2022.12.14
CPSC 547 Information Visualization

BMatrix_Explainer

by Matias I. Bofarull Oddo

Department of Computer Science
The University of British Columbia

Fortran

Paradigm	Multi-paradigm: structured, imperative (procedural, object-oriented), generic, array
Designed by	John Backus
Developer	John Backus and IBM
First appeared	1957; 65 years ago
Stable release	Fortran 2018 (ISO/IEC 1539-1:2018) / 28 November 2018; 3 years ago
Typing discipline	strong, static, manifest
Filename extensions	`.f`, `.for`, `.f90`
Website	fortran-lang.org

Major implementations

Absoft, Cray, GFortran, G95, IBM XL Fortran, Intel, Hitachi, Lahey/Fujitsu, Numerical Algorithms Group, Open Watcom, PathScale, PGI, Silverfrost, Oracle Solaris Studio, others

Influenced by

Speedcoding

Influenced

ALGOL 58, BASIC, C, Chapel,[1] CMS-2, DOPE, Fortress, PL/I, PACT I, MUMPS, IDL, Ratfor

C++

Logo endorsed by the C++ standards committee

Paradigms	Multi-paradigm: procedural, imperative, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC 1 (Joint Technical Committee 1) / SC 22 (Subcommittee 22) / WG 21 (Working Group 21)
First appeared	1985; 37 years ago
Stable release	C++20 (ISO/IEC 14882:2020) / 15 December 2020; 21 months ago
Preview release	C++23 / 17 March 2022; 6 months ago
Typing discipline	Static, nominative, partially inferred
OS	Cross-platform
Filename extensions	.C, .cc, .cpp, .cxx, .c++, .h, .H, .hh, .hpp, .hxx, .h++
Website	isocpp.org

Major implementations

GCC, LLVM Clang, Microsoft Visual C++, Embarcadero C++Builder, Intel C++ Compiler, IBM XL C++, EDG

Influenced by

Ada, ALGOL 68,[1] BCPL,[2] C, CLU,[1] ML, Mesa,[1] Modula-2,[1] Simula, Smalltalk[1]

Influenced

Ada 95, C#,[3] C99, Chapel,[4] Clojure,[5] D, Java,[6] JS++,[7] Lua, Nim,[8] Objective-C++, Perl, PHP, Python,[9] Rust, Seed7

C++ Programming at Wikibooks

MATLAB (programming language)

Paradigm	multi-paradigm: functional, imperative, procedural, object-oriented, array
Designed by	Cleve Moler
Developer	MathWorks
First appeared	late 1970s
Stable release	R2022b[1] / September 15, 2022; 4 days ago
Typing discipline	dynamic, weak
Filename extensions	.m, .p,[2] .mex*,[3] .mat,[4] .fig,[5] .mlx,[6] .mlapp,[7] .mltbx,[8] .mlappinstall,[9] .mlpkginstall[10]
Website	mathworks.com

Influenced by

APL · EISPACK · LINPACK · PL/0 · Speakeasy[11]

Influenced

Julia[12] · Octave[13] · Scilab[14] · INTLAB[15][16][17][18]

MATLAB Programming at Wikibooks

MATLAB (software)

L-shaped membrane logo[19]

Developer(s)	MathWorks
Initial release	1984; 38 years ago
Stable release	R2022b[1] / September 15, 2022; 4 days ago
Written in	C/C++, MATLAB
Operating system	Windows, macOS, and Linux[20][21]
Platform	IA-32, x86-64
Type	Numerical computing
License	Proprietary commercial software
Website	mathworks.com

Python

Paradigm	Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	20 February 1991; 31 years ago[2]
Stable release	3.10.7[3] / 7 September 2022; 12 days ago
Preview release	3.11.0rc2[4] / 12 September 2022; 7 days ago
Typing discipline	Duck, dynamic, strong typing;[5] gradual (since 3.5, but ignored in CPython)[6]
OS	Windows, macOS, Linux/UNIX, Android[7][8] and more[9]
License	Python Software Foundation License
Filename extensions	.py, .pyi, .pyc, .pyd, .pyw, .pyz (since 3.5),[10] .pyo (prior to 3.5)[11]
Website	python.org

Major implementations

CPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, Jython

Dialects

Cython, RPython, Starlark[12]

Influenced by

ABC,[13] Ada,[14] ALGOL 68,[15] APL,[16] C,[17] C++,[18] CLU,[19] Dylan,[20] Haskell,[21][16] Icon,[22] Lisp,[23] Modula-3,[15][18] Perl,[24] Standard ML[16]

Influenced

Apache Groovy, Boo, Cobra, CoffeeScript,[25] D, F#, Genie,[26] Go, JavaScript,[27][28] Julia,[29] Nim, Ring,[30] Ruby,[31] Swift[32]

Python Programming at Wikibooks

julia

Paradigm	Multi-paradigm: multiple dispatch (primary paradigm), procedural, functional, meta, multistaged[1]
Designed by	Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah
Developer	Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and other contributors[2][3]
First appeared	2012; 10 years ago[4]
Stable release	1.8.1[5] / 6 September 2022; 13 days ago and 1.6.7 LTS[8][9] / 19 July 2022; 2 months ago
Preview release	Being worked on: 1.8.2[6] and 1.9.0-DEV with daily updates[7]
Typing discipline	Dynamic,[10] strong,[10] nominative, parametric, optional
Implementation language	Julia, C, C++, Scheme, LLVM[11]
Platform	Tier 1: x86-64, IA-32, CUDA 10.1+[12]/Nvidia GPUs (for Linux and Windows) Tier 2: 64-bit Arm (e.g. Apple M1 Macs, while they also have tier 1 support using Rosetta[13]), 32-bit Windows (64-bit is tier 1) Tier 3: 32-bit Arm, PowerPC, AMD (ROCm) GPUs. Also supports oneAPI/Intel's GPUs and Google's TPUs,[14] and has web browser support (for JavaScript and WebAssembly),[15] and can work in Android. For more details see "supported platforms".
OS	Linux, macOS, Windows and FreeBSD
License	MIT (core),[2] GPL v2;[16][17] a makefile option omits GPL libraries[18]
Filename extensions	.jl
Website	JuliaLang.org

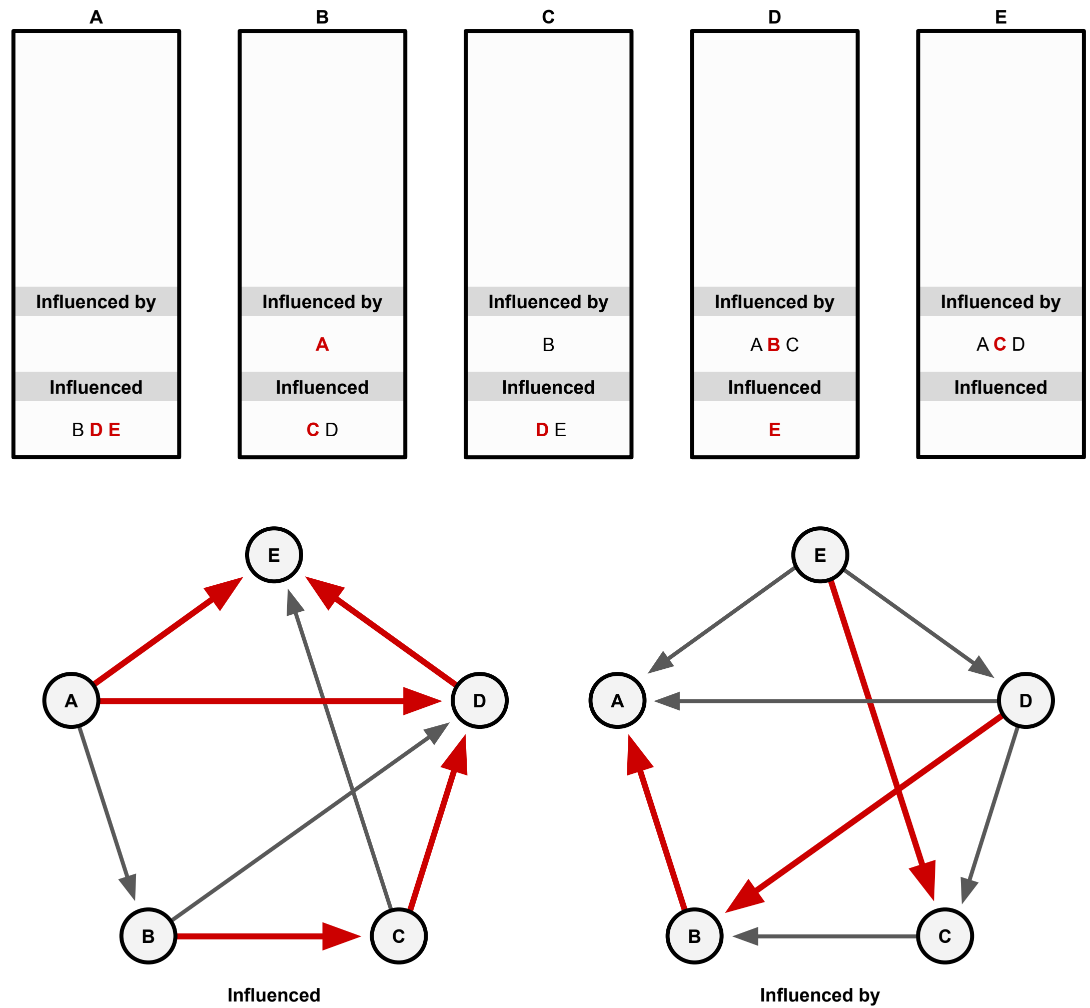
Influenced by

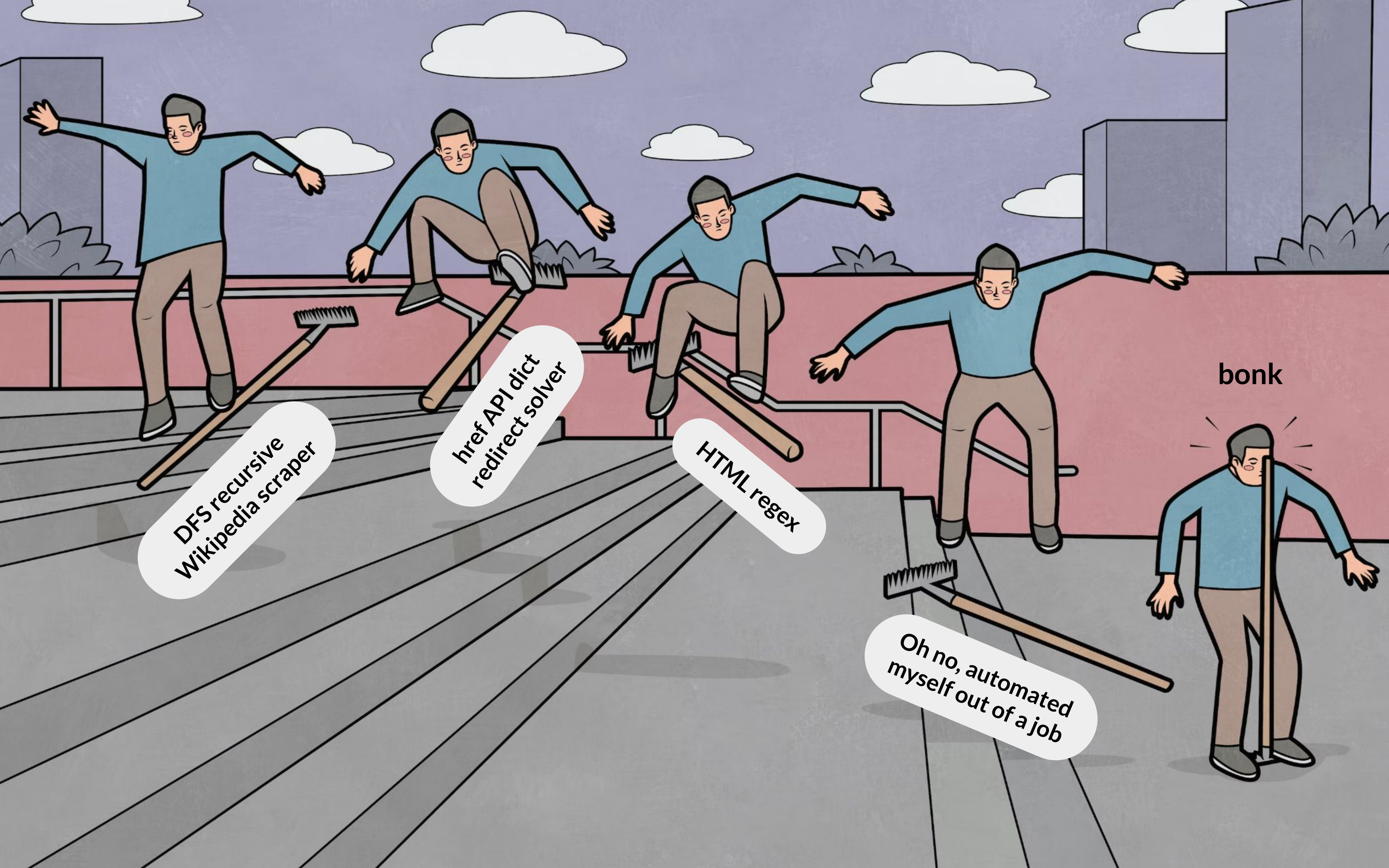
C[4] · Dylan[19] · Lisp[4] · Lua[20] · Mathematica[4] (strictly its Wolfram Language[21][22]) · MATLAB[4] · Perl[20] · Python[20] · R[4] · Ruby[20] · Scheme[23]

Long-story short,
I made a Depth-First Search recursive scraper for Wikimedia API to extract knowledge networks hidden in semantically rich infobox fields.

My goal was to interlink these networks to fill information gaps,
and then create a human-in-the-loop vis tool for Wikipedia editors.

As you can guess,
it didn't go as planned . . .




DFS recursive Wikipedia scraper
href API dict redirect solver
HTML regex
Oh no, automated myself out of a job
bonk

Fortran

From Wikipedia, the free encyclopedia

Fortran (/ˈfɔːrtræn/; formerly **FORTRAN**) is a general-purpose, compiled imperative programming language that is especially suited to numeric computation and scientific computing.

Fortran was originally developed by IBM[2] in the 1950s for scientific and engineering applications, and subsequently came to dominate scientific computing. It has been in use for over six decades in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, geophysics, computational physics, crystallography and computational chemistry. It is a popular language for high-performance computing[3] and is used for programs that benchmark and rank the world's fastest supercomputers.[4][5]

Fortran has had numerous versions, each of which has added extensions while largely retaining compatibility with preceding versions. Successive versions have added support for structured programming and processing of character-based data (FORTRAN 77), array programming, modular programming and generic programming (Fortran 90), High Performance Fortran (Fortran 95), object-oriented programming (Fortran 2003), concurrent programming (Fortran 2008), and native parallel computing capabilities (Coarray Fortran 2008/2018).

Fortran's design was the basis for many other programming languages. Among the better-known is BASIC, which is based on FORTRAN II with a number of syntax cleanups, notably better logical structures,[6] and other changes to work more easily in an interactive environment.[7]

Since August 2021 Fortran has ranked among the top 15 languages in the TIOBE index, a measure of the popularity of programming languages.[8]

Fortran

Fortran logo.svg

Paradigm	Multi-paradigm: structured, imperative (procedural, object-oriented), generic, array
Designed by	John Backus
Developer	John Backus and IBM
First appeared	1957; 65 years ago
Stable release	Fortran 2018 (ISO/IEC 1539-1:2018) / 28 November 2018; 4 years ago
Typing discipline	strong, static, manifest
Filename extensions	`.f`, `.for`, `.f90`
Website	fortran-lang.org

Major implementations

Absoft, Cray, GFortran, G95, IBM XL Fortran, Intel, Hitachi, Lahey/Fujitsu, Numerical Algorithms Group, Open Watcom, PathScale, PGI, Silverfrost, Oracle Solaris Studio, others

Influenced by

Speedcoding

Influenced

ALGOL 58, BASIC, C, Chapel,[1] CMS-2, DOPE, Fortress, PL/I, PACT I, MUMPS, IDL

Contents [hide]

The IBM Blue Gene/P supercomputer

Fortran

From Wikipedia, the free encyclopedia

Fortran (/ˈfɔːrtræn/; formerly **FORTRAN**) is a general-purpose, compiled imperative programming language that is especially suited to numeric computation and scientific computing.

Fortran was originally developed by IBM[2] in the 1950s for scientific and engineering applications, and subsequently came to dominate scientific computing. It has been in use for over six decades in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, geophysics, computational physics, crystallography and computational chemistry. It is a popular language for high-performance computing[3] and is used for programs that benchmark and rank the world's fastest supercomputers.[4][5]

Fortran has had numerous versions, each of which has added extensions while largely retaining compatibility with preceding versions. Successive versions have added support for structured programming and processing of character-based data (FORTRAN 77), array programming, modular programming and generic programming (Fortran 90), High Performance Fortran (Fortran 95), object-oriented programming (Fortran 2003), concurrent programming (Fortran 2008), and native parallel computing capabilities (Coarray Fortran 2008/2018).

Fortran's design was the basis for many other programming languages. Among the better-known is BASIC, which is based on FORTRAN II with a number of syntax cleanups, notably better logical structures,[6] and other changes to work more easily in an interactive environment.[7]

Since August 2021 Fortran has ranked among the top 15 languages in the TIOBE index, a measure of the popularity of programming languages.[8]

Contents [hide]

Fortran

Fortran logo.svg

Paradigm	Multi-paradigm: structured, imperative (procedural, object-oriented), generic, array
Designed by	John Backus
Developer	John Backus and IBM
First appeared	1957; 65 years ago
Stable release	Fortran 2018 (ISO/IEC 1539-1:2018) / 28 November 2018; 4 years ago
Typing discipline	strong, static, manifest
Filename extensions	`.f`, `.for`, `.f90`
Website	fortran-lang.org

Major implementations

Absoft, Cray, GFortran, G95, IBM XL Fortran, Intel, Hitachi, Lahey/Fujitsu, Numerical Algorithms Group, Open Watcom, PathScale, PGI, Silverfrost, Oracle Solaris Studio, others

Influenced by

Speedcoding, Modula-2

Influenced

ALGOL 58, BASIC, C, Chapel,[1] CMS-2, DOPE, Fortress, PL/I, PACT I, MUMPS, IDL, Ratfor, Coral, Dartmouth BASIC, SISAL, S, Verilog

Fortran

From Wikipedia, the free encyclopedia

Fortran (/ˈfɔːrtræn/; formerly **FORTRAN**) is a general-purpose, compiled imperative programming language that is especially suited to numeric computation and scientific computing.

Fortran was originally developed by IBM[2] in the 1950s for scientific and engineering applications, and subsequently came to dominate scientific computing. It has been in use for over six decades in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, geophysics, computational physics, crystallography and computational chemistry. It is a popular language for high-performance computing[3] and is used for programs that benchmark and rank the world's fastest supercomputers.[4][5]

Fortran has had numerous versions, each of which has added extensions while largely retaining compatibility with preceding versions. Successive versions have added support for structured programming and processing of character-based data (FORTRAN 77), array programming, modular programming and generic programming (Fortran 90), High Performance Fortran (Fortran 95), object-oriented programming (Fortran 2003), concurrent programming (Fortran 2008), and native parallel computing capabilities (Coarray Fortran 2008/2018).

Fortran's design was the basis for many other programming languages. Among the better-known is BASIC, which is based on FORTRAN II with a number of syntax cleanups, notably better logical structures,[6] and other changes to work more easily in an interactive environment.[7]

Since August 2021 Fortran has ranked among the top 15 languages in the TIOBE index, a measure of the popularity of programming languages.[8]

Contents [hide]

LESSON
Software that gap-fills Wikipedia is NOT an InfoVis project

Fortran	
Fortran logo.svg	
Paradigm	Multi-paradigm: structured, imperative (procedural, object-oriented), generic, array
Designed by	John Backus
Developer	John Backus and IBM
First appeared	1957; 65 years ago
Stable release	Fortran 2018 (ISO/IEC 1539-1:2018) / 28 November 2018; 4 years ago
Typing discipline	strong, static, manifest
Filename extensions	`.f`, `.for`, `.f90`
Website	fortran-lang.org
Major implementations	
Absoft, Cray, GFortran, G95, IBM XL Fortran, Intel, Hitachi, Lahey/Fujitsu, Numerical Algorithms Group, Open Watcom, PathScale, PGI, Silverfrost, Oracle Solaris Studio, others	
Influenced by	
Speedcoding, Modula-2	
Influenced	
ALGOL 58, BASIC, C, Chapel,[1] CMS-2, DOPE, Fortress, PL/I, PACT I, MUMPS, IDL, Ratfor, Coral, Dartmouth BASIC, SISAL, S, Verilog	

infobox_scraper.py — infobox_interlinker

directed_outgoing_Fortran.tsv | directed_outgoing_Carl_Jung.tsv | run_scraper.py | infobox_scraper.py

Fortran

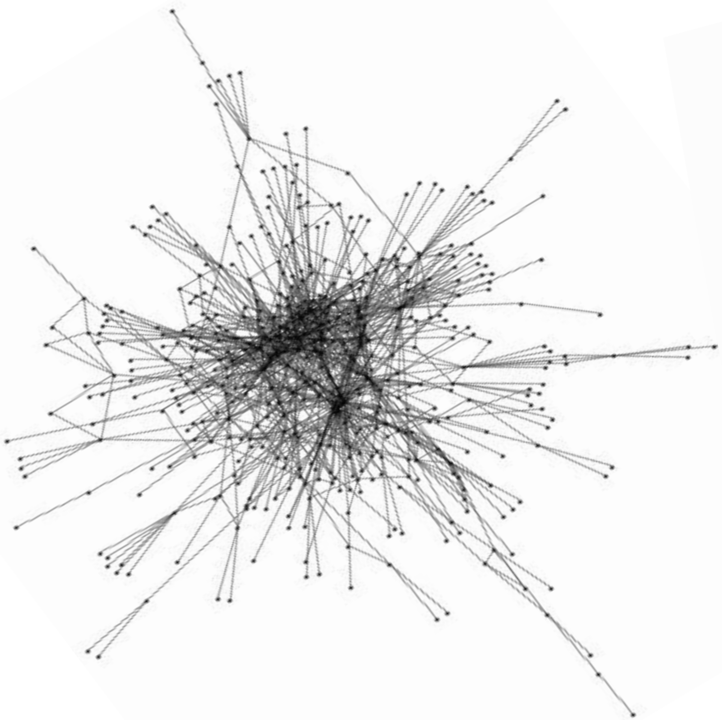
```
F_Sharp_(programming_language)	C_Sharp_(programming_language)
F_Sharp_(programming_language)	Elm_(programming_language)
F_Sharp_(programming_language)	F*_(programming_language)
F_Sharp_(programming_language)	LiveScript_(programming_language)
Forth_(programming_language)	Rebol
Forth_(programming_language)	RPL_(programming_language)
Forth_(programming_language)	Factor_(programming_language)
Forth_(programming_language)	Joy_(programming_language)
Forth_(programming_language)	PostScript
Fortran	IDL_(programming_language)
Fortran	Ratfor
Fortran	PL/I
Fortran	MUMPS
Fortran	Fortress_(programming_language)
Fortran	PACT_I
Fortran	Chapel_(programming_language)
Fortran	C_(programming_language)
Fortran	CMS-2
Fortran	DOPE_(Dartmouth_Oversimplified_Programming_Experiment)
Fortran	BASIC
Fortran	ALGOL_58
GW-BASIC	QBasic
GW-BASIC	QuickBASIC
GW-BASIC	MSX_BASIC
Go_(programming_language)	Crystal_(programming_language)
Go_(programming_language)	Zig_(programming_language)
Gosu_(programming_language)	Kotlin_(programming_language)
Haskell	Mercury_(programming_language)
Haskell	Omega
Haskell	Isabelle_(proof_assistant)
Haskell	Language_Integrated_Query
Haskell	LiveScript_(programming_language)
Haskell	PureScript
Haskell	Visual_Basic_.NET
Haskell	Raku_(programming_language)
Haskell	Rust_(programming_language)
Haskell	Scala_(programming_language)
Haskell	Swift_(programming_language)
Haskell	Wikipedia:Citation_needed
Haskell	Idris_(programming_language)
Haskell	Python_(programming_language)
Haskell	Hack_(programming_language)
Haskell	Java_(programming_language)
Haskell	F_Sharp_(programming_language)
Haskell	Generics_in_Java
Haskell	Agda_(programming_language)
Haskell	Bluespec
Haskell	C_Sharp_(programming_language)
Haskell	Cayenne_(programming_language)
Haskell	Clean_(programming_language)
Haskell	C++11
Haskell	CoffeeScript
Haskell	Concepts_(C++)
Haskell	Curry_(programming_language)
Haskell	Elm_(programming_language)
Haskell	Epigram_(programming_language)
Haskell	Escher_(programming_language)
Haskell	Clojure
HyperTalk	SenseTalk
HyperTalk	ActionScript
HyperTalk	AppleScript
HyperTalk	ECMAScript
HyperTalk	JavaScript
HyperTalk	Lingo_(programming_language)
HyperTalk	LiveCode
HyperTalk	SuperTalk
IBM_BASIC	GW-BASIC
ISWIM	Clean_(programming_language)
```

Carl Jung

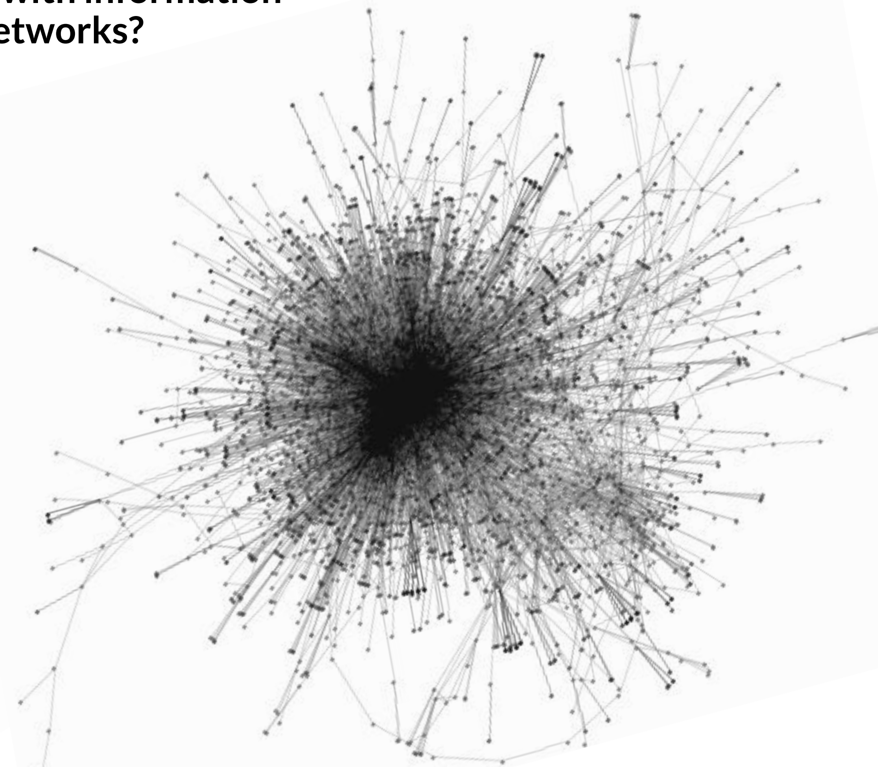
```
Calvin_Seerveld	James_K._A._Smith
Carl_Friedrich_Gauss	Ferdinand_Minding
Carl_Gustav_Hempel	Jaegwon_Kim
Carl_Gustav_Hempel	John_Earman
Carl_Gustav_Hempel	Lawrence_Sklar
Carl_Gustav_Hempel	Nelson_Goodman
Carl_Gustav_Hempel	Peter_Achinstein
Carl_Gustav_Hempel	Philip_Kitcher
Carl_Gustav_Hempel	Richard_Jeffrey
Carl_Gustav_Hempel	Robert_Nozick
Carl_Gustav_Hempel	Wesley_C._Salmon
Carl_Jung	Robert_Anton_Wilson
Carl_Jung	Ross_Nichols
Carl_Jung	Sigmund_Freud
Carl_Jung	Philip_K._Dick
Carl_Jung	Terence_McKenna
Carl_Jung	Wolfgang_Pauli
Carl_Jung	Sonu_Shamdasani
Carl_Jung	Joseph_Campbell
Carl_Jung	Jean_Piaget
Carl_Jung	Jordan_Peterson
Carl_Jung	James_Hillman
Carl_Jung	Hermann_Hesse
Carl_Jung	Gaston_Bachelard
Carl_Jung	Erich_Neumann_(psychologist)
Carl_Jung	Carl_Rogers
Carl_Jung	Arnold_J._Toynbee
Carl_Jung	Alan_Watts
Carl_Schmitt	Jacob_Taubes
Carl_Schmitt	Jacques_Derrida
Carl_Schmitt	Jaime_Guzmán
Carl_Schmitt	Jiang_Shigong
Carl_Schmitt	Jürgen_Habermas
Carl_Schmitt	Liu_Xiaofeng_(academic)
Carl_Schmitt	Mario_Tronti
Carl_Schmitt	Slavoj_Žižek
Carl_Schmitt	Paul_Gottfried
Carl_Schmitt	Reinhart_Koselleck
Carl_Schmitt	Herfried_Münkler
Carl_Schmitt	Susan_Buck-Morss
Carl_Schmitt	Waldemar_Gurian
Carl_Schmitt	Walter_Benjamin
Carl_Schmitt	Wang_Shaoguang
Carl_Schmitt	Mark_Lilla
Carl_Schmitt	Hans_Morgenthau
Carl_Schmitt	Leo_Strauss
Carl_Schmitt	Giorgio_Agamben
Carl_Schmitt	Adam_Wielomski
Carl_Schmitt	Adrian_Vermeule
Carl_Schmitt	Alain_de_Benoist
Carl_Schmitt	Aleksandr_Dugin
Carl_Schmitt	Andrew_Arato
Carl_Schmitt	Antonio_Negri
Carl_Schmitt	Carlo_Galli_(political_scientist)
Carl_Schmitt	Chantal_Mouffe
Carl_Schmitt	Carlo_Lottieri
Carl_Schmitt	Copenhagen_School_(international_relations)
Carl_Schmitt	Curtis_Yarvin
Carl_Schmitt	Ernst_Jünger
Carl_Schmitt	Francis_Parker_Yockey
Carl_Schmitt	Friedrich_Hayek
Carl_Schmitt	Gianfranco_Miglio
Carl_Schmitt	Christopher_Ferrara
Carl_Schmitt	Hannah_Arendt
Carl_Stumpf	Max_Scheler
Carl_Stumpf	Wolfgang_Köhler
Carl_Stumpf	Robert_Musil
Carl_Stumpf	Kurt_Koffka
```

```python

# by Matias I. Bofarull Oddo - 2022.10.30

import re
import requests
from bs4 import BeautifulSoup
from urllib.parse import quote_plus, unquote_plus

rest_API = "https://en.wikipedia.org/api/rest_v1/page/html/"
param_API = "?redirect=true&stash=false"
url_regex = re.compile(r"\/page\/html\/(.*?)[\?|^]")
influenced_by_regex = re.compile(r"Influenced\s*(?:\s*<.*?>\s*){0,2}\s*by")

def scrape_programming_languages(root_href):
    dict_wikigraph = {}
    def wikiscrape_infobox(page_href):
        try:
            if page_href not in dict_wikigraph:
                dict_wikigraph[page_href] = {}
            sesh = requests.Session()
            page = sesh.get(
                rest_API + quote_plus(page_href) + param_API,
                timeout=100,
            )
            url_match = url_regex.search(page.url)
            true_href = unquote_plus(url_match.group(1))

            soup = BeautifulSoup(page.content, "html.parser")
            infobox_HTML = soup.find("table", {"class": "infobox"})
            infobox_rows = [row.prettify() for row in infobox_HTML.find_all("tr")]
            row_index = 0
            data_strings = {
                "incoming": "",
                "outgoing": "",
            }
            for row in infobox_rows:
                if influenced_by_regex.search(row) or "Influences" in row:
                    data_strings["incoming"] += infobox_rows[row_index]
                    data_strings["incoming"] += infobox_rows[row_index + 1]
                elif "Influenced" in row:
                    data_strings["outgoing"] += infobox_rows[row_index]
                    data_strings["outgoing"] += infobox_rows[row_index + 1]
                row_index += 1
            data_incoming = BeautifulSoup(
                data_strings["incoming"],
                "html.parser",
            )
            list_incoming = [
                str(a["href"])[2:]
                for a in data_incoming.find_all(
                    "a",
                    {"rel": True},
                )
            ]
            data_outgoing = BeautifulSoup(
                data_strings["outgoing"],
                "html.parser",
            )
            list_outgoing = [
                str(a["href"])[2:]
                for a in data_outgoing.find_all(
                    "a",
                    {"rel": True},
                )
            ]
            if page_href in dict_wikigraph:
                dict_wikigraph[page_href]["incoming"] = list_incoming
```

What are other ways to understand and better work with information dense networks?



Fortran



Carl Jung

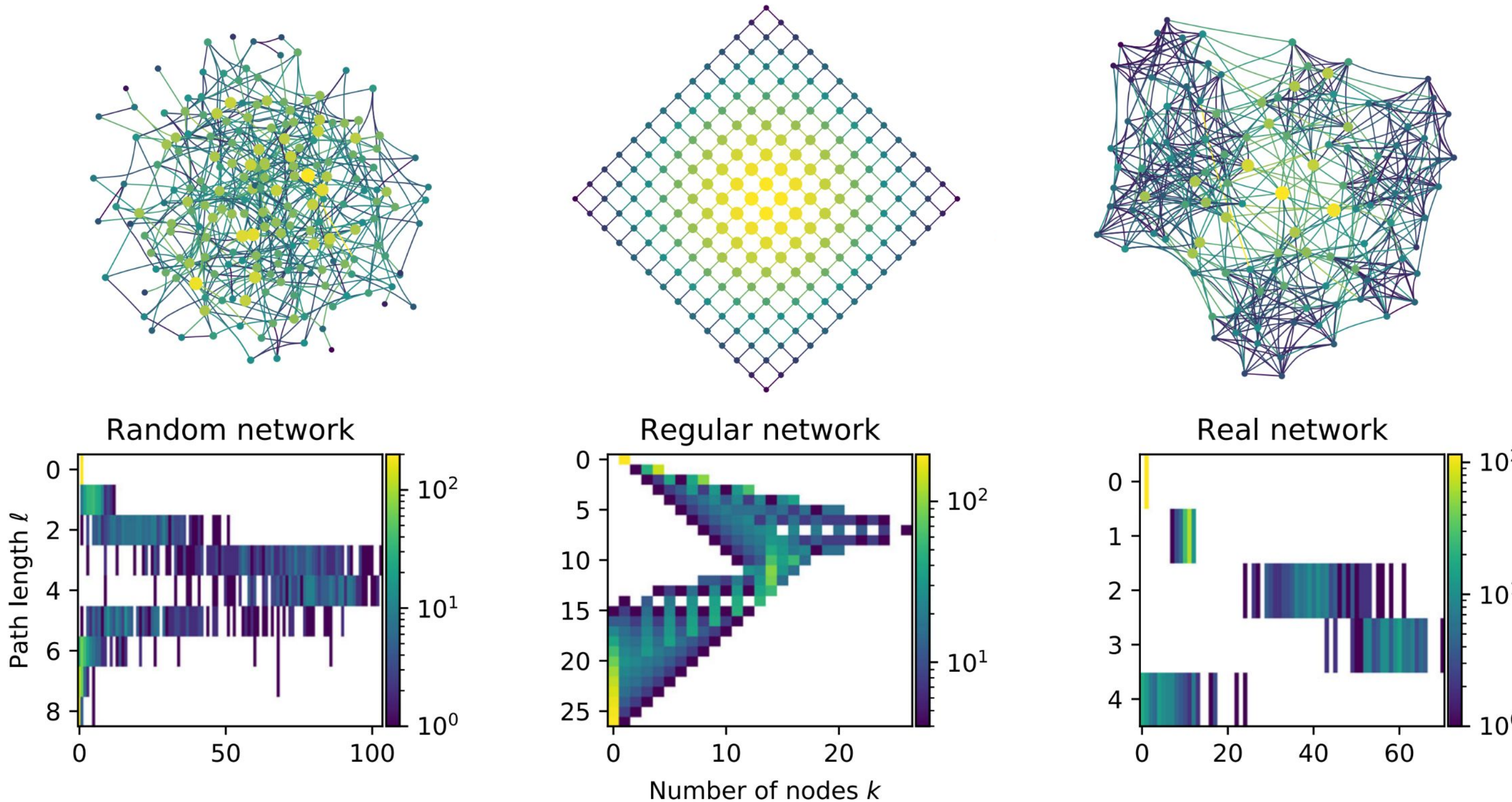


Fig. 1 Example networks and their portraits. The random network is an Erdős-Rényi graph while the real network is the NCAA Division-I football network (Park and Newman 2005). Colors denote the entries of the portrait matrix B (Eq. (2); white indicates $B_{\ell,k} = 0$)

Bagrow, J. P., & Bollt, E. M. (2019).
An information-theoretic, all-scales approach to comparing networks. Applied Network Science, 4(1), 1-15.

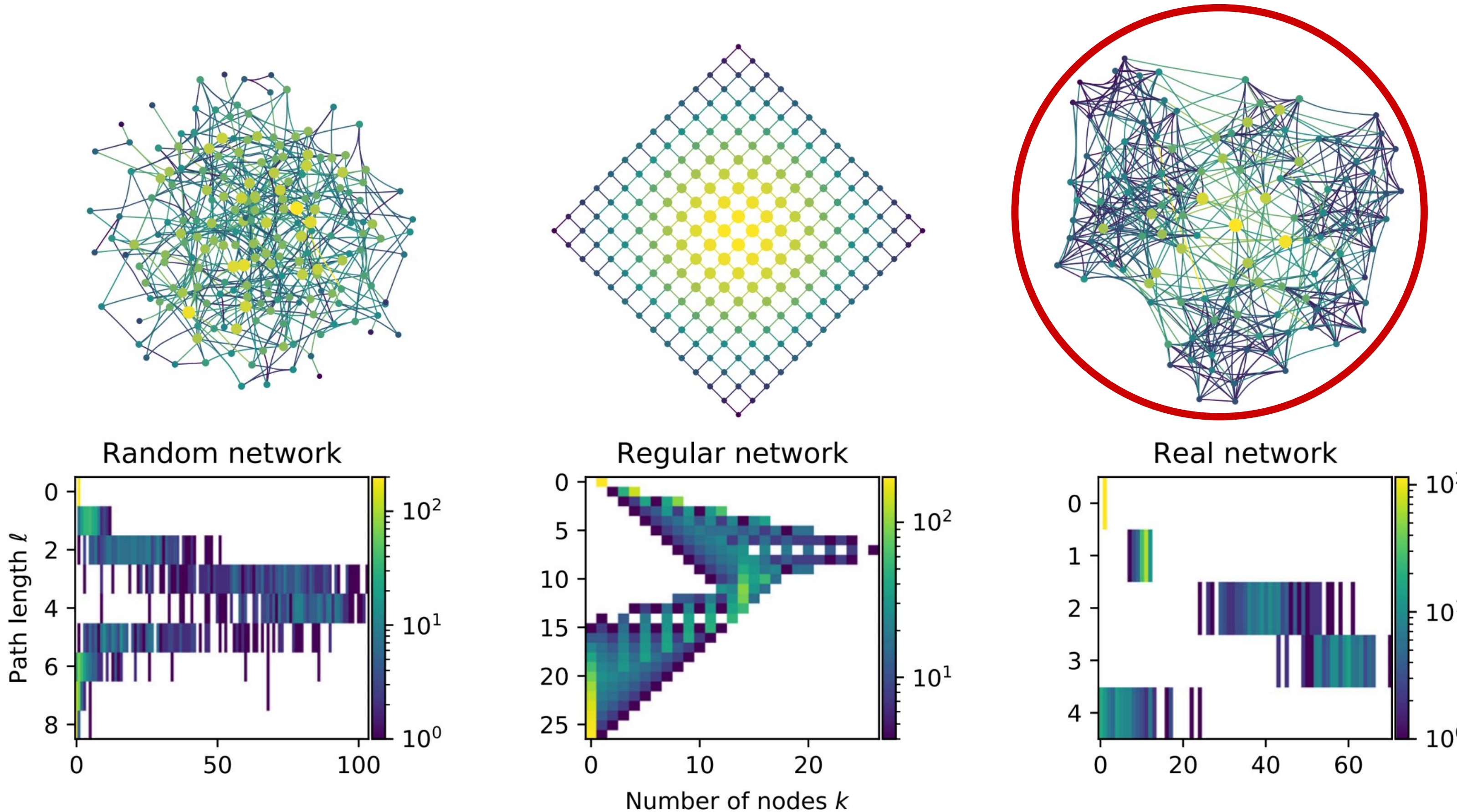


Fig. 1 Example networks and their portraits. The random network is an Erdős-Rényi graph while the real network is the NCAA Division-I football network (Park and Newman 2005). Colors denote the entries of the portrait matrix B (Eq. (2); white indicates $B_{\ell,k} = 0$)

Bagrow, J. P., & Bollt, E. M. (2019).
An information-theoretic, all-scales approach to comparing networks. Applied Network Science, 4(1), 1-15.

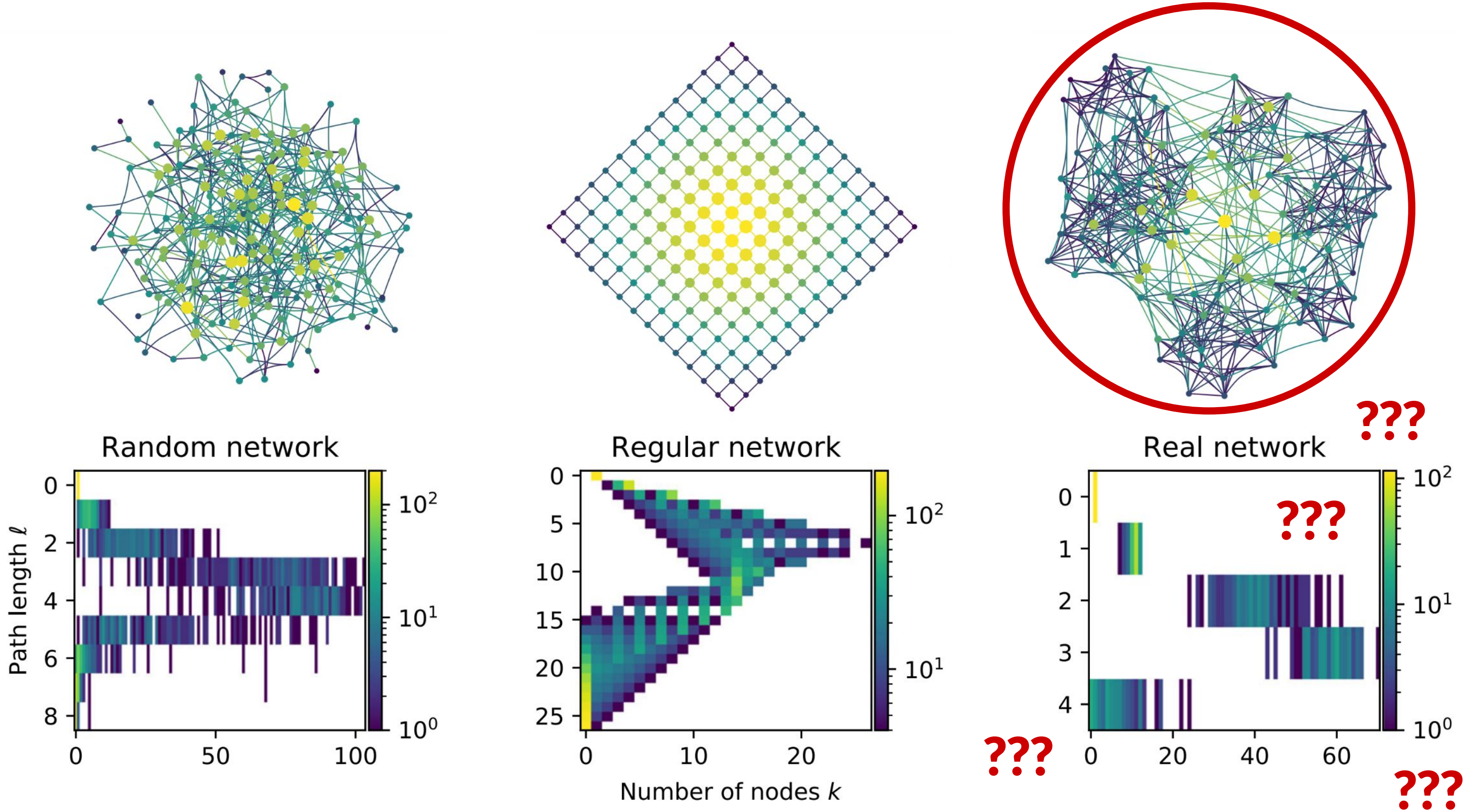
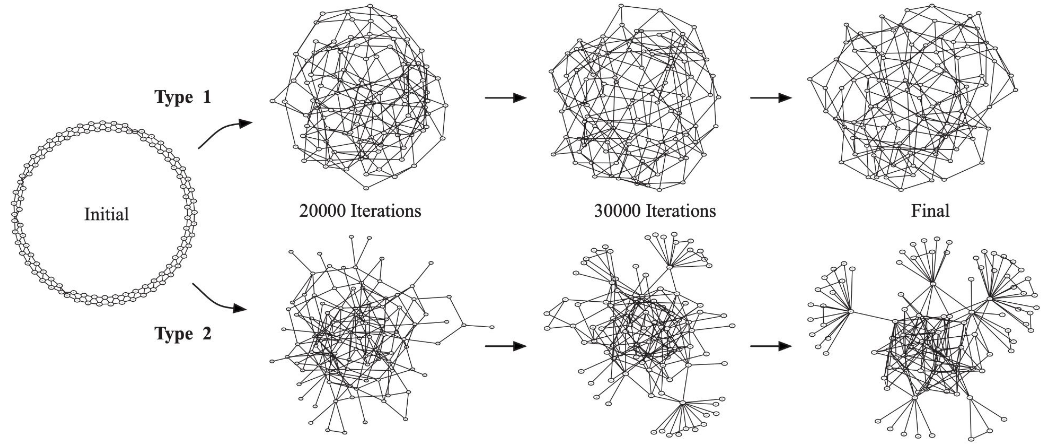


Fig. 1 Example networks and their portraits. The random network is an Erdős-Rényi graph while the real network is the NCAA Division-I football network (Park and Newman 2005). Colors denote the entries of the portrait matrix B (Eq. (2); white indicates $B_{\ell,k} = 0$)

Bagrow, J. P., & Bollt, E. M. (2019).
An information-theoretic, all-scales approach to comparing networks. Applied Network Science, 4(1), 1-15.



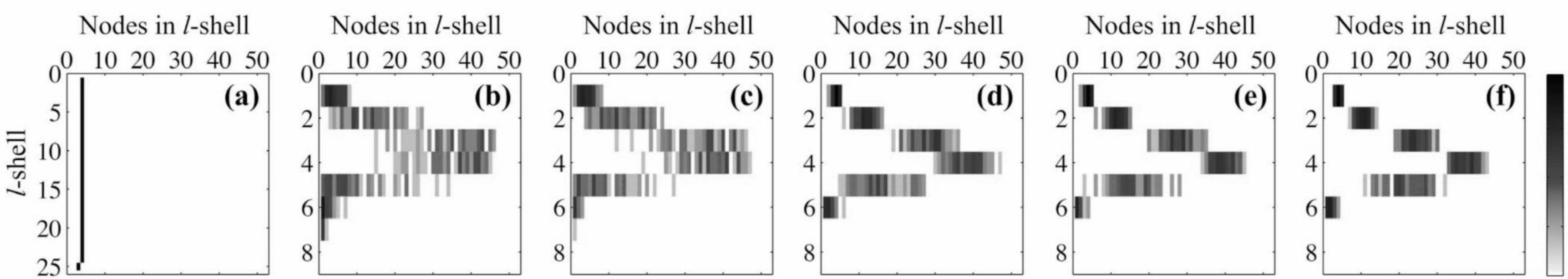
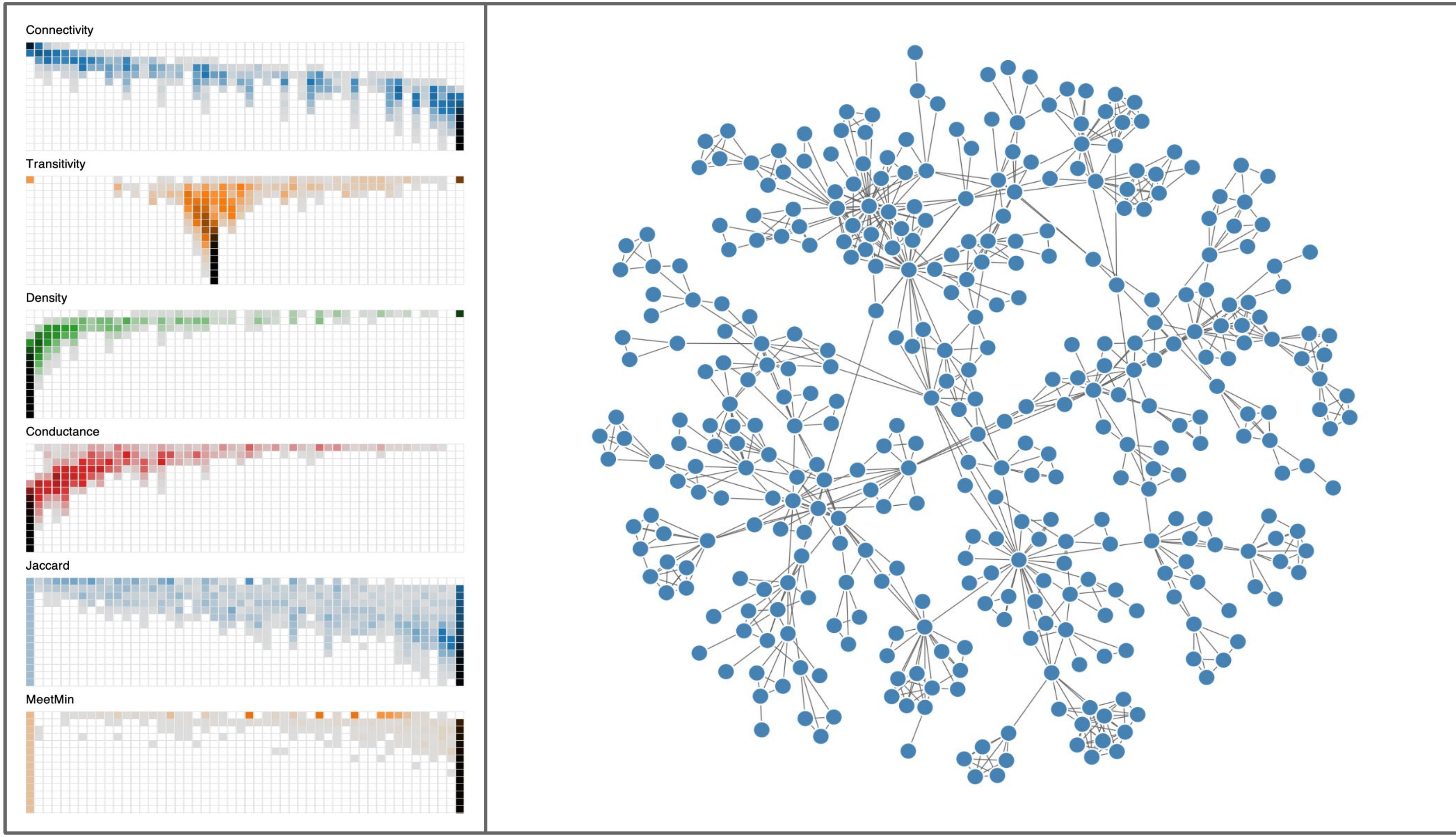


FIG. 12. (Color online) Evolution of a B matrix portrait when using the order-parameter performance measure with $\hat{\sigma}=0.6$. Initial topology was a periodic ring lattice with nearest and next-nearest-neighbor coupling. B matrices were taken at iterations (a) 1, (b) 4000, (c) 8000, (d) 14 000, (e) 20 000, and (f) 180 000. Colors represent the number of nodes at a given index within the B matrix and are plotted on a log scale $[\log(b_{lk})]$.

Gorochowski, T. E., di Bernardo, M., & Grierson, C. S. (2010).
Evolving enhanced topologies for the synchronization of dynamical complex networks. Physical Review E, 81(5), 056212.



Kairam, S., MacLean, D., Savva, M., & Heer, J. (2012, May). GraphPrism: compact visualization of network structure.
In Proceedings of the international working conference on advanced visual interfaces (pp. 498-505).

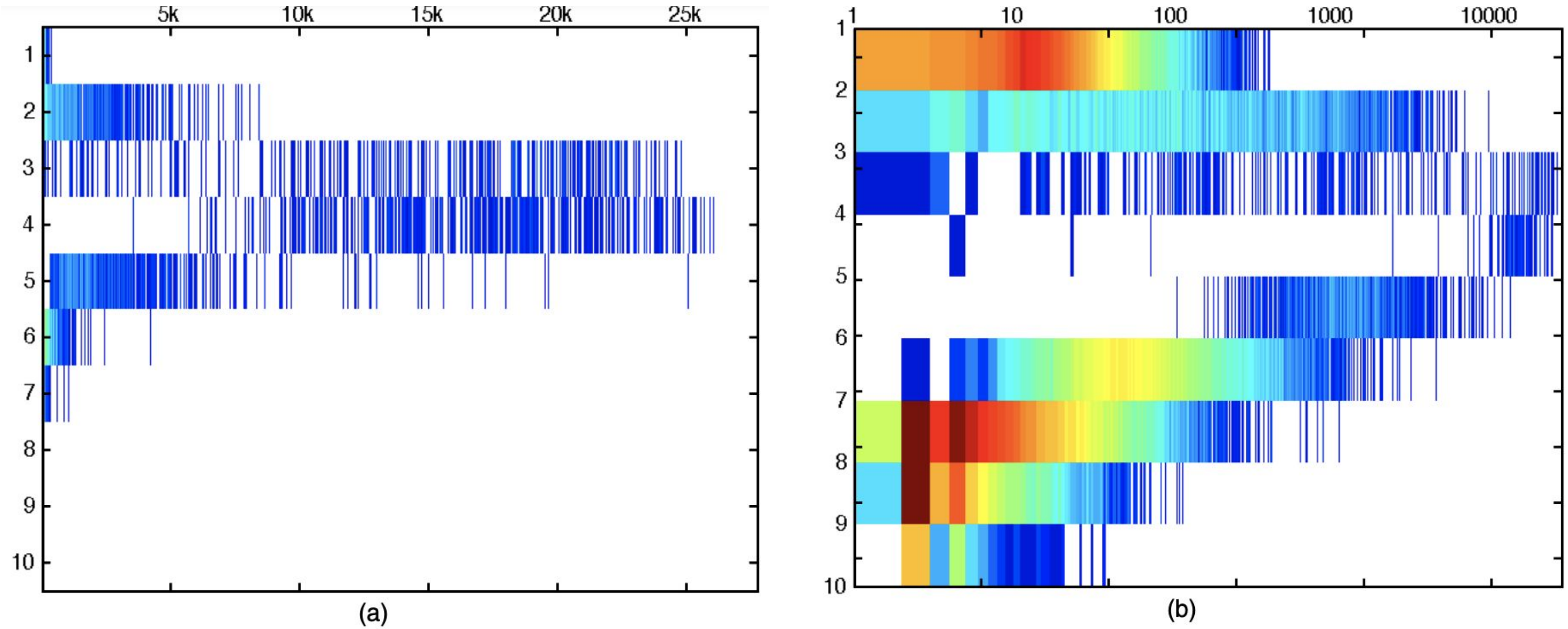


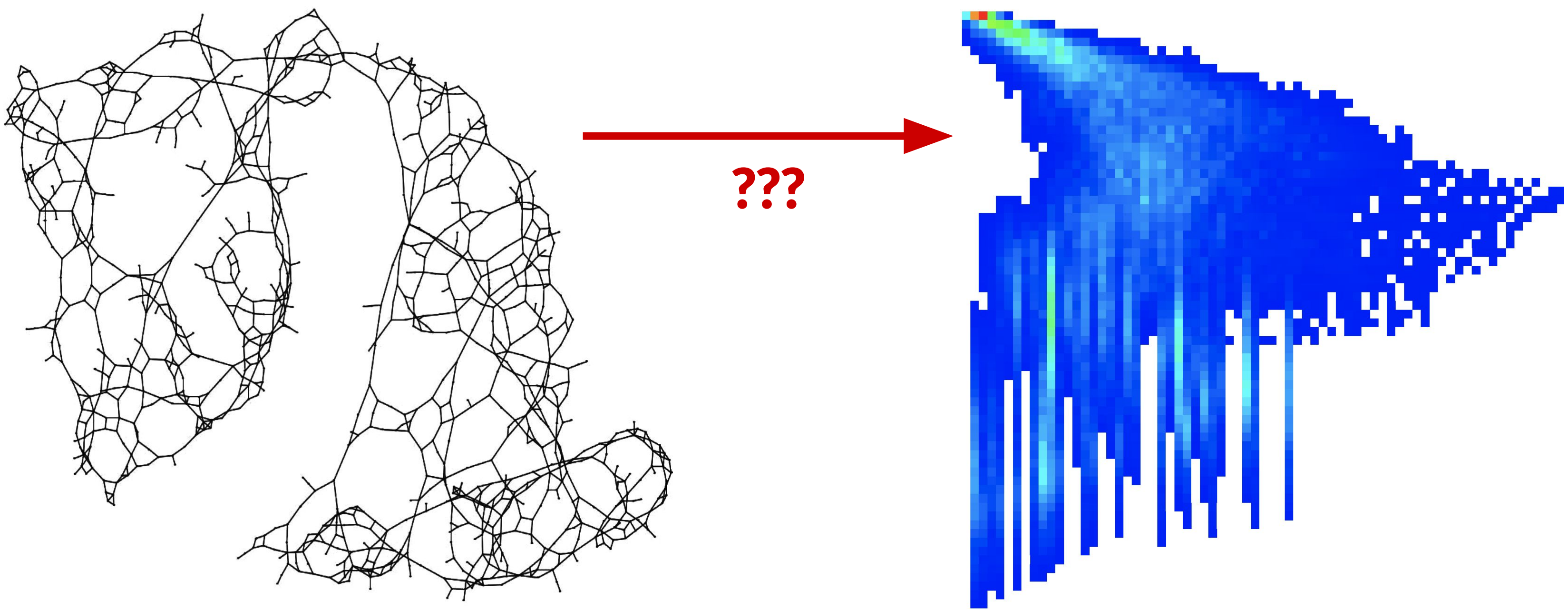
Fig. 2: (Color online) (a) A B-Matrix with a logarithmic color scale (the white background indicates zero elements of B). The degree distribution is slightly visible in the first row. The "turning point" about row 4 represents finite-size effects. Shown is the network of the 10% most connected actors on IMDB [2]. (b) The same matrix with a logarithmic horizontal axis. The degree distribution is now clearly visible.

FUN FACT
This is the
first published
network portrait.

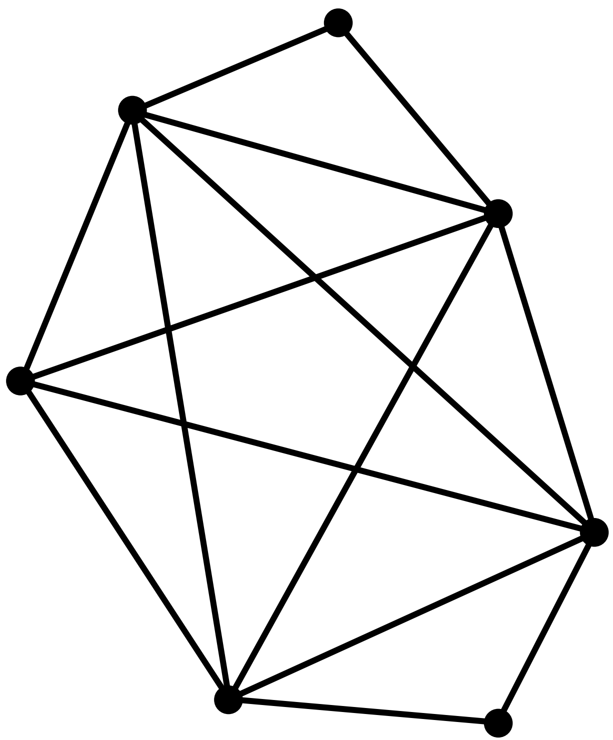
Bagrow, J. P., Bollt, E. M., Skufca, J. D., & Ben-Avraham, D. (2008).
Portraits of complex networks. EPL (Europhysics Letters), 81(6), 68004.

How exactly do we get a graph's B-Matrix? How do we interpret a network portrait?
That's exactly what `BMatrix_Explainer` is all about.

`github.com/diredirebock/BMatrix_Explainer`



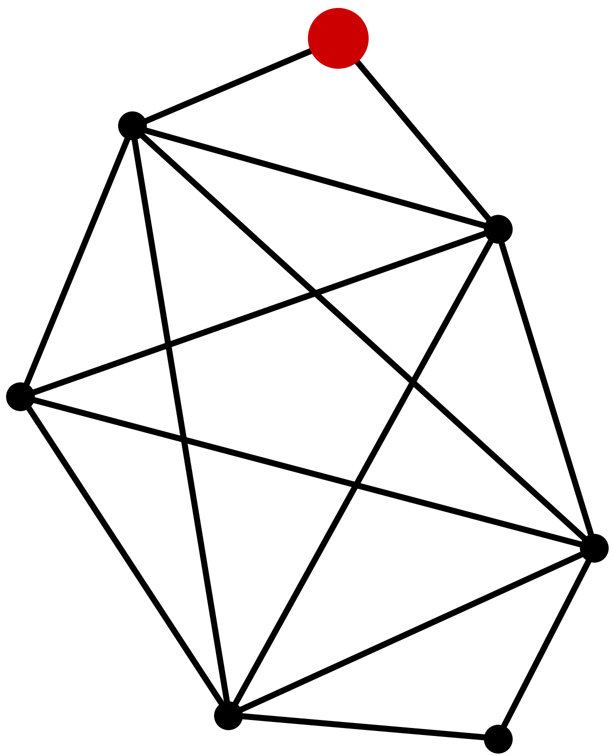
Czech, W., & Yuen, D. A. (2011, August). Efficient graph comparison and visualization using GPU.
In 2011 14th IEEE International Conference on Computational Science and Engineering (pp. 561-566). IEEE.



To get started with `BMatrix_Explainer`, consider this small graph as an explainer example:

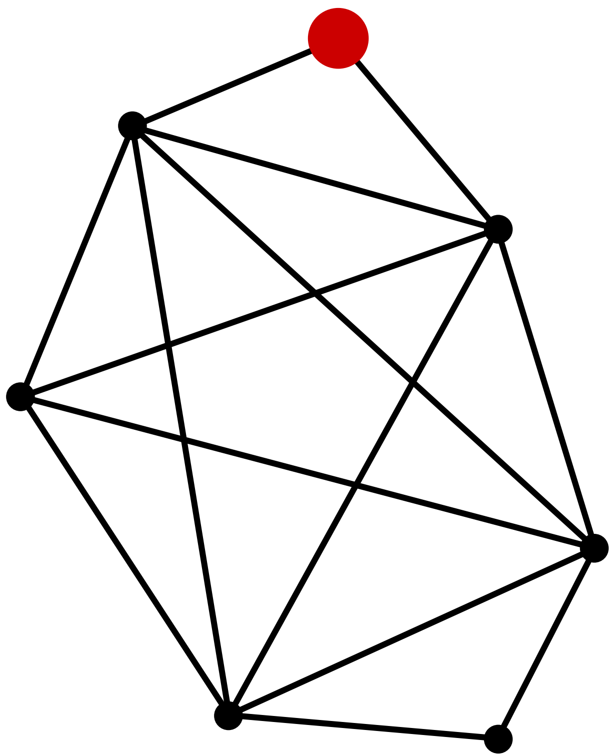
- It has 7 nodes and 14 edges
- Edgelist:

```
1-2
1-3
2-4
2-5
2-6
2-3
3-4
3-5
3-6
4-5
4-6
5-7
5-6
6-7
```



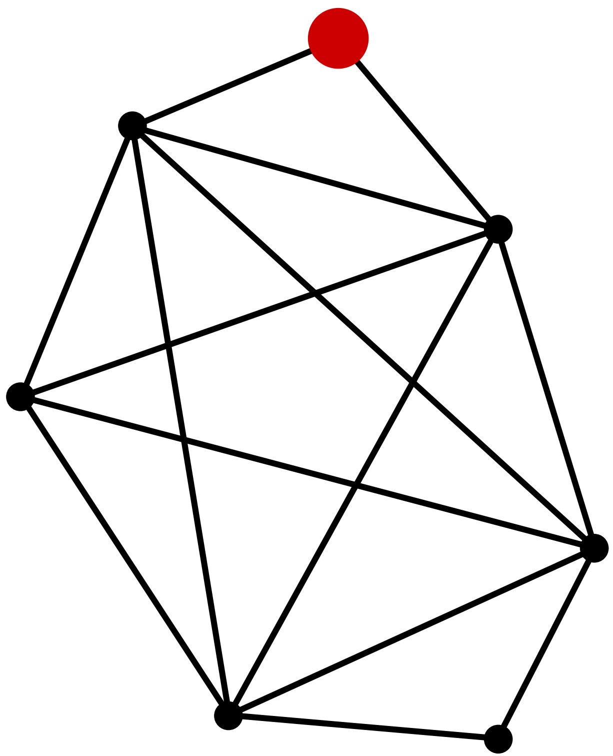
We start by picking a node, and initialize an empty matrix.

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



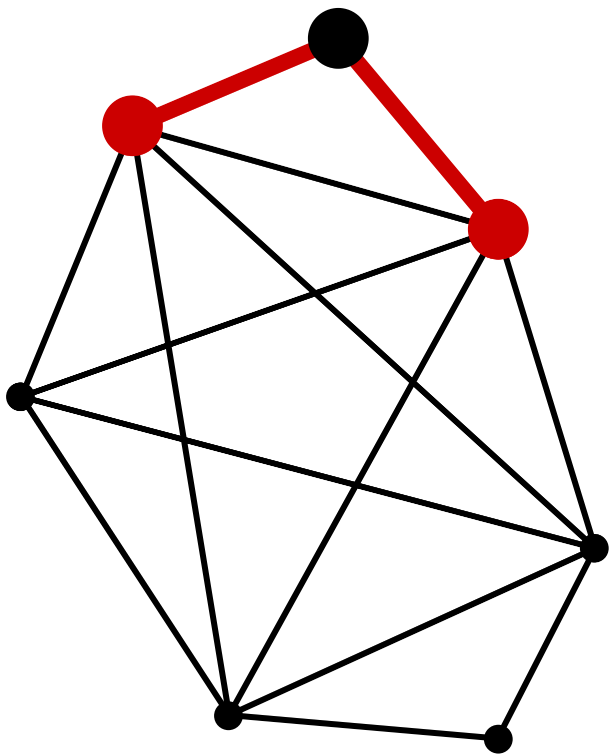
Rows are for number of hops away from this starting node, and columns are for node counts.

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



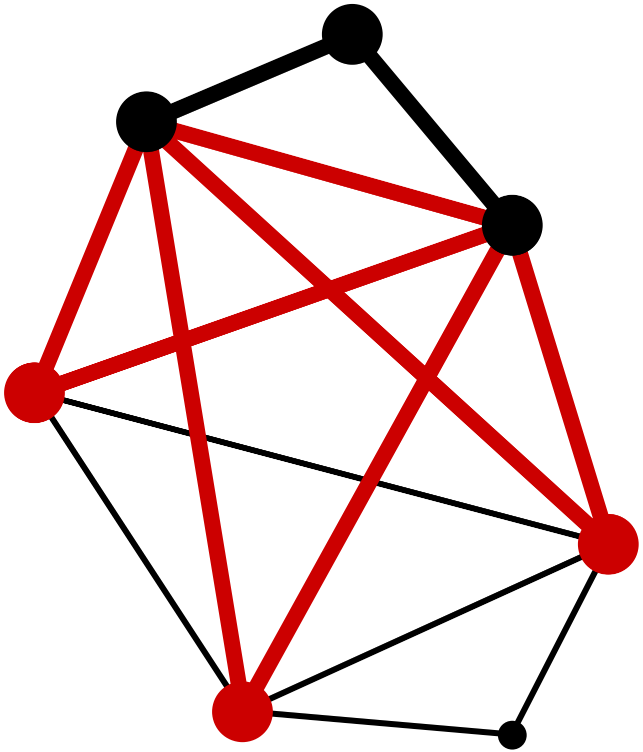
We only have one node at hand (no hops yet), so we flip the bit at zeroth row and first column.

0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



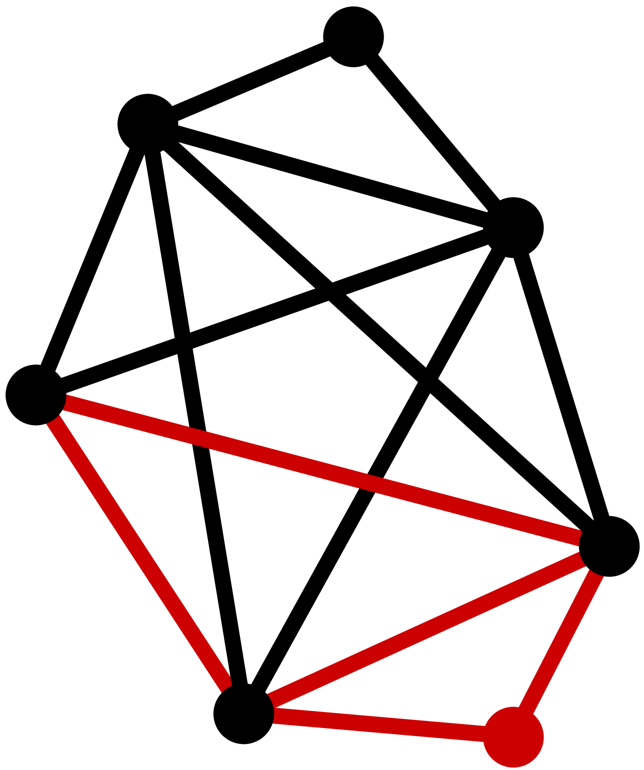
Then at first hop, there are two nodes - so we flip the bit at the first row and second column.

0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



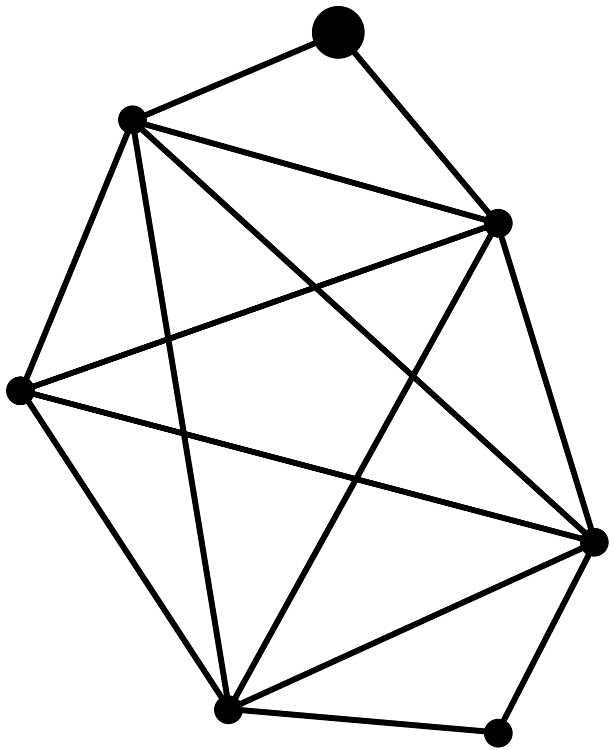
At second hop there are three nodes, so we flip the bit at the second row and third column.

0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	0	0



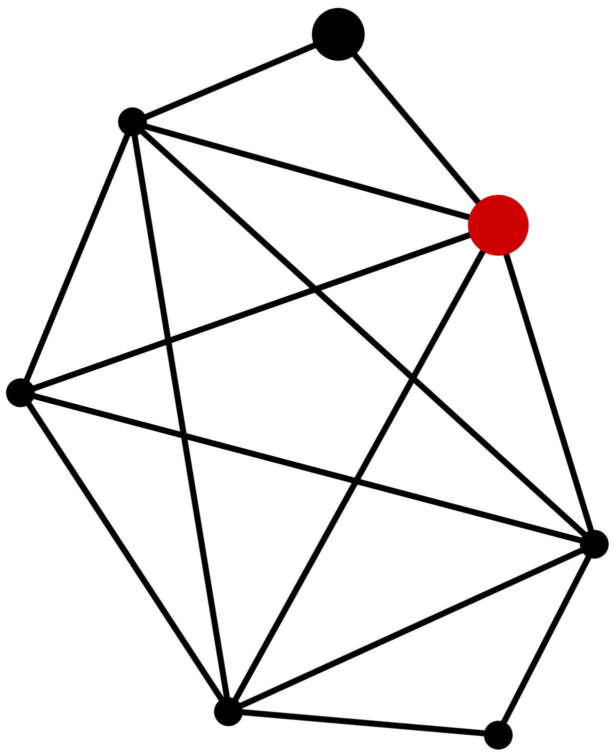
At third hop there is one node (last one), so we flip the bit at the third row and first column.

0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	1	0	0	0	0



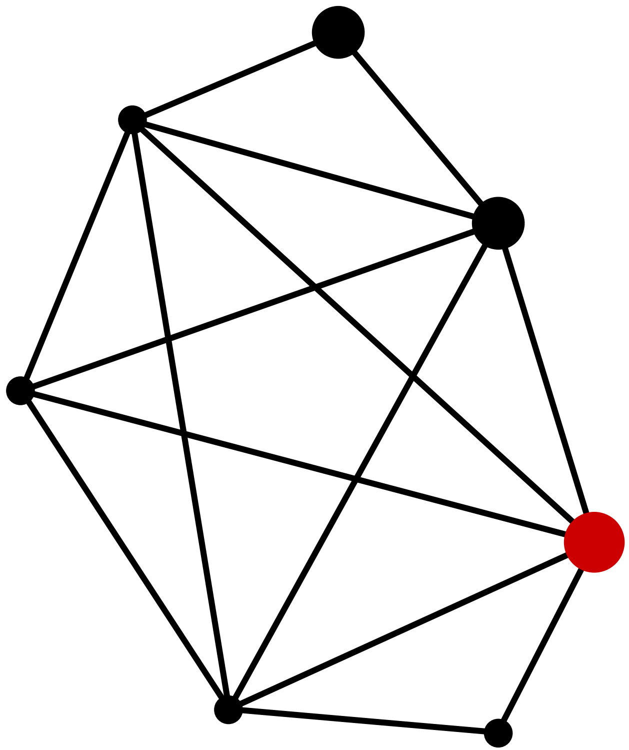
This completes the bit matrix of node 1 (of 7).

0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	1	0	0	0	0



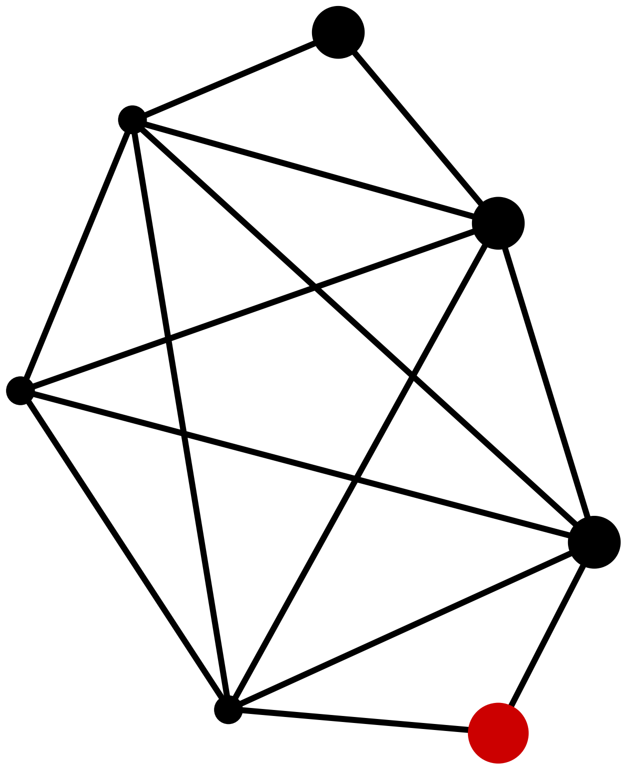
We repeat and get a bit matrix for node 2 (of 7), and we store the already completed matrix.

0	1	0	0	0	0
0	0	0	0	0	1
0	1	0	0	0	0
1	0	0	0	0	0



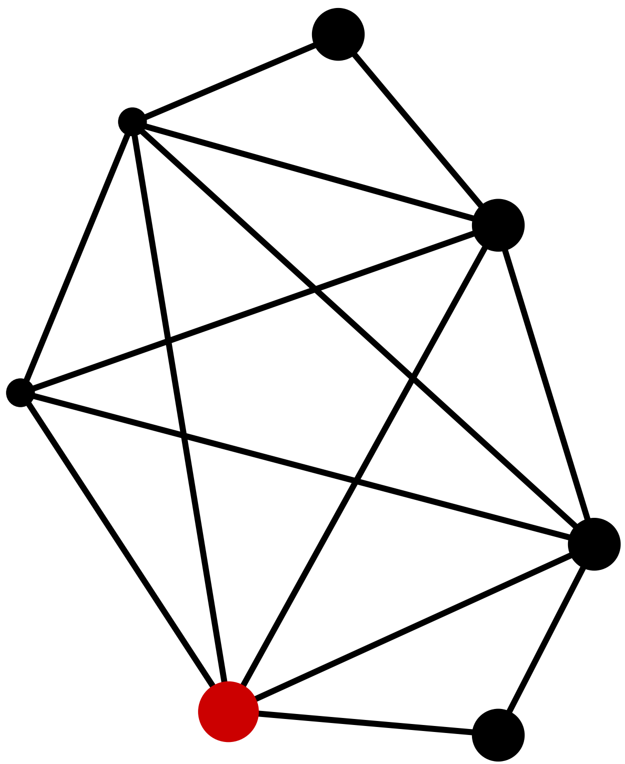
This is the bit matrix of node 3 (of 7), and we save the two already completed matrices.

0	1	0	0	0	0
0	0	0	0	0	1
0	1	0	0	0	0
1	0	0	0	0	0



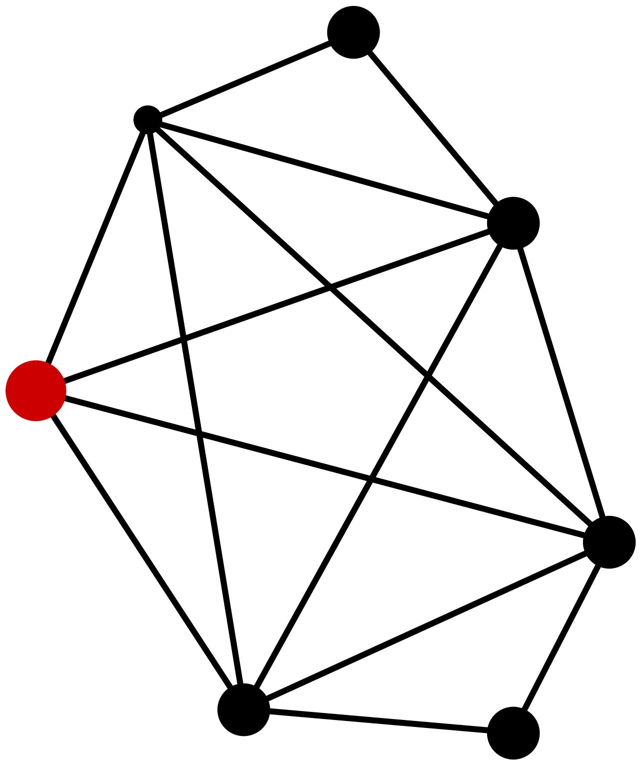
The bit matrix of node 4 (of 7), and we save the three already completed matrices.

0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	1	0	0	0	0



The bit matrix of node 5 (of 7), and we save the four already completed matrices.

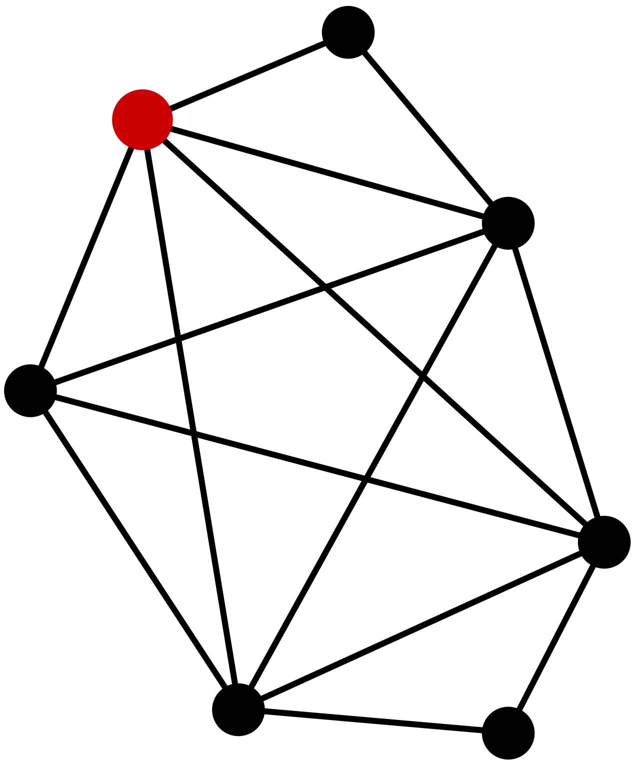
0	1	0	0	0	0
0	0	0	0	0	1
0	1	0	0	0	0
1	0	0	0	0	0



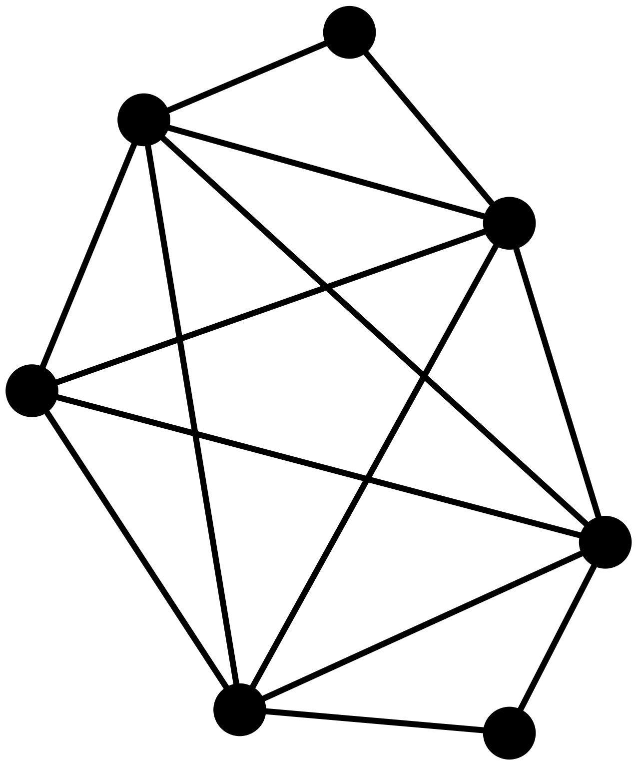
The bit matrix of node 6 (of 7), and we save the five already completed matrices.

0	1	0	0	0	0
0	0	0	0	1	0
0	0	1	0	0	0
1	0	0	0	0	0

Finally, the bit matrix of node 7 (of 7), and we save the six already completed matrices.



0	1	0	0	0	0
0	0	0	0	0	1
0	1	0	0	0	0
1	0	0	0	0	0



We add these seven bit matrices element-wise into a single matrix. This ends the algorithm.

0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

We're done!

The B-Matrix of the graph.

And it has a bunch of properties.

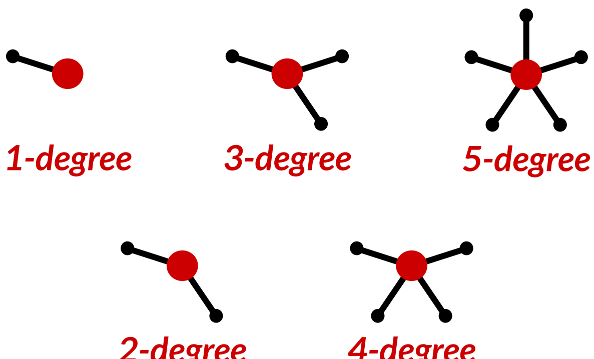
0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

Also each row adds up to the total number of nodes.

7 ◀	0	7	0	0	0	0
7 ◀	0	0	2	0	1	4
7 ◀	0	4	1	2	0	0
7 ◀	5	2	0	0	0	0

The first row marks the number of times different node counts happened at exactly one hop away from the starting node.

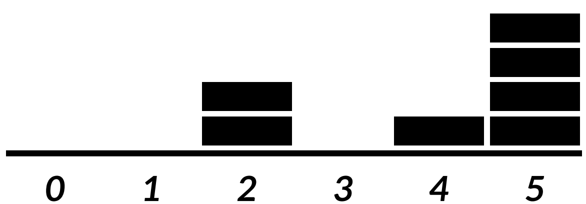
The node degree is how many edges a node has, so this row is effectively a record of frequency of node degrees.



0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

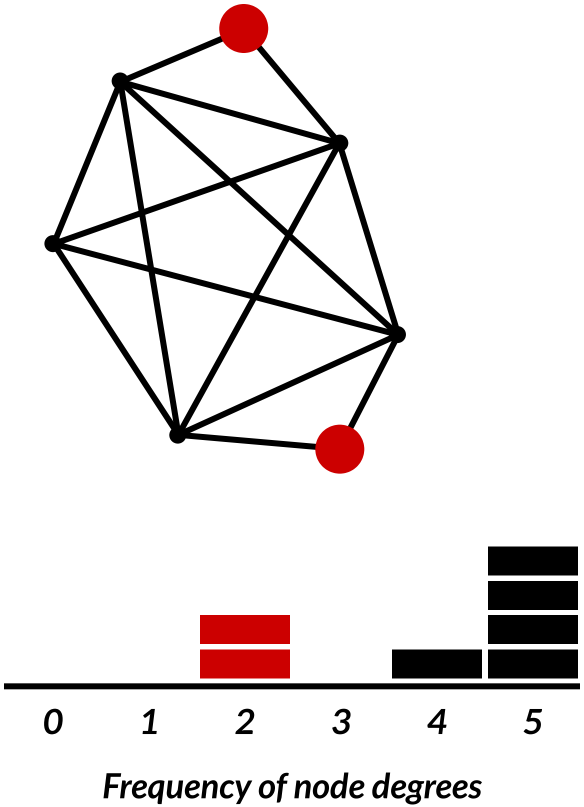
1 2 3 4 5

We can visualize the distribution of node degrees with a bar chart of counts (histogram).



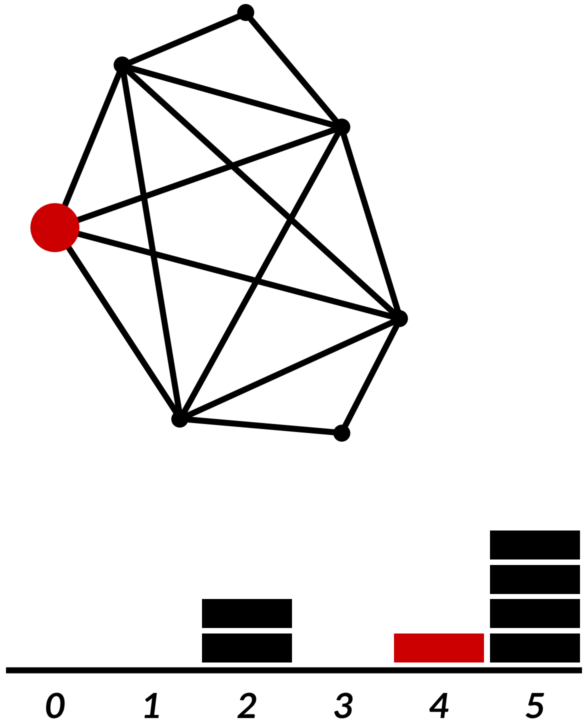
Frequency of node degrees

0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0



[Row 1, Column 2] Two nodes of degree 2

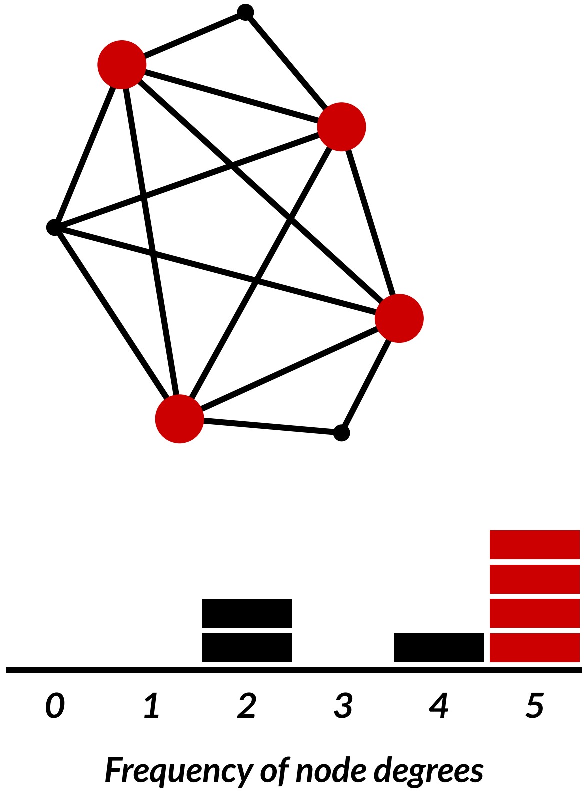
0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0



Frequency of node degrees

[Row 1, Column 4] One node of degree 4

0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

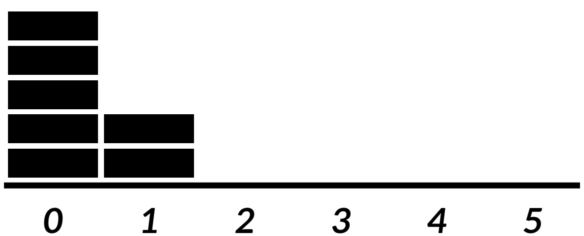


[Row 1, Column 5] Four nodes of degree 5

0	7	0	0	0	0
0	0	2	0	1	**4**
0	4	1	2	0	0
5	2	0	0	0	0

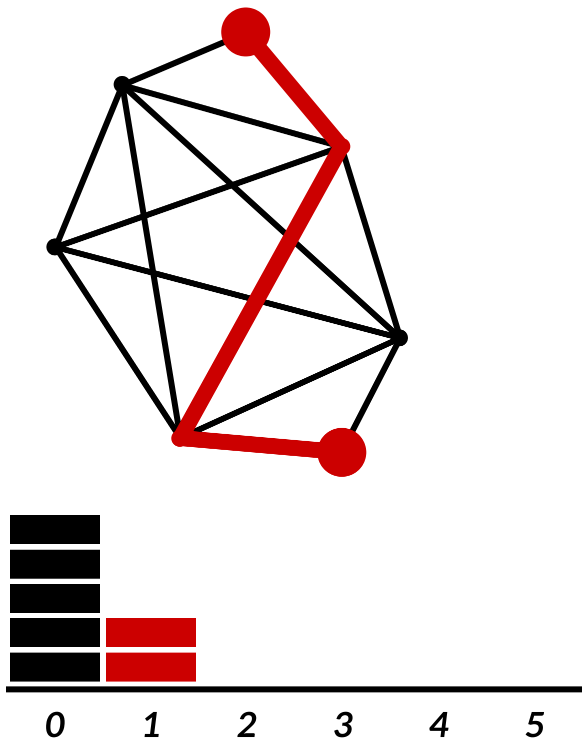
Finally, the last row is the maximum number of hops the algorithm got through before running out of nodes.

In other words, the final hop number is equivalent to the diameter of the graph, *L-shell* of 3 in this case.



Frequency of node 3-shell degrees

0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0



Frequency of node 3-shell degrees

The graph diameter is the length of the shortest path between the two most distanced nodes.

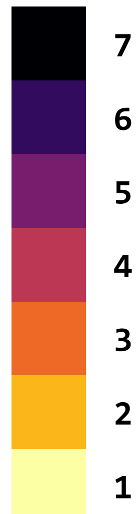
0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

In contrast to this explainer example, the B-Matrix of a real-world graph can be very large.

It is not practical to show this data abstraction directly with numbers, we need a visual encoding.

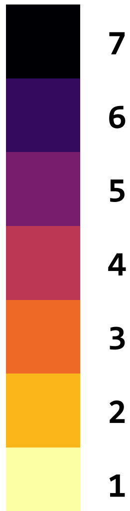
0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

In literature this is solved by mapping the B-Matrix node count to a colormap range.



0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

And the result is a **heatmap**.

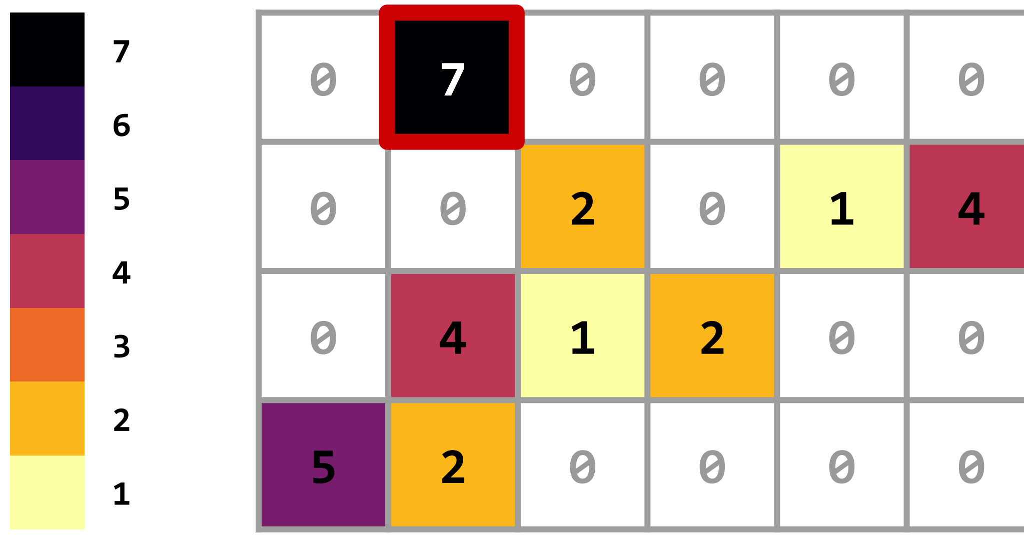


0	7	0	0	0	0
0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

However, we can do one more edit to increase information resolution in this heatmap idiom.

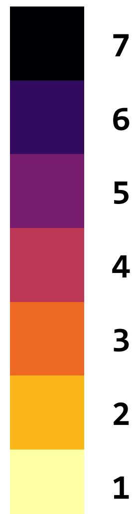
Notice that the zeroth row **always has the highest value** (which sets the colormap extreme).

In large networks this value can be so high that the colormap must be log-transformed.



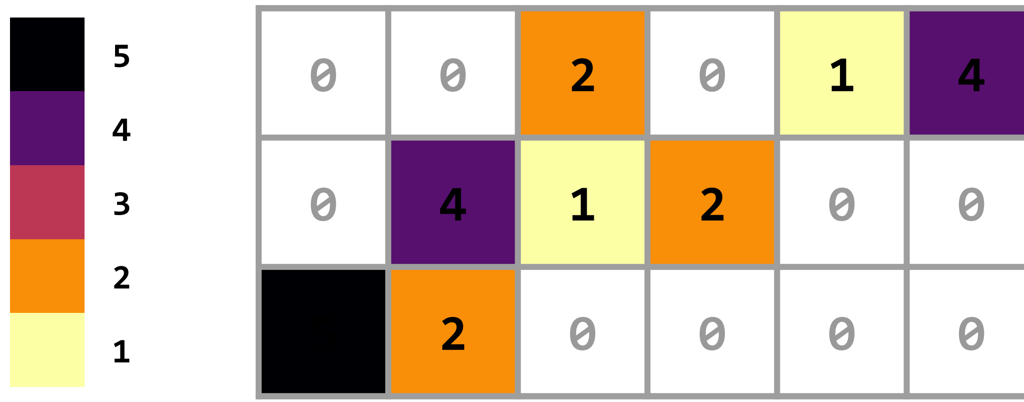
We can safely **remove the zeroth row**.

This is fine because this row only contains redundant data (total node count).

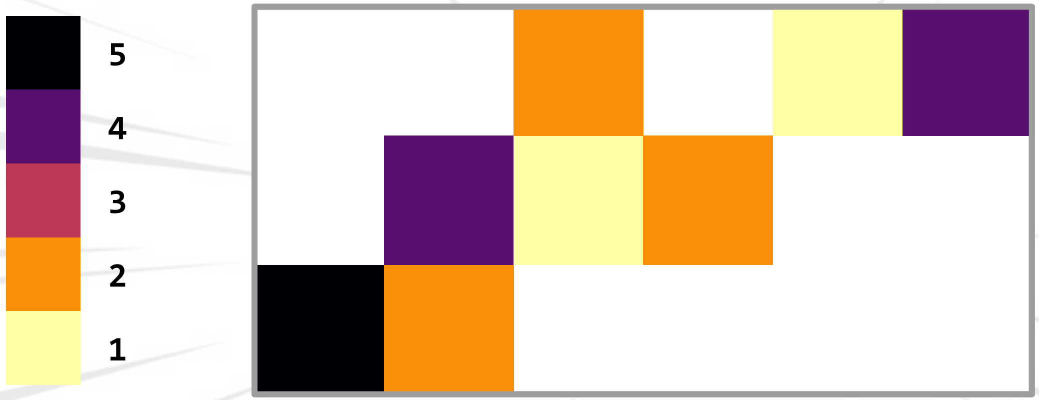


0	0	2	0	1	4
0	4	1	2	0	0
5	2	0	0	0	0

Then we can get higher fidelity by rescaling the colormap.



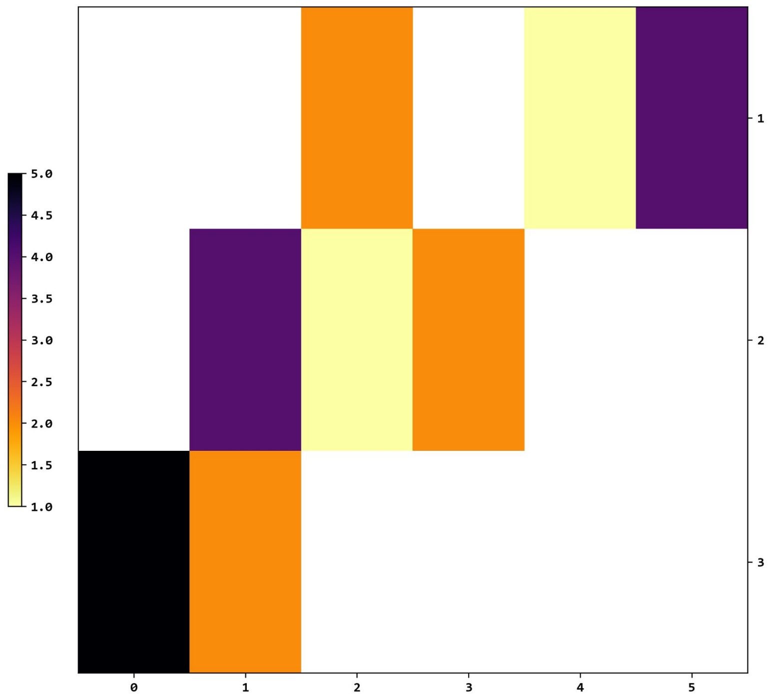
That's it!
This figure is the
network portrait
of the graph.



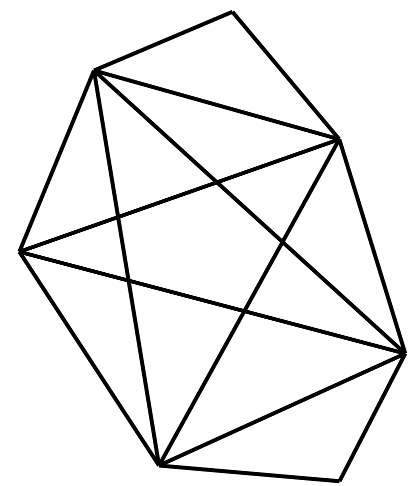
Now let's explore real-world networks with

BMatrix_Explainer

a Python-based B-Matrix visualization GitHub repo.

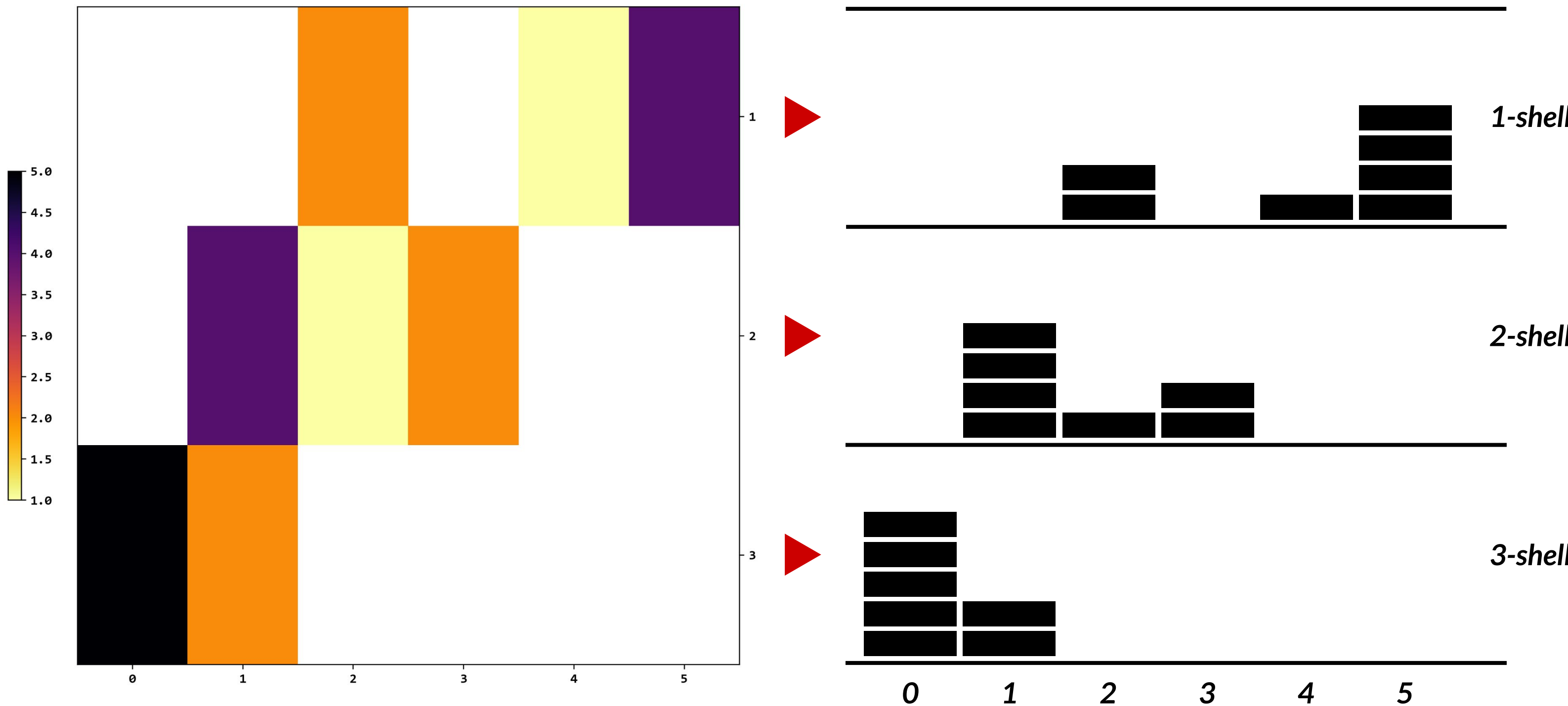


Graph	networkx.graph_atlas(1115)
Nodes	7
Edges	14

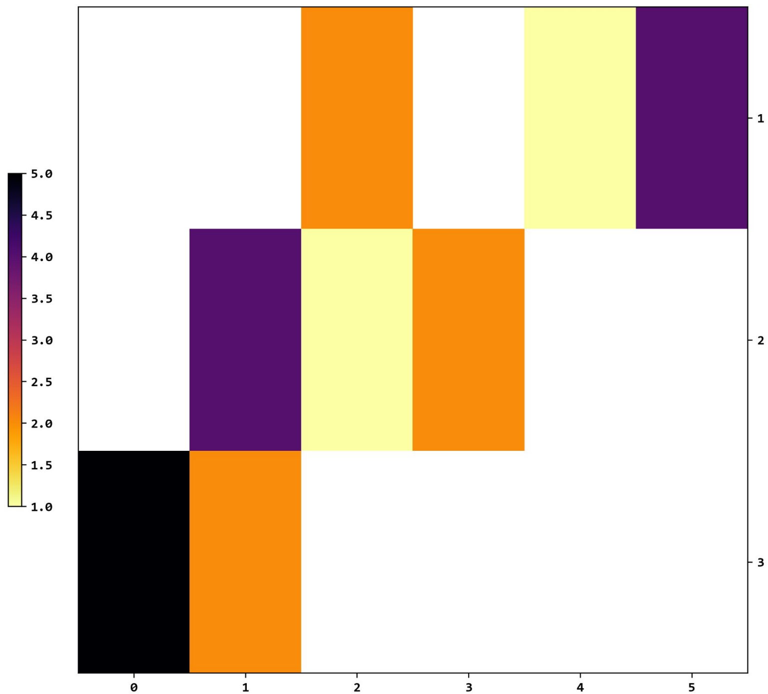


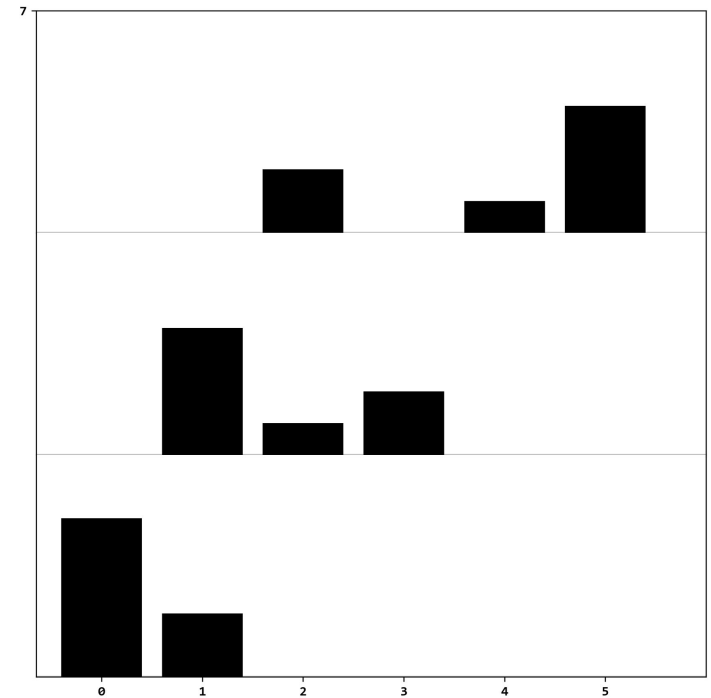
1-2
1-3
2-4
2-5
2-6
2-3
3-4
3-5
3-6
4-5
4-6
5-7
5-6
6-7

Recall that each row of the B-Matrix can be visually encoded as stacked bars for count data.

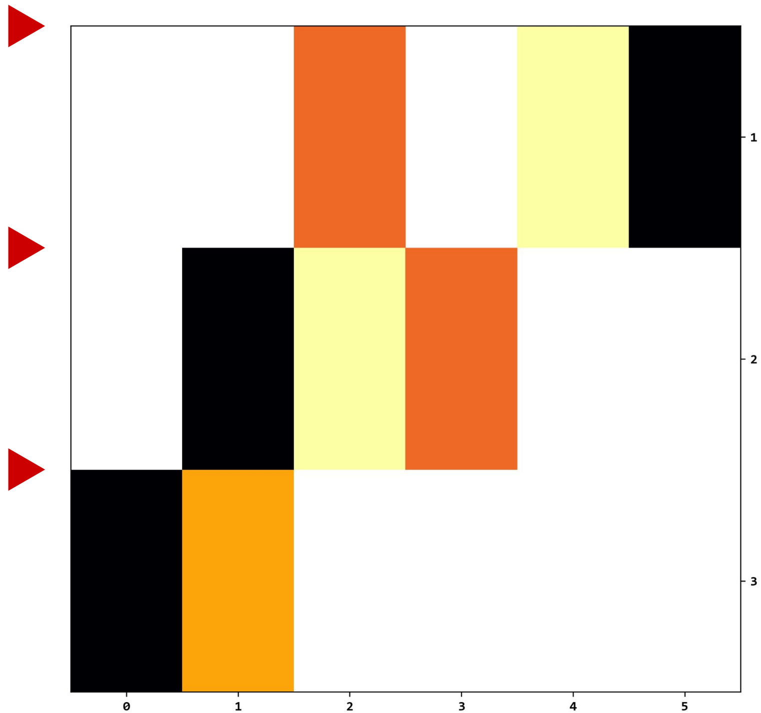


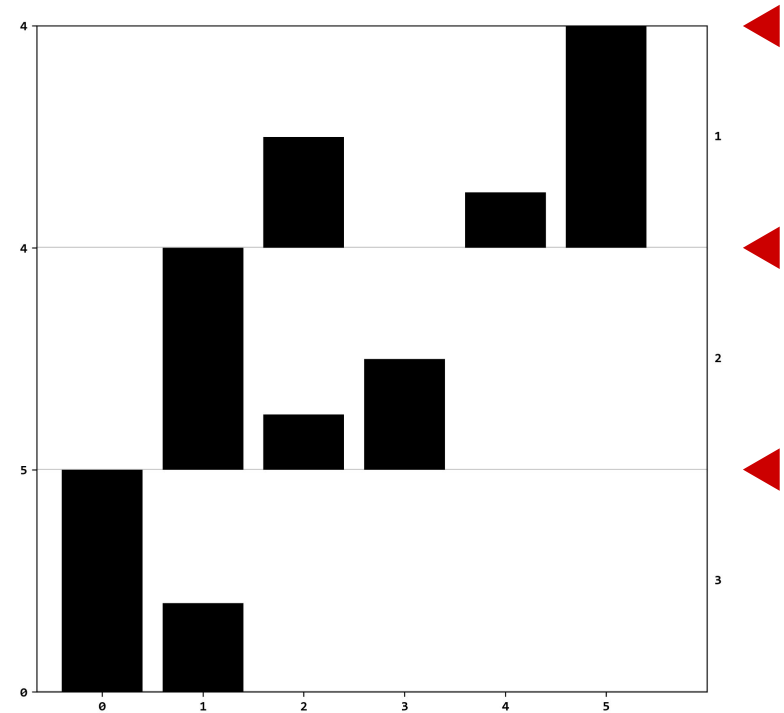
`BMatrix_Explainer` fully features this visualization with bars encoding node counts, or *histograms*.



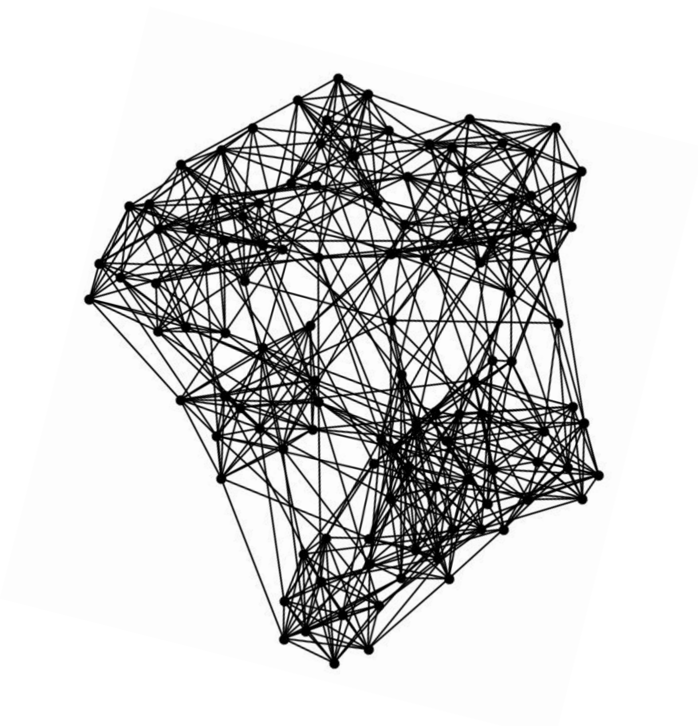


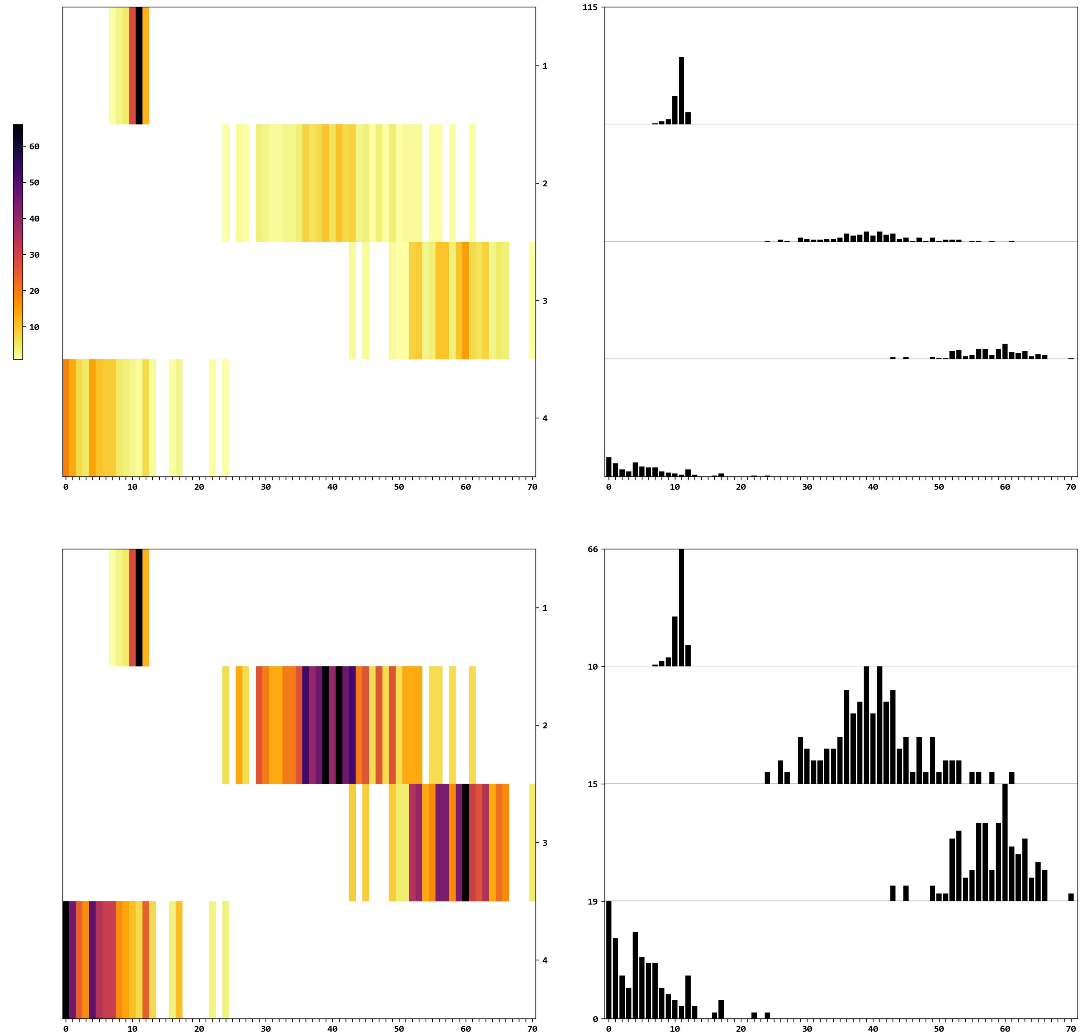
And that is not all! To further support interpretation tasks, in `BMatrix_Explainer` both color and bar encondings can have per-row normalization to enhance information discovery within hop level.





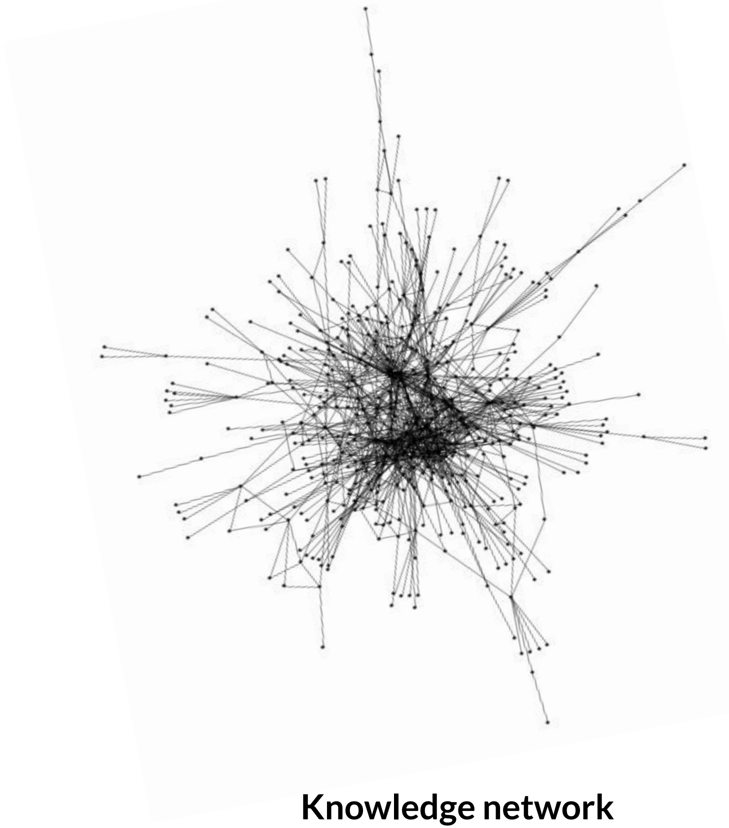
Graph	Fall 2000 College Football Games personal.umich.edu/~mejn/netdata
Nodes	115
Edges	613

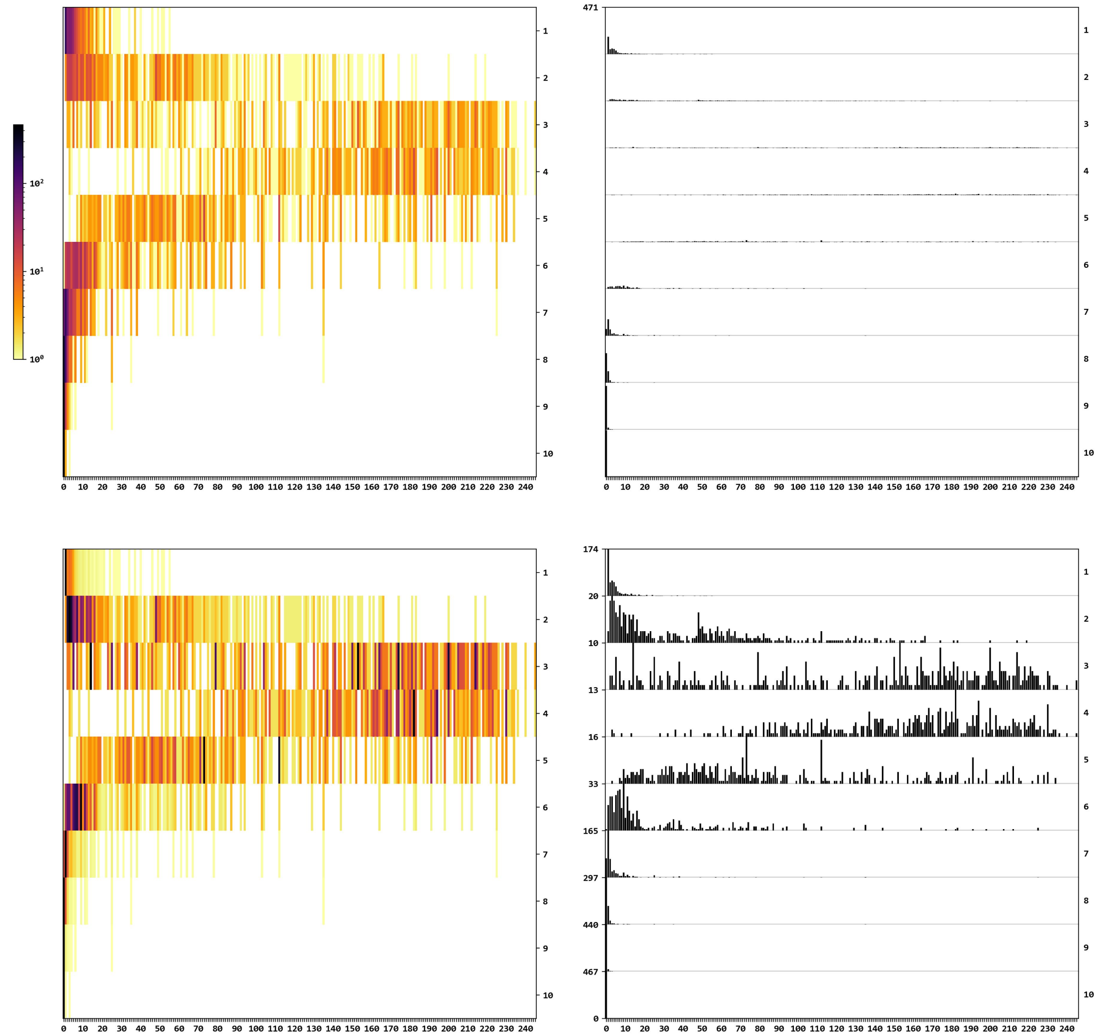




A small “small-world” network.

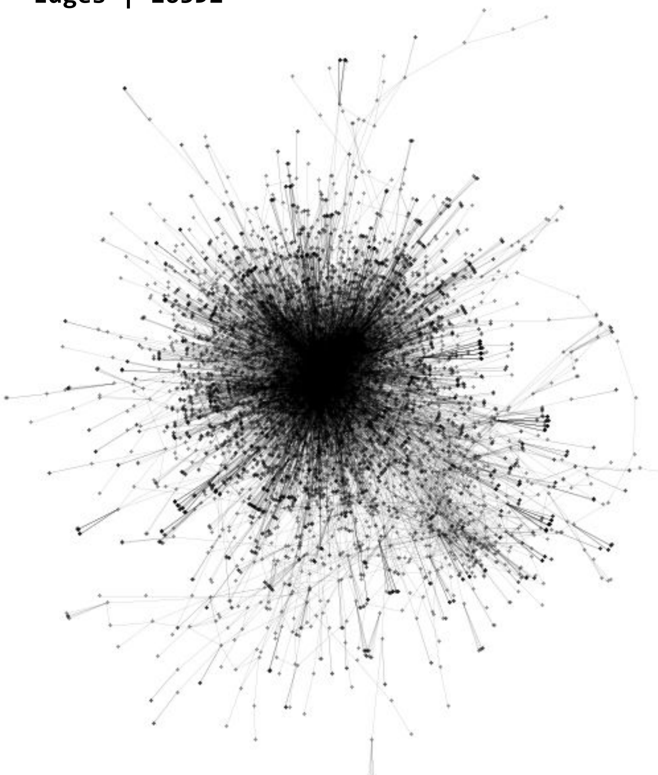
Graph	Semantic Network from "Fortran" direrdiredock/infobox_interlinker
Nodes	471
Edges	1169

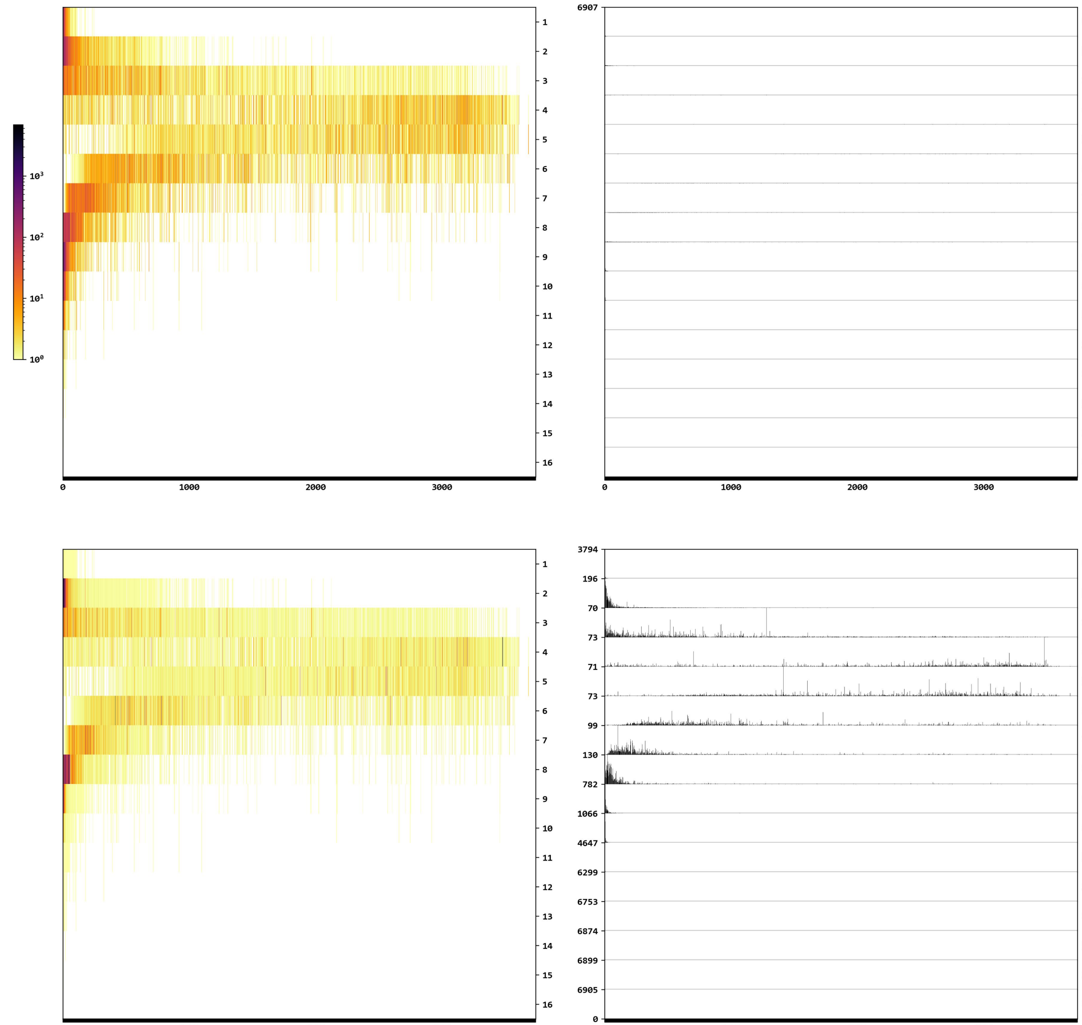




Knowledge network from *Fortran* infobox.

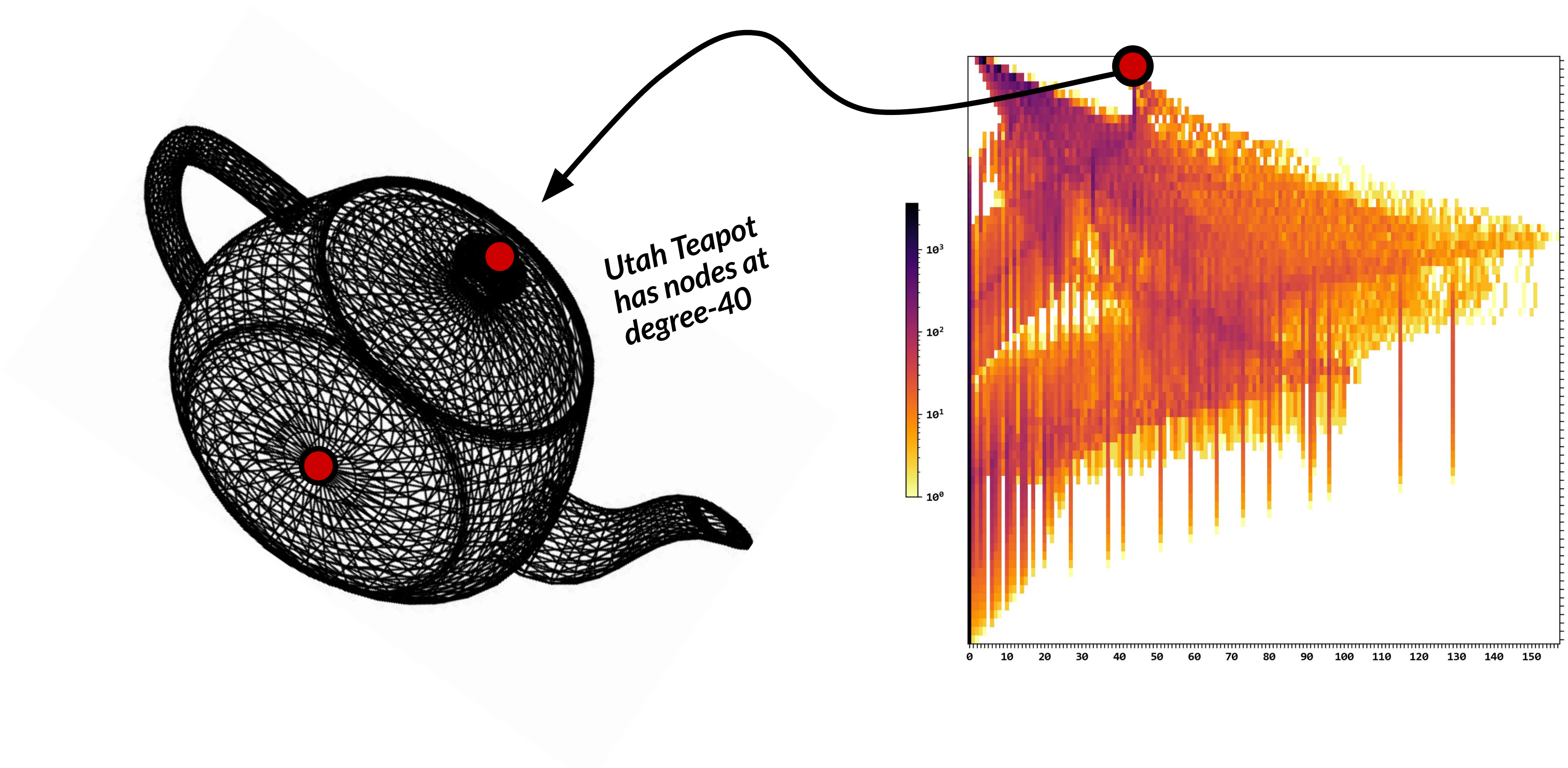
Graph	Semantic Network from "Carl_Jung" diredirеdock/infobox_interlinker
Nodes	6907
Edges	16592





Knowledge network from *Carl Jung* infobox.

For future work, I would like to build a **B-Matrix reverse-highlight visualization system**.
This can help network exploration tasks such as understanding nodes with special properties, where these located, and in relation to what global network features.



Thank You!

To check out code, data, and more figures

`https://github.com/direDireDock/BMatrix_Explainer`
`https://github.com/direDireDock/infobox_interlinker`