Package 'kstIO'

September 7, 2026

Version 0.6-0

Date 2026-09-06

Title Knowledge Space Theory Input/Output

Description Knowledge space theory by Doignon and Falmagne (1999) <doi:10.1007/978-3-642-58625-5> is a set- and order-theoretical framework which proposes mathematical formalisms to operationalize knowledge structures in a particular domain. The 'kstIO' package provides basic functionalities to read and write KST data from/to files to be used together with the 'kst', 'kstMatrix', 'pks', or 'DAKS' packages.

License GPL (>= 3)

Depends R (>= 4.4.0), sets, kstMatrix (>= 3.0-0), openxlsx2, readODS, tools

Suggests litedown

Author Cord Hockemeyer [aut, cre]

Maintainer Cord Hockemeyer <cord.hockemeyer@uni-graz.at>

NeedsCompilation no

Repository CRAN

Encoding UTF-8

Config/roxygen2/version 8.1.0

VignetteBuilder litedown

Date/Publication 2026-09-07 03:50:02 UTC

Contents

as.binaryMatrix 2
as.famset 3
read_kbase 4
read_kdata 5
read_kfamset 6
read_kspace 7

read_kstructure 8
read_surmisefunction 9
read_surmiserelation 10
write_kbase 11
write_kdata 12
write_kfamset 12
write_kspace 13
write_kstructure 14
write_surmisefunction 15
write_surmiserelation 16

Index **17**

`as.binaryMatrix` *Convert a family of sets from set to matrix representation*

Description

Convert a family of sets from set to matrix representation

Usage

```
as.binaryMatrix(x)
```

Arguments

`x`	Set object

Value

Matrix object

See Also

Other Utilities: `as.famset()`

Examples

```
m <- matrix(c(1,0,0,0,
              0,1,0,0,
              1,0,1,0,
              0,1,1,0,
              1,1,1,1), ncol=4, byrow=TRUE)
m
as.binaryMatrix(as.famset(m))
```

`as.famset` *Convert a family of sets from matrix to set representation*

Description

Convert a family of sets from matrix to set representation

Usage

```
as.famset(m, as.letters = TRUE)
```

Arguments

`m`	Binary matrix
`as.letters`	What to do if `is.nuill(colnames(x))`. Defaults to TRUE.

Value

Set object

If `colnames(x)` is set, the colnames are used as item identifiers for the set. Otherwise, letters or numbers are used, depending on the `as.letters` parameter.

See Also

Other Utilities: `as.binaryMatrix()`

Examples

```
m <- matrix(c(1,0,0,0,
              0,1,0,0,
              1,0,1,0,
              0,1,1,0,
              1,1,1,1), ncol=4, byrow=TRUE)
m
as.famset(m)
```

read_kbase *Read a basis from file*

Description

Read a basis from file

Usage

```
read_kbase(
  filename,
  format = "auto",
  as.letters = TRUE,
  header = TRUE,
  sep = ",",
  enforce = TRUE
)
```

Arguments

filename	Name of the basis file
format	File format (default "auto")
as.letters	How to name items (default TRUE)
header	For spreadsheet files: does it include a header line (default TRUE)
sep	Cell separator for CSV files (default ',')
enforce	Do we enforce basis properties (default TRUE)?

Value

List with two elements `matrix` and `sets` containing the respective representations of the basis read

Please note that the terms 'base' and 'basis' are used synonymously and interchangeably.

See Also

Other Functions for reading and writing families of sets: `read_kdata()`, `read_kfamset()`, `read_kspace()`, `read_kstructure()`, `write_kbase()`, `write_kdata()`, `write_kfamset()`, `write_kspace()`, `write_kstructure()`

Examples

```
kstMatrix::xpl$basis
tempfile <- paste0(tempdir(), "/test.bas")
write_kbase(kstMatrix::xpl$basis, tempfile, format="SRBT")
read_kbase(tempfile)
```

read_kdata *Read a data set from file*

Description

Read a data set from file

Usage

```
read_kdata(
  filename,
  format = "auto",
  as.letters = TRUE,
  header = TRUE,
  sep = ","
)
```

Arguments

filename	Name of the data file
format	File format (default "auto")
as.letters	How to name items (default TRUE)
header	For spreadsheet files: does it include a header line (default TRUE)
sep	Cell separator for CSV files (default ',')

Value

kmdata object

Please note that data sets can only be stored in matrix format - sets cannot contain multiple identical response patterns.

See Also

Other Functions for reading and writing families of sets: read_kbase(), read_kfamset(), read_kspace(), read_kstructure(), write_kbase(), write_kdata(), write_kfamset(), write_kspace(), write_kstructure()

Examples

```
head(kstMatrix::xpl$data)
tempfile <- paste0(tempdir(), "/testdata.ods")
write_kdata(kstMatrix::xpl$data, tempfile)
head(read_kdata(tempfile))
```

read_kfamset *Read a family of sets from file*

Description

Read a family of sets from file

Usage

```
read_kfamset(
  filename,
  format = "auto",
  as.letters = TRUE,
  header = TRUE,
  sep = ",",
  enforce = TRUE
)
```

Arguments

filename	Name of the file
format	File format (default "auto")
as.letters	How to name items (default TRUE)
header	For spreadsheet files: does it include a header line (default TRUE)
sep	Cell separator for CSV files (default ',')
enforce	Do we enforce famset properties (default TRUE)?

Value

List with two elements `matrix` and `sets` containing the respective representations of the famset read

See Also

Other Functions for reading and writing families of sets: `read_kbase()`, `read_kdata()`, `read_kspace()`, `read_kstructure()`, `write_kbase()`, `write_kdata()`, `write_kfamset()`, `write_kspace()`, `write_kstructure()`

Examples

```
kstMatrix::xpl$basis
tempfile <- paste0(tempdir(), "/test.bas")
write_kbase(kstMatrix::xpl$basis, tempfile, format="KST")
read_kbase(tempfile)
```

read_kspace *Read a knowledge space from file*

Description

Read a knowledge space from file

Usage

```
read_kspace(
  filename,
  format = "auto",
  as.letters = TRUE,
  header = TRUE,
  sep = ",",
  enforce = TRUE
)
```

Arguments

filename	Name of the file
format	File format (default "auto")
as.letters	How to name items (default TRUE)
header	For spreadsheet files: does it include a header line (default TRUE)
sep	Cell separator for CSV files (default ',')
enforce	Do we enforce knowledge space properties (default TRUE)?

Value

List with two elements `matrix` and `sets` containing the respective representations of the knowledge space read

Please note that enforcing knowledge space properties (i.e. computing the closure under union) is computationally complex and therefore may take quite some time.

See Also

Other Functions for reading and writing families of sets: `read_kbase()`, `read_kdata()`, `read_kfamset()`, `read_kstructure()`, `write_kbase()`, `write_kdata()`, `write_kfamset()`, `write_kspace()`, `write_kstructure()`

Examples

```
kstMatrix::xpl$space
tempfile <- paste0(tempdir(), "/test.spc")
write_kspace(kstMatrix::xpl$space, tempfile, format="SRBT")
read_kspace(tempfile)
```

`read_kstructure` *Read a knowledge structure from file*

Description

Read a knowledge structure from file

Usage

```
read_kstructure(
  filename,
  format = "auto",
  as.letters = TRUE,
  header = TRUE,
  sep = ",",
  enforce = TRUE
)
```

Arguments

`filename`	Name of the file
`format`	File format (default "auto")
`as.letters`	How to name items (default TRUE)
`header`	For spreadsheet files: does it include a header line (default TRUE)
`sep`	Cell separator for CSV files (default ',')
`enforce`	Do we enforce knowledge structure properties (default TRUE)?

Value

List with two elements `matrix` and `sets` containing the respective representations of the knowledge structure read

See Also

Other Functions for reading and writing families of sets: `read_kbase()`, `read_kdata()`, `read_kfamset()`, `read_kspace()`, `write_kbase()`, `write_kdata()`, `write_kfamset()`, `write_kspace()`, `write_kstructure()`

Examples

```
kstMatrix::xpl$space
tempfile <- paste0(tempdir(), "/test.struc")
write_kstructure(kstMatrix::xpl$space, tempfile, format="SRBT")
read_kstructure(tempfile)
```

read_surmisefunction *Read a surmise function from file*

Description

Read a surmise function from file

Read an attribution function from file

Usage

```
read_surmisefunction(filename, sep = ",", enforce = TRUE)

readattributionfunction(filename, sep = ",")
```

Arguments

filename	Name of the file
sep	Cell separator for CSV files (default ',').
enforce	Do we enforce surmise function properties (default TRUE)?

Value

kmsurmisefunction object

Surmise and attribution functions can only be stored in spreadsheet files. The exact file type (CSV, ODS, or XLSX) is automatically inferred from the filename extension.

kmsurmisefunction object

Surmise and attribution functions can only be stored in spreadsheet files. The exact file type (CSV, ODS, or XLSX) is automatically inferred from the filename extension.

See Also

Other Functions for reading and writing surmise and attribution functions and relations: read_surmiserelation(), write_surmisefunction(), write_surmiserelation()

Examples

```
kstMatrix::xpl$sf
tempfile <- paste0(tempdir(), "/test_sf.csv")
write_surmisefunction(kstMatrix::xpl$sf, tempfile)
cat(readLines(tempfile), sep='\n')
read_surmisefunction(tempfile)
```

`read_surmiserelation` *Read a surmise relation from file*

Description

Read a surmise relation from file

Read a attribution relation from file

Usage

```
read_surmiserelation(
  filename,
  format = "auto",
  header = TRUE,
  as.letters = TRUE,
  sep = ",",
  enforce = TRUE
)

read_attributionrelation(
  filename,
  format = "auto",
  header = TRUE,
  as.letters = TRUE,
  sep = ","
)
```

Arguments

`filename`	Name of the file
`format`	File format (default "auto")
`header`	For spreadsheet files: Does it contain a header row (default TRUE)?
`as.letters`	Do we provide numeric or alphanumeric item names if no colnames are specified?
`sep`	Cell separator for CSV files (default ',').
`enforce`	Do we enforce surmise relation properties (default TRUE)?

Value

`kmsurmisefunction` object

`kmsurmisefunction` object

See Also

Other Functions for reading and writing surmise and attribution functions and relations: `read_surmisefunction()`, `write_surmisefunction()`, `write_surmiserelation()`

Examples

```
kstMatrix::xpl$sr
tempfile <- paste0(tempdir(), "/test_sf.csv")
write_surmiserelation(kstMatrix::xpl$sr, tempfile)
cat(readLines(tempfile), sep='\n')
read_surmiserelation(tempfile)
```

`write_kbase` *Write a basis to file*

Description

Write a basis to file

Usage

```
write_kbase(x, filename, format = NULL, sep = ",", enforce = TRUE)
```

Arguments

`x`	Basis (in set (`kbase`) or matrix (`kmbasis`) format)
`filename`	Name of the file to be written
`format`	File format
`sep`	Cell separator for CSV files
`enforce`	Should basis properties be enforced (only for matrix form)?
	If the file format is not specified, `write_kbase()` looks at the filename extension. If it is some spreadsheet type (CSV, ODS, or XLSX), the respective file format is chosen, otherwise SRBT is selected.

See Also

Other Functions for reading and writing families of sets: `read_kbase()`, `read_kdata()`, `read_kfamset()`, `read_kspace()`, `read_kstructure()`, `write_kdata()`, `write_kfamset()`, `write_kspace()`, `write_kstructure()`

Examples

```
kstMatrix::xpl$basis
tempfile <- paste0(tempdir(), "/test.bas")
write_kbase(kstMatrix::xpl$basis, tempfile, format="SRBT")
cat(readLines(tempfile), sep='\n')
```

write_kdata *Write a data set to file*

Description

Write a data set to file

Usage

```
write_kdata(x, filename, format = NULL, sep = ",")
```

Arguments

x	Data set (only in matrix (kmdata) format)
filename	Name of the file to be written
format	File format
sep	Cell separator for CSV files
	If the file format is not specified, write_base() looks at the filename extension. If it is some spreadsheet type (CSV, ODS, or XLSX), the respective file format is chosen, otherwise KST is selected.

See Also

Other Functions for reading and writing families of sets: read_kbase(), read_kdata(), read_kfamset(), read_kspace(), read_kstructure(), write_kbase(), write_kfamset(), write_kspace(), write_kstructure()

Examples

```
head(kstMatrix::xpl$data)
tempfile <- paste0(tempdir(), "/testdata.csv")
write_kdata(kstMatrix::xpl$data, tempfile)
cat(head(readLines(tempfile)), sep='\n')
```

write_kfamset *Write a family f sets to file*

Description

Write a family f sets to file

Usage

```
write_kfamset(x, filename, format = NULL, sep = ",", enforce = TRUE)
```

Arguments

x	Family of sets (in set (kfamset) or matrix (kmfamset) format)
filename	Name of the file to be written
format	File format
sep	Cell separator for CSV files
enforce	Should knowledge structure properties be enforced (only for matrix form)?

If the file format is not specified, write_kfamset() looks at the filename extension. If it is some spreadsheet type (CSV, ODS, or XLSX), the respective file format is chosen, otherwise SRBT is selected.

For larger families of sets, it might be better suitable and way more efficient to save and later load them with the save() and load() functions - if they are to be used only with R.

See Also

Other Functions for reading and writing families of sets: read_kbase(), read_kdata(), read_kfamset(), read_kspace(), read_kstructure(), write_kbase(), write_kdata(), write_kspace(), write_kstructure()

Examples

```
kstMatrix::xpl$basis
tempfile <- paste0(tempdir(), "/test.bas")
write_kbase(kstMatrix::xpl$basis, tempfile, format="KST")
cat(readLines(tempfile), sep='\n')
```

---

write_kspace *Write a knowledge space to file*

Description

Write a knowledge space to file

Usage

```
write_kspace(x, filename, format = NULL, sep = ",", enforce = TRUE)
```

Arguments

x	Knowledge space (in set (kspace) or matrix (kmspace) format)
filename	Name of the file to be written
format	File format
sep	Cell separator for CSV files

enforce	Should knowledge space properties be enforced (only for matrix form)? If the file format is not specified, write_kspace() looks at the filename extension. If it is some spreadsheet type (CSV, ODS, or XLSX), the respective file format is chosen, otherwise SRBT is selected. For larger knowledge spaces, it might be better suitable and way more efficient to save and later load them with the save() and load() functions - if they are to be used only with R.

See Also

Other Functions for reading and writing families of sets: read_kbase(), read_kdata(), read_kfamset(), read_kspace(), read_kstructure(), write_kbase(), write_kdata(), write_kfamset(), write_kstructure()

Examples

```
kstMatrix::xpl$space
tempfile <- paste0(tempdir(), "/test.spc")
write_kspace(kstMatrix::xpl$space, tempfile, format="matrix")
cat(readLines(tempfile), sep='\n')
# Using a non-space object
write_kspace(kstMatrix::xpl$basis, tempfile, format="matrix")
cat(readLines(tempfile), sep='\n')
```

---

write_kstructure *Write a knowledge structure to file*

Description

Write a knowledge structure to file

Usage

```
write_kstructure(x, filename, format = NULL, sep = ",", enforce = TRUE)
```

Arguments

x	Knowledge space (in set (kstructure) or matrix (kmstructure) format)
filename	Name of the file to be written
format	File format
sep	Cell separator for CSV files
enforce	Should knowledge structure properties be enforced (only for matrix form)? If the file format is not specified, write_kstructure() looks at the filename extension. If it is some spreadsheet type (CSV, ODS, or XLSX), the respective file format is chosen, otherwise SRBT is selected. For larger knowledge structures, it might be better suitable and way more efficient to save and later load them with the save() and load() functions - if they are to be used only with R.

See Also

Other Functions for reading and writing families of sets: read_kbase(), read_kdata(), read_kfamset(), read_kspace(), read_kstructure(), write_kbase(), write_kdata(), write_kfamset(), write_kspace()

Examples

```
kstMatrix::xpl$space
tempfile <- paste0(tempdir(), "/test.struc")
write_kstructure(kstMatrix::xpl$space, tempfile, format="matrix")
cat(readLines(tempfile), sep='\n')
# Enforcing knowledge structure properties for a non-structure object
write_kstructure(kstMatrix::xpl$basis, tempfile, format="SRBT")
cat(readLines(tempfile), sep='\n')
```

---

write_surmisefunction *Write a surmise function to file*

Description

Write a surmise function to file

Write a attribution function to file

Usage

```
write_surmisefunction(x, filename, sep = ",")

write_attributionfunction(x, filename, sep = ",")
```

Arguments

x	Attribute function (kmattributionfunction object))
filename	Name of the file to be written
sep	Cell separator for CSV files Attribute functions can only be stored in spreadsheet format. The concrete file format (CSV, ODS, or XLSX) is selected based on the filename extension.

See Also

Other Functions for reading and writing surmise and attribution functions and relations: read_surmisefunction(), read_surmiserelation(), write_surmiserelation()

Examples

```
kstMatrix::xpl$sf
tempfile <- paste0(tempdir(), "/test_sf.csv")
write_surmisefunction(kstMatrix::xpl$sf, tempfile)
cat(readLines(tempfile), sep='\n')
```

`write_surmiserelation` *Write a surmise function to file*

Description

Write a surmise function to file

Write an attribution function to file

Usage

```
write_surmiserelation(x, filename, format = NULL, sep = ",")

write_attributionrelation(x, filename, format = NULL, sep = ",")
```

Arguments

`x`	Attribution relation (`kmattributionrelation` object)
`filename`	Name of the file to be written
`format`	File format
`sep`	Cell separator for CSV files Attribution relations are available only in matrix format. If `format` is `NULL`, SRBT is selected, If the filename has a spreadsheet file type extension, the respective type is selected.#

See Also

Other Functions for reading and writing surmise and attribution functions and relations: `read_surmisefunction()`, `read_surmiserelation()`, `write_surmisefunction()`

Examples

```
kstMatrix::xpl$sr
tempfile <- paste0(tempdir(), "/test_sf.csv")
write_surmiserelation(kstMatrix::xpl$sr, tempfile)
cat(readLines(tempfile), sep='\n')
```

Index

∗ **Functions for reading and writing families of sets**
  read_kbase, 4
  read_kdata, 5
  read_kfamset, 6
  read_kspace, 7
  read_kstructure, 8
  write_kbase, 11
  write_kdata, 12
  write_kfamset, 12
  write_kspace, 13
  write_kstructure, 14
∗ **Functions for reading and writing surmise and attribution functions and relations**
  read_surmisefunction, 9
  read_surmiserelation, 10
  write_surmisefunction, 15
  write_surmiserelation, 16
∗ **Utilities**
  as.binaryMatrix, 2
  as.famset, 3

as.binaryMatrix, 2
as.binaryMatrix(), *3*
as.famset, 3
as.famset(), *2*

read_attributionrelation *(*read_surmiserelation*)*, 10
read_kbase, 4
read_kbase(), *5–8*, *11–15*
read_kdata, 5
read_kdata(), *4*, *6–8*, *11–15*
read_kfamset, 6
read_kfamset(), *4, 5*, *7, 8*, *11–15*
read_kspace, 7
read_kspace(), *4–6*, *8*, *11–15*
read_kstructure, 8
read_kstructure(), *4–7*, *11–15*
read_surmisefunction, 9
read_surmisefunction(), *10*, *15, 16*
read_surmiserelation, 10
read_surmiserelation(), *9*, *15, 16*
readattributionfunction *(*read_surmisefunction*)*, 9

write_attributionfunction *(*write_surmisefunction*)*, 15
write_attributionrelation *(*write_surmiserelation*)*, 16
write_kbase, 11
write_kbase(), *4–8*, *12–15*
write_kdata, 12
write_kdata(), *4–8*, *11*, *13–15*
write_kfamset, 12
write_kfamset(), *4–8*, *11, 12*, *14, 15*
write_kspace, 13
write_kspace(), *4–8*, *11–13*, *15*
write_kstructure, 14
write_kstructure(), *4–8*, *11–14*
write_surmisefunction, 15
write_surmisefunction(), *9, 10*, *16*
write_surmiserelation, 16
write_surmiserelation(), *9, 10*, *15*