NetsJSON

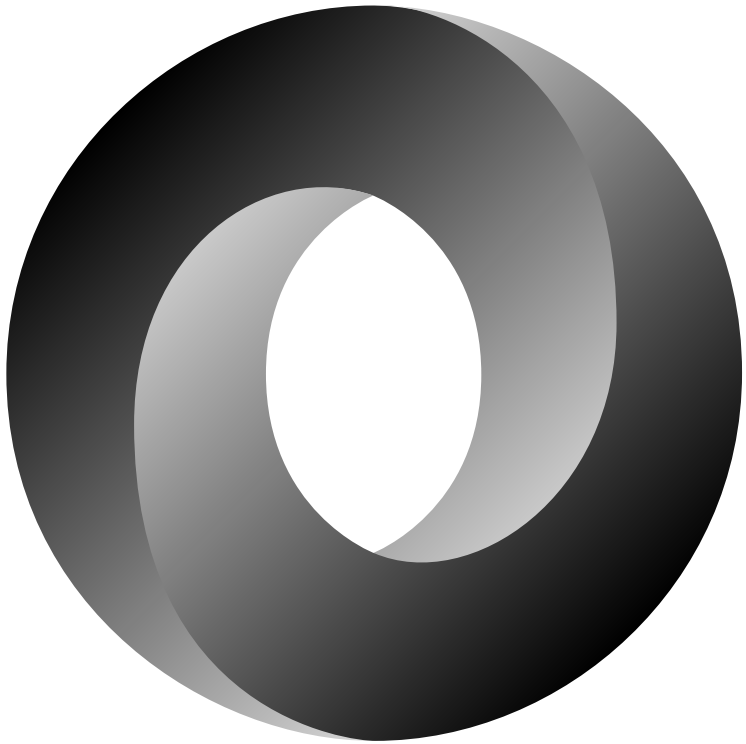
Vladimir Batagelj

IMFM Ljubljana, IAM UP Koper, and NRU HSE Moscow

UP FAMNIT & IAM Computer science seminar and Mathematical research seminar

Koper, April 6 and May 4, 2020

Outline



Vladimir Batagelj: vladimir.batagelj@fmf.uni-lj.si

Version (May 4, 2020, 14 : 39): NetsJSON-UP.pdf

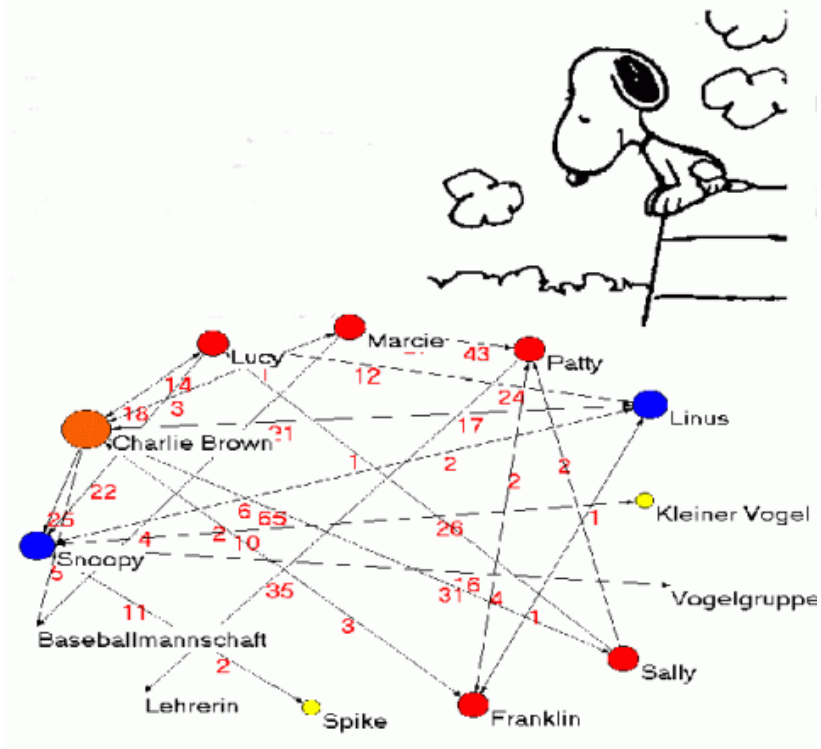
Project

- **NetsJSON**: develop a JSON based format for description of networks. It should be "complete" – it can be used also to describe multi-relational, temporal, two-mode networks, and collections of networks. NetsJSON network description can be extended with a layout information.
- **NetsD3.js**: collect and adapt for NetsJSON selected existing network visualization solutions based on D3.js, and develop new ones.
- **Nets**: provide a support for NetsJSON in the Python library Nets.

NetsJSON could serve as a data exchange format among network analysis programs (conversion program from/to netJSON).

Programers may export their results in NetsJSON and use Nets3D.js for their visualization.

Networks



Alexandra Schuler/ Marion Laging-Glaser:
Analyse von Snoopy Comics

A *network* is based on two sets – set of *nodes* (vertices), that represent the selected *units*, and set of *links* (lines), that represent *ties* between units. They determine a *graph*. A link can be *directed* – an *arc*, or *undirected* – an *edge*.
Additional data about nodes or links can be known – their *properties* (attributes). For example: name/label, type, value, ...

Network = Graph + Data

The data can be measured or computed.

Networks / 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, $\mathcal{A}$ is the set of arcs, $\mathcal{E}$ is the set of edges, and $\mathcal{L} = \mathcal{E} \cup \mathcal{A}$ is the set of links.
  $n = |\mathcal{V}|$, $m = |\mathcal{L}|$
- $\mathcal{P}$ *node value functions* / properties: $p : \mathcal{V} \rightarrow A$
- $\mathcal{W}$ *link value functions* / weights: $w : \mathcal{L} \rightarrow B$

Visual complexity, Icon index, Network Repository

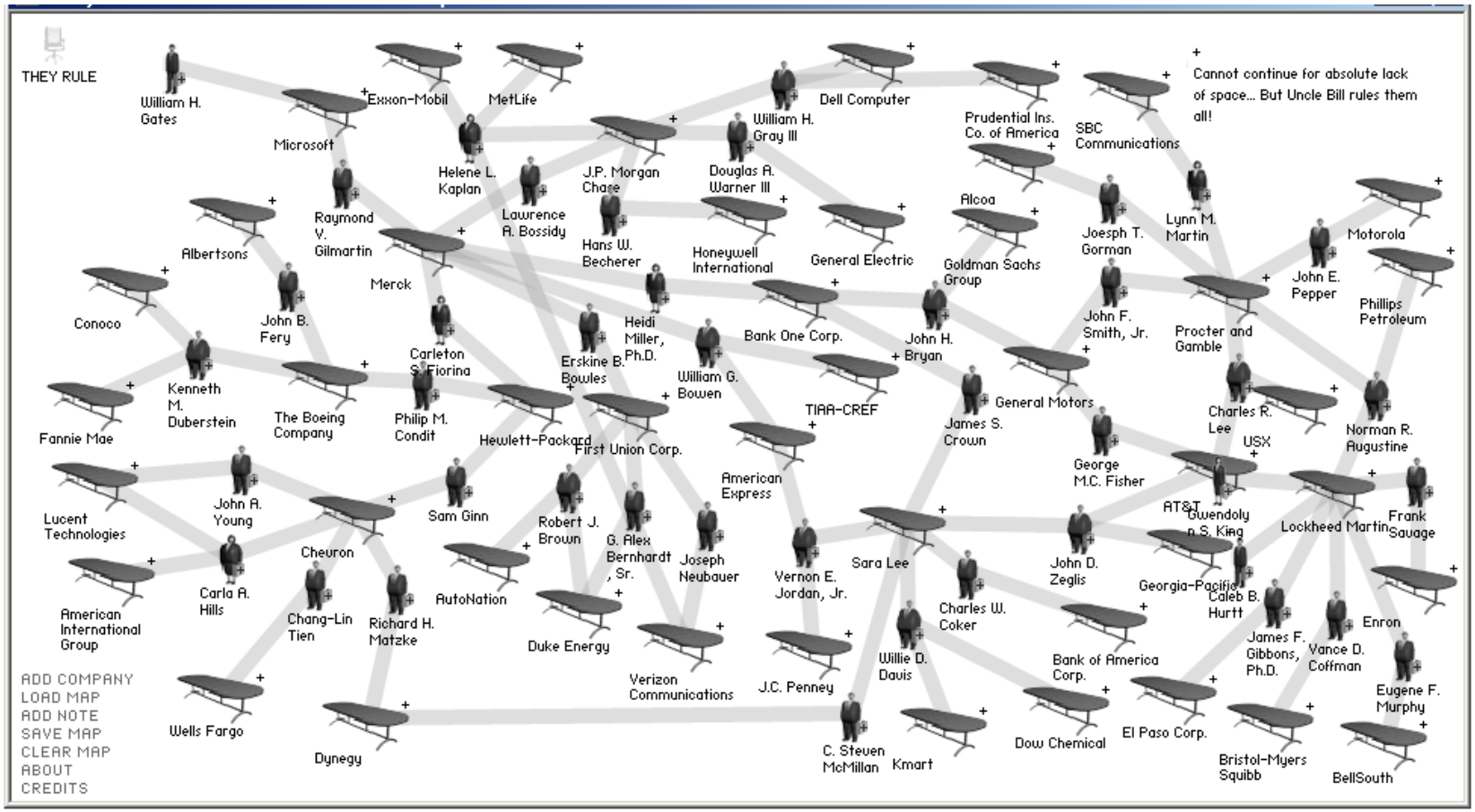
Two-mode networks

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 endnode in $\mathcal{U}$ and the other node in $\mathcal{V}$.

A classical example of two-mode network are the Southern women (Davis 1941).

Names of Participants of Group I	Code Numbers and Dates of Social Events Reported in *Old City Herald*													
	(1) 6/27	(2) 3/2	(3) 4/12	(4) 9/26	(5) 2/25	(6) 5/19	(7) 3/15	(8) 9/16	(9) 4/8	(10) 6/10	(11) 2/23	(12) 4/7	(13) 11/21	(14) 8/3
1. Mrs. Evelyn Jefferson	X	X	X	X	X	X	.	X	X	.	.	.	.	.
2. Miss Laura Mandeville	X	X	X	.	X	X	X	X	.	.	.	.	.	.
3. Miss Theresa Anderson	.	X	X	X	X	X	X	X	X	.	.	.	.	.
4. Miss Brenda Rogers	X	.	X	X	X	X	X	X	.	.	.	.	.	.
5. Miss Charlotte McDowd	.	.	X	X	X	.	X	.	.	.	.	.	.	.
6. Miss Frances Anderson	.	.	X	.	X	X	.	X	.	.	.	.	.	.
7. Miss Eleanor Nye	.	.	.	.	X	X	X	X	.	.	.	.	.	.
8. Miss Pearl Oglethorpe	.	.	.	.	.	X	.	X	X	.	.	.	.	.
9. Miss Ruth DeSand	.	.	.	.	X	.	X	X	X	.	.	.	.	.
10. Miss Verne Sanderson	.	.	.	.	.	.	X	X	X	.	.	X	.	.
11. Miss Myra Liddell	.	.	.	.	.	.	.	X	X	X	.	X	.	.
12. Miss Katherine Rogers	.	.	.	.	.	.	.	X	X	X	.	X	X	X
13. Mrs. Sylvia Avondale	.	.	.	.	.	.	X	X	X	X	.	X	X	X
14. Mrs. Nora Fayette	.	.	.	.	.	X	X	.	X	X	X	X	X	X
15. Mrs. Helen Lloyd	.	.	.	.	.	.	X	X	.	X	X	X	.	.
16. Mrs. Dorothy Murchison	.	.	.	.	.	.	.	X	X	.	.	.	.	.
17. Mrs. Olivia Carleton	.	.	.	.	.	.	.	.	X	.	X	.	.	.
18. Mrs. Flora Price	.	.	.	.	.	.	.	.	X	.	X	.	.	.

They Rule


THEY RULE
William H. Gates
Microsoft
Exxon-Mobil
MetLife
Helene L. Kaplan
Lawrence A. Bossidy
J.P. Morgan Chase
Hans W. Becherer
William H. Gray III
Douglas A. Warner III
Honeywell International
Dell Computer
General Electric
Prudential Ins. Co. of America
SBC Communications
Alcoa
Goldman Sachs Group
Cannot continue for absolute lack of space... But Uncle Bill rules them all!
Lynn M. Martin
Joesph T. Gorman
John F. Smith, Jr.
John H. + Bryan
Bank One Corp.
Heidi Miller, Ph.D.
William G. Bowen
Erskine B. Bowles
TIAA-CREF
James S. Crown
General Motors
Procter and Gamble
John E. Pepper
Motorola
Phillips Petroleum
Charles R. Lee
USX
Norman R. Augustine
George M.C. Fisher
American Express
First Union Corp.
Hewlett-Packard
Albertsons
Raymond V. Gilmartin
Merck
Carleton S. Fiorina
Philip M. Condit
The Boeing Company
John B. Fery
Conoco
Kenneth M. Duberstein
Fannie Mae
Lucent Technologies
John A. Young
Carla A. Hills
American International Group
Cheuron
Chang-Lin Tien
Richard H. Matzke
Sam Ginn
AutoNation
Robert J. Brown
G. Alex Bernhardt, Sr.
Joseph Neubauer
Duke Energy
Verizon Communications
Vernon E. Jordan, Jr.
Sara Lee
Charles W. Coker
Willie D. Davis
J.C. Penney
Bank of America Corp.
John D. Zeglis
AT&T
Gwendoly n S. King
Georgia-Pacific
Caleb B. Hurtt
James F. Gibbons, Ph.D.
Vance D. Coffman
Enron
Eugene F. Murphy
Lockheed Martin
Frank Savage
Dow Chemical
El Paso Corp.
Bristol-Myers Squibb
BellSouth
Kmart
C. Steven McMillan
Wells Fargo
Dynegy
ADD COMPANY
LOAD MAP
ADD NOTE
SAVE MAP
CLEAR MAP
ABOUT
CREDITS

Temporal network

A *temporal network*

$$\mathcal{N}_T = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W}, T)$$

is obtained if the *time* T is attached to an ordinary network. T is a set of *time points* $t \in T$.

In temporal network nodes $v \in \mathcal{V}$ and links $l \in \mathcal{L}$ are not necessarily present or active in all time points. If a link $l(u, v)$ is active in time point t then also its endnodes u and v should be active in time t.

Also the properties of nodes and links can change through time. To describe the changes we introduce the temporal quantities (TQ) [5, 2].

We denote the network consisting of links and nodes active in time $t \in T$ by $\mathcal{N}(t)$ and call it a *time slice* in time point t.

Multirelational networks

In a *multirelational* network

$$\mathcal{N} = (\mathcal{V}, (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k), \mathcal{P}, \mathcal{W})$$

the set of links $\mathcal{L}$ is partitioned into subsets (*relations*) $\mathcal{L}_i$.

Important for encoding textual data according to the S-V-O (*Subject-Verb-Object*) model or its improvements.

Examples: Roberto Franzosi; *KEDS*, *Tabari*, KEDS / Gulf.
This coding can be directly considered as network with *Subjects* ∪ *Objects* as nodes and links (arcs) labeled with *Verbs*.

See also RDF triples in semantic web, SPARQL.

Multi-relational temporal network – KEDS/WEIS

```
% Recoded by WEISmonths, Sun Nov 28 21:57:00 2004
% from http://www.ku.edu/~keds/data.dir/balk.html
*vertices 325
1 "AFG" [1-*]
2 "AFR" [1-*]
3 "ALB" [1-*]
4 "ALBMED" [1-*]
5 "ALG" [1-*]
 ...
318 "YUGGOV" [1-*]
319 "YUGMAC" [1-*]
320 "YUGMED" [1-*]
321 "YUGMTN" [1-*]
322 "YUGSER" [1-*]
323 "ZAI" [1-*]
324 "ZAM" [1-*]
325 "ZIM" [1-*]
*arcs :0 "*** ABANDONED"
*arcs :10 "YIELD"
*arcs :11 "SURRENDER"
*arcs :12 "RETREAT"
 ...
*arcs :223 "MIL ENGAGEMENT"
*arcs :224 "RIOT"
*arcs :225 "ASSASSINATE TORTURE"
*arcs
224: 314 153 1 [4]                 890402  YUG     KSV     224  (RIOT)  RIOT-TORN
212: 314 83 1 [4]                  890404  YUG     ETHALB  212  (ARREST PERSON) ALB ETHNIC JAILED
224: 3 83 1 [4]                    890407  ALB     ETHALB  224  (RIOT)  RIOTS
123: 83 153 1 [4]                  890408  ETHALB  KSV     123  (INVESTIGATE)   PROBING
 ...
42: 105 63 1 [175]                 030731  GER     CYP     042  (ENDORSE)       GAVE SUPPORT
212: 295 35 1 [175]                030731  UNWCT   BOSSER  212  (ARREST PERSON) SENTENCED TO PRIS
43: 306 87 1 [175]                 030731  VAT     EUR     043  (RALLY) RALLIED
13: 295 35 1 [175]                 030731  UNWCT   BOSSER  013  (RETRACT)       CLEARED
121: 295 22 1 [175]                030731  UNWCT   BAL     121  (CRITICIZE)     CHARGES
122: 246 295 1 [175]               030731  SER     UNWCT   122  (DENIGRATE)     TESTIFIED
121: 35 295 1 [175]                030731  BOSSER  UNWCT   121  (CRITICIZE)     ACCUSED
```

Kansas Event Data System *KEDS*

Linked networks

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.

Description of networks using a spreadsheet

How to describe a network $\mathcal{N}$? 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.

As an example, let us describe a part of network determined by the following works:
Generalized blockmodeling, Clustering with relational constraint, Partitioning signed social networks, The Strength of Weak Ties

There are nodes of different types (modes): persons, papers, books, series, journals, publishers; and different relations among them: author_of, editor_of, contained_in, cites, published_by.

Both tables are often maintained in Excel. They can be exported as text in CSV (Comma Separated Values) format.

bibNodes.csv

```
name;mode;country;sex;year;vol;num;fPage;lPage;x;y
"Batagelj, Vladimir";person;SI;m;;;;;809.1;653.7
"Doreian, Patrick";person;US;m;;;;;358.5;679.1
"Ferligoj, Anuška";person;SI;f;;;;;619.5;680.7
"Granovetter, Mark";person;US;m;;;;;145.6;660.5
"Moustaki, Irini";person;UK;f;;;;;783.0;228.0
"Mrvar, Andrej";person;SI;m;;;;;478.0;630.1
"Clustering with relational constraint";paper;;;1982;47;;413;426;684.1;3
"The Strength of Weak Ties";paper;;;1973;78;6;1360;1380;111.3;329.4
"Partitioning signed social networks";paper;;;2009;31;1;1;11;408.0;337.8
"Generalized Blockmodeling";book;;;2005;24;;1;385;533.0;445.9
"Psychometrika";journal;;;;;;;;741.8;086.1
"Social Networks";journal;;;;;;;;321.4;236.5
"The American Journal of Sociology";journal;;;;;;;;111.3;168.9
"Structural Analysis in the Social Sciences";series;;;;;;;;310.4;082.8
"Cambridge University Press";publisher;UK;;;;;;534.3;238.2
"Springer";publisher;US;;;;;;884.6;174.0
```

bibNodes.csv

In large networks, to avoid the empty cells, we split a network to some subnetworks – a collection.

bibLinks.csv

```
from;relation;to
"Batagelj, Vladimir";authorOf;"Generalized Blockmodeling"
"Doreian, Patrick";authorOf;"Generalized Blockmodeling"
"Ferligoj, Anuška";authorOf;"Generalized Blockmodeling"
"Batagelj, Vladimir";authorOf;"Clustering with relational constraint"
"Ferligoj, Anuška";authorOf;"Clustering with relational constraint"
"Granovetter, Mark";authorOf;"The Strength of Weak Ties"
"Granovetter, Mark";editorOf;"Structural Analysis in the Social Sciences
"Doreian, Patrick";authorOf;"Partitioning signed social networks"
"Mrvar, Andrej";authorOf;"Partitioning signed social networks"
"Moustaki, Irini";editorOf;"Psychometrika"
"Doreian, Patrick";editorOf;"Social Networks"
"Generalized Blockmodeling";containedIn;"Structural Analysis in the Soci
"Clustering with relational constraint";containedIn;"Psychometrika"
"The Strength of Weak Ties";containedIn;"The American Journal of Sociolo
"Partitioning signed social networks";containedIn;"Social Networks"
"Partitioning signed social networks";cites;"Generalized Blockmodeling"
"Generalized Blockmodeling";cites;"Clustering with relational constraint
"Structural Analysis in the Social Sciences";publishedBy;"Cambridge Univ
"Psychometrika";publishedBy;"Springer"
```

bibLinks.csv

Factorization and description of large networks

To save space and improve the 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 afterwards replace each its value by the corresponding index in the coding table.

This approach is used in most programs dealing with large networks. Unfortunately the coding table is often a kind of meta-data.

CSV2Pajek.R

```
# transforming CSV file to Pajek files
# by Vladimir Batagelj, June 2016
# setwd("C:/Users/batagelj/work/Python/graph/SVG/EUSN")
# colC <- c(rep("character",4),rep("numeric",7)); nas=c("","NA","NaN")
colC <- c(rep("character",4),rep("numeric",5)); nas=c("","NA","NaN")
nodes <- read.csv2("bibNodes.csv",encoding='UTF-8',colClasses=colC,na.strings=nas)
n <- nrow(nodes); M <- factor(nodes$mode); S <- factor(nodes$sex)
mod <- levels(M); sx <- levels(S); S <- as.numeric(S); S[is.na(S)] <- 0
links <- read.csv2("bibLinks.csv",encoding='UTF-8',colClasses="character")
F <- factor(links$from,levels=nodes$name,ordered=TRUE)
T <- factor(links$to,levels=nodes$name,ordered=TRUE)
R <- factor(links$relation); rel <- levels(R)
net <- file("bib.net","w"); cat('*vertices ',n,'\n',file=net)
clu <- file("bibMode.clu","w"); sex <- file("bibSex.clu","w")
cat('%',file=clu); cat('%',file=sex)
for(i in 1:length(mod)) cat(' ',i,mod[i],file=clu)
cat('\n*vertices ',n,'\n',file=clu)
for(i in 1:length(sx)) cat(' ',i,sx[i],file=sex)
cat('\n*vertices ',n,'\n',file=sex)
for(v in 1:n) {
  cat(v,' "',nodes$name[v],'"\n',sep='',file=net);
  cat(M[v],'\n',file=clu); cat(S[v],'\n',file=sex)
}
for(r in 1:length(rel)) cat('*arcs :',r,' "',rel[r],'"\n',sep='',file=net)
cat('*arcs\n',file=net)
for(a in 1:nrow(links))
  cat(R[a],': ',F[a],' ',T[a],' 1 l "',rel[R[a]],'"\n',sep='',file=net)
close(net); close(clu); close(sex)
```

CSV2Pajek.R

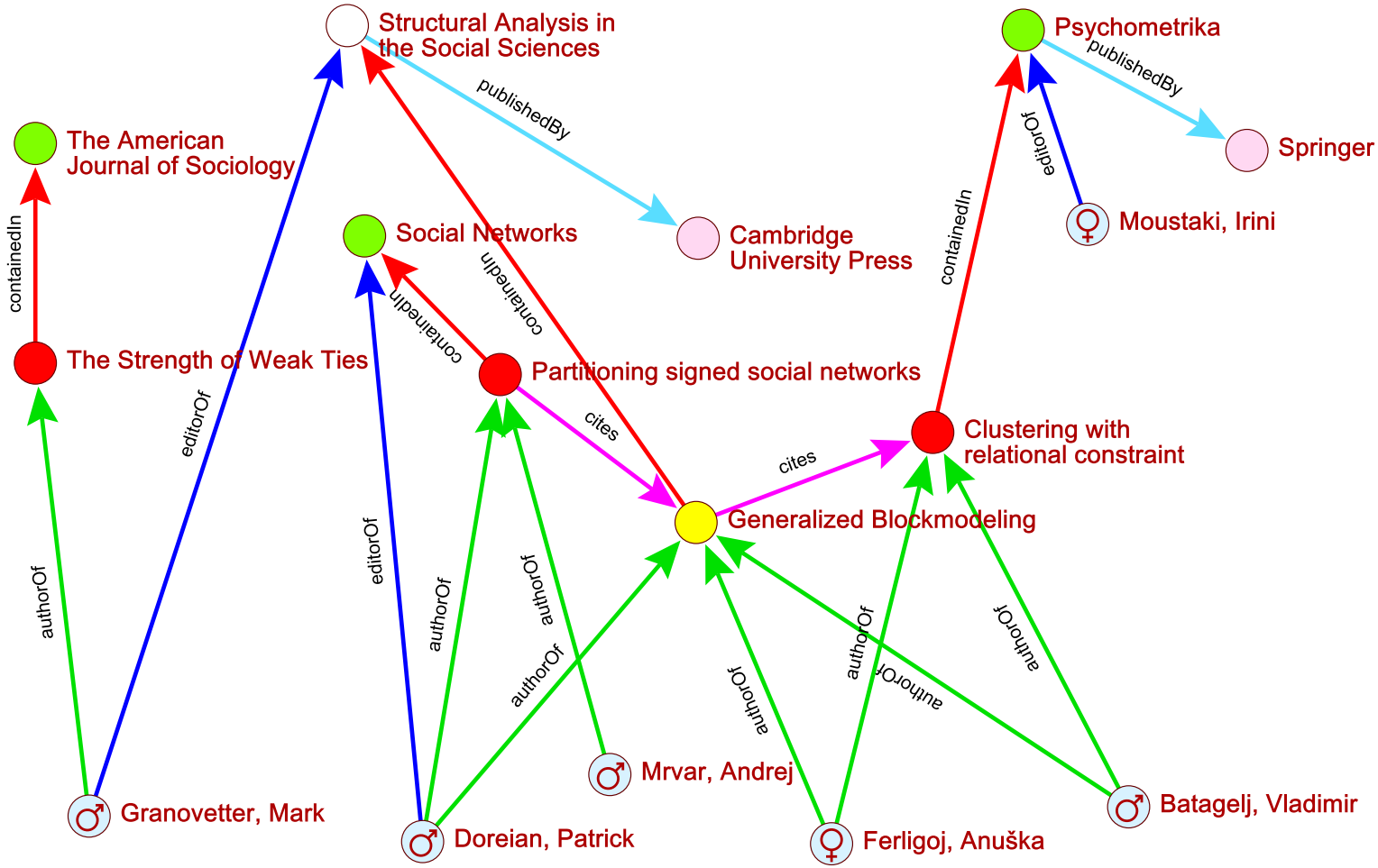
bib.net

```
*vertices  16
1 "Batagelj, Vladimir"
2 "Doreian, Patrick"
3 "Ferligoj, Anuška"
4 "Granovetter, Mark"
5 "Moustaki, Irini"
6 "Mrvar, Andrej"
7 "Clustering with relational constraint"
8 "The Strength of Weak Ties"
9 "Partitioning signed social networks"
10 "Generalized Blockmodeling"
11 "Psychometrika"
12 "Social Networks"
13 "The American Journal of Sociology"
14 "Structural Analysis in the Social Sciences"
15 "Cambridge University Press"
16 "Springer"
*arcs :1 "authorOf"
*arcs :2 "cites"
*arcs :3 "containedIn"
*arcs :4 "editorOf"
*arcs :5 "publishedBy"
```

```
*arcs
1: 1 10 1 l "authorOf"
1: 2 10 1 l "authorOf"
1: 3 10 1 l "authorOf"
1: 1 7 1 l "authorOf"
1: 3 7 1 l "authorOf"
1: 4 8 1 l "authorOf"
4: 4 14 1 l "editorOf"
1: 2 9 1 l "authorOf"
1: 6 9 1 l "authorOf"
4: 5 11 1 l "editorOf"
4: 2 12 1 l "editorOf"
3: 10 14 1 l "containedIn"
3: 7 11 1 l "containedIn"
3: 8 13 1 l "containedIn"
3: 9 12 1 l "containedIn"
2: 9 10 1 l "cites"
2: 10 7 1 l "cites"
5: 14 15 1 l "publishedBy"
5: 11 16 1 l "publishedBy"
```

bib.net, bibMode.clu, bibSex.clu; bib.paj, bib.ini.

Bibliographic network – picture / Pajek



Nets and NetsJSON

For dealing with networks with properties with structured values (for example, temporal quantities) we are developing a Python package Nets [3].

For describing temporal networks we initially, extending Pajek format, defined and used the Ianus format.

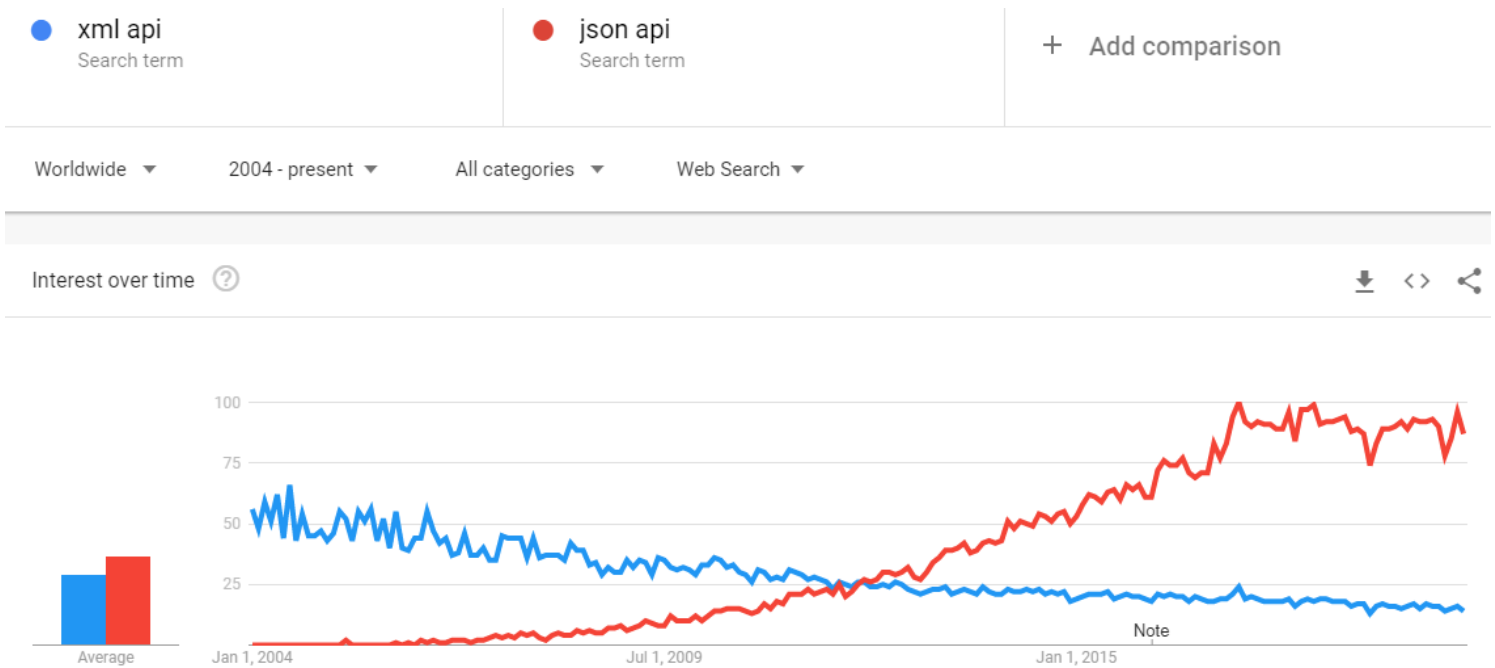
In 2015 we started to develop a new format based on JSON – we named it netJSON. On February 26, 2019 the format was renamed to NetsJSON because of the collision with http://netjson.org/rfc.html.

NetsJSON has two versions: a *basic* and a *general* version. Current implementation of the Nets / TQ library supports only the basic version.

Besides for a *description* of networks with structured values, NetsJSON should *envelope* (most of) existing network description formats [6] (archiving, conversion) and provide input data for D3.js *visualization*s.

XML api – JSON api

In near past, for description of structured data the XML (Extensible Markup Language) was mostly used. In last years a JSON format started to replace it. Google trends (March 2020)



JSON

JSON (JavaScript Object Notation) is a text data format that preserves the structure of data objects. It is "compatible" with basic data structures in modern programming languages.

The initial version of JSON was developed by Douglas Crockford (around 2002). He based it on the Javascript notation. The principal idea is: if we apply on a string (sequence of characters) containing a description of a data object, the Javascript function `eval` we get as its result the corresponding data object. JSON is a programming language independent, open code standard for exchange of data among programs.

Two JSON standards exist:

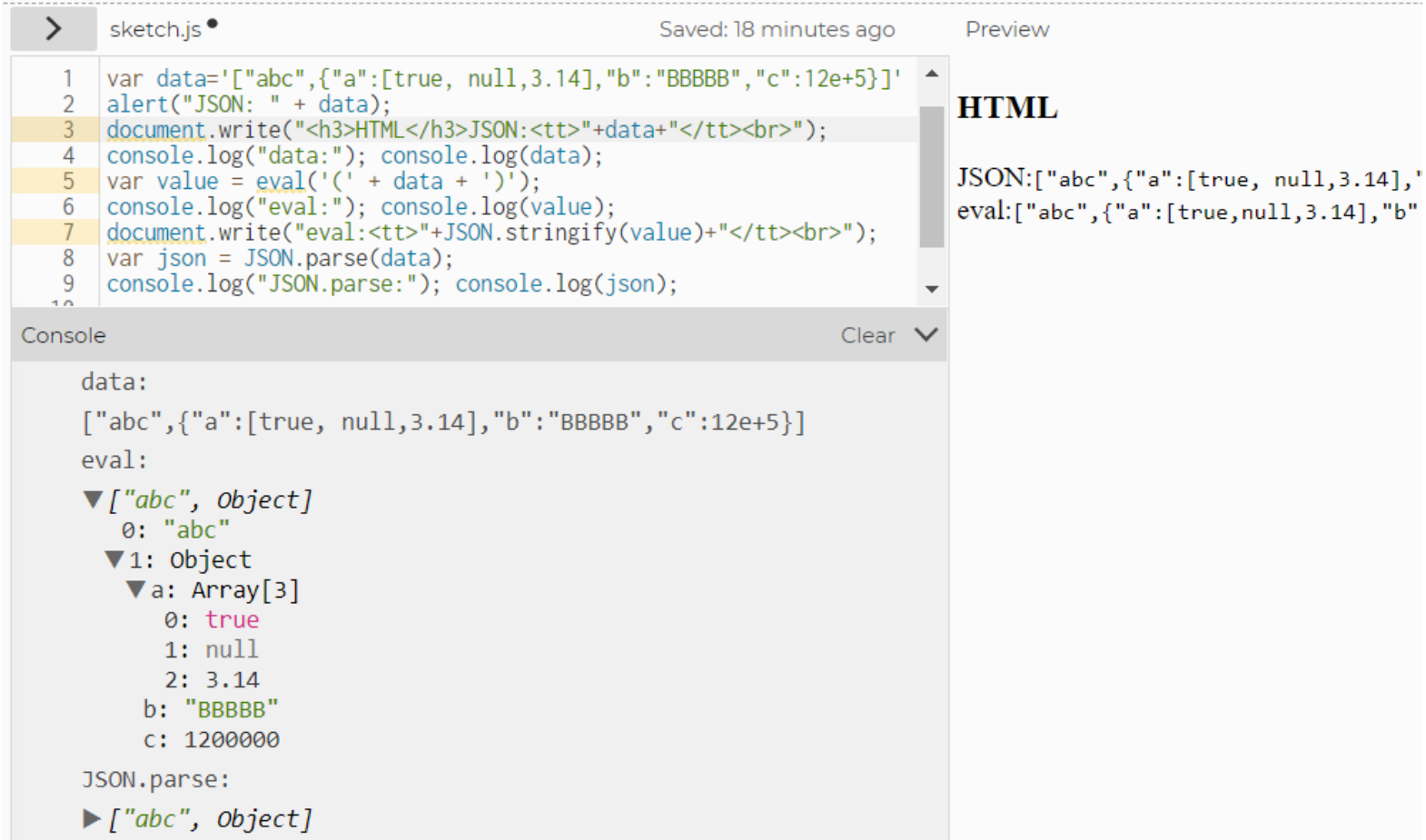
- The JSON Data Interchange Format. Standard ECMA-404, October 2013.
- The JavaScript Object Notation (JSON) Data Interchange Format Request for Comments: 7159, March 2014.

JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Wikipedia

JSON and JavaScript eval



p5.js editor or PLAYCODE - Online JavaScript Editor

JSON

XML is appropriate for describing the structure of textual data, JSON is becoming the first choice for describing structured data.

JSON has much simpler grammar, is more readable and compatible with basic data structures in modern programming languages.

All keys (names of fields) are in double quotes.

JSON files are by default based on the encoding Unicode (UTF-8).

The MIME type for JSON files is `application/json`, the recommended file extension is `.json`.

For work with JSON there exists supporting libraries for all important programming languages http://www.json.org/.

CIA - the world factbook: data in JSON

JSON grammar

value
- *object*
- *array*
- *string*
- *number*
- `true`
- `false`
- `null`

object
- `{}`
- `{` *members* `}`

members
- *pair*
- *pair* , *members*

pair
- *string* : *value*

array
- `[]`
- `[` *elements* `]`

elements
- *value*
- *value* , *elements*

string
- `""`
- `"` *chars* `"`

chars
- *char*
- *char chars*

char
- *any-Unicode-character-except-"-or-\-or-control-character*
- `\"`
- `\\`
- `\/`
- `\b`
- `\f`
- `\n`
- `\r`
- `\t`
- `\u` *four-hex-digits*

number
- *int*
- *int frac*
- *int exp*
- *int frac exp*

int
- *digit*
- *digit1-9 digits*
- *- digit*
- *- digit1-9 digits*

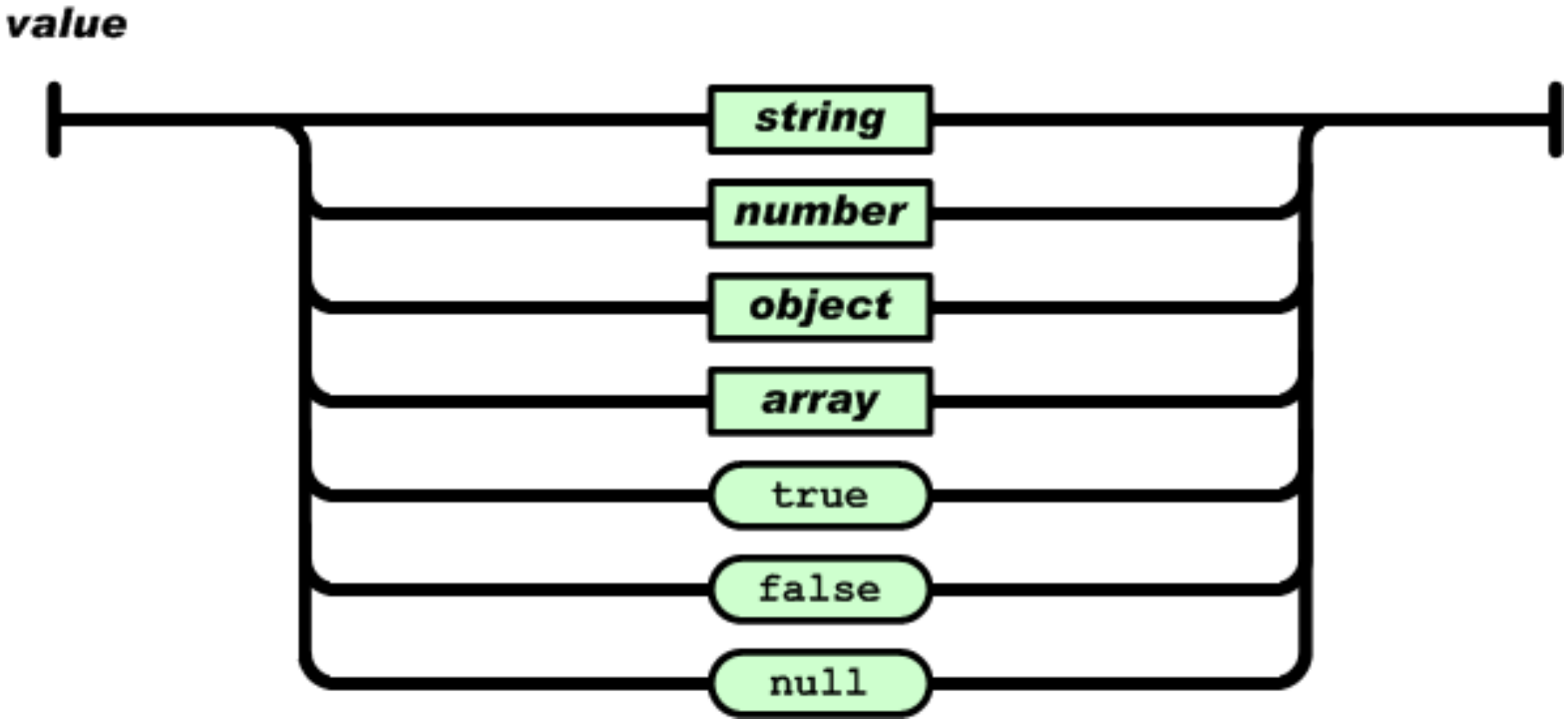
frac
- *. digits*

exp
- *e digits*

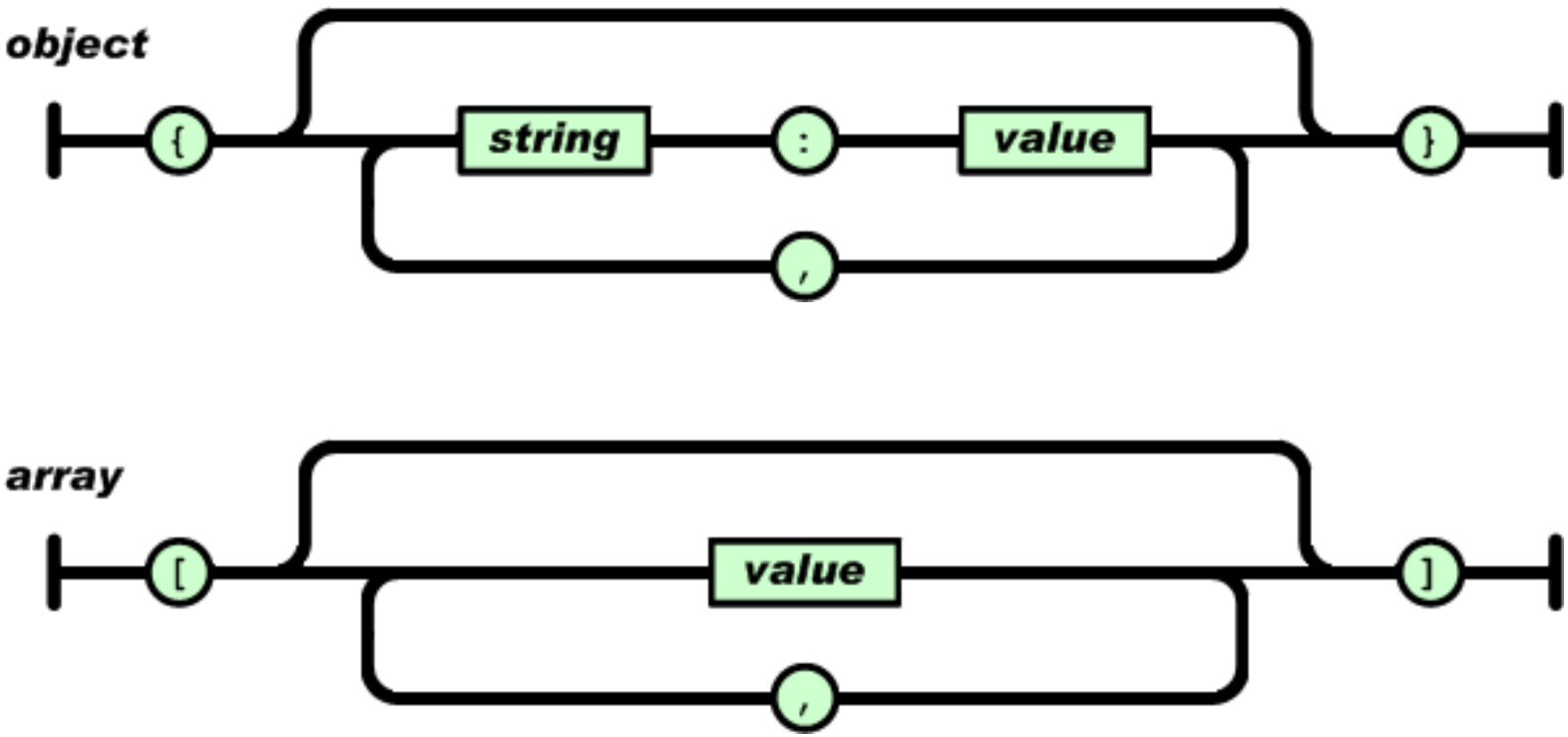
digits
- *digit*
- *digit digits*

e
- `e`
- `e+`
- `e-`
- `E`
- `E+`
- `E-`

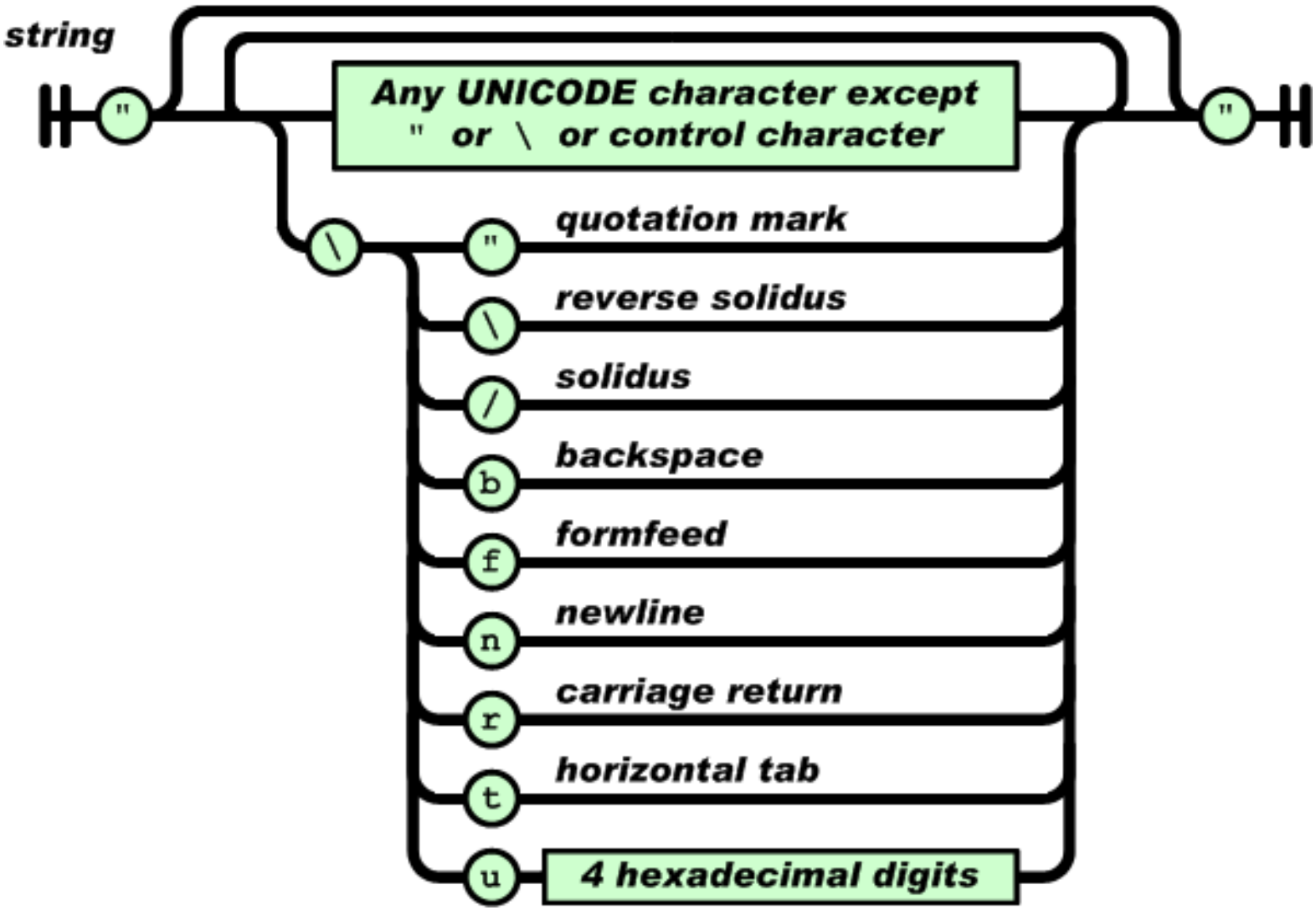
JSON grammar



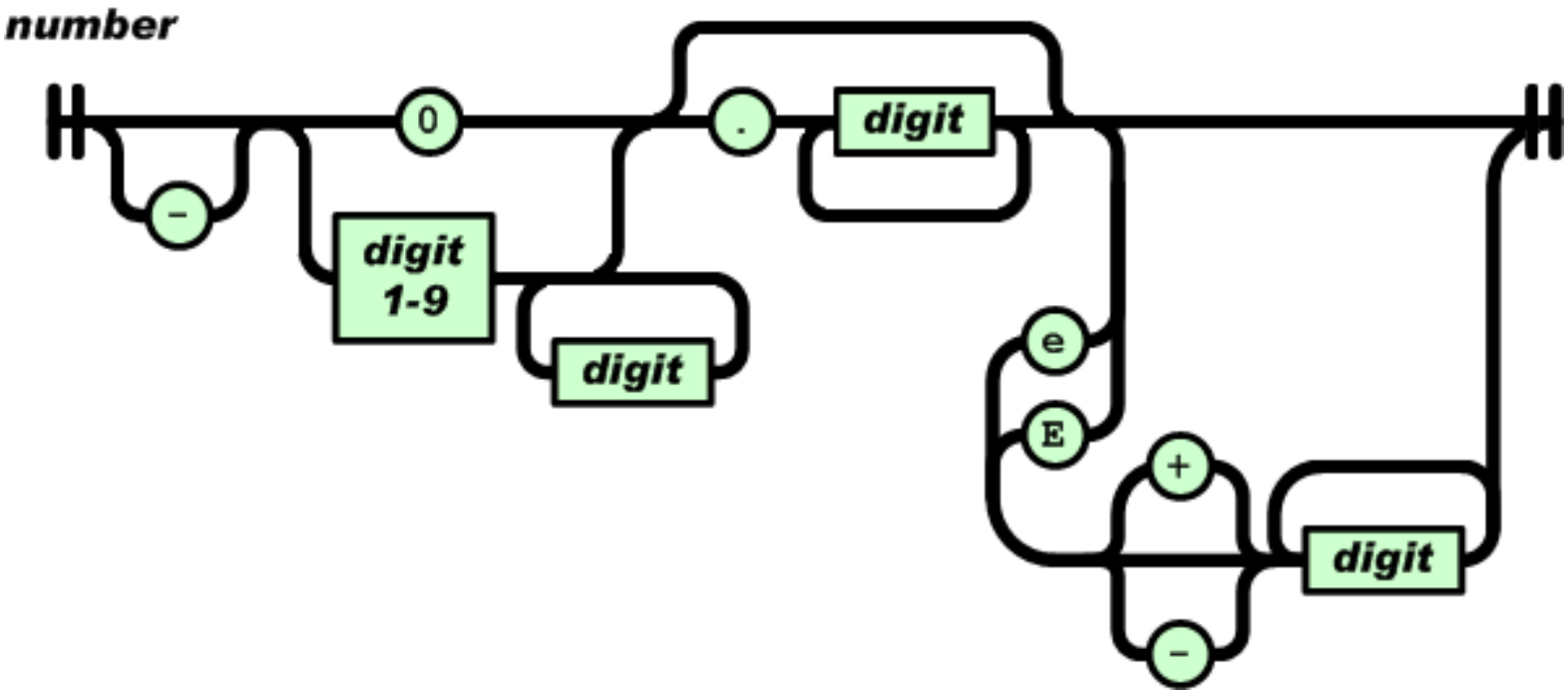
JSON grammar



JSON grammar



JSON grammar



Well formed and valid JSON files

A JSON file is *well formed* iff it respects JSON's grammar. Is my file well formed? service. JSONlint - another checker. We can inspect it using a web browser!!!

JSON editor

Similar to XML's DTD files or schema, we can impose additional restrictions to the structure of JSON files describing special types of data using JSON schema – the JSON files respecting these additional restrictions are called *valid*.

Github, validation, JSON Schema Lint, JSON Schema validator.
R packages jsonlite, rjson and RJSONIO. jsonlite Quick start

JSON in R - jsonlite

```
> wdir <- 'D:/vlado/work/python/graph/JSON/test'; setwd(wdir)
> library(jsonlite)
> (J <- fromJSON(readLines("john.json")))
$firstName
[1] "John"

$lastName
[1] "Smith"

...

$phoneNumbers
    type        number
1   home 212 555-1234
2 office 646 555-4567

$children
list()

$spouse
named list()
> (john <- toJSON(J,auto_unbox=TRUE))
{"firstName":"John","lastName":"Smith","isAlive":true,"age":25,"address"
"city":"New York","state":"NY","postalCode":"10021-3100"},"phoneNumbers"
"number":"212 555-1234"},{"type":"office","number":"646 555-4567"}],"chi
> js <- file("john2.json",encoding="UTF-8")
> write(john,file=js); close(js)
```

JSON in Python - json


```
 9    wdir = 'D:/vlado/work/python/graph/JSON/test'
10    import os; os.chdir(wdir)
11    import json
12    with open('john.json') as jsi: J = json.load(jsi)
13    print(J['phoneNumbers'][1]['number'])
14    with open('johnPy.json', 'w') as jso: json.dump(J, jso)
15    Js = json.dumps(J)
16    print(Js)
17    Jv = json.loads(Js)
```

```
In [14]: json.loads(Js)
Out[14]:
{'firstName': 'John',
 'lastName': 'Smith',
 'isAlive': True,
 'age': 25,
 'address': {'streetAddress': '21 2nd Street',
  'city': 'New York',
  'state': 'NY',
  'postalCode': '10021-3100'},
 'phoneNumbers': [{'type': 'home', 'number': '212 555-1234'},
  {'type': 'office', 'number': '646 555-4567'}],
 'children': [],
 'spouse': {}}
```

NA, NaN, inf, null in R

Not specified in the standards.

```
> (v <- list(a=c(3,7),b=NULL,c=c(NA,NaN,Inf,-Inf)))
$a
[1] 3 7
$b
NULL
$c
[1]   NA  NaN  Inf -Inf
> (s <- jsonlite::toJSON(v))
{"a":[3,7],"b":{},"c":["NA","NaN","Inf","-Inf"]}
> (r <- rjson::toJSON(v))
[1] "{\"a\":[3,7],\"b\":null,\"c\":[\"NA\",\"NaN\",\"Inf\",\"-Inf\"]}"
> cat(r,"\n")
{"a":[3,7],"b":null,"c":["NA","NaN","Inf","-Inf"]}
> (t <- jsonlite::fromJSON(r))
$a
[1] 3 7
$b
NULL
$c
[1]   NA  NaN  Inf -Inf
> (p <- rjson::fromJSON(s))
$a
[1] 3 7
$b
list()
$c
[1] "NA"   "NaN"  "Inf"  "-Inf"
```

NA, NaN, inf, null in Python

Recent versions of the Python library json, default allow_nan = True, encode the float values (nan, inf, -inf) with their JavaScript equivalents (NaN, Infinity, -Infinity).

```
In [26]: v = [None, float('nan'), float('inf'), float('-inf')]

In [27]: v
Out[27]: [None, nan, inf, -inf]

In [28]: s = json.dumps(v)

In [29]: s
Out[29]: '[null, NaN, Infinity, -Infinity]'

In [30]: w = json.loads(s)

In [31]: w
Out[31]: [None, nan, inf, -inf]
```

Duplicate names

From Python json documentation:

Repeated Names Within an Object

The RFC specifies that the names within a JSON object should be unique, but does not mandate how repeated names in JSON objects should be handled. By default, this module does not raise an exception; instead, it ignores all but the last name-value pair for a given name:

```
>>> weird_json = '{"x": 1, "x": 2, "x": 3}'
>>> json.loads(weird_json)
{'x': 3}
```

The *object_pairs_hook* parameter can be used to alter this behavior.

duplicate keys; ECMA.
Python: json; W3.

Informal description of the basic NetsJSON format


```
{
"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, ..
   ***
   ]
}
```

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

Basic NetsJSON format

An event description can contain fields:

```
{  "date": date,
   "title": short description,
   "author": name,
   "desc": long description,
   "url": URL,
   "cite": reference,
   "copy": copyright
}
```

for describing temporal networks a node element and a link element has an additional required property `tq`

Example 1, Franzosi's violence network / UTF-8 no sig

Simple example (*vertices, *arcs).

```
setwd("C:/Users/Batagelj/test/python/2012/amazon")
library(jsonlite)

net2json <- function(netF,cluF,jsonF){
  net <- file(netF,"r");  clu <- file(cluF,"r")
  b <- unlist(strsplit(readLines(net,n=1)," "))
  n <- as.integer(b[length(b)])
  N <- readLines(net,n=n);  nam <- character(n)
  for(i in 1:n) nam[i] <- unlist(strsplit(N[i],'"'))[2]
  skip <- readLines(clu,n=1);  C <- as.integer(readLines(clu,n=n))
  skip <- readLines(net,n=1);  L <- readLines(net,n=-1)
  M <- matrix(as.integer(unlist(strsplit(sub('^\\s+','',L),'\\s+'))),ncol=3,byrow=TRUE)
  nods <- vector('list',n)
  for(i in 1:n) nods[[i]] <- list(id=i,lab=nam[i],group=C[i])
  m <- nrow(M); lnks <- vector('list',m)
  for(i in 1:m) lnks[[i]] <- list(type="arc",n1=M[i,1],n2=M[i,2],w=M[i,3])
  inf <- list(network="simple",org=1,nNodes=n,nArcs=m,title="Simple example")
  data <- list(NetsJSON="basic",info=inf,nodes=nods,links=lnks)
  jstr <- toJSON(data)
  json <- file(jsonF,"w");  cat(jstr,file=json)
  close(json);  close(net); close(clu)
}

net2json("islands.net","islands.clu","islands.json")
```

islands, island 1, island 4, force: islands

CSV2JSON.R

```
# transforming CSV files to JSON file
# by Vladimir Batagelj, June 2016
setwd("C:/Users/batagelj/work/Python/graph/SVG/EUSN")
library(rjson)
colC <- c(rep("character",4),rep("numeric",5)); nas <- c("","NA","NaN")
nodes <- read.csv2("bibNodesXY.csv",encoding='UTF-8',colClasses=colC,na.strings=nas)
M <- factor(nodes$mode); mod <- levels(M); M <- as.numeric(M)
S <- factor(nodes$sex); sx <- levels(S); S <- as.numeric(S); S[is.na(S)] <- 0
links <- read.csv2("bibLinks.csv",encoding='UTF-8',colClasses="character")
F <- as.numeric(factor(links$from,levels=nodes$name,ordered=TRUE))
T <- as.numeric(factor(links$to,levels=nodes$name,ordered=TRUE))
R <- factor(links$relation); rel <-levels(R); R <- as.numeric(R)
n <- nrow(nodes); nods <- vector('list',n)
for(i in 1:n) nods[[i]] <- list(id=i,name=nodes$name[i],mode=M[i],
   sex=S[i],x=as.numeric(nodes$x[i])/1000,y=as.numeric(nodes$y[i])/1000)
m <- nrow(links); lnks <- vector('list',m)
for(i in 1:m) lnks[[i]] <- list(type="arc",n1=F[i],n2=T[i],rel=R[i],w=1)
meta <- list(date="June 11,2016",author="Vladimir Batagelj")
leg <- list(mode=mod,sex=sx,rel=rel)
inf <- list(network="bib",org=1,nNodes=n,nArcs=m,
  title="Example for EUSN'16",legend=leg,meta=meta)
data <- list(NetsJSON="basic",info=inf,nodes=nods,links=lnks)
json <- file("bib.json","w"); cat(toJSON(data),file=json); close(json)
```

bib.json


```
{"NetsJSON":"basic",
 "info":{"network":"bib","org":1,"nNodes":16,"nArcs":19,"title":"Example for EUSN'16",
  "legend":{
   "mode":["book","journal","paper","person","publisher","series"],
   "sex":["f","m"],
   "rel":["authorOf","cites","containedIn","editorOf","publishedBy"]},
  "meta":{"date":"June 11,2016","author":"Vladimir Batagelj"}},
 "nodes":[
  {"id":1,"lab":"Batagelj, Vladimir","mode":4,"sex":2,"x":0.8091,"y":0.6537},
  {"id":2,"lab":"Doreian, Patrick","mode":4,"sex":2,"x":0.3585,"y":0.6791},
  {"id":3,"lab":"Ferligoj, Anu\u0161ka","mode":4,"sex":1,"x":0.6195,"y":0.6807},
  {"id":4,"lab":"Granovetter, Mark","mode":4,"sex":2,"x":0.1456,"y":0.6605},
  {"id":5,"lab":"Moustaki, Irini","mode":4,"sex":1,"x":0.783,"y":0.228},
  {"id":6,"lab":"Mrvar, Andrej","mode":4,"sex":2,"x":0.478,"y":0.6301},
  {"id":7,"lab":"Clustering with relational constraint","mode":3,"sex":0,"x":0.6841,"y":0.3801},
...
  {"id":15,"lab":"Cambridge University Press","mode":5,"sex":0,"x":0.5343,"y":0.2382},
  {"id":16,"lab":"Springer","mode":5,"sex":0,"x":0.8846,"y":0.174}],
 "links":[
  {"type":"arc","n1":1,"n2":10,"rel":1,"w":1},
  {"type":"arc","n1":2,"n2":10,"rel":1,"w":1},
  {"type":"arc","n1":3,"n2":10,"rel":1,"w":1},
  {"type":"arc","n1":1,"n2":7,"rel":1,"w":1},
  {"type":"arc","n1":3,"n2":7,"rel":1,"w":1},
  {"type":"arc","n1":4,"n2":8,"rel":1,"w":1},
  {"type":"arc","n1":4,"n2":14,"rel":4,"w":1},
...
  {"type":"arc","n1":10,"n2":7,"rel":2,"w":1},
  {"type":"arc","n1":14,"n2":15,"rel":5,"w":1},
  {"type":"arc","n1":11,"n2":16,"rel":5,"w":1}
]}
```


bib.json, picture: bib

Temporal networks

Temporal networks are described in our papers [5] and [2].

They are supported in Python by libraries TQ and Nets.

Python: Set Up; Temporal analysis of authors.

D3.js

There exists an excellent library D3.js for interactive data visualization on the web (and locally) in SVG format. Most of the network data for D3.js are prepared in the JSON format. Many nice D3.js based network visualization solutions were developed:

- Force: Force-Directed Graph, Force Layout & Matrix Market Format, 3D Force Layout; An A to Z of extra features for the d3 force layout
- Directed: Directed Graph Editor, Directed Edges (Curves and Arrow Markers), Mobile Patent Suits
- Other: Co-occurrence Matrix, Hive Plots, Chord Diagram, Hierarchical Edge Bundling
- Applications: Linked JAZZ, Ontology Visualization, Visualizing Package Dependencies, Connectome explorer for the "brain" of C. elegans, Gene functional interaction networks
- More: D3 gallery, The Big List of D3.js Examples - Christophe Viau, Over 2000 D3.js Examples and Demos

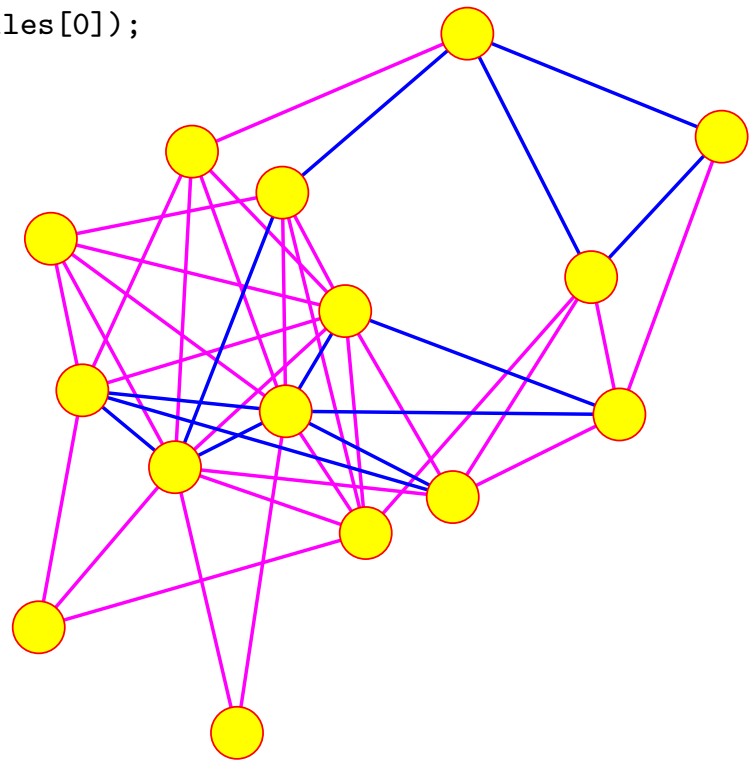
Reading NetsJSON files and displaying a network with given nodes' coordinates


```
<!DOCTYPE html>
<head>
<meta charset="utf-8">
<script src="http://d3js.org/d3.v3.min.js"></script>
</head>
<body>
<input type='file' accept='application/json' onchange='openFile(event)'>
<script>
function process(graph) {
// set up the drawing area
var width = 500,
  height = 500; s = graph.attributes.org;
var svg = d3.select("body").append("svg")
  .attr("width", width)
  .attr("height", height)
  .attr("xmlns", "http://www.w3.org/2000/svg");
// draw the links
var link = svg.selectAll("line")
  .data(graph.links).enter().append("line")
  .style("stroke", function(d,i) {return((d.type=="arc" ? "magenta" : "blue"))})
  .attr("stroke-width", 2)
  .attr("x1", function(d) {return(graph.nodes[d.n1-s].x*width);})
  .attr("y1", function(d) {return(graph.nodes[d.n1-s].y*height);})
  .attr("x2", function(d) {return(graph.nodes[d.n2-s].x*width);})
  .attr("y2", function(d) {return(graph.nodes[d.n2-s].y*height);});
// draw the nodes
var node = svg.selectAll("circle")
  .data(graph.nodes).enter().append("circle")
  .attr("r", 15)
  .attr("cx", function(d,i) {return(d.x*width);})
  .attr("cy", function(d,i) {return(d.y*height);})
  .attr("fill", "yellow")
  .attr("stroke", "red");
}
```

. . . reading

adapted from Matt West Reading Files Using The HTML5 FileReader API

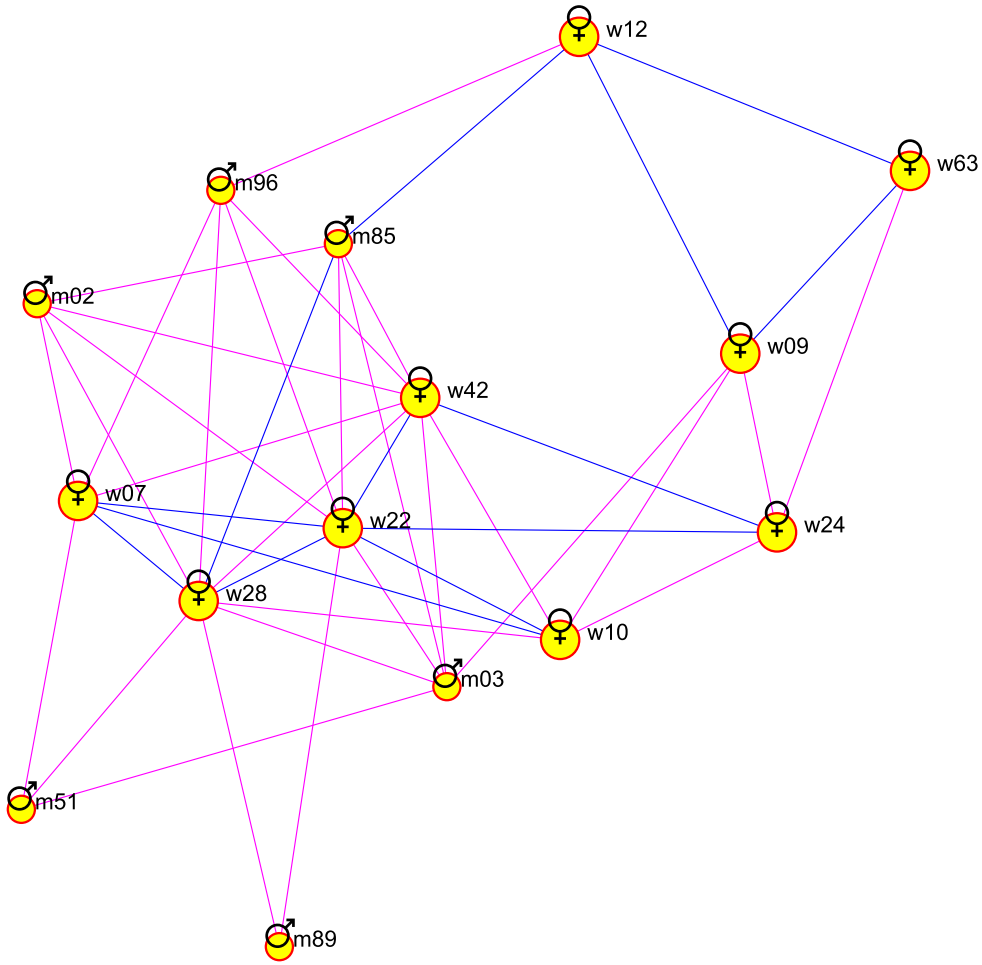
```
var openFile = function(event) {
  var input = event.target;
  var reader = new FileReader();
  reader.onload = function(){
    process(JSON.parse(reader.result));
  };
  reader.readAsText(input.files[0]);
};
</script>
</body>
```



graphRead.html

Access to client side files !!!

Network displayer graphXY



drawNet.js; Bavla/NetsJSON; Seminar 2016.

To do

- extend the list of info-attributes: `type` (simple, temporal), `twoModeOrg`, `nStrong`, `nRelations`, `planar`, `trace` (network's "genealogy"), `style` (default), `legend`, ...
- `style` can be attached as an attribute also to an element (node, link) thus changing the default settings;
- functions in JavaScript (or some other language) can be assigned to nodes/links. Define computation in such network.
- icons can be used for visualization of nodes Font Awesome, Material Icons , ..., tests;
- links can be visually represented in many different ways that can be described in `style`;
- NetsJSON can be used for archiving and exchanging networks – converters from/to NetsJSON.

. . . to do

- add the visualization of arcs with arrows Directed Graph Editor, D3 Tips and Tricks;
- to be included in NetsD3.js: spring embedder Force and editor Springy. Matrix with permutations.
- can some attributes be renamed: `from`, `tail`, `nodeA`, `source` → `n1` ; `to`, `head`, `nodeB`, `target` → `n2` ; . . . Maybe the simplest solution is a replace in some text editor;
- implement saving of the obtained SVG picture to a file: Export SVG with Style, d3js/SVG Export demo, . . .
- include in NetsJSON elements useful in some applications, such as, hooks or in/out-ports; background image, etc.

References I


Batagelj, V.: Python Packages for Networks. Encyclopedia of Social Network Analysis and Mining. Reda Alhajj, Jon Rokne (Eds.), 2nd Ed, Springer, 2017. PDF


Batagelj, V., Maltseva, D.: Temporal bibliographic networks. Journal of Informetrics, 14(2020)1, 101006. PDF


Batagelj, V.: Nets. https://github.com/bavla/Nets


Batagelj, V.: NetsJSON. https://github.com/bavla/netsJSON


Batagelj, V., Praprotnik, S.: An algebraic approach to temporal network analysis based on temporal quantities. Social Network Analysis and Mining, 6(2016)1, 1-22. PDF


Bodlaj, J., Cerinšek, M.: Network Data File Formats. Encyclopedia of Social Network Analysis and Mining. Reda Alhajj, Jon Rokne (Eds.), 2nd Ed, Springer, 2018. PDF

References II

Brath, R., Jonker, D.: Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data. John Wiley & Sons, Indianapolis, Indiana, 2015.

Groß, B., Bohnacker, H., Laub, J., Lazzeroni, C.: Generative Design: Visualize, Program, & Create with JavaScript in p5.js. Princeton Architectural Press, 2018.

Meeks, E.: D3.js in Action: Data visualization with JavaScript, 2nd Edition. Manning, 2017.