Towards a Format for Describing Networks

Vladimir Batagelj
IMFM
Ljubljana, Slovenia
UP, IAM and FAMNIT
Koper, Slovenia
UL, FMF
Ljubljana, Slovenia
vladimir.batagelj@fmf.uni-lj.si

Tomaž Pisanski
UP, FAMNIT
Koper, Slovenia
IMFM
Ljubljana, Slovenia
tomaz.pisanski@upr.si

Iztok Savnik
UP, FAMNIT
Koper, Slovenia
iztok.savnik@upr.si

Ana Slavec
UP, FAMNIT
Koper, Slovenia
UP, IAM InnoRenew CoE
Koper, Slovenia
ana.slavec@famnit.upr.si

Nino Bašić
UP, FAMNIT
Koper, Slovenia
IMFM
Ljubljana, Slovenia
nino.basic@famnit.upr.si

Abstract

The article provides an overview of the most important network analysis resources and the various types of networks encountered in their use. Based on experience in developing the NetsJSON format, we present components that an exchange/archive format for describing networks should contain.

Keywords

Network analysis, Network types, Identification, Format, Exchange, Archive, Data repository, Factorization, JSON, FAIR.

1 Introduction

Open data plays a crucial role in ensuring the computational reproducibility and verifiability of published results. The obtained results can be verified or supplemented with other methods. Collections of similar and well-documented datasets are also crucial for developing new methods to analyze specific types of data. It is good to test a new method on several datasets and check whether it gives meaningful/expected results. When preparing such data, it is essential to adhere to the FAIR principles – Findability, Accessibility, Interoperability, and Reusability. To facilitate ease of use, data should ideally be stored in a text format that preserves the structure of the data and includes relevant metadata. Datasets are alive. Their connection to open repositories is important for their accessibility and maintenance.

In 2023, the International Network for Social Network Analysis (INSNA) requested that Zachary Neal form a working group to develop **recommendations for sharing network data and materials**. They were published in *Network Science* in 2024 [21] accompanied by the *Endorsement page* [20].

Network analysis is an area where data is often stored in diverse file formats. It would be highly beneficial to adopt a common "archiving/exchange" format that can describe (almost) all networks and support authoring, deposit, exchange, visualization, reuse, and preservation of network data. Such a format would allow us to obtain the specific descriptions required by various network analysis programs using relatively simple scripts.

We have many years of experience in developing formats for describing graphs and networks [11, 10, 4]. We will present the NetsJSON format currently used to describe networks with structured data, and some ideas for improving it. This could be a starting point for the development of a common format for exchanging and archiving networks.

2 Support for network analysis

The concept of a network is an extension of the concept of a graph. A graph describes the structure of a network. Network analysis is a branch of data analysis that draws heavily on the concepts and results of graph theory. The difference between the two is that networks are usually "irregular", while most problems and results of graph theory assume some "regularity".

There are many tools and programs for network analysis. For example, UCINET, Pajek, Gephi, NetMiner, Cytoscape, NodeXL, E-Net, Tulip, PUCK, GraphViz, SocNetV, Kumu, Polinode, etc. Programmers can use network analysis packages/libraries in a variety of programming languages (Python, R, Julia, C++, etc.).

They are supporting various network description formats: CSV, UCINET DL, Pajek NET, Gephi GEXF, GDF, GML, GraphML, GraphX, GraphViz DOT, Tulip TPL, Netdraw VNA, Spreadsheet, etc. [13, 25, 16].

In addition, network data appears in several application areas such as chemistry and genealogy. There are many formats for describing these data.

Network datasets are available in multiple repositories. Some repositories only provide metadata about an individual network and a link to the actual dataset. At the same time, others also store the data itself and offer users a display of basic network characteristics. None of them explicitly adheres to FAIR data principles.

Interesting networks can also be found on general data repositories such as Kaggle. Networks can also be created programmatically from selected data. For example, from bibliographic data from the free OpenAlex service, we can create collections of bibliographic networks on a selected topic using the OpenAlex2Pajek library in R.

For detailed lists of network analysis resources with links to web pages, see GitHub/bavla/NetsJSON/Info [4].

Information Society 2025, 6–10 October 2025, Ljubljana, Slovenia

https://doi.org/10.70314/is.2025.sikdd.9

3 Graphs and networks

3.1 Unit identification

The fundamental task in transforming data about the selected topic into a structured dataset to be used in further analyses is the *identification of units* (entity recognition). Often, the source data are available as unstructured or semi-structured text. In this case, the transformation is a task of the *computer-assisted text analysis* (CaTA). *Terms* considered in TA are collected in a *dictionary* (it can be fixed in advance, or built dynamically). The two main problems with terms are: *equivalence* – different words representing the same term, and *ambiguity* – same words representing different terms. Because of these, the *coding* – the transformation of raw text data into formal *description* – is often done manually or semiautomatically.

We assume that unit identification assigns a unique identifier (ID) to each unit. For some types of units, such IDs are standardized: ISO 3166-1 alpha-2 two-letter country codes, ISO 9362 Bank Identifier Codes (BIC), ORCID – Open Researcher and Contributor ID, ISSN – International Standard Serial Number, DOI – Digital Object Identifier, URI – Uniform Resource Identifier, etc.

Often, in data displays, IDs are replaced by corresponding (short) labels/names.

Besides the semantic units or *concepts* related to the selected topic, we can also identify in the raw data syntactic units – parts of the text. As *syntactic units* of TA, we usually consider clauses, statements, paragraphs, news, messages, etc.

In thematic TA, the units are coded as a rectangular matrix *Syntactic units* × *Concepts* which can be considered as a two-mode network.

In semantic TA the units (often clauses) are encoded according to the S-V-O (*Subject-Verb-Object*) model or its improvements. This coding can be directly considered as network with *Subjects* ∪ *Objects* as nodes and links (arcs) labeled with *Verbs*. This is also a basis for the semantic web and knowledge networks.

3.2 Networks

A *network* is based on two sets – a set of *nodes* (vertices) that represent the selected *units*, and a set of *links* (lines) that represent *ties* between units. They determine a *graph*.

Additional data about nodes or links can be known – their *properties* (attributes). For example: name/label, type, value, etc.

Network = Graph + Data

The data can be measured or computed.

Formally, a *network* $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W})$ consists of:

- a *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{L} = \mathcal{E} \cup \mathcal{A}$ is the set of links. A link $e \in \mathcal{L}$ is either directed – an arc $e \in \mathcal{A}$, or undirected – an edge $e \in \mathcal{E}$, $n = |\mathcal{V}|$, $m = |\mathcal{L}|$
- $\mathcal{P}$ is a set of *node value functions* / properties: $p : \mathcal{V} \to A$
- $\mathcal{W}$ is a set of *link value functions* / weights: $w : \mathcal{L} \to B$

Sometimes, implicit additional information/data about values is provided in the specifications of properties: (a) how can we compute with values – algebraic structures, semigroup, monoid, group, semiring, etc., and (b) properties of values – in a molecular graph, an atom is assigned to each node; properties of relevant atoms are such additional data.

The terminology in the field of network analysis is not unified. Different application areas use other terms. For example: node – vertex, actor, unit; link – line, tie, edge, connection; etc.

3.3 Types of networks

Besides ordinary (directed, undirected, mixed) networks, some special types of networks are also used:

- *2-mode networks*, bipartite (valued) graphs – networks between two disjoint sets of nodes.
- *multi-relational networks*.
- *linked networks* and *collections of networks*.
- *multilevel networks*.
- *temporal networks*, dynamic graphs – networks changing over time.
- specialized networks: representation of genealogies as *p-graphs*; *Petri's nets*, molecular graphs, etc.

Network (input) file formats should provide the means of expressing all of these types of networks. All interesting data should be recorded (respecting privacy).

In a *two-mode* network $\mathcal{N} = ((\mathcal{U}, \mathcal{V}), \mathcal{L}, \mathcal{P}, \mathcal{W})$ the set of nodes consists of two disjoint sets of nodes $\mathcal{U}$ and $\mathcal{V}$, and all the links from $\mathcal{L}$ have one end node in $\mathcal{U}$ and the other in $\mathcal{V}$.

A *multi-relational network* $\mathcal{N} = (\mathcal{V}, (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k), \mathcal{P}, \mathcal{W})$ contains different relations $\mathcal{L}_i$ (sets of links) over the same set of nodes. Also, the weights from $\mathcal{W}$ are defined on different relations or their union.

In a *linked* or *multimodal* network $\mathcal{N} = ((\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_j), (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k), \mathcal{P}, \mathcal{W})$ the set of nodes $\mathcal{V}$ is partitioned into subsets (*modes*) $\mathcal{V}_i$, $\mathcal{L}_s \subseteq \mathcal{V}_p \times \mathcal{V}_q$, and properties and weights are usually partial functions.

A set of networks $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$ in which each network shares a (sub)set of nodes with some other network is called a *collection* of networks.

A linked network can be transformed into a collection of networks and vice versa.

Bibliographical information is usually represented as a collection of bibliographical networks {**Cite**, **WA**, **WK**, **WC**, **WI**, ...} (W – works, A – authors, K – keywords, C – countries, I – institutions) [7].

Another example of multimodal multirelational networks are knowledge graphs. They can have a very diverse structure (a large number of types of units (modes) and predicates (relations)), which allows for a fairly accurate description of facts from a selected field and solving problems about it. Network analysis methods are particularly useful in analyzing one or a few relational (sub)networks of a knowledge graph.

In a *temporal network*, the presence/activity of a node/link can change through time $\mathcal{T}$. The basic division of temporal network descriptions is into cross-sectional and longitudinal. A cross-sectional description usually consists of a sequence of time slices – ordinary networks that describe the state at a selected moment or time interval. A longitudinal description is based on *temporal quantities* [12, 9] or on a sequence of events.

4 Description of traditional networks

How to describe a network $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W})$? In principle the answer is simple – we list its components $\mathcal{V}$, $\mathcal{L}$, $\mathcal{P}$, and $\mathcal{W}$.

The simplest way is to describe a network $\mathcal{N}$ by providing $(\mathcal{V}, \mathcal{P})$ and $(\mathcal{L}, \mathcal{W})$ in a form of two tables. Both tables are often maintained in some spreadsheet program. They can be exported as text in CSV (Comma Separated Values) format. In large networks, we split a network into some subnetworks – a collection, to avoid the empty cells.

To save space and improve computing efficiency, we often replace values of categorical variables with integers. In R, this

encoding is called a *factorization*. We enumerate all possible values of a given categorical variable (coding table) and afterward replace each value with the corresponding index in the coding table. Since node labels/IDs can be considered a categorical variable, factorization is also usually applied to them. In data analysis, indices start with 1, but real computer scientists start counting from 0. Therefore, it is desirable to include information about the minimal index value in the description.

This approach is used in most programs dealing with large networks. Unfortunately, the coding table is often considered as a kind of metadata and is omitted from the description.

In Pajek [15], node property can be represented in the associated file as a vector (numbers, **.vec**), a partition (nominal, **.clu**), or a permutation (order, **.per**). All network files can be combined into a single project file (**.paj**). Metadata can be added as comments written on lines starting with the **%**. An example of transforming CSV tables into Pajek files is available at GitHub/bavla/netsJSON/example/bib [4].

5 Nets and NetsJSON

We were satisfied with the ”traditional” network description, as implemented in Pajek [15], until we became interested in networks with node/link properties that are not measured in standard scales (ratio, interval, ordinal, nominal), but have structured values (text, subset, interval, distribution, time series, temporal quantity, function, etc.). In topological graph theory, an embedding is described by assigning a rotation to each node [23]. For describing temporal networks, we initially extended the Pajek format, defined and used the Ianus format [12].

For a format supporting structured values, there were two obvious choices for its base – XML and JSON. They are both widely known and suitable as structured data formats. However, JSON can represent the same data as XML more concisely. We chose JSON and in 2015 started developing and using the NetJSON format and the Nets Python package to handle networks with structured-valued properties or weights [5, 4, 3]. On February 26, 2019, the format was renamed to NetsJSON because of the collision with http://netjson.org/rfc.html. NetsJSON has two versions: a *basic* and a *general* version. The current implementation of the Nets library supports only the basic version.

In addition to describing networks with structured values, NetsJSON is expected to offer the capabilities of (most) existing network description formats [13, 25] (archiving, conversion) and provide input data for D3.js visualizations.

A network description in NetsJSON follows the JSON (JavaScript) syntax and consists of five main fields (`netsJSON`, `info`, `nodes`, `links`, `data`).

```
{
"netsJSON": "basic",
"info": {
   "org":1, "nNodes":n, "nArcs":mA, "nEdges":mE,
   "simple":TF, "directed":TF, "multirel":TF, "mode":m,
   "network":fName, "title":title,
   "time": { "Tmin":tm, "Tmax":tM, "Tlabs": {labs} },
   "meta": [events], ...
   },
"nodes": [
   { "id":nodeId, "lab":label, "x":x, "y":y, ... },
   ***
   ],
"links": [
   { "type":arc/edge, "n1":nodeID1, "n2":nodeID2, "rel":r, ... },
   ***
   ],
"data": {
   "data1":description1,
   ***
   }
}
```

where ... are user-defined properties and *** is a sequence of such elements.

The `netsJSON` field identifies the format, the `info` field contains metadata, the `nodes` field contains a table $(\mathcal{V}, \mathcal{P})$ and the `links` field contains a table $(\mathcal{L}, \mathcal{W})$. In recent years, we also analyzed bike systems (link weight is a daily number of trips distribution), bibliographies (yearly distributions of publications or citations), and multiway networks [8, 9, 1]. It turned out that it was necessary to add another main field, `data`, to the basic NetsJSON format, in which we provide additional data about the properties of values (translations of labels in selected languages, algebraic structure, etc. [6]).

An event description can contain the following fields: type, date, title, author, desc, url, cite, copyright, etc. It is intended to provide information about the "life" of the dataset – collection/creation, changes, releases, uses, publications, etc.

For describing temporal networks, a node element and a link element have an additional required property `tq` – a temporal quantity. For example, see `violenceU.json` at `GitHub/bavla/Graph/JSON` describing the Franzosi's violence network.

The general NetsJSON format is also expected to support the description of network collections.

6 Elements of a common network format

Our experience with network analysis to date is summarized in the following recommendations on the elements of a common format for describing networks.

Combining data and its metadata into a single file is a robust approach for ensuring data integrity. A JSON-based format is particularly well-suited for this purpose, as it fits well with the data structures of modern programming languages. JSON also supports Unicode.

We would also encourage the provision, as metadata, of information about the context of the network, additional knowledge about it, articles or notebooks on its analysis, comments of users, etc. Kaggle is a good example. An improved ICON repository or Network Repository (we disagree with their "citation request") could be the way to go. Existing metadata standards should be taken into account (Dublin Core, FAIR, Schema). Data has a "life". When selecting data, its age is often important. Metadata should include at least the collection/creation date and the last modification date.

By FAIR principles, the format should support: **F**indability: Globally unique and persistent identifier, rich metadata. **A**ccessibility: Open, free, and universally implementable standardized communication protocol. **I**nteroperability: Formal, accessible, shared, and proudly applicable language for knowledge representations. **R**eusable: Metadata are richly described and associated with detailed provenance.

The format must support all types of networks (simple, 2-mode, linked, multi-relational, multi-level, temporal). The network can contain both arcs and edges, as well as parallel links. To describe some knowledge graphs, it would be necessary to allow other links to act as the end nodes of a link [18].

As mentioned earlier, using factorization produces a more concise description of the network. In cases where the node names are not too long and are readable, we sometimes want to avoid factorization. This can be achieved by using a switch that indicates whether factorization is used. We can also shorten the description length by introducing default values of selected

properties. If we also allow counting from 0, it makes sense to add information about the smallest index.

Long labels cause problems when printing/visualizing (parts of) networks and results. Therefore, it is useful to have abbreviated versions of labels available. For language-based labels, it is sometimes useful to offer additional versions in selected other languages, which increases the accessibility of the data and the understandability of the results.

Most of the network datasets produced by network science have no node labels. Node labels are not needed if you study distributions, but they are essential in the interpretation of the obtained "important substructures". We would encourage providing node labels, or at least some typological information, in cases where privacy issues arise.

The common format should support descriptions of networks specific to specialized fields of application, such as molecular graphs, genealogies (p-graphs) [29], and topological graph embeddings [23, 24], among others. The format must be extensible. In addition to the agreed-upon fields, the users can add their own, allowing for a comprehensive description of their data.

It is also interesting to ask whether and, if so, how to include descriptions of its displays in the network description. Perhaps it would be worth relying on VEGA-lite [26, 28] and D3.js [14]. Some ideas can also be taken from the section on "defining visualization parameters in the input file" in the Pajek manual / 5.3 [19, p. 89].

Although we are committed to a single-file approach, there may be times when external files are needed (for example, images to display nodes). Consideration should be given to how to support this option. Given the basic purpose of the common format, standard tools (ZIP) can be used to compress large networks.

We have not yet started working on a general format. It is supposed to enable descriptions of collections of networks. The question arises about the scope of validity of IDs – does the same ID in different networks represent the same or other units? This is important for operations such as the union or intersection of networks. Which way to go – introducing contexts or using matchings? Maybe some ideas from the Open Archives Initiative Object Reuse and Exchange (OAI-ORE) and GraphX could be used [22, 17]. An interesting option is the constructive network description – building a network from smaller components [10] or describing a network by its construction sequence [2].

Additional ideas may be found on the page "*A Python Graph API?*" [27]. For now, we would leave aside descriptions of generalizations of networks (multiway networks and hypernets), but we must not forget about them.

The agreed format must be well documented and supported by examples of the use of supported options.

7 Conclusions

Yet another format only makes sense as a project of a larger community of users in the field of network analysis.

Acknowledgements

The computational work reported in this paper was performed using programs R and Pajek [15]. The code and data are available at Github/Bavla [4].

V. Batagelj is supported in part by the Slovenian Research Agency (research program P1-0294 and research project J5-4596) and prepared within the framework of the COST action CA21163 (HiTEc).

T. Pisanski is supported in part by the Slovenian Research Agency (research program P1-0294 and research projects BI-HR/23-24-012, J1-4351, and J5-4596).

N. Bašić is supported in part by the Slovenian Research Agency (research program P1-0294 and research project J5-4596).

References

[1] Vladimir Batagelj. 2024. Cores in multiway networks. *Social Network Analysis and Mining*, 14, 1, 122.

[2] Vladimir Batagelj. 1985. Inductive classes of graphs. In *Proceedings of the 6th Yugoslav Seminar on Graph Theory*, 43–56.

[3] Vladimir Batagelj. 2016. Nets – Python package for network analysis. accessed: 2025-03-18. (2016). https://github.com/bavla/Nets.

[4] Vladimir Batagelj. 2016. NetsJSON – a JSON format for network analysis. accessed: 2025-03-18. (2016). https://github.com/bavla/netsJSON.

[5] Vladimir Batagelj. 2016. Network visualization based on JSON and D3.js. slides. (2016). https://github.com/bavla/netsJSON/blob/master/doc/netVis.pdf.

[6] Vladimir Batagelj. 2021. Semirings in network data analysis / an overview. slides. (2021). https://github.com/bavla/semirings/blob/master/docs/semirings.pdf.

[7] Vladimir Batagelj and Monika Cerinšek. 2013. On bibliographic networks. *Scientometrics*, 96, 3, 845–864.

[8] Vladimir Batagelj and Anuška Ferligoj. 2016. Symbolic network analysis of bike sharing data / Citi Bike. accessed: 2025-03-18. (2016). https://github.com/bavla/Bikes/blob/master/bikes.pdf.

[9] Vladimir Batagelj and Daria Maltseva. 2020. Temporal bibliographic networks. *Journal of Informetrics*, 14, 1, 101006.

[10] Vladimir Batagelj and Andrej Mrvar. 1995. NetML. accessed: 2025-03-18. (1995). https://github.com/bavla/netsJSON/blob/master/doc/snetml.pdf.

[11] Vladimir Batagelj and Andrej Mrvar. 2018. Pajek and pajekxxl. In *Encyclopedia of Social Network Analysis and Mining*. Springer, 1–13.

[12] Vladimir Batagelj and Selena Praprotnik. 2016. An algebraic approach to temporal network analysis based on temporal quantities. *Social Network Analysis and Mining*, 6, 1–22.

[13] Jernej Bodlaj and Monika Cerinšek. 2014, 2017. Network data file formats. In *Encyclopedia of Social Network Analysis and Mining*. Reda Alhajj and Jon Rokne, editors. Springer New York, New York, NY, 1076–1091. ISBN: 978-1-4614-7163-9. DOI: 10.1007/978-1-4614-7163-9_298-1.

[14] Bostock, Mike and Davies, Jason and Heer, Jeffrey and Ogievetsky, Vadim. [n. d.] The JavaScript library for bespoke data visualization. accessed: 2025-08-29. (). https://d3js.org/.

[15] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. 2018. *Exploratory social network analysis with Pajek: Revised and expanded edition for updated software*. Vol. 46. Cambridge university press.

[16] Gephi. 2022. Supported graph formats. accessed: 2025-03-18. (2022). https://gephi.org/users/supported-graph-formats/.

[17] Joseph E Gonzalez, Reynold S Xin, Ankur Dave, Daniel Crankshaw, Michael J Franklin, and Ion Stoica. 2014. {Graphx}: graph processing in a distributed dataflow framework. In *11th USENIX symposium on operating systems design and implementation (OSDI 14)*, 599–613.

[18] Aidan Hogan et al. 2021. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54, 4, 1–37.

[19] Andrej Mrvar and Vladimir Batagelj. 2025. Pajek reference manual. *Version 6.01*. http://mrvar.fdv.uni-lj.si/pajek/pajekman.pdf.

[20] Zachary P Neal and et al. 2024. Recommendations for sharing network data and materials – endorsement page. accessed: 2025-03-18. (2024). https://www.zacharyneal.com/datasharing.

[21] Zachary P Neal et al. 2024. Recommendations for sharing network data and materials. *Network Science*, 12, 4, 404–417.

[22] Open Archives Initiative. 2014. Object Reuse and Exchange (OAI-ORE). 2014-08-14. accessed: 2025-03-18. (2014). https://www.openarchives.org/ore/.

[23] Tomaž Pisanski. 1980. Genus of cartesian products of regular bipartite graphs. *Journal of Graph Theory*, 4, 1, 31–42.

[24] Tomaž Pisanski and Arjana Žitnik. 2004. Representations of graphs and maps. In *26th International Conference on Information Technology Interfaces, 2004.* IEEE, 19–25.

[25] Matthew Roughan and Jonathan Tuke. 2015. Unravelling graph-exchange file formats. *arXiv preprint arXiv:1503.02781*.

[26] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: a grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23, 1, 341–350.

[27] The Python Wiki. 2011. A Python Graph API? accessed: 2025-03-18. (2011). https://wiki.python.org/moin/PythonGraphApi.

[28] University of Washington Interactive Data Lab. [n. d.] Vega-Lite – A Grammar of Interactive Graphics. accessed: 2025-08-29. (). https://vega.github.io/vega-lite/.

[29] Douglas R White, Vladimir Batagelj, and Andrej Mrvar. 1999. Anthropology: analyzing large kinship and marriage networks with pgraph and pajek. *Social Science Computer Review*, 17, 3, 245–274.