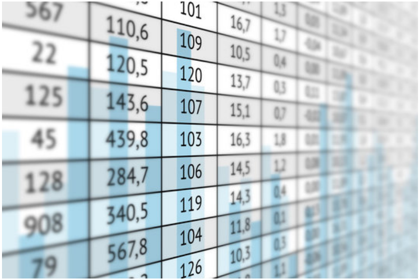
Generalized data tables

Vladimir Batagelj
IMFM Ljubljana, IAM & FAMNIT UP Koper, FMF UL Ljubljana

1365. sredin seminar
On Zoom, 21. May 2025

Outline



Vladimir Batagelj: vladimir.batagelj@fmf.uni-lj.si
Current version of slides (May 21, 2025 at 06 : 28): slides PDF
https://github.com/bavla/symData/

Data tables

Traditional data analysis is based on a (simple) *data table* $\mathbf{T}_{U \times V}$, over a set of *units* U and a set of unit properties or *variables* V. The entry $T[u, v]$ contains the (measured) *value* of a property $v \in V$ at a unit $u \in U$. The values are simple data: numbers, logical values, dates, and character strings.

LOREOM IPSUMAT CONSTRUM ETRIM KIST	Lorem ipsum dolor	Amister umarkl finish	Gatolep odio un accums	Tortores remus justica
LOREM DOLOR SIAMET	8 288	123 %	YES	$89
CONSECTER ODIO	123	87 %	NO	$129
GATOQUE ACCUMS	1 005	12 %	NO	$99
SED HAC ENIM REM	56	69 %	N/A	$199
REMPUS TORTOR JUST	5 554	18 %	NO	$999
FCELISQUE SED MORBI	12 569	112 %	NO	$123
SENECTUS URNA MOSTUM	779	33 %	N/A	$56
VESIBU LORIS SET MURTIL	6 112	27 %	YES	$684

Data tables

When encoding data, sometimes there is a need for unusual values such as unknown, meaningless, and infinite. Spreadsheet programs such as Excel can be used to prepare, maintain, and perform simple analyses of such tables.

In recent times, there are more and more examples of data that go beyond simple tables - the values can be composite data: time series, sequences of events, sets of strings, intervals, distributions, graphs, etc. Sometimes we add one or more (weighted) relations between the units – we get a network. If we convert the table **T** into triples $(u, v, T[u, v])$, we get a knowledge graph.

In the seminar, we will look at examples in R to see how generalized tables are represented, read, used, and saved to a file in modern programming languages, and can be exchanged between programs written in different programming languages.

```
> wdir <- "C:/Users/vlado/docs/papers/2025/SDA/test"
> setwd(wdir)
> nm <- c("Anna", "Betty", "Charles", "Doris", "Edward")
> sx <- c("F", "F", "M", "F", "M")
> ag <- c(29, 30, 28, 33, 27)
> D <- data.frame(name=nm,sex=factor(sx,levels=c("M","F")),
+    age=ag)
> D
     name sex age
1    Anna   F  29
2   Betty   F  30
3 Charles   M  28
4   Doris   F  33
5  Edward   M  27
> write.csv2(D,file="DFex1.csv")
```

CSV files.

A variable can also be a list of structured (composed) values

```
> ph <- list(
+   data.frame(loc=c("home","work"),
+     num=c("051123456","051654321")),
+   data.frame(loc="home",num="051121212"),
+   data.frame(loc=c("work","home"),
+     num=c("051987654","051456789")),
+   data.frame(loc="work",num="051356356"),
+   data.frame(loc="home",num="051717171"))
> D$phone <- ph
> D
     name sex age                              phone
1    Anna   F  29 home, work, 051123456, 051654321
2   Betty   F  30                  home, 051121212
3 Charles   M  28 work, home, 051987654, 051456789
4   Doris   F  33                  work, 051356356
5  Edward   M  27                  home, 051717171
> (P <- D$phone[1][[1]])
   loc       num
1 home 051123456
2 work 051654321
> P[P$loc=="home",]$num
[1] "051123456"
> write.csv2(D,file="DFex2.csv")
Error in utils::write.table(D, file = "DFex2.csv", col.names = N
  unimplemented type 'list' in 'EncodeElement'
```

JSON

Data tables
V. Batagelj
Data tables
The World Factbook
JSON
References
References

WWW, GitHub/JSON, Kaggle, Ian Coleman.

```
> library(jsonlite)
> F <- fromJSON("C:/Users/vlado/DL/data/kaggle/CIA/factbook.json")
> names(F)
[1] "countries" "metadata"
> length(names(F$countries))
[1] 259
> head(names(F$countries))
[1] "world"          "afghanistan"    "akrotiri"       "albania"        "algeria"
[6] "american_samoa"
> tail(names(F$countries))
[1] "west_bank"      "western_sahara" "yemen"          "zambia"         "zimbabwe"
[6] "european_union"

> str(F$countries$slovenia,max.level=2)
> F$countries$slovenia$data$energy$electricity
> F$countries$slovenia$data$energy$electricity$exports
$kWh
[1] 7.972e+09
$global_rank
[1] 26
$date
[1] "2017"
> D <- as.data.frame(F$countries[["slovenia"]]$data$energy$electricity)

> names(F$countries$slovenia$data)
> names(F$countries$slovenia$data$people)
> P <- F$countries$slovenia$data$people$age_structure
> names(P)
[1] "0_to_14"     "15_to_24"     "25_to_54"     "55_to_64"     "65_and_over" "date"
> P$date
[1] "2020"
> P[[2]]
$percent
[1] 9.01
$males
[1] 98205
$females
[1] 91318
```

Data frames

The World Factbook / age structure

```
> N <- names(F$countries); n <- length(N)
> C <- NULL
> for(i in 1:n){
+   P <- F$countries[[i]]$data$people$age_structure
+   d <- NULL
+   for(j in 1:5){d <- rbind(d,P[[j]])}
+   row.names(d) <- names(P)[1:5]
+   C <- rbind(C,list(name=N[i],year=P$date,pop=as.data.frame(d)))
+ }
> head(C)
     name               year   pop
[1,] "world"            "2020" data.frame,3
[2,] "afghanistan"      "2020" data.frame,3
[3,] "akrotiri"         NULL   data.frame,0
[4,] "albania"          "2020" data.frame,3
[5,] "algeria"          "2020" data.frame,3
[6,] "american_samoa" "2020" data.frame,3
> C <- as.data.frame(C)
> str(C[1:2,])
'data.frame':   2 obs. of  3 variables:
 $ name:List of 2
  ..$ : chr "world"
  ..$ : chr "afghanistan"
 $ year:List of 2
  ..$ : chr "2020"
  ..$ : chr "2020"
 $ pop :List of 2
  ..$ :'data.frame':    5 obs. of  3 variables:
  .. ..$ percent:List of 5
  .. .. ..$ 0_to_14     : num 25.3
  .. .. ..$ 15_to_24    : num 15.4
  .. .. ..$ 25_to_54    : num 40.7
  .. .. ..$ 55_to_64    : num 9.09
  .. .. ..$ 65_and_over: num 9.49
  .. ..$ males   :List of 5
  .. .. ..$ 0_to_14     : int 1005229963
  .. .. ..$ 15_to_24    : int 612094887
  .. .. ..$ 25_to_54    : int 1582759769
  .. .. ..$ 55_to_64    : int 341634893
  .. .. ..$ 65_and_over: int 326234036
  .. ..$ females:List of 5
  .. .. ..$ 0_to_14     : int 941107507
```

Data frames

The World Factbook / age structure

```
> as.data.frame(C[1,3])
             percent      males    females
0_to_14        25.33 1005229963  941107507
15_to_24       15.42  612094887  572892123
25_to_54       40.67 1582759769 1542167537
55_to_64        9.09  341634893  357176983
65_and_over     9.49  326234036  402994685
> (sp <- as.data.frame(C[which(N=="slovenia"),3]))
             percent  males females
0_to_14        14.84 160134  151960
15_to_24        9.01  98205   91318
25_to_54       40.73 449930  406395
55_to_64       14.19 148785  149635
65_and_over    21.23 192420  253896
> (vp <- unname(unlist(sp$percent)))
[1] 14.84  9.01 40.73 14.19 21.23
> sp[3,]
          percent  males females
25_to_54    40.73 449930  406395

> write(toJSON(C,auto_unbox=TRUE),file="popAge.JSON")

> Q <- fromJSON("popAge.json")
> names(Q)
[1] "name" "year" "pop"
> dim(Q)
[1] 259   3
> (p1 <- as.data.frame(Q$pop[1]))
             percent      males    females
0_to_14        25.33 1005229963  941107507
15_to_24       15.42  612094887  572892123
25_to_54       40.67 1582759769 1542167537
55_to_64        9.09  341634893  357176983
65_and_over     9.49  326234036  402994685
> (ps <- as.data.frame(Q$pop[which(Q$name=="slovenia")]))
             percent  males females
0_to_14        14.84 160134  151960
15_to_24        9.01  98205   91318
25_to_54       40.73 449930  406395
55_to_64       14.19 148785  149635
65_and_over    21.23 192420  253896
```

Tidyverse / Tibble

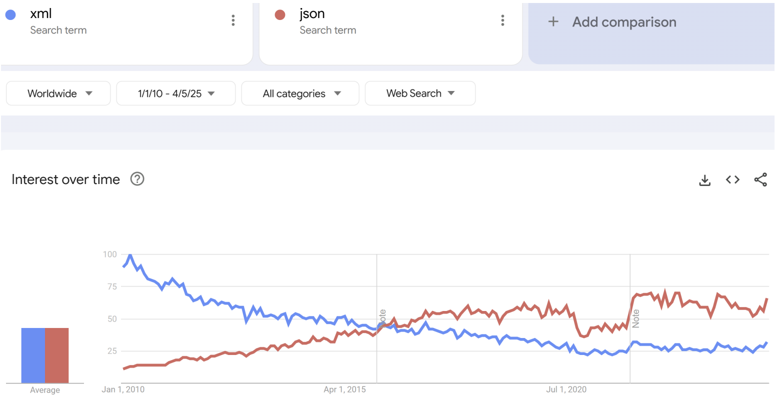
Open science

In Open Science (Wikipedia, 2025), there is a growing emphasis on publishing research data following the FAIR principles (Findable, Accessible, Interoperable, Reusable) (GoFAIR, 2016). Adhering to these standards ensures the verifiability of results and enables alternative analyses. Additionally, open data contributes to greater diversity in datasets, supporting the development and testing of new methodologies.

In symbolic data analysis, the starting point is usually a generalized (symbolic) data table, where variable values can be structured (combinations of primitive values). These require specialized external (file-based) and internal (in-memory) representations. Ideally, the two representations would be compatible.

This presentation focuses on file-based descriptions of symbolic data tables, which can facilitate seamless data exchange between symbolic data analysis tools.

Google trends XML : JSON



Google trends XML : JSON

Most formats for structured data are based on XML or JSON, with JSON increasingly favored in modern applications – see Figure 1. JSON description is not only a valid JavaScript expression but also uses data structures that are natively supported by most programming languages (e.g., *R*, *Python*, *Julia*, *C++*) (JSON, 2017; ECMAScript, 2024; Batagelj, 2016).

To understand why JSON is lightweight, consider the representation of a person in XML and it's JSON equivalent:

```
XML:

<person>
<first-name>Janez</first-name>
<last-name>Novak</last-name>
</person>
```

```
JSON:

{
"firstname":"Janez",
"lastname":"Novak"
}
```

JSON

The basic idea of the JSON (JavaScript Object Notation) format (RFC 8259 and ECMA-404) is that in JavaScript, a JSON data description is evaluated into a JavaScript data value (object).

There are two problems related to numerical values

- most programming languages support the IEEE 754 IEEE Standard for Floating-Point Arithmetic that includes (section 6) also special values **Infinity**, **-Infinity**, and **Not_a_Number** (`+Inf`, `-Inf`, `NaN`). JavaScript allows numbers of unlimited precision, but doesn't support these special values.
- in data analysis, the value **Not_Available** (`NA`) is used to indicate a missing value

See also: Infinity and JSON; JSON status in ECMAScript; JSON in Python 3; Issue 98.

JSON

The new Javascript standard ECMAScript® 2026 finally introduced (section 6.1.6.1.) values **+Infinity**, **-Infinity**, **NaN**, and **Undefined**.

In R the library `jsonlite` already supports `+Inf`, `-Inf`, `NaN`, and `NA`.

JSON variants:

JSON WP, JSON-LD WP, UBJSON WP, Smile WP

JSON

Inf, NA, NaN

```
> (M <- matrix(c(1:4,NaN,NA,Inf,+Inf,-Inf),ncol=3,byrow=TRUE))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4  NaN   NA
[3,]  Inf  Inf -Inf
> (m <- toJSON(M))
[[1,2,3],[4,"NaN","NA"],["Inf","Inf","-Inf"]]
> fromJSON(m)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4  NaN   NA
[3,]  Inf  Inf -Inf
> t <- '[[1,2,3],[4,"NaN","NA"],["Inf","Infinity","-Inf"]]'
> fromJSON(t)
     [,1]  [,2]       [,3]
[1,] "1"   "2"        "3"
[2,] "4"   "NaN"      NA
[3,] "Inf" "Infinity" "-Inf"
>
```

JSON tools

Oxygen,

json-editor online, jsonformatter jsoneditor, json-editor, Altova json-tools, jsonlint, json-buddy,

phcode

kate editor Windows

jsoncrack.

Data table format

Beyond the raw data, it is essential to incorporate metadata in the file description. When designing such descriptions, it is advisable to rely on established standards, such as persistent identifiers (DOIs, ORCID, ROR) (DPC, 2025), ISO standards(ISO, 2025), schema.org (Schema, 2025), Dublin Core (DCMI, 2025), etc.

Adopting these practices ensures better interoperability, reusability, and long-term preservation of symbolic data.

Acknowledgments

The computational work reported in this presentation was performed in R. The code and data are available at https://github.com/bavla/symData/.

This work is partly supported by the Slovenian Research Agency ARIS (research program P1-0294 and research project J5-4596), and prepared within the framework of the COST action CA21163 (HiTEc).

References I

Batagelj, V. (2016). Network visualization based on JSON and D3.js (slides). *Second European Conference on Social Networks. June 14-17, 2016, Paris*. https://github.com/bavla/netsJSON/blob/master/doc/netVis.pdf.

DCMI (3 April 2025). The Dublin Core Metadata Initiative. https://www.dublincore.org/.

Digital Preservation Coalition (2025). Persistent identifiers. In Digital Preservation Handbook. https://www.dpconline.org/handbook/technical-solutions-and-tools/persistent-identifiers.

ECMA (December 2017). Introducing JSON – ECMA-404: The JSON data interchange syntax. 2nd edition. https://www.json.org/json-en.html.

ECMA (June 2024). ECMA-262: ECMAScript® 2024 Language Specification. 15th Edition. https://ecma-international.org/publications-and-standards/standards/ecma-262/.

Go FAIR (2016). FAIR Principles. https://www.go-fair.org/fair-principles/.

ISO (2025). The International Organization for Standardization. https://www.iso.org/.

References II

Schema.org (24 March 2025). Schemas for structured data. https://schema.org/.

Wikipedia (1 April 2025). Open science. https://en.wikipedia.org/wiki/Open_science.