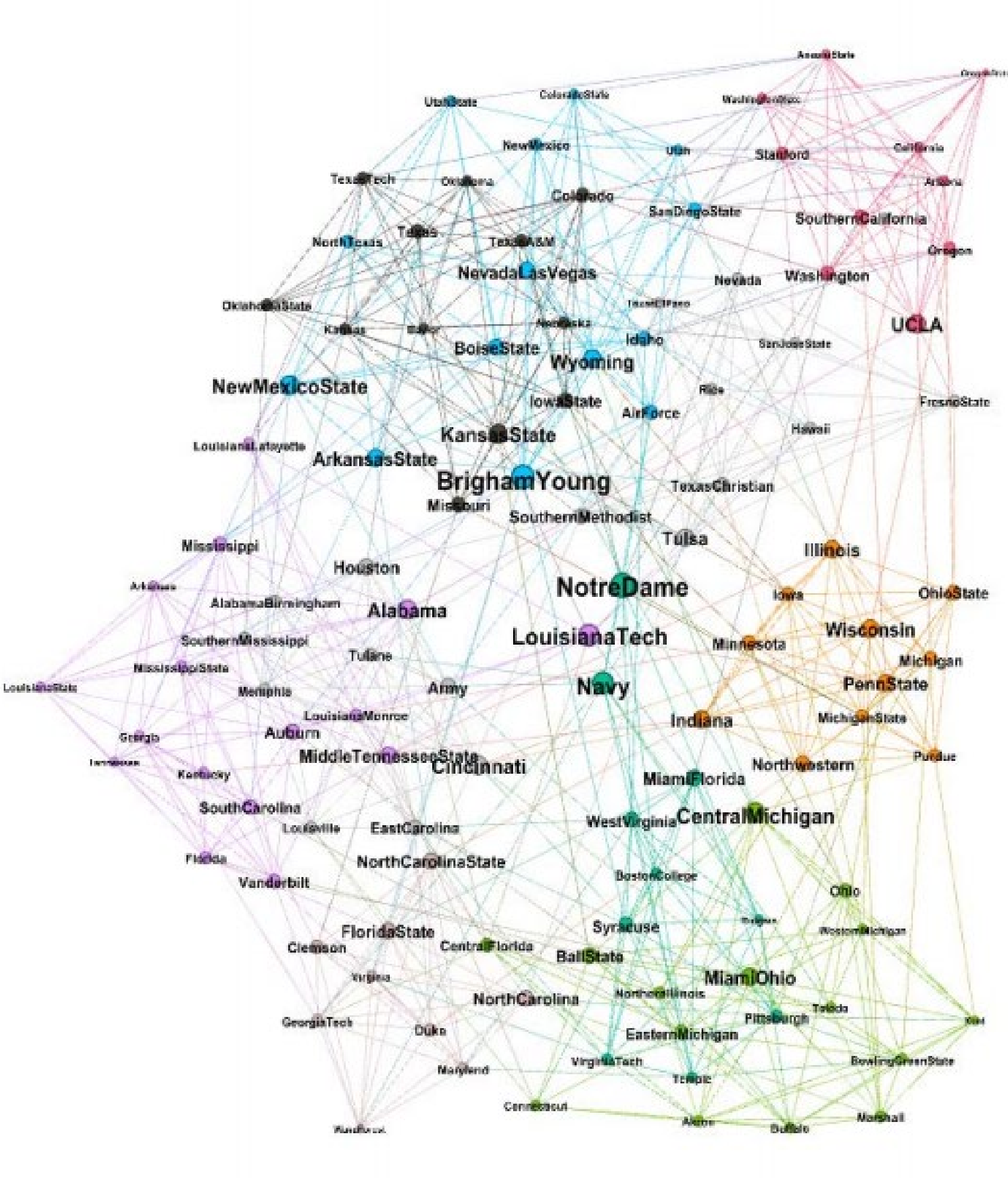
SIGNATURE: Joseph Zuccarelli

Objectives:
1. Visualize your network using Gephi.
2. Create a null model for your network using NetworkX.
3. Visualize and compare network degree distributions using NetworkX.

Process:
1. Create a folder on your computer just for this Lesson.
2. Download the files for this Assessment from Blackboard's Content\Lessons\L36 folder

PART 1: JOURNAL 3

American College Football Network



VISUALIZATION DISCUSSION:

The nodes in the graph above are colored by community. I chose to color the nodes by community because this network incorporates a known community structure, as the teams in the network are divided into conferences, and games are more frequent between members of the same conference than between members of different conferences. Based on the coloring, we see that Gephi found communities of teams that closely resemble the actual conference structure. Note that the edges are colored in the same manner as the nodes, as this best displays the community structure detected by Gephi.

The nodes in the graph are sized by betweenness centrality, a metric used to measure how much a given node is in-between others. I chose to size the nodes by betweenness centrality in order to highlight which teams play more games against opponents from different conferences. From the graph included on the previous page, we see that Notre Dame, BYU, and Navy play many teams from different conferences, as they are the largest nodes in the graph. On the other hand, we see that Oregon State, Kent, and Wake Forest mainly play teams from the same conference, as they are the smallest nodes in the graph.

ASSOCIATED PAPER OR PUBLICATION: M. Girvan and M. E. J. Newman, "Community Structure in Social and Biological Networks." Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002).

GENERAL BACKGROUND INFO:

A network of American college football teams was studied by Girvan and Newman during the early 2000s as part of a report concerning community structure in social and biological networks. In this study the two researchers review existing methods for detecting community structure and propose their own method of doing so that avoids some of the shortcomings of the traditional techniques. The focus of the researchers' proposed method is the edges in the network that are least central, or, in other words, the edges that are most "between" communities.

In order to test their proposed method, Girvan and Newman apply it to a network of 115 Division 1A college football teams with links between teams that played against each other during the 2000 Fall season. The teams within the network are divided into conferences of about 8-12 teams each. Teams often play more games against members of their same conference than members of a different conference. The researchers found that when applied to this network, their proposed method identifies the football conference structure with a high degree of success, as almost all teams were correctly grouped with the other teams in their conference. Overall, the proposed method mainly failed in cases where the network structure genuinely did not correspond to the conference structure. In all other respects, however, the method worked remarkably well.

MY NETWORK METRICS

Nodes: $N = 115$, Nodes represent Division 1A college football teams
Edges: $L = 613$, Edges represent games between teams during the 2000 Fall season
The Network is Undirected, with 1 Connected Component
Noteworthy Metrics:

Highest Degree: 12 (Note that there are 12 nodes of degree 12 in the network, which indicates that 12 teams played 12 games during the 2000 Fall season)

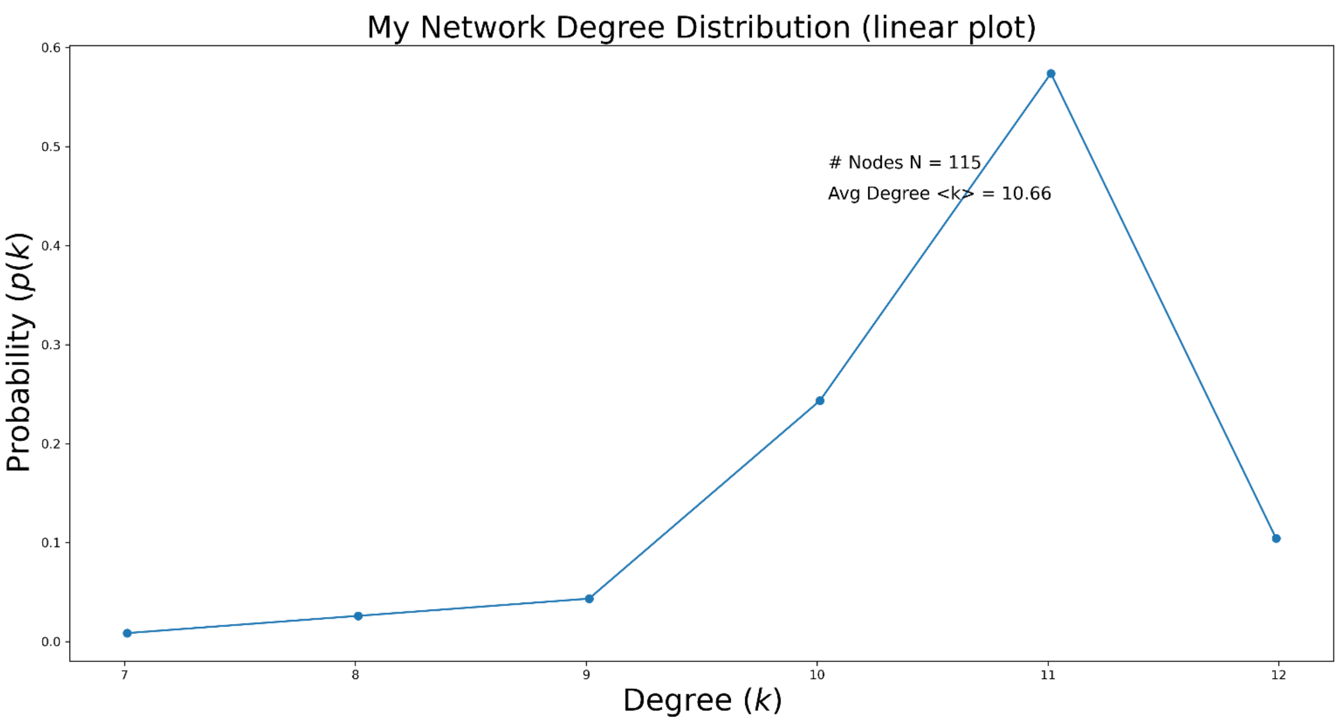
Average Degree $< k > \ = 10.661$
Average Clustering Coefficient $< C > \ = 0.403$
Network Density $\rho = 0.094$
Graph Modularity Q = 0.601 with 9 communities
Girvan-Newman modularity:
# of communities: 10
Maximum modularity: 0.600
Network Diameter $d_{max} = 4$
Average Path Length $< d > \ = 2.508$
Highest Betweenness Centrality (x_i): Is Node “Notre Dame”, with degree $k_{NotreDame} = 11$, and BC $x_{NotreDame} \ = \ 215.99$

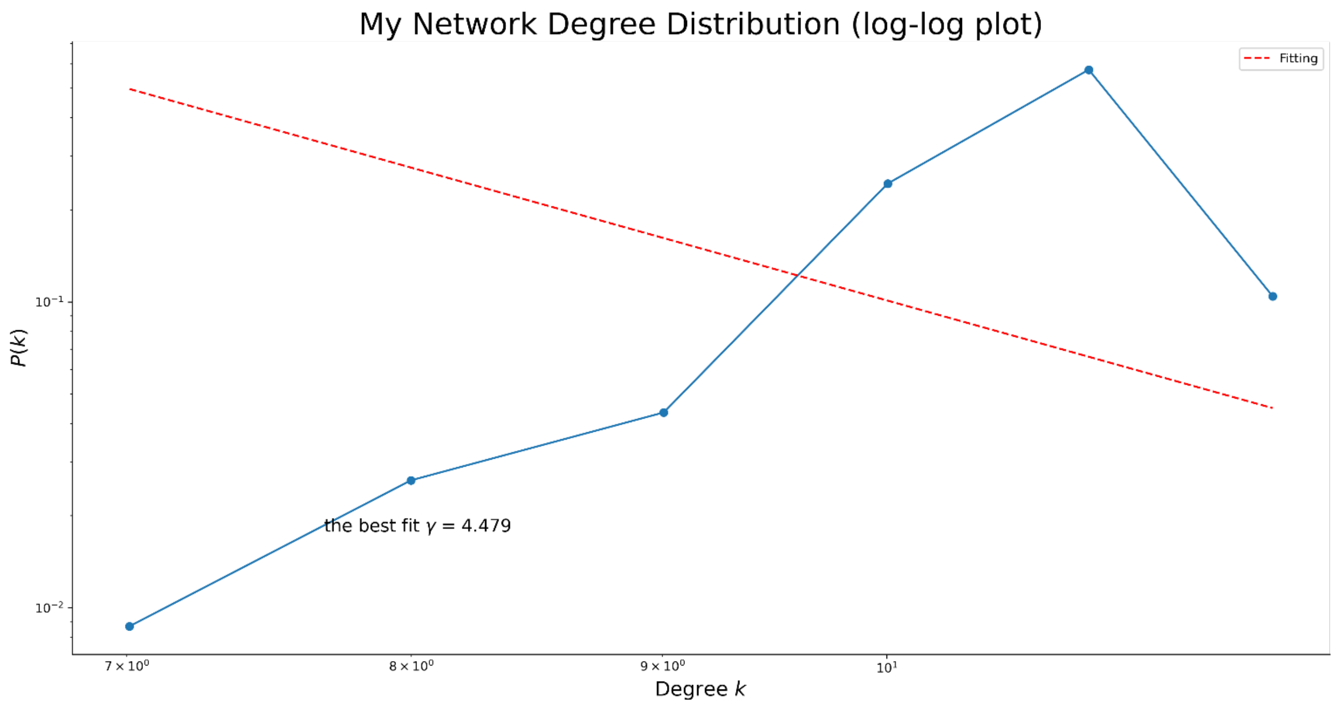
Discussion:

One metric that stands out in this network is the communities found in Gephi. As mentioned earlier, one thing that makes this network interesting is that it incorporates a known community structure. The teams in the network are divided into conferences of 8-12 teams each, and games are more frequent between members of the same conference than between members of different conferences. Therefore, Gephi found communities of teams that closely resemble the actual conference structure. However, Gephi’s community detection was not perfect at aligning to the actual conferences due to some nuances in the scheduling of games. Some conferences allow their teams to schedule more games with teams from different conferences, which causes Gephi’s algorithm to group some teams with non-conference members. Also note that using the Girvan-Newman method for detecting community structure resulted in a different number of communities within the network. The Girvan-Newman method detected ten communities within the network, one more community than the default algorithm used by Gephi. The actual number of conferences represented within the networks was twelve; therefore, the Girvan-Newman method provides a slightly more accurate estimate of the number of communities in this case.

PART 2: DEGREE DISTRIBUTIONS

3. If your data did not come as a .gml file, and you need help getting it into Gephi, see me ASAP! If your data came as a .gml file:
   a. Using "Data Laboratory" and the "Export Table" function, export your node degrees into a .csv file.
   b. Using Excel, eliminate all columns but the "Degree" column, then also delete the first row with the "Degree" word in it so you just have a column of numbers that represent the degrees of each of your nodes. Save this as a .csv file, specifically "MyNetDegs.csv" in your working directory where you saved your downloaded python code from step 2. for python to ingest. and then using the python code provided, plot the degree distribution using the provided code of your network. Plot in a linear plot first, then try a log-log plot. Massage the number of bins until you get a distribution.
   c. Put these two plots in this document here and DISCUSS! Which is better? Why? Are both bad? Are both good?

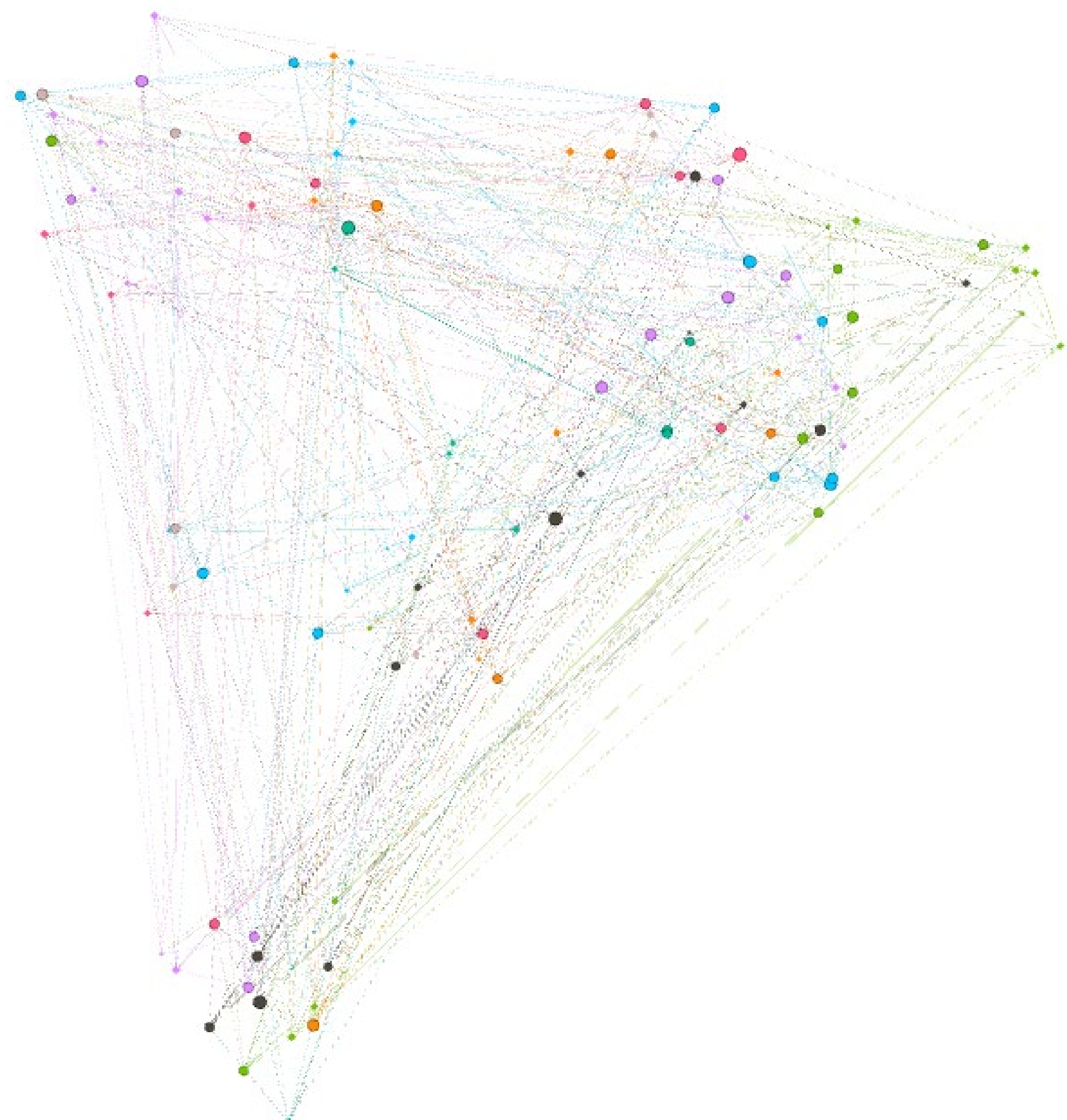




Note that both degree distributions are skewed-left. Note that the nodes in this network range from degree 7-12. The average degree of the network is 10.66. Therefore, the skewedness of the distributions makes sense, as the majority of the nodes in the network are of degree 10-11 since teams typically play 10-11 games per season. Overall, both plots accurately depict the skewedness of the degree distribution that represents the college football network.

4. Review Configuration Model, Barabási pp 138-139.
    a. Now using the python code, use the configuration model to generate a similar graph to yours.
    b. Export your Null Model to a .gexf file and then import into Gephi.
    c. Visualize using Gephi here:

Null Network



NULL NETWORK METRICS

Nodes: $N = 115$,
Edges: $L = 585$,
The Network is Undirected, with 1 Connected Component
Noteworthy Metrics:
- Highest Degree: 12 (eight degrees in the network are of degree 12)
- Average Degree $< k > = 10.174$
- Average Clustering Coefficient $< C > = 0.079$
- Network Density $\rho = 0.089$
- Graph Modularity Q = 0.267 with 7 communities
- Girvan-Newman modularity:
  - # of communities: 29
  - Maximum modularity: 0.202
- Network Diameter $d_{max} = 4$
- Average Path Length $< d > = 2.28$
- Highest Betweenness Centrality (x_i): Is Node 5, with degree $k_5 = 12$, and BC $x_5 = 111.31$

Which values are the same and which are different? Explain why in your own words.

Note that the following values are the same in the null network as the actual network: number of nodes, number of connected components, highest degree, and network diameter. The following values are different in the null network than the actual network: number of edges, average degree, average clustering coefficient, network density, graph modularity, number of communities, maximum modularity, average path length, and highest betweenness centrality. The configuration model helps us build a network with a pre-defined degree sequence. In the null network generated by the model, each node has a pre-defined degree, but otherwise the network is wired randomly. Therefore, the null network should have the same number of nodes and edges, yet our code removed any self-loops in the network and thus there is a small discrepancy between the number of edges in the null network and the actual network.

Should the number of nodes be the same?

The number of nodes should be the same, as the configuration model is built using the degree sequence from the actual network (thus preserving all nodes from the actual network).

Should the number of edges be the same?

The number of edges should be the same, as the configuration model is built using the degree sequence from the actual network. However, this is not the case here because our code removed any self-loops in the null network.

Should the average clustering be the same?

The average clustering need not be the same, as the average clustering coefficient of the null network is much lower than that of the actual network. There is much less clustering in the null network than the actual network, which makes sense considering that the actual network contains a built-in community structure. Remember that the configuration model builds a network through random wiring based on the degree sequence of the actual network.

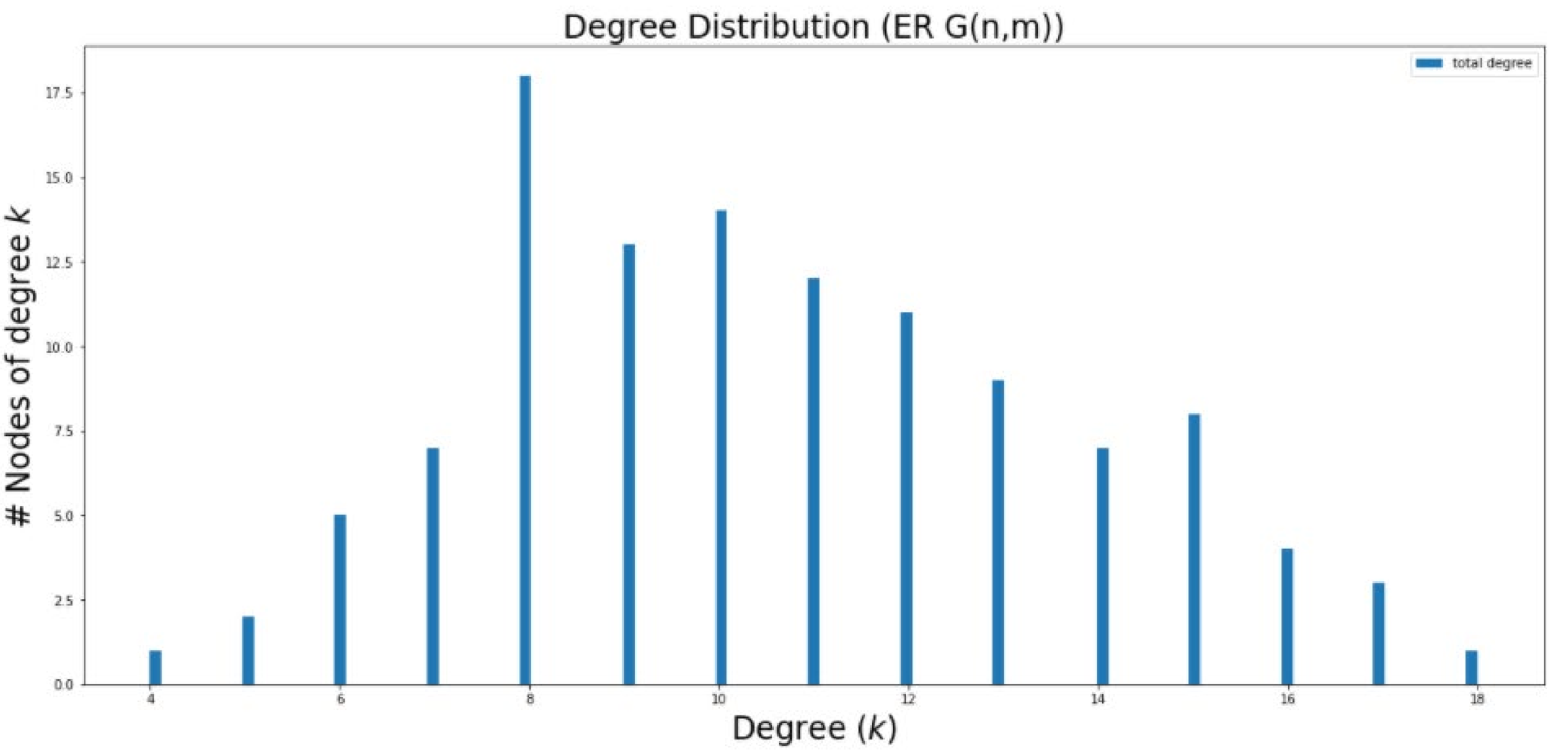
Should the communities and modularity be the same?

The communities and modularity need not be the same, as the number of communities and the modularity of the null network is much lower than that of the actual network. There are less communities in the null network than the actual network, which makes sense considering that the actual network contains a built-in community structure. Remember that the configuration model builds a network through random wiring based on the degree sequence of the actual network.

What happened to the diameter, average path length, and highest betweenness centrality?

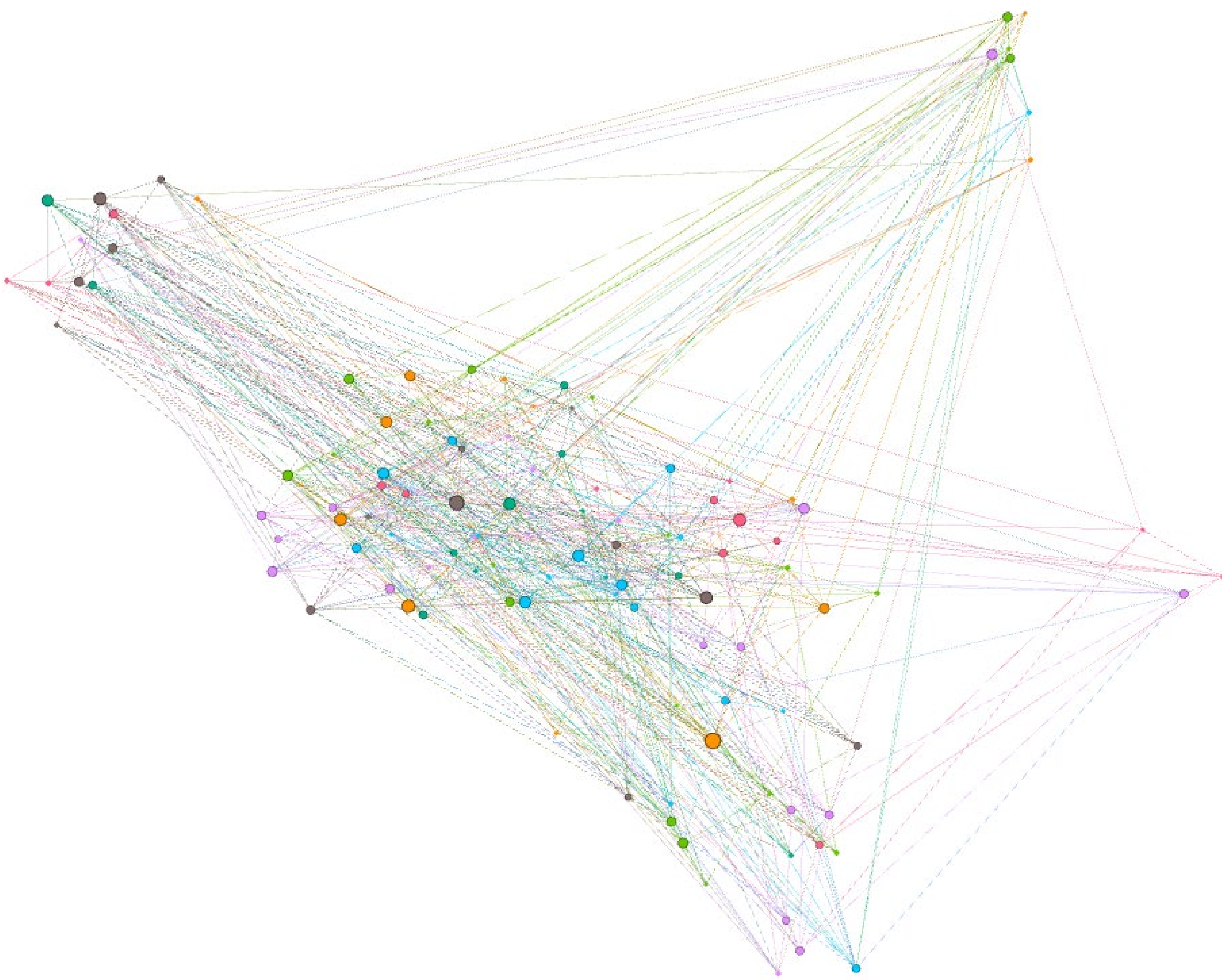
The diameter of the network remained the same, yet the average path length and the highest betweenness centrality decreased slightly.

5. Now use either the ER G(n,m) model to also create a null model with the same average degree and number of edges.
    a. Plot the degree sequence and insert here. Is it similar to your original network?
    b. Export your Random Null Model to a .gexf file and then import into Gephi.
    c. Visualize here.



Note that the degree distribution displayed above is not very similar to that of the original college football network. The nodes in this random null network range from degree 4-17, which is a much larger degree range than that of the original network. The random null network also appears much more normally distributed than the original network, as the original network was heavily left skewed since most nodes in the network were of degree 10-11.

Random Null Network



In [1]:
```
import numpy as np
from numpy import linalg as LA
from numpy.random import choice
import networkx as nx
from networkx.readwrite import json_graph
import matplotlib.pyplot as plt
import scipy as sp
import scipy.sparse as ss
from scipy import sparse
from scipy.sparse import dok_matrix
from scipy.special import binom
# import seaborn as sns
import pandas as pd
import powerlaw as pwl
# import sys
# from IPython.display import IFrame
# from IPython.core.display import display, HTML
# display(HTML("<style>.container { width:95% !important; }</style>"))

from functions import *

# from time import gmtime, localtime, strftime, sleep
# from collections import defaultdict
# from collections import Counter
# import copy
# from collections import deque, defaultdict
# from itertools import combinations
```

In [2]:
```
## This function returns a list of the degrees from your csv file
def get_degree_list(degfile = 'MyNetDegs.csv'):
    import csv
    degrees = []
    with open(degfile, newline='') as inputfile:
        for row in csv.reader(inputfile):
            degrees.append(int(row[0]))
    return degrees
```

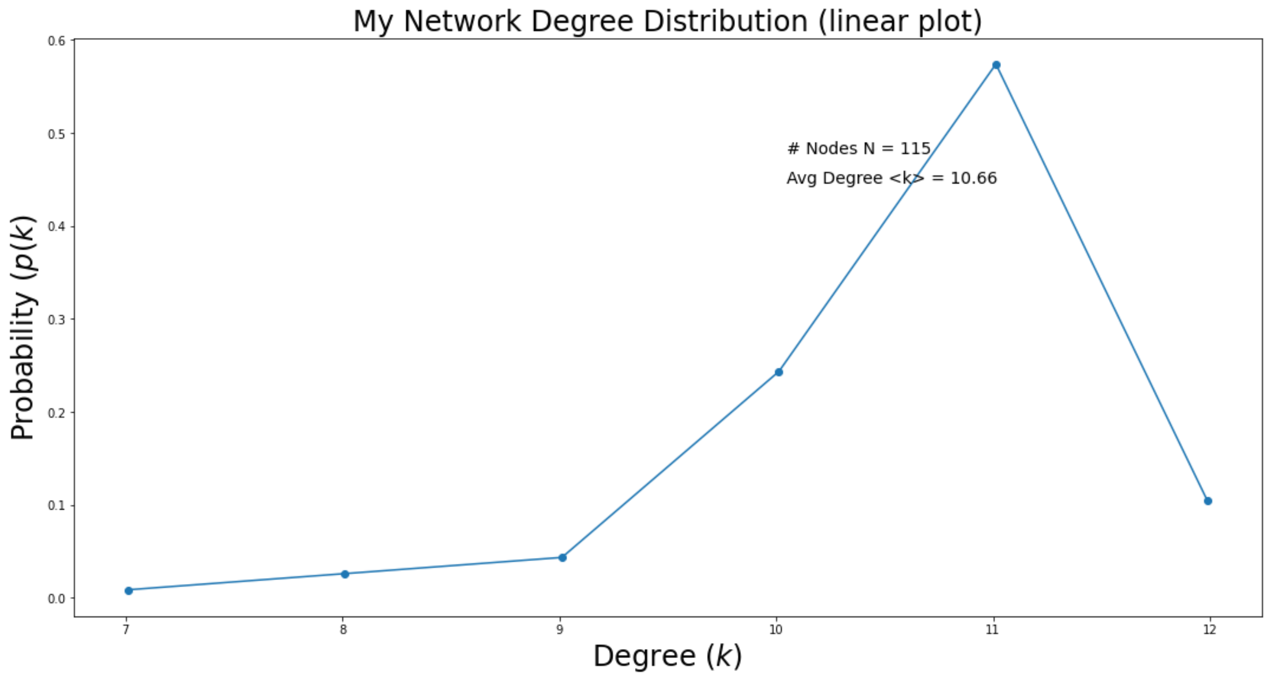
In [7]:
```
## Plot linear degree distribution
## DegreeSequence is a list of degreees, nb is number of bins
def plot_lin_deg_dist(title,DegreeSequence,nb):
    import numpy as np
    avg_deg = np.mean(DegreeSequence)
    sizer = (18,9)
    fig, ax = plt.subplots(figsize=(sizer))
    x,p,resid_N = get_binning3(DegreeSequence, num_bins=nb, log_binning=False, is_pmf=T
    plt.plot(x,p,'o',ls = '-')
    ax.set_xlabel('Degree ($k$)', fontsize=24)
    ax.set_ylabel('Probability ($p(k)$', fontsize=24)
    ax.set_title('{} Degree Distribution (linear plot)'.format(title), fontsize=24)
    plt.text(.6, 0.75,"Avg Degree <k> = {:.2f}".format(avg_deg), transform=ax.transAxes
    plt.text(.6, 0.8,"# Nodes N = {}".format(len(degrees)), transform=ax.transAxes, fon
#     plt.legend()
    plt.savefig("{}_degree_sequence_linear.png".format(title), dpi=300)
    plt.show()
```

In [8]:
```
## Plot linear degree distribution
## DegreeSequence is a list of degreees, nb is number of bins
def plot_log_deg_dist(title,DegreeSequence,nb):
```

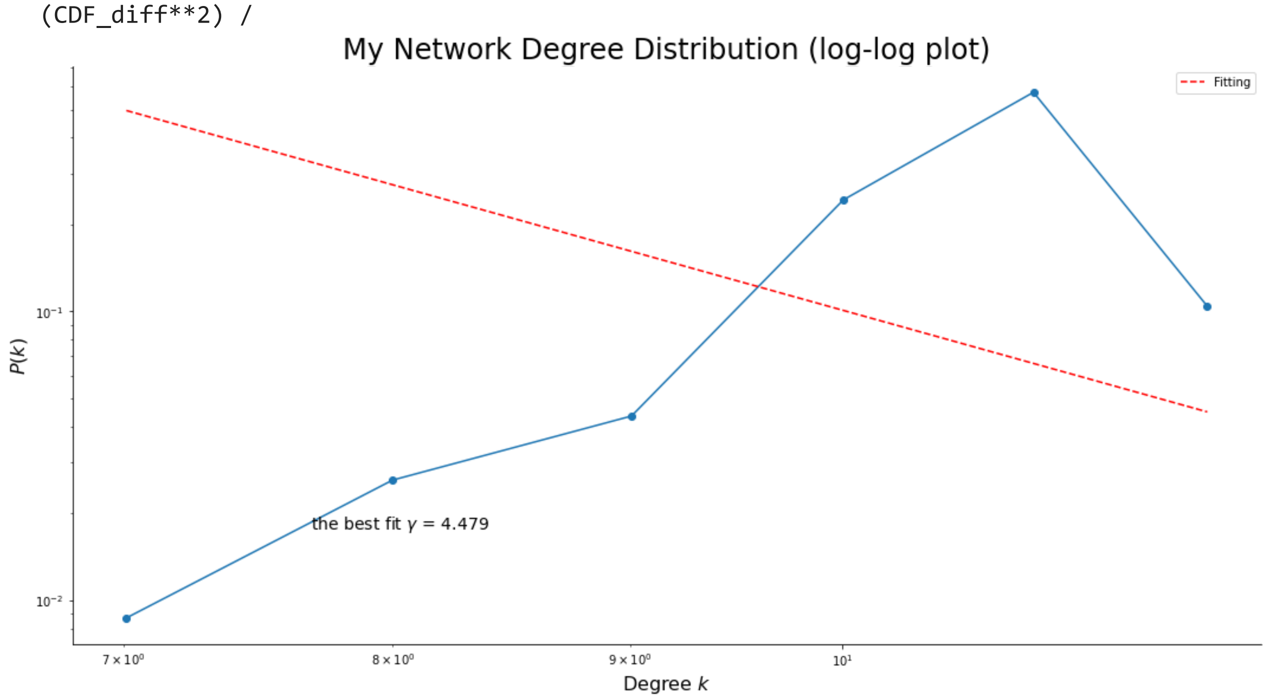
```
    import numpy as np
    import powerlaw as pwl
    avg_deg = np.mean(DegreeSequence)
    sizer = (18,9)
    fig, ax = plt.subplots(figsize=(sizer))
    x,p,resid_N = get_binning3(DegreeSequence, num_bins=nb, log_binning=True, is_pmf=Tr
    plt.plot(x,p, marker='o')
    ax.set_yscale('log')
    ax.set_xscale('log')
    plt.xlabel(r"Degree $k$", fontsize=16)
    plt.ylabel(r"$P(k)$", fontsize=16)
    # remove right and top boundaries because they're ugly
    ax = plt.gca()
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)
    ax.yaxis.set_ticks_position('left')
    ax.xaxis.set_ticks_position('bottom')
    ax.set_title('{} Degree Distribution (log-log plot)'.format(title), fontsize=24)
    # Show the plot
    data = x
    fit = pwl.Fit(data)
    gamma = fit.power_law.alpha
    fitlabel = 'Fitting'
    fit.power_law.plot_pdf(ax=ax, color='r', linestyle='--',label=fitlabel)
    plt.legend()
    plt.text(0.2, 0.2,"the best fit $\gamma$ = {:.3f}".format(gamma), transform=ax.tran
    plt.savefig("{}_degree_sequence_log-log.png".format(title), dpi=300)
    plt.show()
```

In [10]:

```
# Get the degrees from your .csv file
degrees = get_degree_list('MyNetDegs.csv')
DegreeSequence = sorted(degrees, reverse=True)
# Give your network a title
title = "My Network"
# nb is number of bins. I set it at the number of nodes,
# but experiment with other numbers until you get something good.
nb = 200
# nb=25

#Plot the degree ditribution linearly and save the image as a .png file
plot_lin_deg_dist(title,DegreeSequence,nb)
#Plot the degree ditribution log-log and save the image as a .png file
plot_log_deg_dist(title,DegreeSequence,nb)
```




```
Calculating best minimal value for power law fit
C:\Users\joseph.zuccarelli\anaconda3\lib\site-packages\powerlaw.py:699: RuntimeWarning:
invalid value encountered in true_divide
  (CDF_diff**2) /
```



If your Network has UNWEIGHTED EDGES use the following code:

In [11]:
```
#  If your Network has UNWEIGHTED EDGES:
# Build a NULL MODEL of your network based on the configuration model
H = nx.configuration_model(DegreeSequence, create_using=None, seed=None)
# Remove any duplicate edges!
H = nx.Graph(H)
# remove any self-loops!
H.remove_edges_from(nx.selfloop_edges(H))
# output it to a .gexf file that can be read by Gephi
nx.write_gexf(H, "{}_Null_Model.gexf".format(title))
```

If your Network has WEIGHTED EDGES use the following code:

In [ ]:
```
#  If your Network has UNWEIGHTED EDGES:
# Build a NULL MODEL of your network based on the configuration model
H = nx.configuration_model(DegreeSequence, create_using=None, seed=None)
# remove any self-loops!
H.remove_edges_from(nx.selfloop_edges(H))
# output it to a .gexf file that can be read by Gephi
nx.write_gexf(H, "{}_Null_Model.gexf".format(title))
```

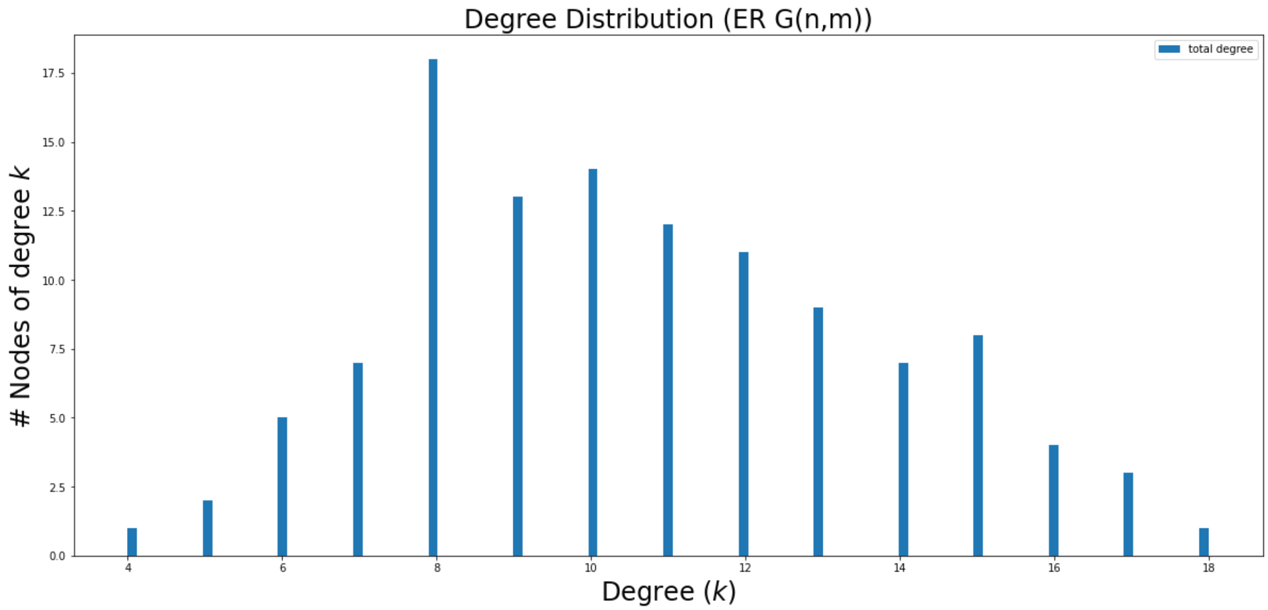
In [ ]:

In [13]:
```
#Erdos-Reyni G(n,m) model
n=115 # number of nodes
m=613 # number of edges

G = nx.gnm_random_graph (n, m, seed=None, directed=False)
nx.draw_networkx(G,pos=nx.spring_layout(G))
limits=plt.axis('off')
plt.show()

nx.write_gexf(G, "Gnp_N-{}_m-{}.gexf".format(n,m))
print("the network has {} edges".format(len(nx.edges(G))))
DegreeSequence = sorted(dict(nx.degree(G)).values(),reverse=True)
#print(DegreeSequence)
```

the network has 613 edges

In [20]:
```
# Erdos-Reyni Plot specifications
# Experiment with the number of bins (nb) for the degree distribution!
nb=115 # number of bins on the x-axis
sizer = (20,9)

fig, ax = plt.subplots(figsize=(sizer))
plt.hist(DegreeSequence,bins = nb, histtype='bar',rwidth=1, label='total degree')
# x,p,resid_N = get_binning2(DegreeSequence, num_bins=nb, log_binning=False, is_pmf=Tru
# plt.plot(x,p,'o',label='total degree')
ax.set_xlabel('Degree ($k$)', fontsize=24)
# ax.set_ylabel('Probability ($p(k)$', fontsize=24)
ax.set_ylabel('# Nodes of degree $k$', fontsize=24)
ax.set_title('Degree Distribution (ER G(n,m))', fontsize=24)
plt.legend()
plt.savefig("deg_dist_Gnp_N-{}_m-{}.png".format(n,p), dpi=300)
```

```
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-20-9bdbfe2915e2> in <module>
     13 ax.set_title('Degree Distribution (ER G(n,m))', fontsize=24)
     14 plt.legend()
---> 15 plt.savefig("deg_dist_Gnp_N-{}_m-{}.png".format(n,p), dpi=300)

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    857 def savefig(*args, **kwargs):
    858     fig = gcf()
--> 859     res = fig.savefig(*args, **kwargs)
    860     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    861     return res

~\anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent,
 **kwargs)
   2309                 patch.set_edgecolor('none')
   2310 
-> 2311         self.canvas.print_figure(fname, **kwargs)
   2312 
   2313         if transparent:

~\anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filenam
e, dpi, facecolor, edgecolor, orientation, format, bbox_inches, pad_inches, bbox_extra_a
rtists, backend, **kwargs)
   2208 
   2209             try:
-> 2210                 result = print_method(
   2211                     filename,
   2212                     dpi=dpi,

~\anaconda3\lib\site-packages\matplotlib\backend_bases.py in wrapper(*args, **kwargs)
   1637             kwargs.pop(arg)
   1638 
-> 1639         return func(*args, **kwargs)
   1640 
   1641     return wrapper

~\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, file
name_or_obj, metadata, pil_kwargs, *args)
    508         """
    509         FigureCanvasAgg.draw(self)
--> 510         mpl.image.imsave(
    511             filename_or_obj, self.buffer_rgba(), format="png", origin="upper",
    512             dpi=self.figure.dpi, metadata=metadata, pil_kwargs=pil_kwargs)

~\anaconda3\lib\site-packages\matplotlib\image.py in imsave(fname, arr, vmin, vmax, cma
p, format, origin, dpi, metadata, pil_kwargs)
   1603         pil_kwargs.setdefault("format", format)
   1604         pil_kwargs.setdefault("dpi", (dpi, dpi))
-> 1605         image.save(fname, **pil_kwargs)
   1606 
   1607 

~\anaconda3\lib\site-packages\PIL\Image.py in save(self, fp, format, **params)
   2146                 fp = builtins.open(filename, "r+b")
   2147             else:
-> 2148                 fp = builtins.open(filename, "w+b")
   2149 
   2150         try:

FileNotFoundError: [Errno 2] No such file or directory: 'deg_dist_Gnp_N-115_m-[0.0086956
5 0.0173913  0.04347826 0.06086957 0.15652174 0.11304348\n 0.12173913 0.10434783 0.09565
217 0.07826087 0.06086957 0.06956522\n 0.03478261 0.02608696 0.00869565].png'
```



In [21]:

```
# Erdos-Reyni Plot specifications
# Experiment with the number of bins (nb) for the degree distribution!
nb=50 # number of bins on the x-axis
sizer = (20,9)

fig, ax = plt.subplots(figsize=(sizer))
# plt.hist(DegreeSequence,bins = nb, histtype='bar',rwidth=1, label='total degree')
x,p,resid_N = get_binning2(DegreeSequence, num_bins=nb, log_binning=False, is_pmf=True)
plt.plot(x,p,'o',label='total degree')
ax.set_xlabel('Degree ($k$)', fontsize=24)
ax.set_ylabel('Probability ($p(k)$', fontsize=24)
# ax.set_ylabel('# Nodes of degree $k$', fontsize=24)
ax.set_title('Degree Distribution (ER G(n,m))', fontsize=24)
plt.legend()
plt.savefig("deg_dist_Gnp_N-{}_m-{}.png".format(n,p), dpi=300)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-21-c48f5bc6e582> in <module>
     13 ax.set_title('Degree Distribution (ER G(n,m))', fontsize=24)
     14 plt.legend()
---> 15 plt.savefig("deg_dist_Gnp_N-{}_m-{}.png".format(n,p), dpi=300)

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    857 def savefig(*args, **kwargs):
    858     fig = gcf()
--> 859     res = fig.savefig(*args, **kwargs)
    860     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    861     return res

~\anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent, **kwargs)
   2309                 patch.set_edgecolor('none')
   2310 
-> 2311         self.canvas.print_figure(fname, **kwargs)
   2312 
   2313         if transparent:

~\anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, bbox_inches, pad_inches, bbox_extra_artists, backend, **kwargs)
   2208
```

```
   2209             try:
-> 2210                 result = print_method(
   2211                     filename,
   2212                     dpi=dpi,

~\anaconda3\lib\site-packages\matplotlib\backend_bases.py in wrapper(*args, **kwargs)
   1637             kwargs.pop(arg)
   1638 
-> 1639         return func(*args, **kwargs)
   1640 
   1641     return wrapper

~\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, filename_or_obj, metadata, pil_kwargs, *args)
    508         """
    509         FigureCanvasAgg.draw(self)
--> 510         mpl.image.imsave(
    511             filename_or_obj, self.buffer_rgba(), format="png", origin="upper",
    512             dpi=self.figure.dpi, metadata=metadata, pil_kwargs=pil_kwargs)

~\anaconda3\lib\site-packages\matplotlib\image.py in imsave(fname, arr, vmin, vmax, cmap, format, origin, dpi, metadata, pil_kwargs)
   1603         pil_kwargs.setdefault("format", format)
   1604         pil_kwargs.setdefault("dpi", (dpi, dpi))
-> 1605         image.save(fname, **pil_kwargs)
   1606 
   1607 

~\anaconda3\lib\site-packages\PIL\Image.py in save(self, fp, format, **params)
   2146                 fp = builtins.open(filename, "r+b")
   2147             else:
-> 2148                 fp = builtins.open(filename, "w+b")
   2149 
   2150         try:

FileNotFoundError: [Errno 2] No such file or directory: 'deg_dist_Gnp_N-115_m-[0.00869565 0.0173913  0.04347826 0.06086957 0.15652174 0.11304348\n 0.12173913 0.10434783 0.09565217 0.07826087 0.06086957 0.06956522\n 0.03478261 0.02608696 0.00869565].png'
```