Prioritizing Visual Encoding in Networks

Using the rDynamo and rNetVisor packages

Nikhil Gopal

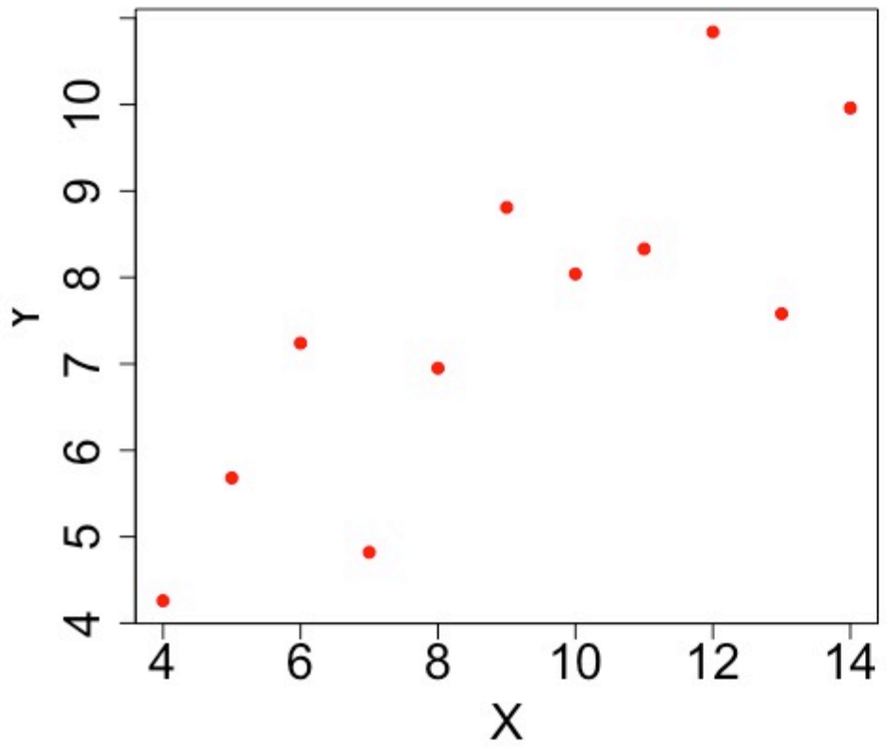
A Brief Story About Me

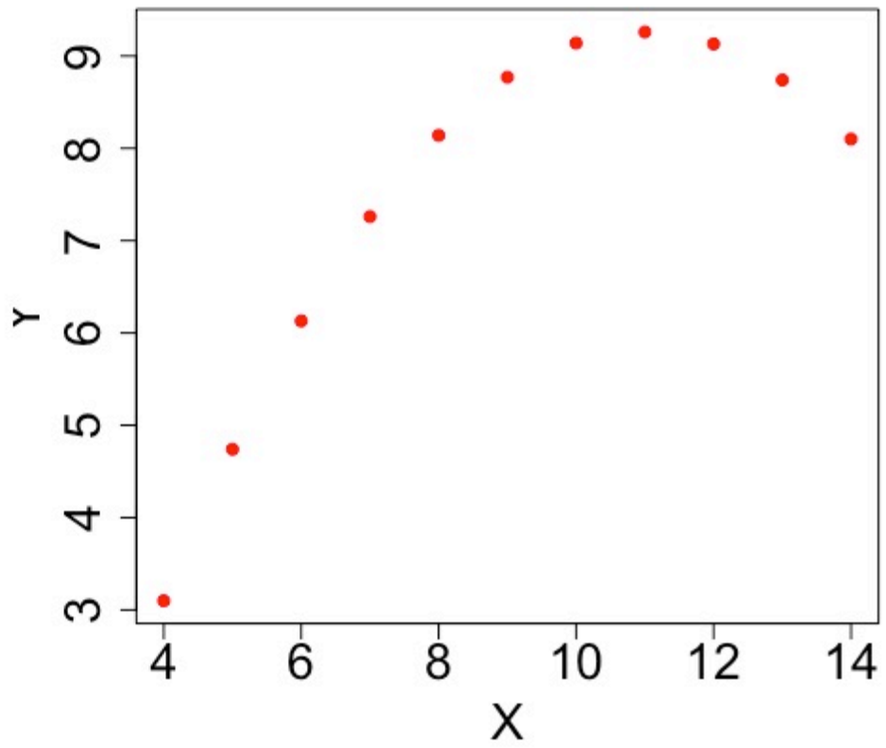
Trained in bioinformatics and biomedical informatics

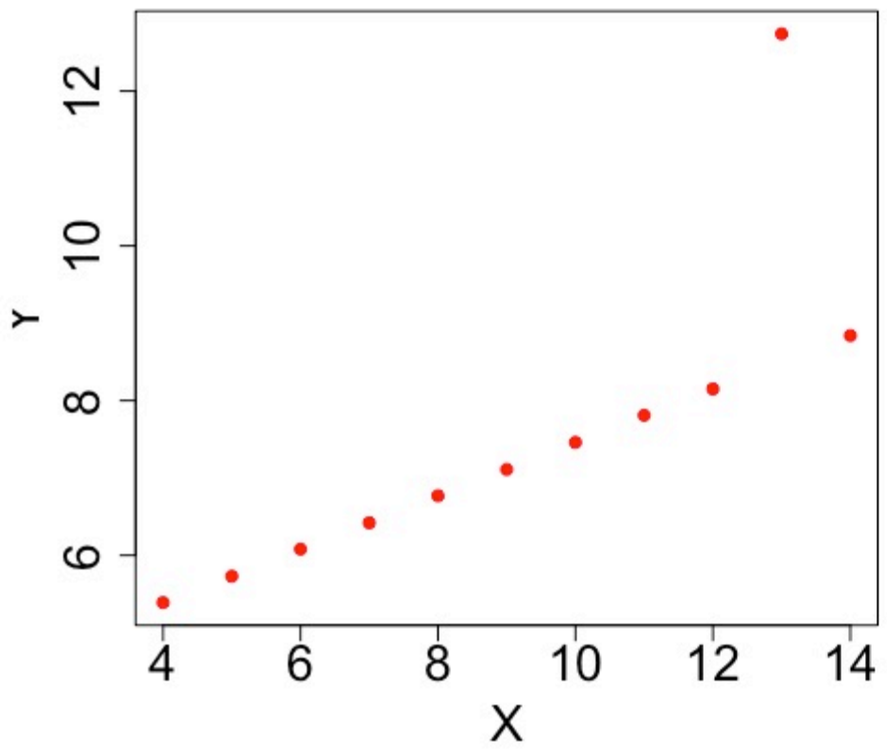
Formerly worked at Affymetrix and Illumina

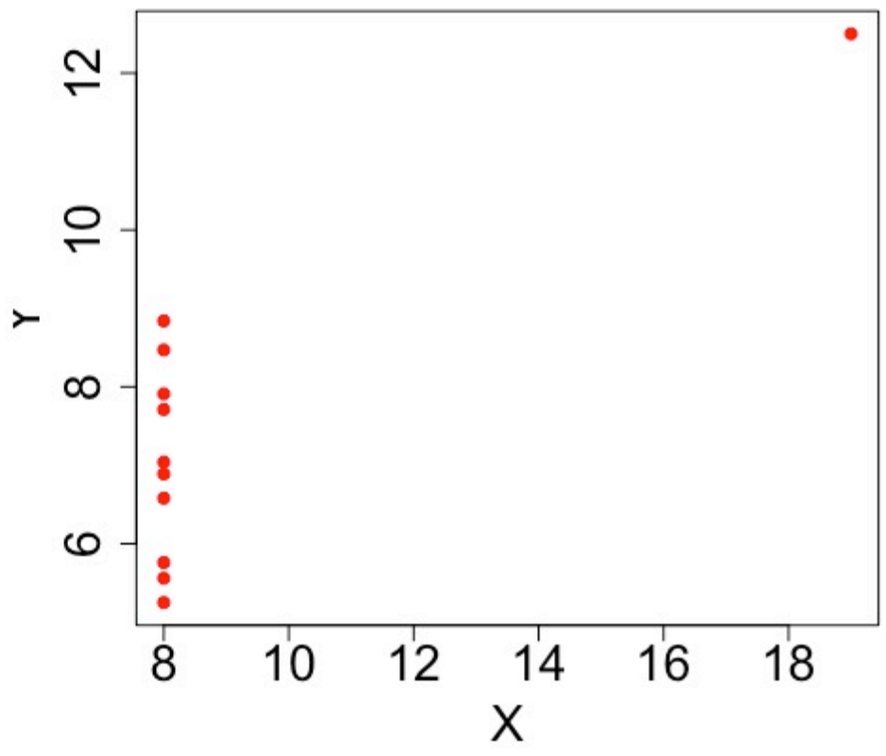
Found that analysis can be limited, we need to make sense of underlying complexity!

Anscombe's Quartet









Network Visualizations are useful

PANDEMIC

INFECTION RATE

2 2 2 3 3

OUTBREAKS

1 2 4 5 6 7

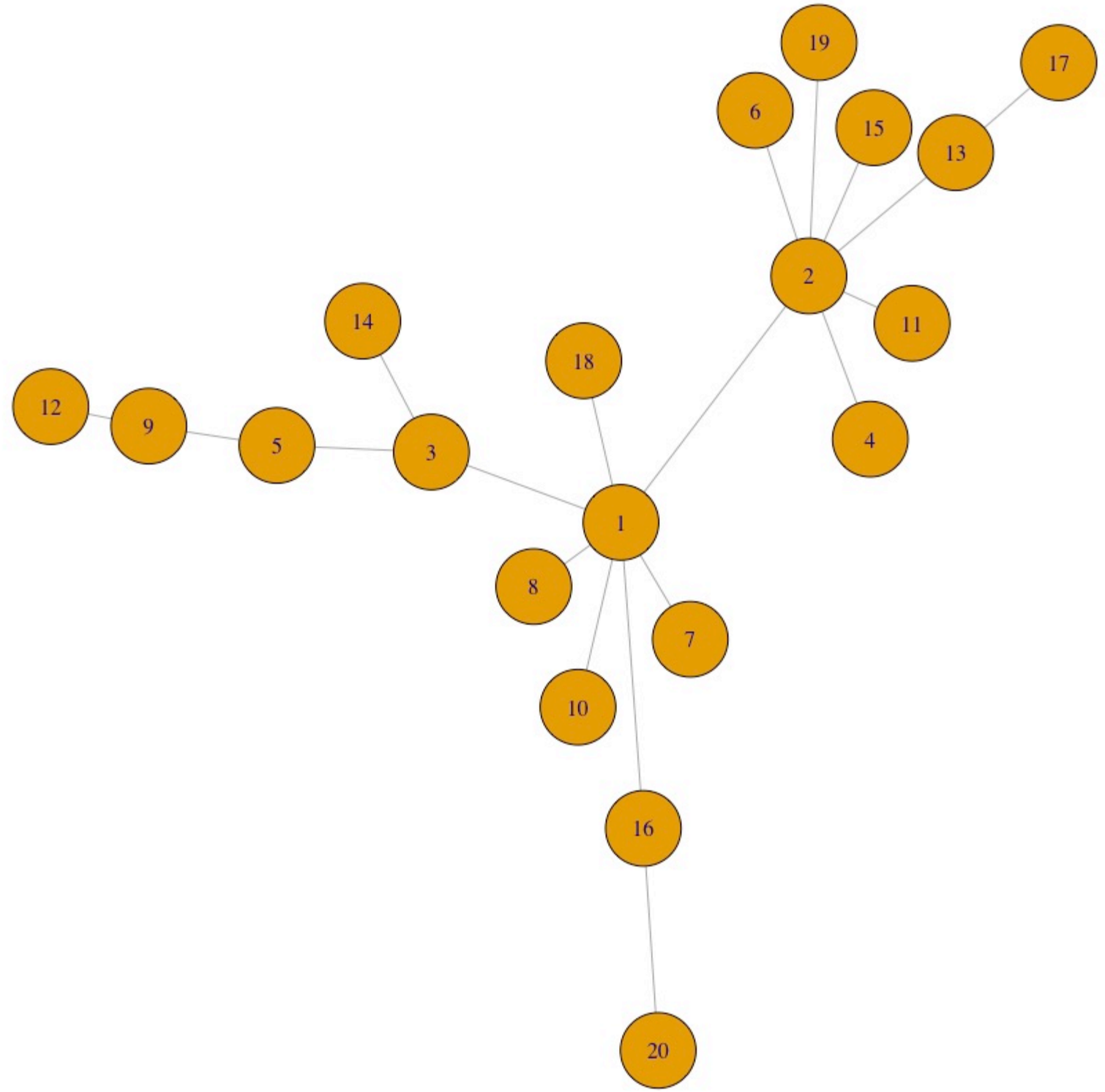
Cures Discovered

Medic
Dispatcher
Operations Expert
Researcher
Scientist

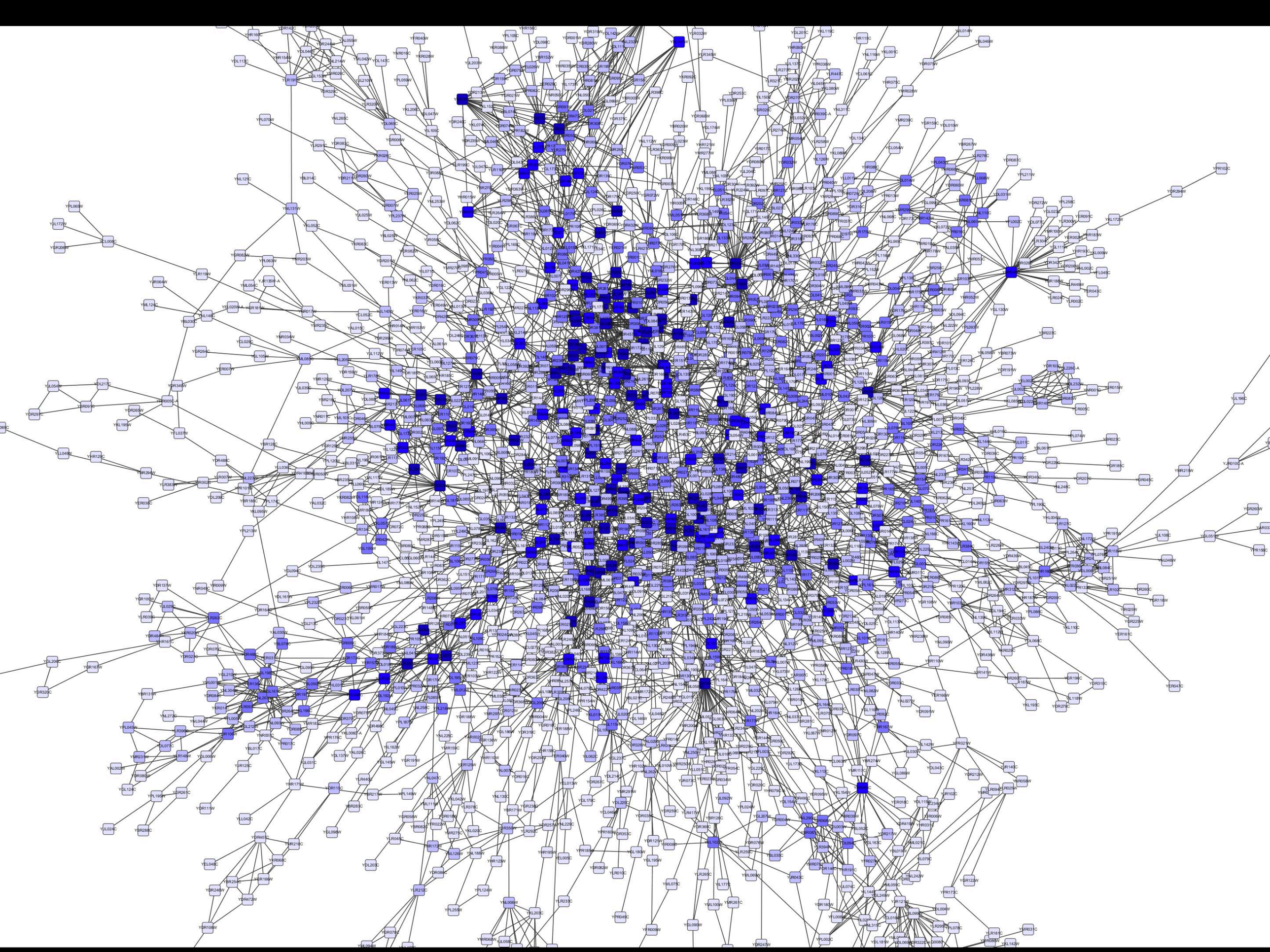
OSAKA
SANTIAGO
KOLKATA
TORONTO

EPIDEMIC!
1. INCREASE
2. INFECT
3. INTENSIFY

Khartoum
Osaka


19
17
6
15
13
2
14
11
18
12
9
5
3
4
1
8
7
10
16
20

But they become unreadable



A portion of my research

- Perception study to evaluate *noticeability* of visual attributes while *visually scanning* a network
- Presented networks to participants, and asked them to click on the most *noticeable* node or edge in rapid succession
- 34 controlled encoding pairs — same data presented via competing visual channels (e.g. size & shape, shape & color, etc)

A predictive model

- Created a random forest model to explain/ predict attention
  - SelectedNode ~ NodeSize + NodeShape + …
  - SelectedEdge ~ EdgeWidth + EdgePattern + …
- Performed reasonably well (AUC of 0.78, 0.86)

So, R package?

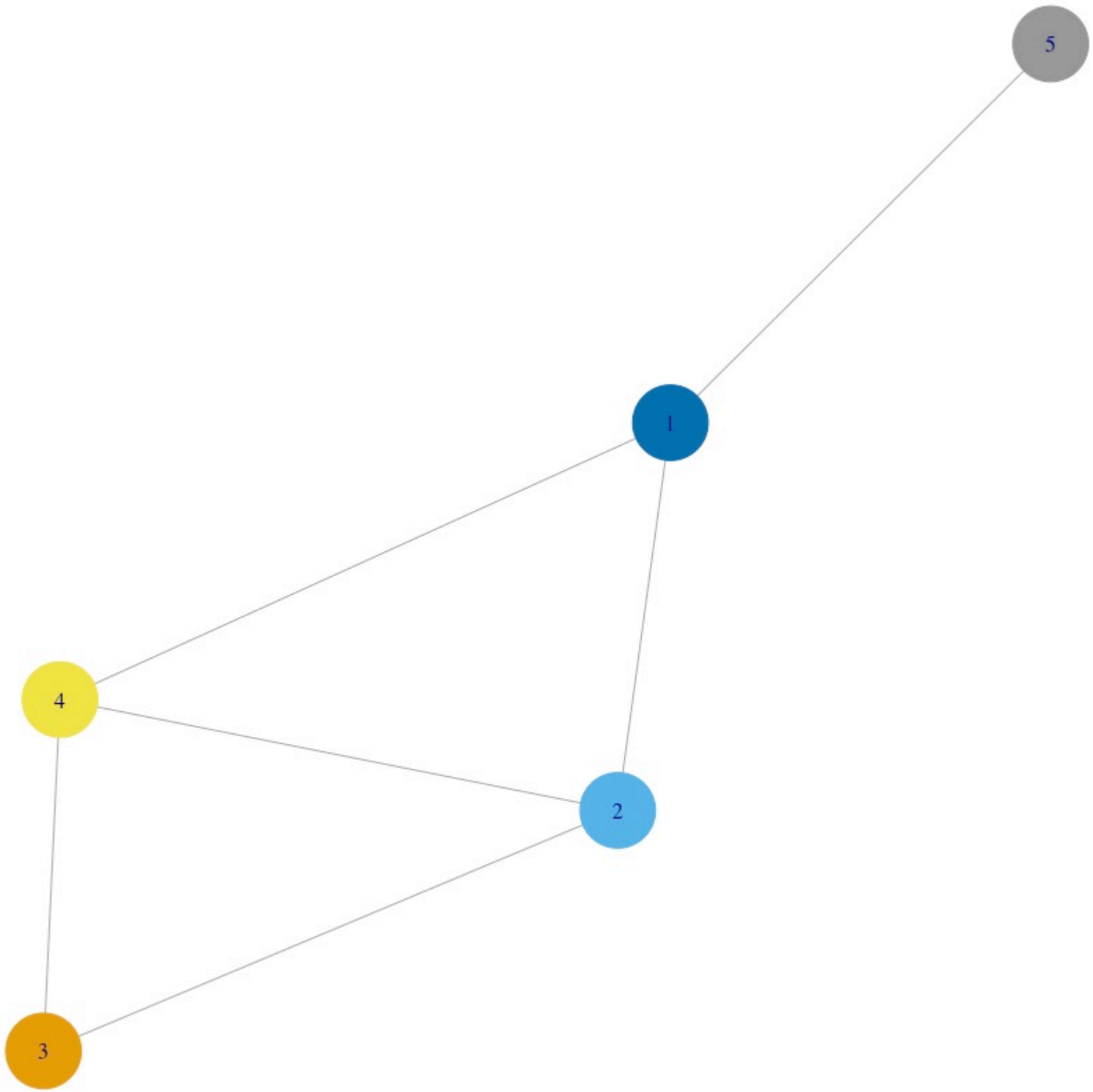
- Put the ML model into an R package to help network visualization designers “see” what a reader sees.
- rDynamo for prioritizing visual encodings (mapping of data attribute to visual attributes)
- rNetVisor for generating an attention heat map

rDynamo Usage

- Handles prioritization of mapping between data attributes and visual attributes
- Uses linear programming to make assignments

> renderGraph(igraphObj, prioritizationMatrix)

	Data Attribute$_1$ (N)	Data Attribute$_2$ (N)	Data Attribute$_3$ (E)	Data Attribute$_4$ (E)
Visual Attribute$_1$	10	4	1	1
Visual Attribute$_2$	3	3	6	2
Visual Attribute$_3$	6	9	3	3
Visual Attribute$_4$	3	3	4	7


5
1
4
2
3

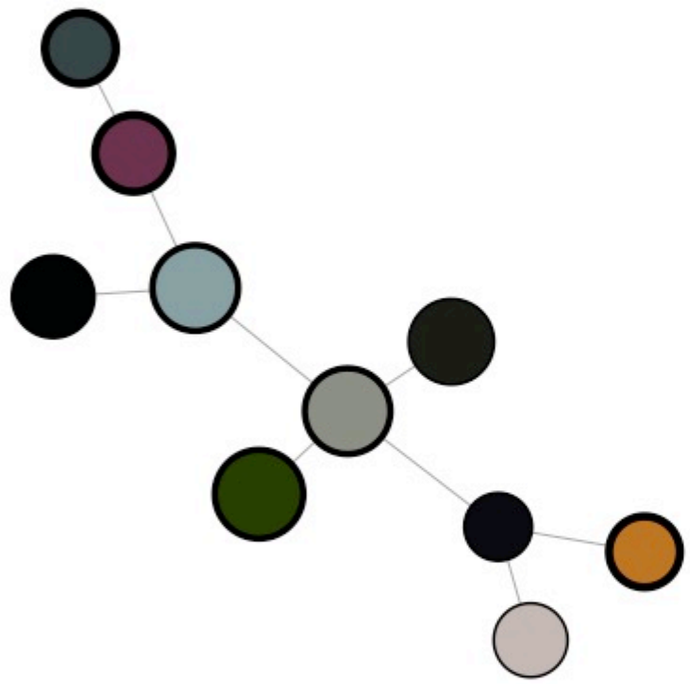
rNetVisor Usage

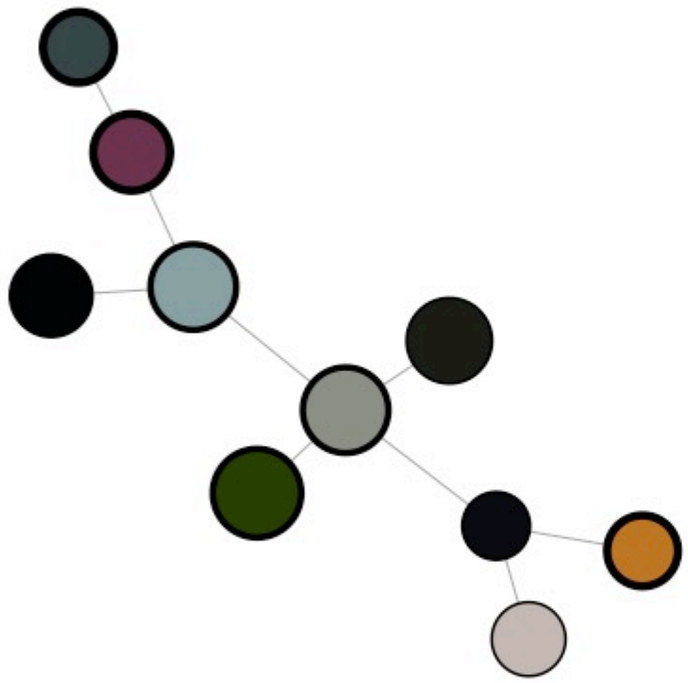
Uses the RF model to draw attention heat map

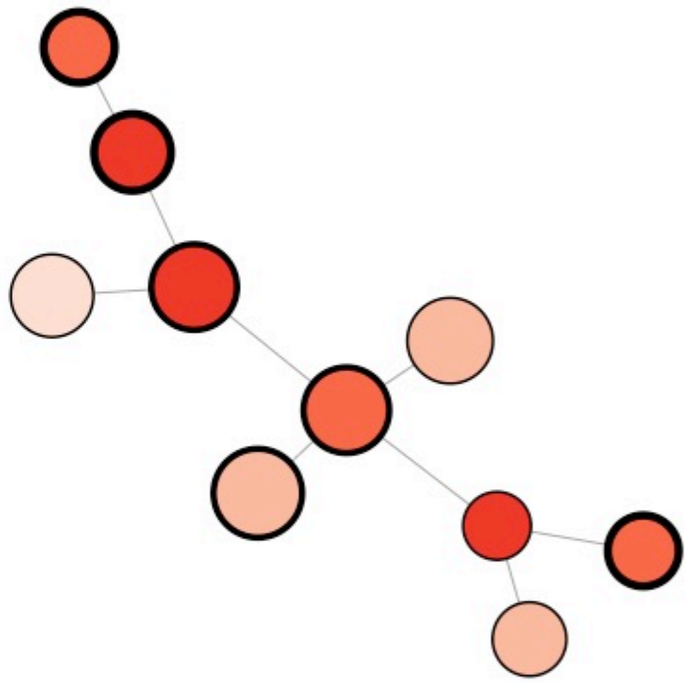
> generateNetValidation(igraphObj, paneled = T)

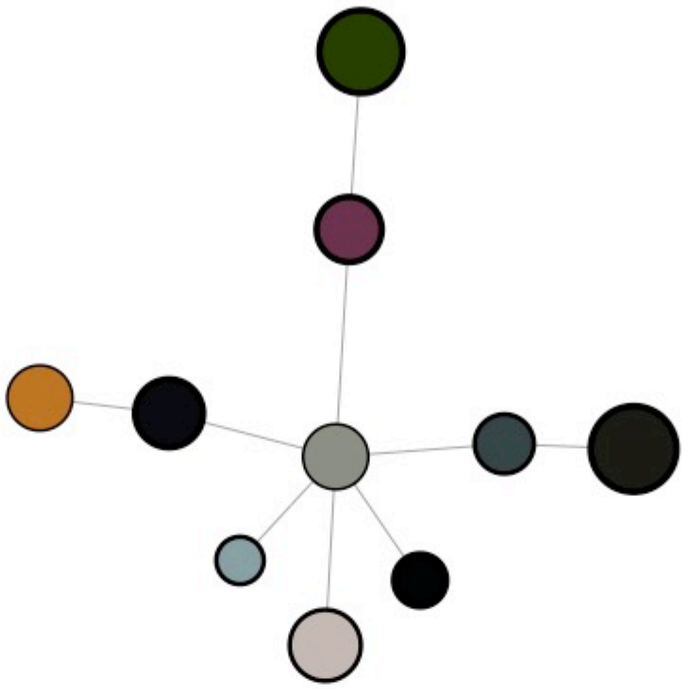
One panel has network, adjacent has attention heat map

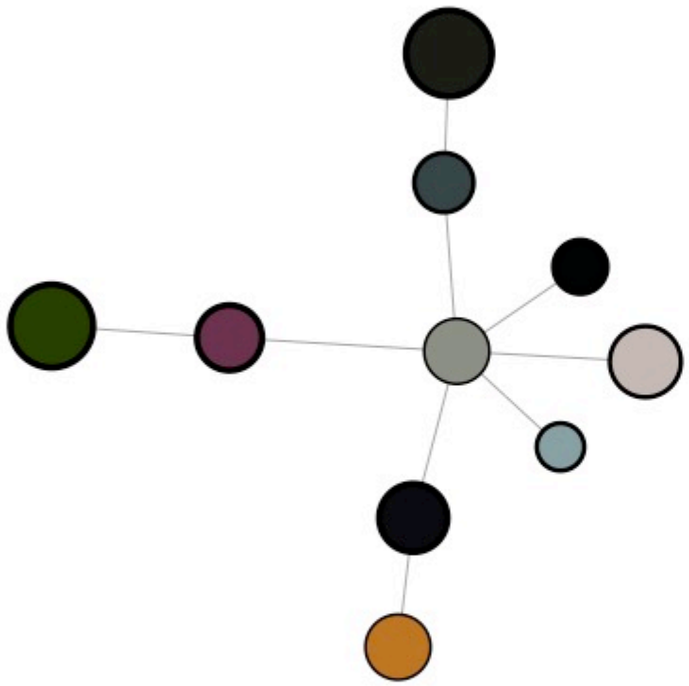
Let’s see it action!

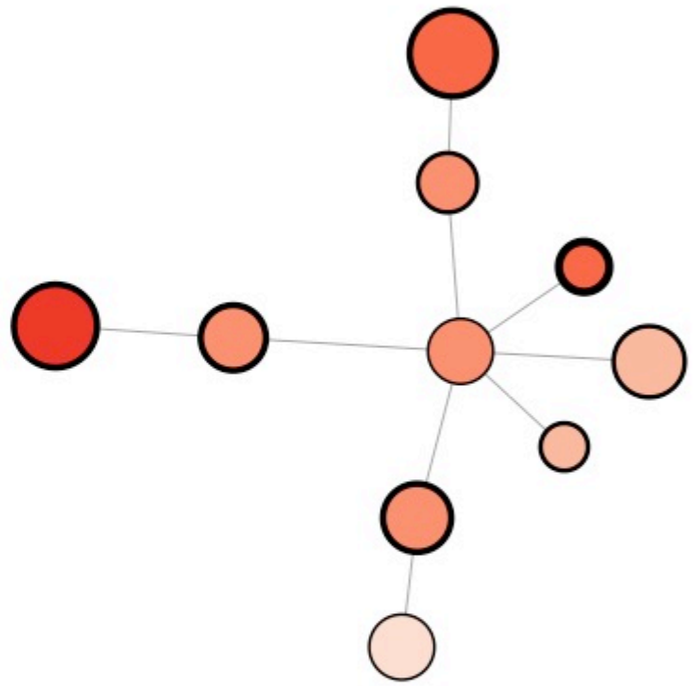


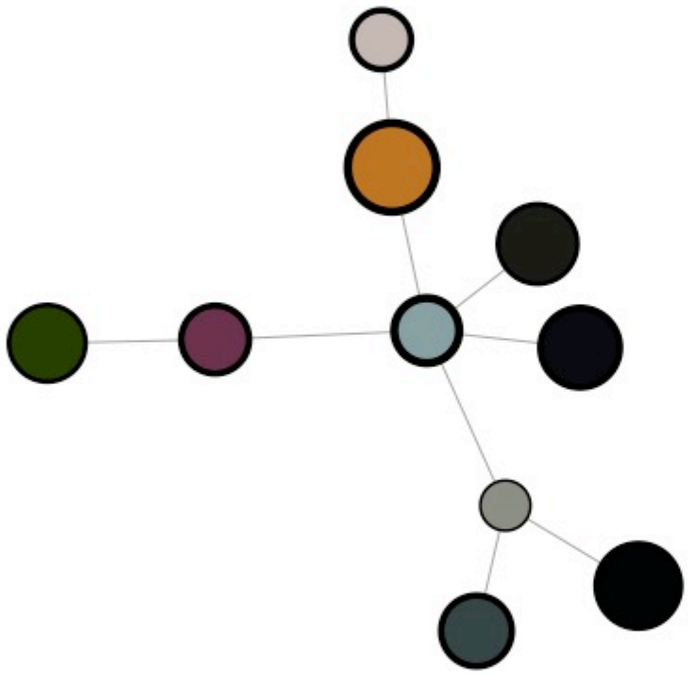


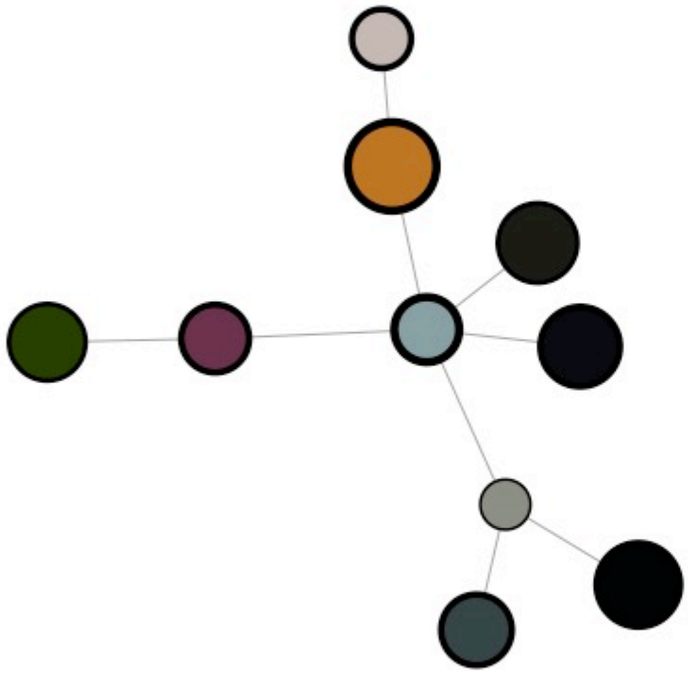


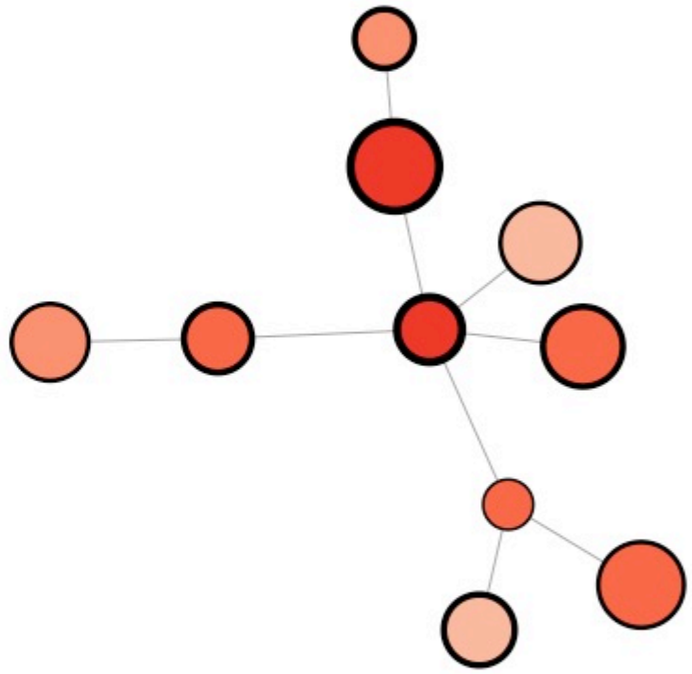


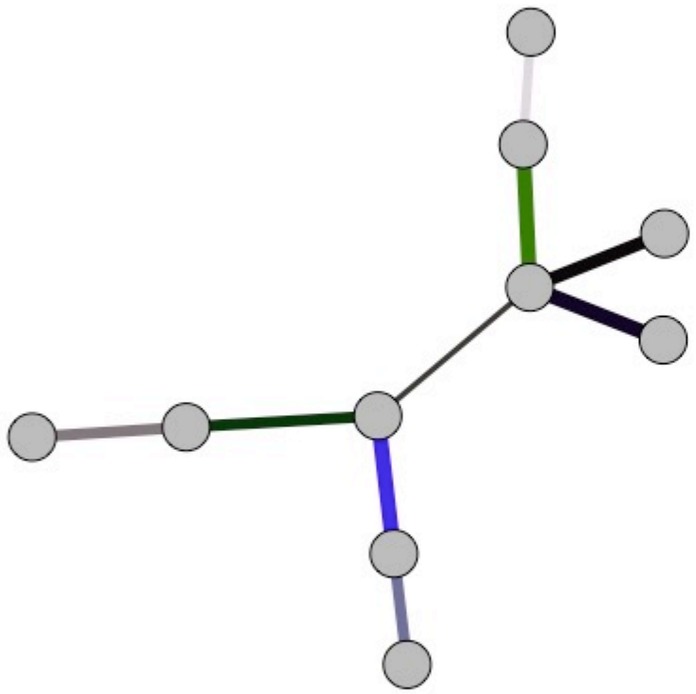


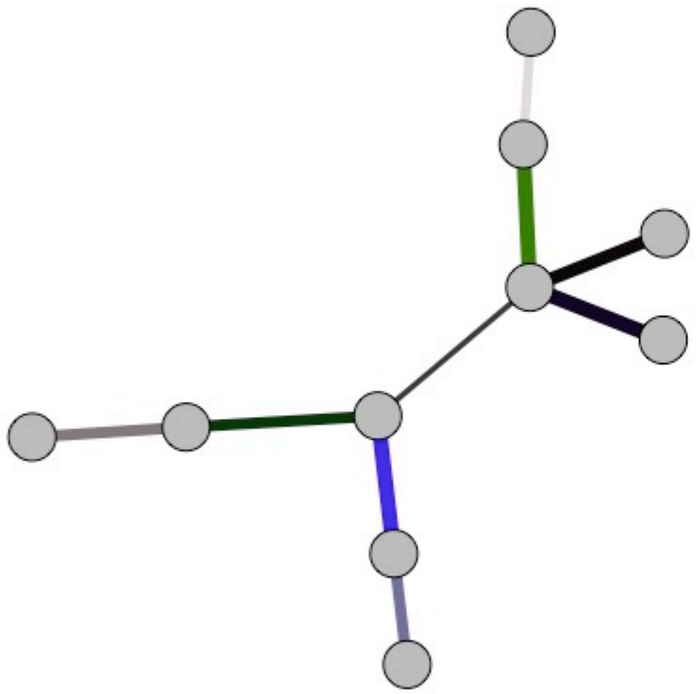


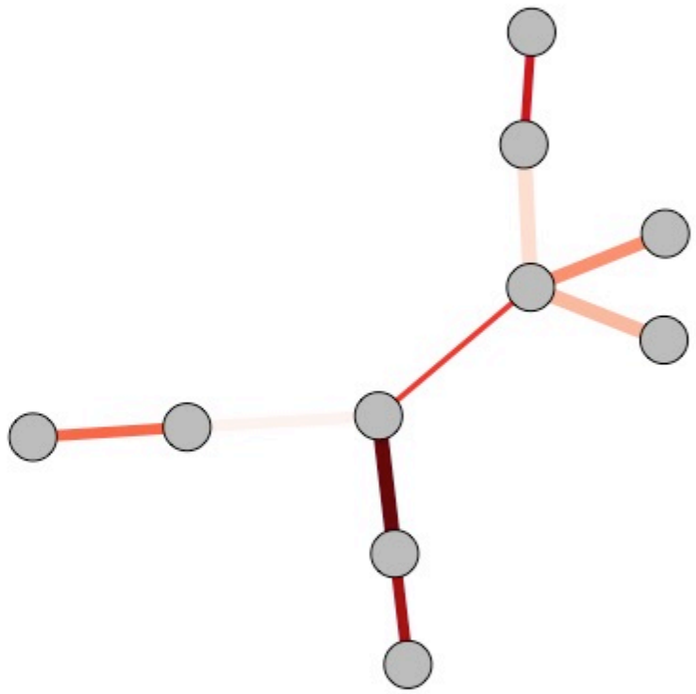


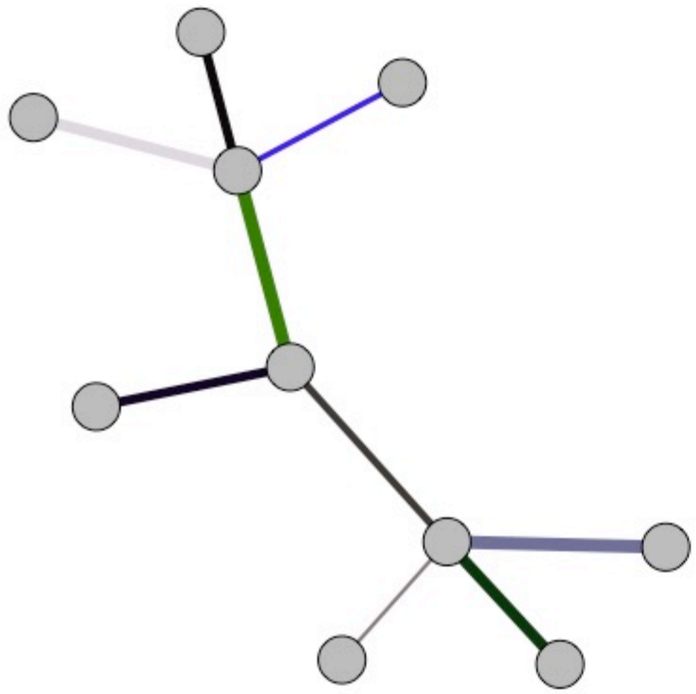


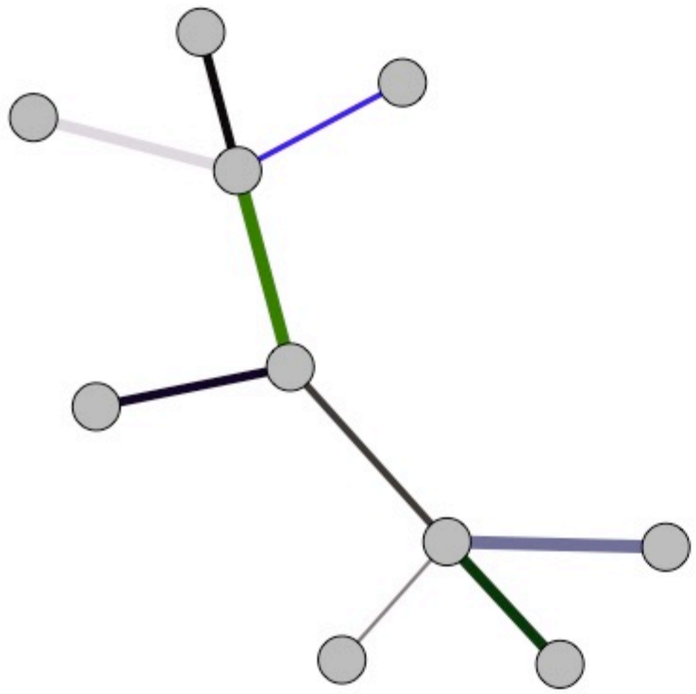


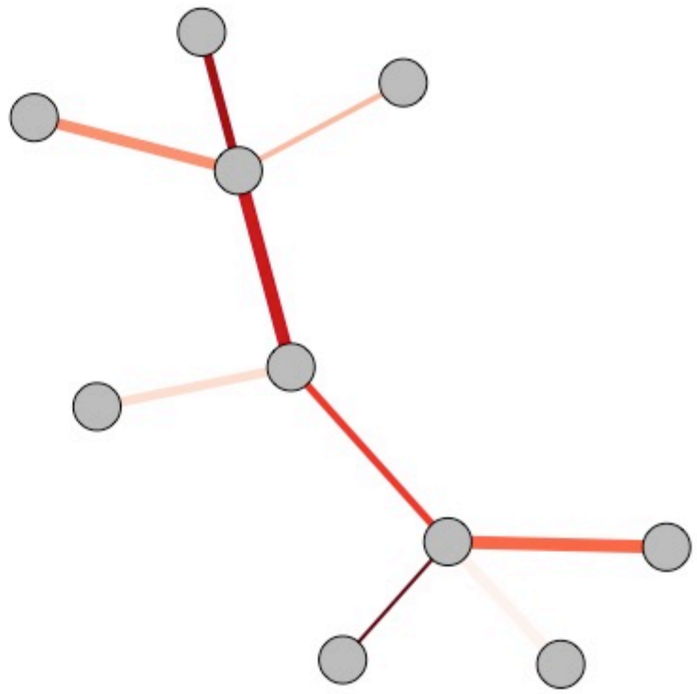


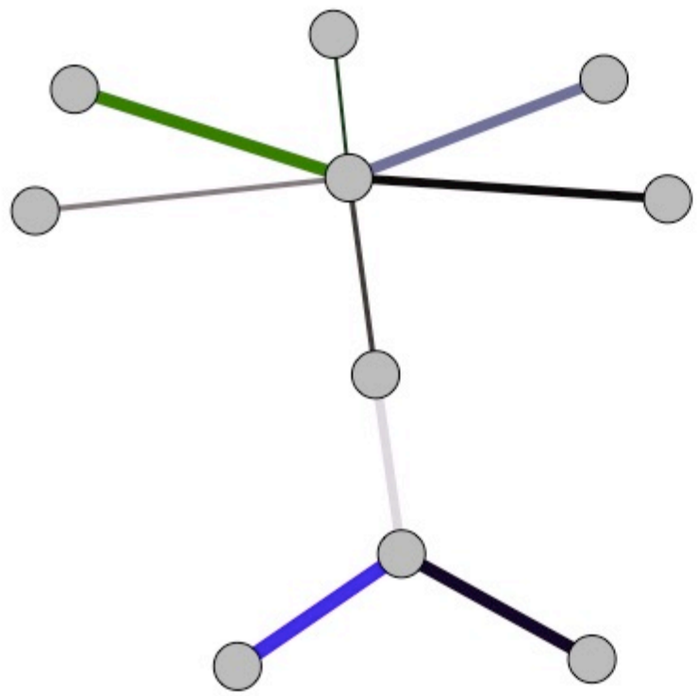


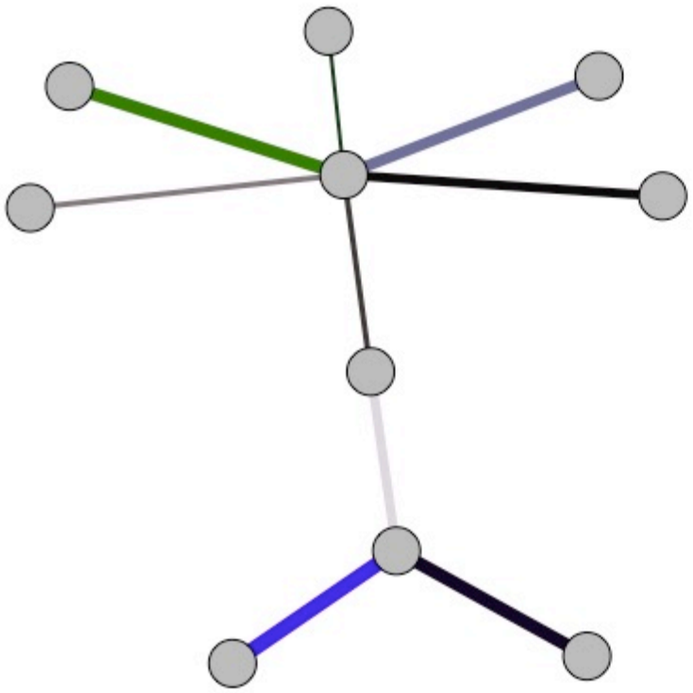


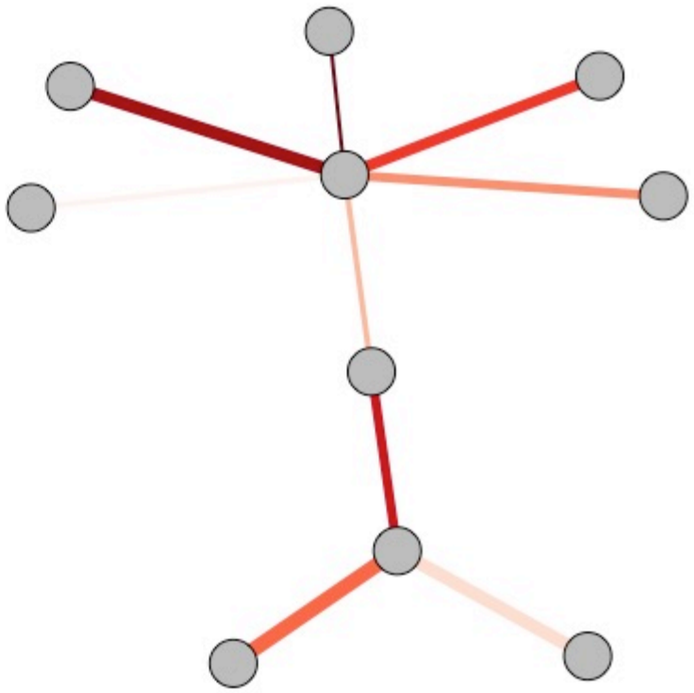












Benefits

- Provides a prediction on how users may read a network
- Reproducible network visualization designs
- Supports transparency of design decisions
- Extensible

Limitations and Future Work

- Designed to be used with iGraph objects
- Works best for numerical data attributes
- Underlying model assumes very small networks
- Plan to run future studies to build models for larger networks and additional tasks
- Post-hoc analysis shows linear models possible

Where to get rNetVisor?

- GitHub link

install_github(“ngopal/rNetVisor”)

install_github(“ngopal/rDynamo”)

Thank You

- Yay! CascadiaRConf!
- I'd love help with this!
  - Fork the repo, help identify/fix bugs
  - Send me an email!
  - Raising funds/resources for future studies
- Thanks to Neil Abernethy, John Gennari, and Jeff Heer

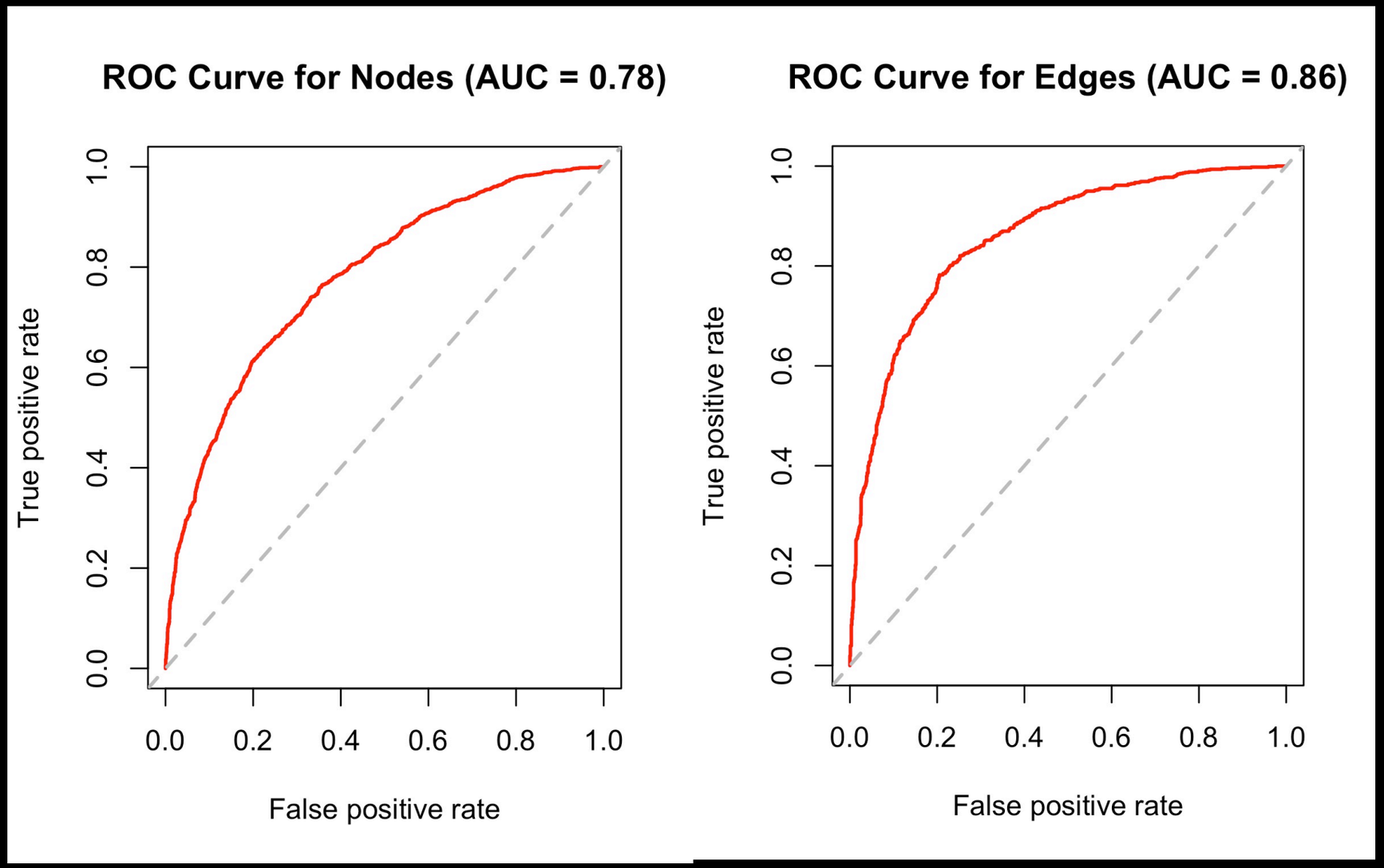
Variable Importances for Nodes Model

1. Network
2. Node Border Size
3. Node Size
4. Node Color Saturation
5. Node Color Value
6. Node Shape
7. Node Color Hue
8. Node Degree

Variable Importances for Edge Model

1. Edge Width
2. Network
3. Edge Color Saturation
4. Edge Color Value
5. Edge Color Hue
6. Edge Pattern

Model	Error	Sensitivity	Specificity	PPV	NPV	AUC
Nodes	0.29	0.61	0.80	0.75	0.67	0.78
Edges	0.21	0.78	0.79	0.79	0.78	0.86



```
# Generate Graph

g = random.graph.game(5, 0.70)

plot(g)

V(g)$variableOne <- sample(10, 5, replace = T)

V(g)$variableTwo <- sample(10, 5, replace = T)


# Specify Encoding Options

encodingOptions = c("vertex.frame.color",
                    "vertex.size",
                    "vertex.color",
                    "vertex.shape")


# Generate a matrix with synthetic matrix values

synMat <- matrix(round(rnorm(length(vertex_attr_names(g))*length(encodingOptions), 10, 2)),
length(vertex_attr_names(g)), length(encodingOptions))

colnames(synMat) <- encodingOptions

rownames(synMat) <- vertex_attr_names(g)
```