Knowledge Space Theory Input and Output

Cord Hockemeyer

September 12, 2024

Abstract

This document explains basic read and write operations for knowledge structures and knowledge spaces available in R through the **kstIO** package.

Contents

1 Introduction

Knowledge Space Theory (Doignon and Falmagne, 1999) is a set- and order-theoretical framework, which proposes mathematical formalisms to operationalize knowledge structures in a particular domain. There exist several R packages for knowledge space theory, namely **kst**, **kstMatrix**, **pks**, and **DAKS** which use different forms of representations for knowledge spaces and structures. The **kstIO** package provides functions for reading and writing those structures from/to files.

2 File Formats

Over time and in different research groups with knowledge space theory, different file formats have evolved.

2.1 Matrix Format

The probably simplest and most direct approach is to store the information in a binary ASCII matrix where a "1" in row i and column j means that item j is element of state/response pattern i.

There is no separating character between the columns, and there should be no trailing whitespace at the end of the line. The last line of the matrix must carry an EndOfLine - in most editors (except vi) this means an empty line after the matrix.

2.2 KST Tools Format

This format (Hockemeyer, 2001) extends the matrix format by two preceding header lines containing the number of items and the number of states/response patterns, respectively.

2.3 SRBT Tools Format

This format (Pötzi and Wesiak, 2001) extends the KST tools format by yet another preceding header line with format and content metadata. This new header line has the format

```
#SRBT v2.0 <struct> ASCII <comment>
```

where `<struct>` specifies the type of data stored in the file and `<comment>` is an optional arbitrary comment.

The following data types are supported by the respective `kstIO` functions:

- basis
- data
- space
- structure
- relation

2.4 CSV Format

In CSV format, there is a separating comma between the columns. The matrix is preceded by a head line containing the item IDs.

3 Specific File Types

3.1 Base Files

For base files, some special rules apply. They are available only in KST, SRBT tools, and CSV format. Their matrix part differs from the other files in that it contains "0", "1", and "2". A "1" means that the state is minimal for the item and a "2" means that it is not (but contains the item). A "0" stands (as always) for the state not containing the item.

For `kbase` files, the encoding information `"ASCII"` is missing because `kbase` files are always in ASCII format.

Note: While the respective functions in **kst** and **kstIO** use the term *base* in their names, the `struct>` term in the SRBT header line is *basis*.

3.2 Surmise Function Files

In surmise unction files, the matrix is preceded by an additional column containing the ID of the item for which the respective matrix row is a clause.

3.3 (Surmise) Relation Files

Also for relation files, some special rules apply. As relation file formats were never defined in KST tools format, these files are available in SRBT tools, matrix, and CSV format only. Like for `base` files, the encoding information `"ASCII"` is missing because `relation` files are always in ASCII format.

3.4 Example

Below, you see an example of a small knowledge structure file in SRBT format.

```
#SRBT v2.0 structure ASCII
3
5
000
100
110
101
111
```

3.5 Binary File Formats

The KST and SRBT Tools User Manuals (Hockemeyer, 2001; Pötzi and Wesiak, 2001) define also binary file formats. These formats are not supported by the **kstIO** package.

4 Output functions

There are five output functions in the **kstIO** package.

- `write_kbase()`
- `write_kdata()`
- `write_kspace()`
- `write_kstructure()`
- `write_surmiserelation()`

These functions have the same calling scheme

```
write_XXX(x, filename, format="SRBT")
```

where `x` denotes the data structure to be written, `filename` the name of the file to be created, and `format` the file format (`"SRBT"`, `"KST"`, or `"matrix"` as described in Section 2 above. The knowledge structure or knowledge space can be in set-based format (classes `kspace` or `kstructure`) or in matrix format. Please note that for bases, only the `SRBT` and `KST` formats are valid.

```
> # Obtain data from the pks package
> data(DoignonFalmagne7)
> ksp <- kspace(kstructure(as.pattern(DoignonFalmagne7$K, as.set=TRUE)))
> b <- kbase(ksp)
> d <- as.binmat(DoignonFalmagne7$N.R, uniq=FALSE)
> r <- as.relation(ksp)
> ksp

{{}, {"a"}, {"b"}, {"a", "b"}, {"a", "b", "c"}, {"a", "b", "d"}, {"a",
 "b", "c", "d"}, {"a", "b", "c", "e"}, {"a", "b", "c", "d", "e"}}
```

```
> b

{{"a"}, {"b"}, {"a", "b", "c"}, {"a", "b", "d"}, {"a", "b", "c", "e"}}

> head(d)

     a b c d e
[1,] 0 0 0 0 0
[2,] 0 0 0 0 0
[3,] 0 0 0 0 0
[4,] 0 0 0 0 0
[5,] 0 0 0 0 0
[6,] 0 0 0 0 0

> # Write data to files
> write_kstructure(ksp, "DF7.struct")
> write_kspace(ksp, "DF7.space", format="matrix")
> write_kbase(b, "DF7.bas", format="KST")
> write_kdata(d, "DF7.data", format="SRBT")
> write_surmiserelation(r, "DF7.rel")
```

The resulting base file, for example, looks like the following:

```
> txt <- readLines("DF7.bas")
> for (i in txt)
+   cat(paste(i, "\n", sep=""))

5
5
10000
01000
22100
22010
22201
```

5 Input Functions

There are six input functions in the **kstIO** package.

- `read_kbase()`
- `read_kdata()`
- `read_kfamset()`
- `read_kspace()`
- `read_kstructure()`
- `read_surmiserelation()`

These functions have a similar calling scheme. For bases, data, famsets, and knowledge structures it is

```
d <- read_kXXX(filename, format="SRBT")
```

where `filename` denotes the file to be read and `format` the file format (`"SRBT"`, `"KST"`, or `"matrix"` as described in Section 2 above, or `"auto"` (default) for automatic format detection. Please note that automatic format detection works slightly heuristically and therefore might err between `"KST"` and `"matrix"` formats under rare circumstances.

For surmise relations and knowledge spaces, there is an additional (optional) parameter:

```
d <- read_YYYY(filename, format="SRBT", close=FALSE)
```

The return values depend on the type of file to be read: for `read_kfamset()`, `read_kspace()`, `read_kstructure()`, and `read_kbase()`, it is a list containing two elements, `matrix` and `sets` which contain the read knowledge structure/space/base as a binary matrix and in set-based form (i.e. as object of class `kspace`, `kstructure`, or `kbase`), respectively. For `read_kdata()`, a binary matrix is returned. For `read_surmiserelation()`, a list with two elements, `relation` and `matrix` is returned which contain the surmise relation and its incedence matrix, repectively.

If `close` is `TRUE`, the respective structure is closed, i. e. in case of a knowledge space, it is closed under union, and in case of a surmise relation, it is closed under reflexivity and transitivity.

```
> # Read the data files stored before
> read_kfamset("DF7.space")

$matrix
       a  b  c  d  e
 [1,] NA NA NA NA NA
 [2,]  0  0  0  0  0
 [3,]  1  0  0  0  0
 [4,]  0  1  0  0  0
 [5,]  1  1  0  0  0
 [6,]  1  1  1  0  0
 [7,]  1  1  0  1  0
 [8,]  1  1  1  1  0
 [9,]  1  1  1  0  1
[10,]  1  1  1  1  1

$sets
{{}, {NA}, {"a"}, {"b"}, {"a", "b"}, {"a", "b", "c"}, {"a", "b", "d"},
 {"a", "b", "c", "d"}, {"a", "b", "c", "e"}, {"a", "b", "c", "d", "e"}}

> read_kstructure("DF7.struct", format="SRBT")

$matrix
       a  b  c  d  e
 [1,] NA NA NA NA NA
 [2,]  0  0  0  0  0
 [3,]  1  0  0  0  0
 [4,]  0  1  0  0  0
 [5,]  1  1  0  0  0
 [6,]  1  1  1  0  0
 [7,]  1  1  0  1  0
 [8,]  1  1  1  1  0
 [9,]  1  1  1  0  1

$sets
{{}, {NA}, {"a"}, {"b"}, {"a", "b"}, {"a", "b", "c"}, {"a", "b", "d"},
 {"a", "b", "c", "d"}, {"a", "b", "c", "e"}}

> read_kspace("DF7.space", format="matrix")
```

```
$matrix
       a  b  c  d  e
 [1,] NA NA NA NA NA
 [2,]  0  0  0  0  0
 [3,]  1  0  0  0  0
 [4,]  0  1  0  0  0
 [5,]  1  1  0  0  0
 [6,]  1  1  1  0  0
 [7,]  1  1  0  1  0
 [8,]  1  1  1  1  0
 [9,]  1  1  1  0  1
[10,]  1  1  1  1  1

$sets
{{}, {NA}, {"a"}, {"b"}, {"a", "b"}, {"a", "b", "c"}, {"a", "b", "d"},
 {"a", "b", "c", "d"}, {"a", "b", "c", "e"}, {"a", "b", "c", "d", "e"}}
```

```
> read_kbase("DF7.bas", format="auto")
```

```
$matrix
     a b c d e
[1,] 1 0 0 0 0
[2,] 0 1 0 0 0
[3,] 1 1 1 0 0
[4,] 1 1 0 1 0
[5,] 1 1 1 0 1

$sets
{{"a"}, {"b"}, {"a", "b", "c"}, {"a", "b", "d"}, {"a", "b", "c", "e"}}
```

```
> head(read_kdata("DF7.data"))
```

```
     a b c d e
[1,] 0 0 0 0 0
[2,] 0 0 0 0 0
[3,] 0 0 0 0 0
[4,] 0 0 0 0 0
[5,] 0 0 0 0 0
[6,] 0 0 0 0 0
```

```
> read_surmiserelation("DF7.rel")
```

```
$relation
A binary relation of size 5 x 5.

$matrix
   a  b  c  d  e
a NA NA NA NA NA
b  1  0  1  1  1
c  0  1  1  1  1
d  0  0  1  0  1
e  0  0  0  1  0
```

References

J.-P. Doignon and J.-C. Falmagne. *Knowledge Spaces.* Springer–Verlag, Berlin, 1999.

C. Hockemeyer. *KST Tools User Manual*, 2nd edition, 2001. https://kst.hockemeyer.at/techreports/KST-Tools_TechRep_FWF01.pdf.

S. Pötzi and G. Wesiak. *SRbT Tools User Manual*, 2001. https://kst.hockemeyer.at/techreports/SRBT-Tools_TechRep_FWF01.pdf.

Index