Using R in an Applied Graph Theory Course

Amir Barghi[1*]

[1] Department of Mathematics and Statistics, Saint Michael's College

Colchester, VT 05439, USA

[*] abarghi@smcvt.edu

Abstract

This paper provides a brief summary of various resources I used in teaching network analysis using R in an undergraduate applied graph theory course. I taught the course in Spring 2020 at Saint Michael's College, which is a liberal arts college in the Northeast of the United States. In particular, I include hyperlinks to and descriptions of the R code I wrote for classroom use, along with an example of a final exam. The R code covers some of the examples in Chapter 3, 4, and 5 of *Network Science*, by Albert-László Barabási, which was one of the textbooks that I used for the course.

1 Introduction

In Spring 2020, I taught an undergraduate applied graph theory at Saint Michael's College, which is a liberal art college in Colchester, Vermont, USA. The prerequisite for the course is a first-semester calculus course. There were twelve students enrolled in the course and they came from diverse backgrounds within STEM (Science, Technology, Engineering, and Mathematics) fields but mainly from mathematics and data science. Moreover, each semester is about fifteen weeks long at Saint Michael's College and this course met three times a week, with each session being about an hour long. The course is usually offered every other year.

In the first half of the semester, I covered the usual topics for an introductory graph theory course such as the basic properties of graphs and trees, Eulerian circuits and Hamiltonian cycles, planarity, and coloring, with emphasis on both computation and mathematical arguments. Since the prerequisite for the course is a first-semester calculus course, I had to briefly introduce students to basic matrix operations and counting techniques. As the title of the course suggests, I also needed to dedicate some portion of the course to applications of graph theory. As Sayama

et al. [2] suggest, the ubiquitousness of networks is one of the seven essential concepts of network literacy. Since students in the course came from different STEM fields, studying networks seemed like an appropriate approach as it would captivate the attention of a diverse audience because of this ubiquitousness. So in the second half of the semester, I pivoted the focus of the course to network science and its applications.

2 Resources

For the second portion of the course, I used the book *Network Science*[1], by Albert-László Barabási [1]. The book is written as a textbook with each chapter containing homework problems and advance topics. In terms of the material, it is comprehensive and, for the most part, self-contained—it requires a working knowledge of differentiation and integration and a basic familiarity with matrix operations and counting techniques. Moreover, what makes this book an attractive option is that the author has made it available online under a Creative Commons Attribution-NonCommercial 3.0 Unported License[2]. Datasets used in the book, along with slides for classroom use, are also available from the book's website.

According to Sayama et al. [2], another essential concept of network literacy is that "Today's computer technology allows you to study real-world networks." So to make the course pedagogically and technologically interactive, I decided to use R[3] in the classroom. I chose R because it is an open-source and free statistical and programming software and, along with Python, is one of the popular choices in the data science community. To this end, I created R Markdown[4] files for the chapters I covered (Chapters 3, 4, and 5), with a focus on the yeast protein interaction network[5] from `igraph`[6] package as the main real-world network. The yeast protein interaction network is an undirected network where the nodes are proteins inside a yeast cell and the links are the interactions between these proteins.

To run these files, students installed R and RStudio[7] Desktop (also available for free under an open source license) on their computers. Since this course does not have a prerequisite where learning R, particularly utilizing `tidyverse`[8], is one of

[1] http://networksciencebook.com/

[2] "This book's text and illustrations are licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported License."

[3] https://www.r-project.org/

[4] https://rmarkdown.rstudio.com/

[5] https://rdrr.io/cran/igraphdata/man/yeast.html; this network is from an article by Von Mering et al. [3].

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

[7] https://www.rstudio.com/

[8] https://www.tidyverse.org/

the main objectives, I used the following DataCamp[9] courses as online assignments to bring everyone to the same level: Introduction to R; Intermediate R; Network Analysis in R; Introduction to the Tidyverse; Case Studies: Network Analysis in R. DataCamp is an online platform for learning a plethora of data science topics, with a focus on R and Python, and my students had free access to these courses through DataCamp for Classrooms[10]. Having access to these courses, to some extent, addressed the challenge of students coming from "diverse computational and coursework backgrounds" as one of the challenges pointed out by Porter [4] in teaching an undergraduate network science course.

I find learning the basics of R and using `tidyverse` functions less challenging for someone who has had little to no coding experience in comparison to learning how to do similar tasks in Python. And RStudio provides an interactive platform to work, especially the R Markdown feature allows users to generate attractive documents. However, I find Jupyter Notebook[11], using the R package `IRkernel`[12], provides a more interactive alternative to R Markdown for creating narratives and lecture notes. This is the reason as to why I posted the Jupyter Notebook counterparts of the R Markdown files that I wrote for this course on the GitHub[13] page as well. In addition to a local installation of Anaconda Distribution[14], which is open source, there are several cloud computing platforms where Jupyter Notebooks can be run for free. Moreover, one possible alternative to `igraph`, which is also available in Python and C, is `NetworkX`[15]. However, `NetworkX` is only available in Python, and for the purposes of this course and similar courses, I find `igraph` is more than sufficient.

To emphasize the ideas students learned in class from a more "big picture" perspective, I assigned readings from *Linked* [5]. *Linked* is a popular science book written by Albert-László Barabási as well. Students read the entire book in segments (spanning two to three weeks each), wrote short summaries of what they learned, and shared their observations with their peers in class. The book was electronically available to students through Saint Michael's College Library. There were three reading discussions and each reading discussion was conducted by a different group of three-to-four students, who were responsible with coming up with a set of questions based on the chapters covered. *Linked* meshed perfectly with *Network Science* by reinforcing the students' intuition about what the more theoretical

[9] https://www.datacamp.com/

[10] https://www.datacamp.com/groups/classrooms

[11] https://jupyter.org/

[12] https://github.com/IRkernel/IRkernel

[13] https://github.com/abarghi/Using_R_in_an_Applied_Graph_Theory_Course

[14] https://www.anaconda.com/products/distribution

[15] https://networkx.org/

material they were learning in class.

3 Assessment

I should point out that, due to the COVID-19 pandemic, I had to conduct the second half of the semester, which coincided with covering topics from *Network Science* and reading *Linked*, remotely.

In the first half of the semester, there were seven weekly written assignments and seven short in-class quizzes based on these assignments. There were two mid-semester exams and a final exam. On the mid-semester exams, with the exception of perhaps one problem, the problems were computational or conceptual. Moreover, the first mid-semester exam was in person while students had to take the second mid-semester exam and the final exam remotely. In the second half of the semester, there were three readings assigned from *Linked* and five assignments from DataCamp. At the end of the semester, I used the following distribution to compute an overall numeric grade: participation: 5%; DataCamp assignments: 10%; assignments: 15%; in-class quizzes: 10%; Exam 1: 18%; Exam 2: 18%; Final: 24%.

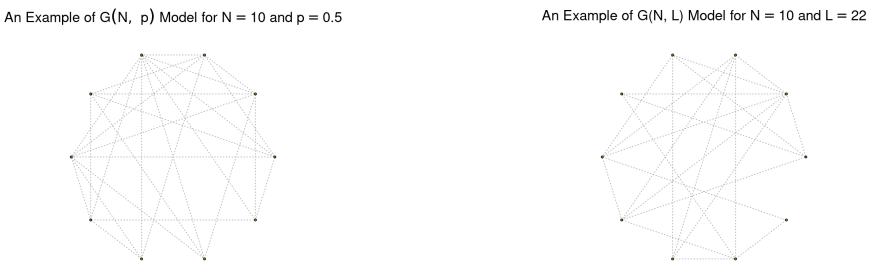
For the final exam, as a guide, students used the yeast protein interaction network file I shared with them, and created a new R Markdown file to answer questions about the immunoglobulin interaction network[16] that comes with `igraph` as well. The immunoglobulin interaction networks is an undirected and connected network modeling the interactions between amino-acids in the immunoglobulin protein. An edited version of the exam is included in Appendix B. As showcased in the fly (*drosophila medulla*) connectome network[17] file below (see Appendix A), this final exam can be used for individualized final projects as well, where students find a network of their choice and explore it by answering the provided questions. Even though the examples used here are from life sciences, networks from other fields such as social networks can be selected by students to analyze for final projects.

4 Code Overview

Real-world networks have randomness embedded in them to varying degrees depending on the context from which they arise. Therefore, creating models to study real-world networks require the use of tools from probability. By studying the syllabi or course schedules of thirty network science courses, mostly postgraduate, Sayama [8] shows that small-world networks and random networks are the top two

[16] https://rdrr.io/cran/igraphdata/man/immuno.html; this network is from an article by Gfeller [6].

[17] https://neurodata.io/project/connectomes/; this network is from an article by Takemura et al. [7]. According to this website, "A 'connectome' is a specific, cell-to-cell mapping of axonal tracts between neurons, created from cellular data like electron microscopy."



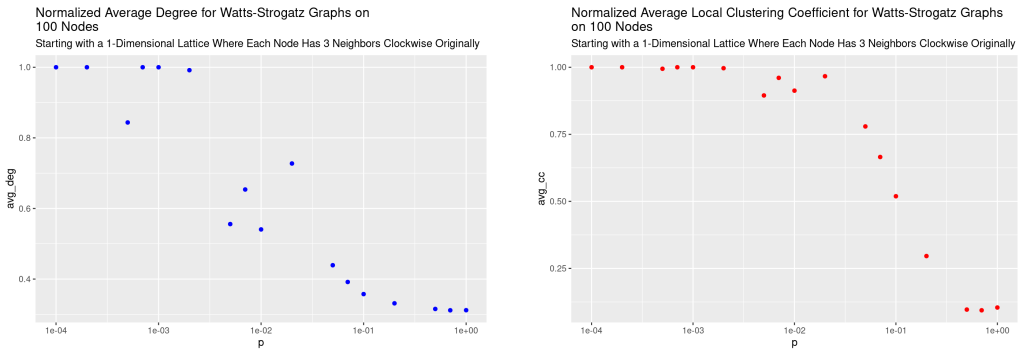
(a) (b)

Figure 1: Why it is important to use logarithmic binning instead of linear binning when visualizing power law degree sequence in a scatter plot on a log-log scale

topics covered in these courses. Moreover, preferential attachment, power-law distribution, and scale-free networks rank six to eight, respectively. These are also the main topics that are discussed in Chapters 3 through 5 of *Network Science*, which I covered in my course.

Chapter 3 introduces Erdős-Rényi random networks. There are two alternative ways to define an Erdős-Rényi random network. Suppose N is the number of nodes in a random network. In the first model, known as the $G(N, P)$ model, a link between a pair of nodes exists with a fixed probability p. In the second model, know as the $G(N, L)$ model, L links are randomly selected from all $\binom{N}{2}$ possibilities, where L is fixed. The code provides examples of both $G(N, p)$ and $G(N, L)$ models; see Figure 1 for $N = 10$, $p = 0.5$, and $L = 22$. Since the course did not have a probability or statistics prerequisite, the code discusses binomial and Poisson distributions (and their relation) in detail visually before exploring the degree distribution in a Erdős-Rényi random network. Next, the code, as does the book, explores how $G(N, p)$ random networks evolve as p increases. In particular, the code visually shows the threshold for which a $G(N, p)$ model becomes connected, i.e., $p > \ln(N)/N$. Moreover, the code demonstrates that the degree distribution for a $G(N, p)$ random network follows a Poisson distribution.

Next, examples of Watts-Strogatz model are generated, with a focus on eccentricity and local clustering coefficients. In the Watts-Strogatz model, one starts with a circular regular lattice in which each node has a fixed number of neighbors. Then, each link is rewired with a fixed probability p, called the rewiring probability. The eccentricity of a node v in a network is the maximum path length where one end of a path is v. Moreover, the local clustering coefficient of v measures the interconnectivity of v's neighbors. One point of emphasis here is how examples of this

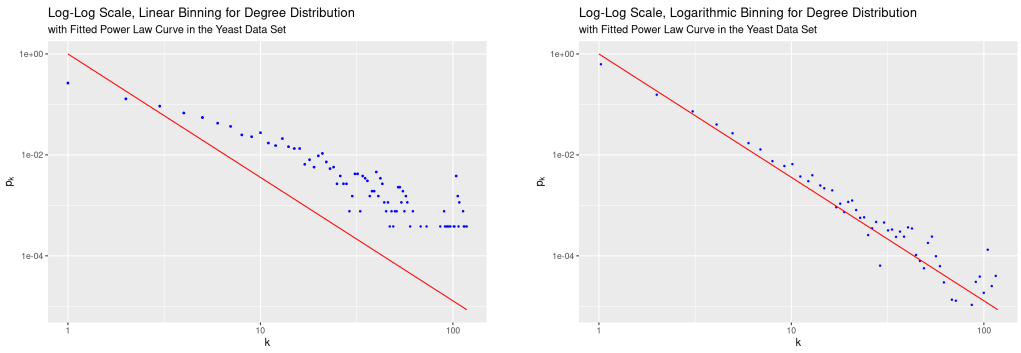


(a) (b)

Figure 2: Replicating Figure 3.14(d) on p. 97 in *Network Science*

model evolve as one increases the rewiring probability. In particular, Figure 3.14(d) on p. 97 in *Network Science*, which explores the relation between average degree and average local clustering coefficient as functions of rewiring probability in the Watts-Strogatz model, is replicates. As we see in Figure 2, as p increases from zero to one, there is an earlier sharp drop in average degree while the sharp drop for local clustering coefficient happens for higher values of p.

I used the yeast protein interaction network as the real-world example to cover some of the topics in Chapter 4 such as hubs, degree exponent, and power laws. In a network G, let p_k be the probability that the degree of a node is k. When the degree sequence in G follows a power law distribution, then $p_k \sim k^{-\gamma}$ for some fixed γ, which is called the degree exponent. Moreover, hubs are nodes that are in the long tail of a power law distribution. The book also emphasizes the importance of logarithmic binning as opposed to linear binning when plotting power law degree sequence in a scatter plot on a log-log scale. The code demonstrates the difference between the two types of binning; see Figure 3. In linear binning, all the bins have the same width, while in logarithmic binning, the width of bins increase logarithmically. Since both axes are on a logarithmic scale in the scatter plots in Figure 3, we see that, even though the underlying data is the same in both plots, the fitted power law curve is closely following the dots in the scatter plot with logarithmic binning. The last topic the code covers from Chapter 4 is generating random networks with pre-defined degree distributions. The code uses the yeast protein interaction network to provide examples of the two models that generate random networks with pre-defined degree sequences: the configuration model and the degree-preserving randomization model. In the configuration model, we start with an existing network or a pre-defined degree sequence for a network, and rewire the nodes randomly while preserving the existing degree sequence. This rewiring



(a) (b)

Figure 3: The importance of using logarithmic binning instead of linear binning when visualizing power law degree sequence in a scatter plot on a log-log scale

does not prevent us from creating self-loops (a link connecting a node to itself) or multi-links (multiple links connecting the same pair of nodes). In the degree-preserving randomization model, at each step, we swap a random pair of existing links as far as this swap does not create a multi-link.

In addition to hubs, degree exponent, and power laws, the code for the yeast protein interaction network creates visualizations of this network along with some of its summary statistics, degree distribution, and local clustering coefficient distribution. Moreover, the code visually explores the relation between eccentricity and local clustering coefficient on the one hand and degree on the other hand.

Lastly, the code provides examples of Barabási-Albert model, which is covered in Chapter 5. Barabási-Albert random networks are generated by adding a new node at each step under two conditions: each new node is connected to m existing nodes in the network (this is called growth), where m is fixed, and the probability that the new node is connected to an existing node depends on that node's degree (this is called preferential attachment). A good number of real-world networks have hubs and their degree sequences follow a power law distribution. Preferential attachment guarantees that the generated network has both of these properties. The code in particular shows how such networks evolve when the number of links added at each step or the power of preferential attachment increases while the number of nodes is fixed. These are two of the arguments in the `igraph` function `sample_pa` that generates Barabási-Albert random networks.

5 Reflection

As a great number of undergraduate data science programs has been established in recent years, it is essential that, in addition to the more established topics such as cleaning, analyzing, modeling, and visualizing data, students are exposed to concepts and ideas from network science as well. Even though this course is an elective for the mathematics and data science programs at my college, I think it played an important role in introducing my students to not only graph theoretic concepts but also topics from network science and in creating an opportunity for them to gain hands-on experience experimenting with these topics—something that is traditionally missing from introductory undergraduate graph theory courses. As Porter [4] concludes, "An introductory mathematics course about networks is an important component of a 'discrete structures and data science' pathway through an undergraduate degree in the mathematical sciences ... ."

One of my objectives for this course was to use as many open-source and freely available resources as possible. With the exception of the textbook that I used for the topics that I covered in the first half of Spring 2020, students had access to all the resources for this course without needing to purchase access codes or licenses:

- *Network Science* is available online under a Creative Commons Attribution-NonCommercial 3.0 Unported License.
- Students had electronic access to *Linked* through Saint Michael's College Library.
- Students downloaded and installed R, RStudio, and R packages for free, under different licenses.
- Students had free access to DataCamp courses and material for six months through DataCamp for Classrooms.

Of course, one of the main challenges that I had to face was conducting the second half of Spring 2020 remotely. Despite this hindrance, the course went well overall, especially with the availability of technological tools such as video conferencing software. Students reported that they learned a great deal in the course and they enjoyed the diversity of the material that was covered. The availability of *Network Science* as an online textbook was an incredible resource as well. Teaching this course fully in person again will certainly provide me with a better gauge of what needs to be improved in this course.

6 Acknowledgments

I would like to thank the Office of the Dean of the Faculty at Saint Michael's College for the Merit-based Course Reduction Award in Fall 2020 that allowed me to organize the R code mentioned on this page. I would also like to thank my colleagues Christopher Desjardins, who is an invaluable resource when I have an R-related question, and Joanna Ellis-Monaghan, who gave me careful and thorough feedback on earlier drafts of this page.

References

[1] A.-L. Barabási. *Network Science*. Cambridge University Press, 2016.

[2] H. Sayama, C. Cramer, M. A. Porter, L. Sheetz, and S. Uzzo. What are essential concepts about networks? *Journal of Complex Networks*, 4(3):457–474, 11 2015.

[3] C. Von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399–403, 2002.

[4] M. A. Porter. An undergraduate mathematics course on networks. In C. B. Cramer, M. A. Porter, H. Sayama, L. Sheetz, and S. M. Uzzo, editors, *Network Science In Education: Transformational Approaches in Teaching and Learning*, pages 3–21. Springer International Publishing, 2018.

[5] A.-L. Barabási. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Basic Books, 2014.

[6] D. Gfeller. *Simplifying complex networks: from a clustering to a coarse graining strategy*. PhD thesis, 2007.

[7] S. Takemura, A. Bharioke, Z. Lu, A. Nern, S. Vitaladevuni, P. K. Rivlin, W. T. Katz, D. J. Olbris, S. M. Plaza, P. Winston, et al. A visual motion detection circuit suggested by drosophila connectomics. *Nature*, 500(7461):175–181, 2013.

[8] H. Sayama. Mapping the curricular structure and contents of network science courses. In C. B. Cramer, M. A. Porter, H. Sayama, L. Sheetz, and S. M. Uzzo, editors, *Network Science In Education: Transformational Approaches in Teaching and Learning*, pages 101–116. Springer International Publishing, 2018.

A Code

The R Markdown files (along with their Jupyter Notebook counterparts) may be found on GitHub[18] and their rendered HTML files on RPubs[19].

- **Chapter 3: Random Networks – Erdős-Rényi**

  Code on GitHub:

  - `Chap3--Erdos-Renyi_Model.Rmd`
  - `Chap3--Erdos-Renyi_Model.ipynb`

  Rendered HTML on RPubs: Chapter 3: Random Networks – Erdős-Rényi

- **Chapter 3: Random Networks – Small Worlds**

  Code on GitHub:

  - `Chap3--Small_World_Model.Rmd`
  - `Chap3--Small_World_Model.ipynb`

  Rendered HTML on RPubs: Chapter 3: Random Networks – Small Worlds

- **Chap. 4: The Scale Free Property**

  Code on GitHub:

  - `Chap4.Rmd`
  - `Chap4.ipynb`

  Rendered HTML on RPubs: Chap. 4: The Scale Free Property

- **Chap. 5: The Barabási-Albert Model**

  Code on GitHub:

  - `Chap5.Rmd`
  - `Chap5.ipynb`

  Rendered HTML on RPubs: Chap. 5: The Barabási-Albert Model

- **Yeast Protein Interaction Network**

  Code on GitHub:

[18] https://github.com/abarghi/Using_R_in_an_Applied_Graph_Theory_Course
[19] https://rpubs.com/abarghi/

  - `Yeast.Rmd`
  - `Yeast.ipynb`

  Rendered HTML on RPubs: Yeast Protein Interaction Network

- **Fly (*Drosophila Medulla*) Connectome Network**

  Code on GitHub:

  - `Fly.Rmd`
  - `Fly.ipynb`

  Rendered HTML on RPubs: Fly (*Drosophila Medulla*) Connectome Network

To run the above files, the following packages are needed:

- CRAN packages: `tidyverse`, `igraph`, `ggraph`, `latex2exp`, `e1071`
- GitHub package: `qdread/forestscaling` (requires installing the package `remotes`)

B Final Exam

Instructions:

Please submit your work as an R Markdown file which contains your code and a written report which contains your answers, observations, and the main outputs. Your written report should be a PDF document and it can be generated using the `Knitr` button.

1. Create an R Markdown (`.Rmd`) file: `File` > `New File` > `R Markdown` and select PDF as the default output format.
2. Save this file as `Final_[your_username].Rmd`.
3. Delete everything below the first chunk of code, which contains code for `Knitr`.
4. Create a code chunk and load the following packages: `tidyverse`, `igraph`, `ggraph`, `latex2exp` and `e1071`.
5. Make sure that for all chunks of code that require a random process, you use a random seed generator. For example, use `set.seed(42)`.

6. For the written report, please do not include your code or its output unless a question explicitly asks for it.

Questions:

`Yeast.Rmd` as a guide, load the immunoglobulin interaction network (`immuno`) that comes with `igraph` and answer the following questions. You can read more about this network here[20]. If you are running `Yeast.Rmd` concurrently with your file to compare results, you want to use a short network name other than `g` for the `immuno` network, e.g. `h`.

1. What is the order of the immunoglobulin interaction network? What is the size of this network? How many components does it have? Is the network directed or undirected? Do the nodes have attributes? Do the links have attributes? Include your answers to these questions in your written report, but not your code or its output.

2. Using `lgl`, `mds`, and `drl` layouts separately, what do you observe when you visualize the immunoglobulin interaction network ? Describe your observations for each of these layouts. Use at most 25 words for each layout. Include your observations in your written report, but not your code or its output.

3. Create a `tibble` that contains eccentricity, betweenness, degree, transitivity, eigen centrality, and hub score for the immunoglobulin interaction network. You need to look up the documentation for the last two statistics in Help. Note that both `eigen_centrality` and `hub_score` return three lists and you need to access one of them, i.e., `vector,` before wrapping it in `enframe`. Make sure that you make the appropriate changes to the code. Do not include your code or its output in your report.

4. Create a `tibble` that includes the average degree, maximum degree, diameter, average clustering coefficient, average betweenness, average eigen centrality, and average hub score for the immunoglobulin interaction network . Include the output in your report but not your code. Do you notice anything interesting in this `tibble`? Is this observation you are making also true when you look at the `tibble` in Question 3?

5. Find the average distance in the immunoglobulin interaction network and create the plot for the distribution of distance in this network. What do you notice about this plot? Describe your observations in at most 50 words. Include your answer and the plot in your report, but not your code or other outputs.

[20] https://rdrr.io/cran/igraphdata/man/immuno.html

6. Create the plot for the degree distribution in the immunoglobulin interaction network . What do you notice about this plot? Describe your observations in at most 25 words. Include your answer and one plot in your report, but not your code or other outputs.

7. Based on Question 6, do you think that the degree distribution follows a power law in the immunoglobulin interaction network ? In other words, is this is a scale-free network? If so, find its degree exponent.

8. Create a plot (using dots rather histograms) containing both the degree distribution for the immunoglobulin interaction network (use blue) and the Poisson distribution with the correct parameter (use red). Include the plot in your report, but not your code.

9. Create a plot that visualizes the average of clustering coefficients based on the degree in the immunoglobulin interaction network . Is the average clustering coefficient matches what the Erdős-Rényi model predicts? Include your answer and one plot in your report, but not your code or other outputs.

10. Create a plot that visualizes the relationship between betweenness and degree in the immunoglobulin interaction network . What do you notice about this plot? Describe your observations in at most 50 words. Include your answer and the plot in your report, but not your code or other outputs.

11. Create a plot that visualizes the relationship between eccentricity and degree in the immunoglobulin interaction network . What do you notice about this plot? Describe your observations in at most 50 words. Include your answer and the plot in your report, but not your code or other outputs.

12. Create a plot that visualizes the relationship between clustering and degree in the immunoglobulin interaction network . What do you notice about this plot? Describe your observations in at most 50 words. Include your answer and the plot in your report, but not your code or other outputs.

13. Create a plot that visualizes the relationship between hub score and degree in the immunoglobulin interaction network . What do you notice about this plot? Describe your observations in at most 50 words. Include your answer and the plot in your report, but not your code or other outputs.