Project 1

Computational Analysis of Social Complexity - CAP 6318

Due Date: Three weeks from assignment date **Groups:** 2-3 students (randomly assigned) **Weight:** 15% of course grade

Overview

This project is your opportunity to apply the computational and theoretical concepts from weeks 1-6 to a real-world problem. You will work in teams to analyze a network using Julia, demonstrate your understanding of graph theory concepts, and present clear, well-documented findings.

The project is intentionally open-ended to allow you to explore topics that interest you. You may focus on any combination of concepts covered so far: basic graph theory, strong/weak ties, homophily, weighted graphs, structural balance, production networks, or agent-based models.

Instructions

1. **Select a domain and dataset**: Choose a real-world network to analyze. This could be:
   - Social networks (Twitter/X connections, Facebook friendships, collaboration networks)
   - Economic networks (trade relationships, supply chains, input-output tables)
   - Biological networks (protein interactions, food webs, neural connections)
   - Infrastructure networks (transportation, power grids, communication networks)
   - Any other network that interests you and has publicly available data
2. **Formulate research questions**: Develop 2-3 specific questions about your network. Examples:
   - "Do high-degree nodes in this social network exhibit homophily in political affiliation?"
   - "What is the impact of disruptions to key sectors in a regional economic network?"
   - "How does the clustering coefficient vary across different communities in this network?"
   - "Can we identify weak ties that serve as bridges between communities?"
3. **Conduct computational analysis**: Use Julia to:
   - Load and preprocess your network data
   - Compute relevant network metrics (degree distribution, clustering coefficients, centrality measures, etc.)

- Implement or apply appropriate algorithms (BFS, shortest path, community detection, etc.)
- Generate visualizations of your network and findings
- Test hypotheses or explore patterns related to your research questions

4. **Document your work**: Create a Jupyter notebook (`.ipynb`) that includes:
   - Clear markdown sections explaining your approach
   - Well-commented Julia code
   - Visualizations with appropriate labels and captions
   - Interpretation of your results
5. **Write supporting code**: If your analysis requires custom functions or data processing:
   - Organize reusable code into `.jl` files
   - Use modules to structure your code cleanly
   - Include docstrings for your functions
   - Demonstrate good software engineering practices

Deliverables

Your team will submit:

1. **Jupyter Notebook** (`project1.ipynb`): Your main analysis document containing:
   - Introduction: domain background, dataset description, research questions
   - Methods: description of your computational approach and algorithms used
   - Results: visualizations, computed metrics, and findings
   - Discussion: interpretation of results and connection to course concepts
   - Conclusion: summary of key insights and limitations
2. **Julia Scripts** (`*.jl` files): Any supporting code files containing:
   - Custom functions you developed
   - Data preprocessing scripts
   - Module definitions (if applicable)
3. **Data** (if not too large): Include your dataset or provide clear instructions for obtaining it
   - If dataset is large (>10 MB), include a `data_README.md` explaining how to obtain it
   - Include any preprocessing scripts needed to go from raw data to analysis-ready format
4. **Project Environment** (`Project.toml`): Your Julia project dependencies
   - Ensure your code can be reproduced by running `Pkg.instantiate()`
5. **README** (`README.md`): Brief overview including:
   - Project title and team members
   - Brief description (2-3 sentences)
   - Instructions to run your analysis

- List of key findings (3-5 bullet points)

Resources

- **Course materials**: Lecture notebooks from weeks 1-6 provide examples and reference implementations
- **Julia packages**: Consider using:
  - `Graphs.jl` and `SimpleWeightedGraphs.jl` for network analysis
  - `GraphPlot.jl`, `NetworkLayout.jl`, or `PlotlyBase.jl` for visualization
  - `CSV.jl` and `DataFrames.jl` for data handling
  - `LinearAlgebra` for matrix operations
  - Any other packages relevant to your analysis
- **Datasets**:
  - Stanford Large Network Dataset Collection (SNAP)
  - Network Repository
  - UCI Network Data Repository
  - Government data portals (BEA, Census, etc.)
  - Domain-specific repositories in your area of interest
- **Julia documentation**: docs.julialang.org
- **Course textbooks**:
  - Easley & Kleinberg - Networks, Crowds, and Markets (chapters 2-5)
  - QuantEcon Networks lectures (chapters 1-2)

Grading Rubric

Your project will be evaluated on three equally-weighted dimensions:

1. Julia Proficiency (33 points)

Excellent (30-33 points)

- Code is clean, well-organized, and follows Julia best practices
- Effective use of Julia-specific features (multiple dispatch, broadcasting, etc.)
- Custom functions are well-designed with appropriate type annotations
- Code is efficient and makes good use of available packages
- Project environment is properly configured with `Project.toml`
- All code runs without errors and produces expected outputs

Good (24-29 points)

- Code is mostly clean and functional
- Uses Julia packages appropriately
- Some custom functions defined, though may lack polish
- Minor inefficiencies or style issues present
- Code runs with only minor issues

Satisfactory (18-23 points)

- Code is functional but may be poorly organized
- Limited use of Julia-specific features
- Heavy reliance on examples without adaptation
- Some errors or required manual intervention to run
- Minimal code documentation

Needs Improvement (0-17 points)

- Code is difficult to understand or poorly structured
- Significant errors prevent execution
- Little evidence of Julia understanding beyond copying examples
- Missing or incomplete documentation

2. Theoretical Understanding (33 points)

Excellent (30-33 points)

- Research questions are thoughtful and well-motivated
- Analysis demonstrates deep understanding of relevant graph theory concepts
- Correctly applies multiple concepts from course material
- Shows insight into why certain metrics or approaches are appropriate
- Makes meaningful connections between theory and empirical findings
- Discusses limitations and assumptions clearly

Good (24-29 points)

- Research questions are reasonable and relevant
- Correctly applies course concepts to the problem
- Shows understanding of the theory behind the methods used
- Makes some connections between theory and findings
- Acknowledges some limitations

Satisfactory (18-23 points)

- Basic research questions stated
- Uses course concepts but with limited depth
- Some theoretical understanding evident but superficial
- Limited discussion of why methods are appropriate
- Minimal discussion of limitations

Needs Improvement (0-17 points)

- Unclear or poorly motivated research questions
- Misapplies theoretical concepts
- Little evidence of understanding beyond mechanics
- No discussion of appropriateness or limitations

3. Presentation and Communication (34 points)

Excellent (31-34 points)

- Notebook has clear narrative structure with logical flow
- Visualizations are publication-quality with proper labels and captions
- Writing is clear, concise, and free of errors
- Code is well-commented and easy to follow
- Results are interpreted thoughtfully
- README provides all necessary information
- Overall professional presentation

Good (25-30 points)

- Notebook structure is logical and easy to follow
- Visualizations effectively communicate findings
- Writing is generally clear with few errors
- Code comments are present and mostly helpful
- Results are interpreted reasonably
- README is complete

Satisfactory (19-24 points)

- Basic structure present but may lack coherence
- Visualizations present but may lack polish or clarity
- Writing is understandable but may have errors or be verbose
- Code comments are sparse or unhelpful
- Limited interpretation of results
- README is minimal

Needs Improvement (0-18 points)

- Poorly organized or difficult to follow
- Visualizations are missing, unclear, or misleading
- Writing is unclear or has significant errors
- Code lacks comments or documentation
- Results presented without interpretation
- README is missing or incomplete

Tips for Success

1. **Start early**: Three weeks may seem like a lot of time, but finding the right dataset and getting it into the right format can take longer than you expect.
2. **Communicate with your team**: Establish regular meeting times and use version control (Git) if possible to coordinate your work.
3. **Keep it focused**: It's better to do a thorough analysis of 2-3 well-chosen questions than a superficial analysis of many questions.

4. **Test your code frequently**: Don't wait until the end to run your entire notebook. Make sure each cell works before moving on.
5. **Use the discussion board**: If you're stuck on a technical issue, post to the course discussion forum. Your classmates and TAs are resources!
6. **Iterate on visualizations**: Your first plot is rarely your best plot. Take time to make your figures clear and informative.
7. **Document as you go**: Don't leave all the writing for the end. Document your thought process as you work.
8. **Review course materials**: The lecture notebooks contain many examples and patterns you can adapt for your project.
9. **Cite your sources**: If you use external resources (datasets, code snippets, ideas), cite them appropriately.
10. **Have fun**: This is your chance to explore something you find interesting. Choose a problem you're genuinely curious about!

Academic Integrity

Remember the course AI policy: you are encouraged to use GenAI tools (ChatGPT, Claude, Copilot, etc.) to help with your project, but you must:

- Disclose all AI usage in your notebook
- Include prompts you used
- Take responsibility for the accuracy of any AI-generated content
- Ensure you understand any code or concepts produced with AI assistance

You are responsible for your work. You should be able to explain every line of code and every conclusion in your project.

Questions?

If you have questions about the project:

- Post to the course discussion forum for technical questions
- Contact the instructor or TA during office hours for guidance on scope or approach
- Discuss with your team members to clarify requirements

Good luck, and enjoy exploring the fascinating world of network analysis!