SI649 Final Assignment

Fall 2023

Important Dates and Deliverables:

- **Friday, November 18, 2023, 9pm** - Check-in. (1) Design: we will ask you to run a vizitcard exercise for yourselves in your group and report on what you have, (2) Implementation: We want to see a dashboard that can talk to the server and display a couple of plots based on received data
    - Submit to Canvas: Google Slides of VizItCards Activity
    - Submit to Canvas: 5 minute video
    - Submit to Canvas: Video of a working demo
- **Friday, Dec 1, 9pm** - Check-in (1) Design: An updated set of target designs for your interface, (2) Implementation: Updates on implementation should be nearly complete version
    - Submit to Canvas: Short video update
    - We will ask you to meet with schedule a meeting with us during office hours that week to make sure you can connect to the server and things are working
- **Monday, Dec 4**, Final tournament - details TBD, but we will be using some subset of labs
    - Bring with you: Final team dashboard, your game face
- **Saturday, Dec. 10, midnight**: Final Report Due
    - Submit to Canvas: Short final video that has a demo of your design
    - Submit to Canvas: Your code needed to run it
- **Sunday, Dec. 11**, midnight: Peer-review
    - Submit to qualtrics: answers to short survey

Organizing Groups

Organize into teams of 3-5 people on this google spreadsheet:
https://docs.google.com/spreadsheets/d/1rA42Mydph7xMVxdxjGtyxN-qj9lid0iiANbisXjSNio/edit#gid=0
Please come up with a name for your team on the left column, then add team member names in the row. Feel free to add a note on the right side column.

Deliverable 1 - VizItCards Exercise, and basic functions (10%)

Deadline: November 18 (9pm)
Activity:

- We want your team to complete a VizItCard design exercise for this project. You should record your domain cards, sketches, and team solution in a slide deck (you can use the vizitcard lab template). Please don't wait until the last minute to do this, as you should be already implementing the designs into code to prepare for the following week. We also want you to record a short video (no longer than 5 minutes!) explaining your domain cards, team solution sketch, and any strategies your team is considering for the game. This deliverable will be worth 10% of the Robogames grade.

- We want you to implement a base dashboard. We will give you some starter code using Panel, but you're welcome to use something else (Streamlit or a Javascript-based solution are options). We will ask you to replace at least a couple of the dataframes that you'll see in our sample with actual visualizations. Your code should connect to the demo server and your visualizations should update.

Submit: VizItCards activity slides, a short video describing your current approach, and a second short video showing that your code works.

Deliverable 2 - Design Update and Functionality test (10%)

Deadline: December 1 (9pm)
Activity: By this deadline, we will ask that your team meet with us to show us that your code can connect with the server and run. You can meet with us during office hours to do this. We will also ask for you to submit a short video updating us on your current functionality/design.
Submit: a short video showing us your updated design and functionality
Bring with you to office hours/scheduled meeting: A running prototype that will be ready to connect to the server. Note that this milestone is required.

Final Tournament

Attend: Monday, Dec 4, time TBD, but likely at night
Activity: The final tournament! This session will be hosted as a bracket tournament where teams will face off head to head in a battle for ultimate victory. This is the final session, your team should be ready to play Robogames F23 with your final dashboard. We will also bring pizza, please let us know if you have any dietary restrictions.
Bring with you: Final team dashboard, your game face

Final Report (80% total - 60% for dashboard, 20% for video)

Deadline: December 10th, midnight
Your team will create a short final video that has a demonstration of your design. Please keep this to 10-12 minutes maximum. We'd like to see an explanation of your system (a short demo/walkthrough) but also some details about why you made design choices. A good report will also describe things you chose not to implement and things that inspired your design.
Your dashboard/system itself will be worth 60% of your grade. Full credit will be given for a working implementation which is clearly expressive/effective, is well designed (aesthetic), etc. The video will be worth 20% of the Roboviz grade.
Submit: Short final video and your code needed to run it

Before you start: Please note that while there are many complex rules and strategies, you do not need to build a system that does everything. We have designed the game so that you can build different kinds of visualizations (time, networks, trees, multidimensional). Each has a benefit, but you can choose to focus on some over the others. **Whatever you choose, you must pick *at least* two of the data types (e.g., time + multidimensional).**

Rubric

- 10% - VizItCards exercise for the Robogames project and basic functions: slides and video
- 10% - Updated video demo and functionality test
  - We expect a significant dashboard improvement from deliverable 1
- 20% - Final video report
  - Explanation of your system (short demo/walkthrough)
  - Why you made your design choices (with connections to course theory)
  - What you chose not to implement
  - Things that inspired your design
- 60% - Robogames system
  - Functionality (code functions without issues)
  - Visualization Design, expressiveness/effectiveness, includes at least two of the data types
  - Interactivity is well thought out and useful to game play
  - Aesthetics, well-designed
  - No extraneous visualizations–if they're not part of your strategy you should remove them from the final project (we'd still like to hear about them–but just because you built it, doesn't mean it needs to be included).
- Bonus Points (i.e., Extra Credit): There are a number of ways to earn extra credit: (1) We will award bonus points based on your team's performance during the final tournament, (2) We will also offer bonus points for dashboards/communicative vis that go above and beyond (e.g., implementing good views for all 4 data types), and (3) creating visualizations that combine datatypes (e.g., a single view where the timeseries and multivariate datasets are combined visually and/or interactively).

Robogames Instructions

Planet X Game Objective

Recruit as many (good) robot miners as you can in a limited amount of time (10 minutes).

You will need to build a dashboard to help your team recruit these robots.

Overview

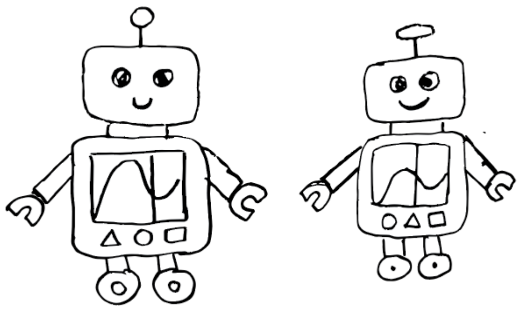
Welcome to Planet X421ZZ (Planet X for short)! You are a representative of an interdimensional company tasked with mining the resources of the planet. Bad news first: your competitors have also landed on the planet!

You won't be able to accomplish this goal without some help. There are robots that have been stuck on the planet for over 100 years. The colonists who used to live here left them when they moved on. The robots evolved, developed some personality quirks, and emulated some aspects of human society (with questionable accuracy). They've also evolved some serious abandonment issues so you'll need to work hard to convince them to join your team. There are 100 robots and not a lot of time (10 minutes) for you to work so you'll need to be strategic!

To recruit robots to your team, you can visualize two to four types of data:

- The robot friendship game (time series) - recruit the robot to your team by guessing the robot's number
- The social network (network) - recruit popular robots to your team to influence their friends
- Robot productivity (multidimensional) - determine which robots are productive and which are unproductive
- The family tree (hierarchical) - robots that are closely related will have similar number generators for the robot friendship game

Your class project will be to build an interface/dashboard that lets you play a match (you'll be playing against other teams). You don't have to implement an interface for all the strategies (more below) but you can if you want. If you choose to play, your team can use one or more players during the game (so a distributed dashboard). One note, the robots don't like AI competition: So one thing you ***can't do*** is implement an ***automated*** AI/ML/stats based solution (emphasis is on the vis, not building an automated system to play). For example: you

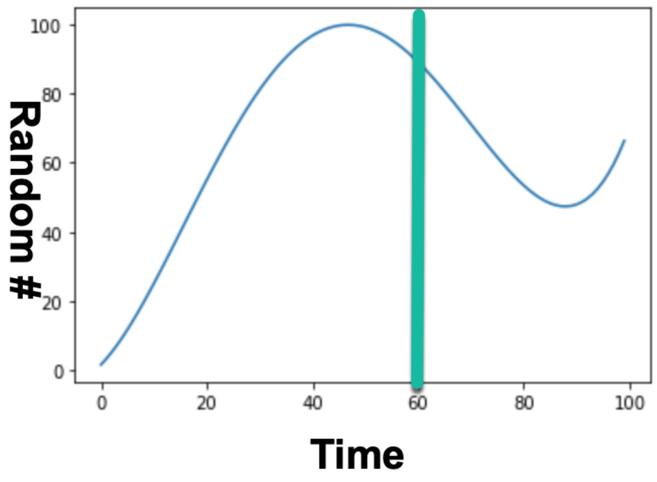


can write a program that will rank the robots by likely productivity and then visualize that data to enable the human to decide if that robot is worth friending. You ***can't*** write a program that automatically decides which robots are likely best and then automatically tries to friend them. **If you have any questions about whether your solution is using automation, please ask us.**

Each match lasts 100 X421ZZ time units (XTU). Each time unit is 6 earth seconds so the entire match lasts 10 earth minutes (100 XTUs).

The Robot Friendship Game (Time Vis)

Did we mention the robots have abandonment issues? The robots would like to make sure you're on the same wave-length as they are. They don't want you to leave them (like the last colonists) and have reasoned that if you think alike, they'll trust you more. To test you, they want you to play their favorite game: "guess my number" (they needed to do something to keep themselves entertained… give them a break!).



The friendship game is pretty basic. Each robot has a number generator that generates a new number (between 1 and 100) every XTU. Their random number generator is based on the amount of light one of their sensors detected, so it's not quite random (this will help you).

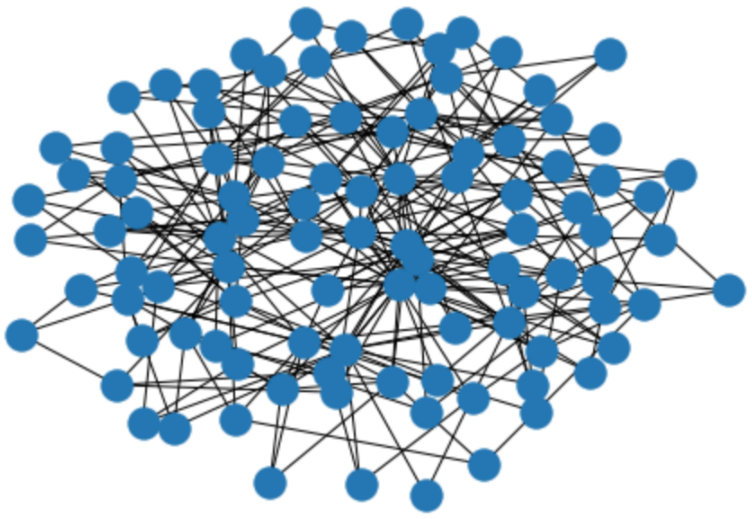
They've also randomly decided a time that you need to submit your guess by.

For example, Pushwalker Botson (each robot has a name and ID, so Pushwalker is also known as robot 87) has the random number generator and guess time displayed in the figure above. Pushwalker has decided it wants you to guess by time unit 60. It happens to have the "correct answer" 92. When 60 time units have gone by, Pushwalker will look at your guess and your opponent's guess and decide who it wants to join. Closer is better if you want to *win* that robot and have it "declare" for you (a robot that declares for you is part of your team).

Each robot has a different random number generator and expiration date. But because it's based on the light sensor, you'll notice from the figure that the generator isn't actually that random. If you know the value at time 52, you can probably have a good guess about 53 or 51. Even with a few sample points, you might be able to fit a reasonable line to have a good guess.

You can change your guess up to the deadline. Knowing nothing about the robots, you may as well guess something like “50” for all of them but then you’re leaving the game up to chance (if your competitor does the same). As you’ll see below, there are ways about gaining information about the random number generators for each robot (see “the hacker”) so you don’t need to randomly guess.

- If you guess “-1” that means you don’t want the robot (more on why below)
- If both teams guess -1, the robot is sad and powers down (no one gets the robot)
- If your guess is closest to the correct answer and the other team is not within 10 of the correct answer, you will get the robot
- If both teams are within 10 of each other, the robot will decide using the social network strategy (more below)



The Social Network (Network Vis)

The robots have evolved their own social network (we told you they liked to behave like humans). Their network looks something like the figure to the left. They’re very proud of their network and are happy to tell you who all their friends are before the game even starts.

The robots rely on their social network to help them decide which team to join. So if both teams are close to each other’s answer in the friendship game (or there’s a tie), the robot will look to their social network to help them decide.

The way they do this is by considering their immediate neighbors that have already committed to one of the two teams. They will then take the weighted average of their neighbors based on how popular each neighbor is. For example, let’s say Pushwalker Botson is friends with Wallminer Botberg Jr., Stoneminer Boterson, Rockhauler Botsky I. Wallminer is very popular and has 10 connections (Wallminer went with team 1). Stoneminer and Rockhauler have 7 and 4 friends respectively (both have gone with team 2).

Looking at these numbers, Pushwalker will go with team 2 (11 > 10).

What this tells you, is that if you can, it may help to identify popular robots with earlier expiration dates. If you can somehow get them on your team, it may help you get other robots. Again, you *don't need* to adopt this strategy, but it can help.

Note that the robot's social network may not look like a human social network. While the network is guaranteed to be connected, it may take any shape (a ring, a lattice, a small-world graph, etc.).

Robot Productivity (Multidim Vis)

Wouldn't it be great if all robots were productive? Why yes it would. Unfortunately, that's not the way things have emerged on planet X421ZZ. The robots have been around for a while and things have started falling apart. They have started using parts from older, powered down robots to try and repair themselves. Because some parts are great and some are bad, some robots will be productive (they'll be good at mining rocks!) whereas others will have negative productivity (they'll make you lose rocks!). On average, robots are on average more productive than not (in these cases: if you somehow win all the robots you will have positive productivity). However, you will occasionally experience "hard mode" games–these are more likely the further you are in the tournament. In hard mode games, the robots may not have an average positive productivity: if you win them all, you'll lose.

It turns out there are 10 parts/features of a robot you care about: 'Astrogation Buffer Length', 'InfoCore Size', 'AutoTerrain Tread Count', 'Polarity Sinks', 'Cranial Uplink Bandwidth', 'Repulsorlift Motor HP', 'Sonoreceptors', 'Arakyd Vocabulator Model', 'Axial Piston Model', 'Nanochip Model'. The first 7 are quantitative features (e.g., 7.3 or -92) and the last 3 are nominal/categorical (e.g., Alpha, Beta, Gamma).

Variable Name	**Type**
Astrogation Buffer Length	Quantitative
InfoCore Size	Quantitative
AutoTerrain Tread Count	Quantitative
Polarity Sinks	Quantitative
Cranial Uplink Bandwidth	Quantitative
Repulsorlift Motor HP	Quantitative
Sonoreceptors	Quantitative
Arakyd Vocabulator Model	Nominal
Axial Piston Model	Nominal
Nanochip Model	Nominal

A robot's productivity is some function of these parts. For example, you might find that the higher the *Astrogation Buffer Length* the higher the *Productivity*. On the other hand, model Alpha of the *Axial Piston Model* yields negative productivity while Beta is positive!

Because you only want productive robots, your best strategy is to try and figure out which robots are productive and which are not. You may even want to try and get the other team to win unproductive robots. Not all parts/features will be important and some will be highly correlated and not give you new information.

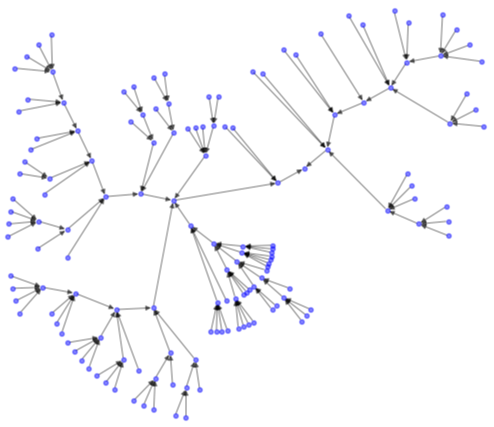
Once a robot has been declared for one team or the other, both teams will know its productivity and can use this for future guesses.

Again, you start the game not knowing much about which parts each robot has, but that's where the helpful hacker will come in. Also note that these things can change between matches so don't count on anything you learn in one holding against your next opponent.

The Family Tree (Hierarchical Vis)

In mimicking human societies, one robot, long ago, decided that it wanted to "evolve." What this means is that over time, old robots built newer ones and a family tree of sorts developed (see image below). All the robots living on the planet today are evolved from that robot. As with the social network, the robots are happy to give you their genealogy when you start the game.

Why do you care? When a robot creates another robot, it will build it nearby and possibly of similar parts. That means that the "child" robot parts will be very similar to the "parent." In turn, this means that the random number generator for that child will be similar (but not the same) as the parent. This also means that the productivity for the child will be similar as well. If you know a parent's random number generator, you have a better chance of getting the child's. Similarly, if you know one robot's generator, you have a better guess about their "sibling." The closer robots are in this tree (number of hops) the more similar their random number generator and productivity will be.



One strategy you might consider is using data from one robot's generator to "fill in the blanks." For example, you might find out that Pushwalker Botson will say "72" at time 31. You can guess that Pushwalker's brother Extrahauler will also say 72 +/- some amount at time 31. The more robots you learn about, the better you can triangulate this value.

Similarly, Pushwalker's creator, Wallbot (who does not exist anymore), was also likely to be 72 +/- some amount at time 31.

While there are only 100 robots on the ground today (ids are 0 to 99, inclusive), there were more before. Records for them exist and you will be able to get some of that information using the hacker (see below). These robots will have IDs >= 100. You don't need to guess their values, but they'll be useful to know about.

The Hacker

At the start of the game, you will be given:

- The robot names and ids
- The social network
- The family tree
- Each robot's deadline for the friendship game

At any time during a running game, you can get:

- Which robot has declared for which team
- The productivity of your declared robots

Additionally, during the game you have access to your hacker!

The robots won't give you details about their random number generator, parts, or productivity level. So you start knowing very little about what to do. But the good news is that you have a hacker at your disposal. Every XTU (X421ZZ time units) you can ask the hacker what they found out about the robots. The hacker will be able to deliver:

- 10 data points to you about a robot's random number generator (for example, they might tell you that Wallhauler's random number at time 54 will be 27). A way of thinking of this is that there are ~150 robots (the 100 in the game plus the ancestors) and 100 time points. This is ~15000 possible cells. The hacker will give you 10 of these at a time.
- They will also be able to give you 6 data points about the parts for a robot (for example, they might tell you that Pushwalker's *Cranial Uplink Bandwidth* is 921). This will only apply to robots in the game, not the ancestors. So 100 robots x 10 part columns. The hacker will give you 6 of these at a time.

While there is some randomness to what the hacker can tell you, you can register interest in specific robots and/or specific parts (this is done through the API we describe below). For example, you can tell the hacker you prefer information about robots 8, 23, and 75 and would like more data on *InfoCore Size* and *Nanochip Models*. Once you register interest, you will get some random information *and* some information about the things you care about. So one strategy might be to change what you ask the hacker as the number of undeclared robots changes.

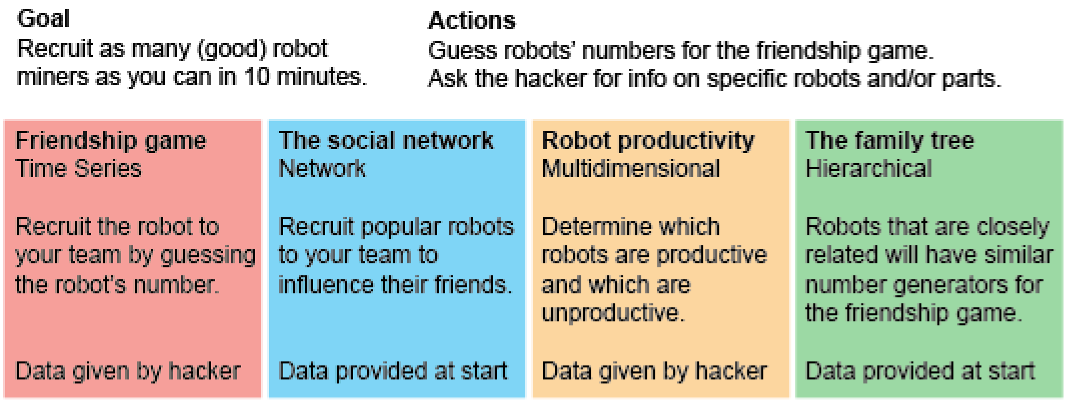
Some more details

You DO NOT need to build a dashboard for everything or even try to grapple with all the strategies. Your team can decide what they want to focus on. One strategy might be to focus on mainly the friendship game and try to build the best time series visualizations to help you decide what to guess.

To help you practice, we'll be giving you some sample datasets of different matches, some easier and some harder. We'll also give you the server code (more below) to help you simulate the game.

- For now, you'll find two gexf files (https://gephi.org/gexf/format/) corresponding to the social network and family tree. These are compatible with networkx (https://networkx.org/, a Python package) for visualization/analysis (and you can load them into Gephi to see what they look like: https://gephi.org/).
- You will also find two json files for the social network and family tree. This is the same data as the gexf in a format compatible with D3 (https://networkx.org/documentation/stable//reference/readwrite/json_graph.html).
- You will also get a csv file that has all the data for that match instance (in a real game you won't have this). Robots 1-100 are the robots you need to recruit. Any other robots in the dataset are no longer around, but will be useful for you.

Game Summary



Goal
Recruit as many (good) robot miners as you can in 10 minutes.

Actions
Guess robots' numbers for the friendship game.
Ask the hacker for info on specific robots and/or parts.

Friendship game Time Series	**The social network** Network	**Robot productivity** Multidimensional	**The family tree** Hierarchical
Recruit the robot to your team by guessing the robot's number.	Recruit popular robots to your team to influence their friends.	Determine which robots are productive and which are unproductive.	Robots that are closely related will have similar number generators for the friendship game.
Data given by hacker	Data provided at start	Data given by hacker	Data provided at start

The Server

We've implemented the game engine as a web service. We'll share how to use it in javascript soon, but we've packaged up everything you need for Python in canvas. You will want to download the example files. We have provided some easier gameplay examples and some harder gameplay examples. We will make this available on github: https://github.com/eytanadar/si649robogames. We will let you know about updates or bug fixes. You can load up Example.ipynb in the client directory using jupyter and follow the instructions (you'll need to run the server in a separate terminal). We may update the server code over the course of the project, but the client-side API will be largely stable.

Frequently Asked Questions

Can I automate my system to make the guesses for me? No, you cannot have your code automatically make the guesses for you.

What software can I use to create my system? We recommend something like Panel (https://panel.holoviz.org/), Streamlit (https://streamlit.io/), or Voila (https://voila.readthedocs.io/en/stable/). You do not have to use Altair and are welcome to use whatever libraries work for you. That said, you can take advantage of supplemental libraries like nx_altair (https://github.com/Zsailer/nx_altair) for visualizing networks in Altair. While you can do your development in Jupyter notebook, we do not recommend keeping your dashboard in that environment. You're also welcome to use Javascript-based solutions if you like (we've given you some example code for that).

Teams have tried to use Tableau in the past and have had problems refreshing the live data. There are some workarounds to using live refreshing data in Tableau, but it's more complicated and can lead to problems. We won't forbid the use of Tableau, but be prepared for some headaches if you go that route.

Can I get a brief recap of the game? Here's a brief overview in video form: https://www.youtube.com/watch?v=DzAGYCH8ZEU